THE CAUCHY-SCHWARZ DIVERGENCE AND ENTROPY-BASED DENSITY-WEIGHTED ACTIVE LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET ENES ŞEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2022

Approval of the thesis:

THE CAUCHY-SCHWARZ DIVERGENCE AND ENTROPY-BASED DENSITY-WEIGHTED ACTIVE LEARNING

submitted by **MEHMET ENES ŞEN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Date: 02.09.2022

ABSTRACT

THE CAUCHY-SCHWARZ DIVERGENCE AND ENTROPY-BASED DENSITY-WEIGHTED ACTIVE LEARNING

Şen, Mehmet Enes
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Barış Nakiboğlu

September 2022, 70 pages

The general framework for active learning is explained. The existing active learning strategies are surveyed. The information-theoretic measures such as the entropy and the mutual information are analyzed as active learning objectives. The use of divergence measures in density-weighted active learning is discussed. A novel density-weighted active learning algorithm, based on Cauchy-Schwarz divergence and entropy, is introduced and compared with the state-of-the-art active learning strategies.

Keywords: Active Learning, Cauchy-Schwarz Divergence, Entropy, Mutual Information, Information-Theory, Machine Learning

ÖZ

CAUCHY-SCHWARZ UZAKLIĞI VE ENTROPİ TABANLI YOĞUNLUK AĞIRLIKLI AKTİF ÖĞRENME

Şen, Mehmet Enes
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Barış Nakiboğlu

Eylül 2022 , 70 sayfa

Genel aktif öğrenme mimarisi açıklanmıştır. Literatürdeki mevcut aktif öğrenme stratejileri özetlenmiştir. Entropi ve ortak enformasyon miktarı gibi bilgi-teorisi temelli ölçülerin aktif öğrenme hedefleri olarak nasıl kullanılabileceği incelenmiştir. Bilgi-teorisi temelli uzaklık ölçülerinin yoğunluk ağırlıklı aktif öğrenmede yoğunluk ağırlığı olarak kullanımı tartışılmıştır. Cauchy-Schwarz uzaklığı ve entropiye dayanan yeni bir yoğunluk ağırlıklı aktif öğrenme algoritması önerilmiştir. Önerilen yöntem güncel aktif öğrenme stratejileri ile karşılaştırılmıştır.

Anahtar Kelimeler: Aktif Öğrenme, Cauchy-Schwarz Uzaklığı, Entropi, Ortak Enformasyon Miktarı, Bilgi-Teorisi, Makine Öğrenmesi

To my family

ACKNOWLEDGMENTS

Foremost, I would like to thank Assist. Prof. Dr. Barış Nakiboğlu. Without his endless support, completing this thesis would not be possible. He always guided me in this work.

I would like to thank Assist. Prof. Dr. Serkan Sarıtaş for his suggestions and corrections for the thesis.

I would like to express my gratitude to my beloved family. They kept me motivated during this long work.

Finally, I would like to thank all my friends and colleagues who supported me during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Ω	The set of all points in the data pool
ω	The dummy index for a point in the data pool (i.e., an element of Ω)
Q	The dummy index for a set of points in the data pool (i.e., a subset of Ω)
X_ω/X_Ω/X_Q	The random column vector for the data of the point ω / of the points in Ω / of the points in Q
x_ω/x_Ω/x_Q	The realization of the random column vector for the data of the point ω / of the points in Ω / of the points in Q
d	The number of features (i.e., the dimension) of the data of any point
$z_{1,\omega} \ldots z_{d,\omega}$	The features of x_ω (i.e., $x_\omega = [z_{1,\omega} \ldots z_{d,\omega}]^T$)
Y_ω/Y_Ω/Y_Q	The random variable for the label of the point ω / of the points in Ω / of the points in Q
y_ω/y_Ω/y_Q	The realization of the random variable for the label of the point ω / of the points in Ω / of the points in Q
(x_ω, y_ω)	The data-label pair associated with the point ω
$(x_\mathsf{Q}, y_\mathsf{Q})$	The data-label pairs associated with the set of points Q
c	The number of distinct labels
Q_t	The set of points selected at iteration t
$\hat{y}(\theta, x)$	The learner's prediction for the label of x
L_0	The set of the labeled points in the data pool initially
L_t	The set of the labeled points in the data pool at the end of iteration t

U_t	The set of the unlabeled points in the data pool at the end of iteration t (i.e., $\Omega \setminus \mathsf{L}_t$))
θ	The realization of the random parameter vector of the learner
$\theta^{(t)}$	The parameter vector of the learner at the end of iteration t
Θ	The random variable for the parameter vector of the learner
ϑ	The set of possible values of the parameter vector
$\ell(\mathsf{L}, \theta)$	The loss function of the learner
θ_{L}	The parameter vector of the learner trained by the data//label pairs in the labeled set L (i.e., $(y_{\mathsf{L}}, x_{\mathsf{L}})$)
x_{test}	The set of the data in the test set
y_{test}	The set of x_{test}'s actual labels
$\hat{y}_{test}$	The set of the learner's label predictions for x_{test}
$\mathbf{f}(\cdot)$	The surrogate objective function for an active learning strategy

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The supervised machine learning is the task of learning a function that maps an input to an output. A machine learning model figures out the mapping function using a training set. The training set consists of the representative input-output pairs. Supervised learning problems can be divided into classification problems and regression problems. The desired output (i.e., the label) is an element of a finite set in the classification problems; the label is a real number in the regression problems. The focus of this thesis will be the classification problems.

Each machine learning model has a parameter vector (i.e., the constants for the learning function) and a loss function. The parameters that minimize the loss function are determined in the training process. The task of the loss function is to evaluate how well the learner models the given training set: we minimize the loss function on the training set with the hope that this will lead to the best performance (i.e., the learner's accuracy on actual inputs). Therefore, different training sets yield different parameters. If the training set enlarges, one should train the learner with the new training set.

The supervised learning requires many representative input-output pairs, as the training set, to learn the function precisely. To train a classifier, one needs a reasonable amount of correctly classified data-label pairs. Nowadays, one can find enormous amount of unlabeled data for a supervised machine learning task. We will refer to the gathered data for a specific task as the data pool. However, there is no way to use this tremendous amount of data for a supervised learning task without a labeling process which is expensive, time-consuming, and requires a human annotator.

Active learning is an important field that significantly reduces the labeling effort. Passive learning, the main alternative to active learning, is the traditional way. In passive learning, the training set is formed without any strategy, and the training set is fixed at the beginning. An oracle (i.e., an external source that reveals actual labels) labels the unlabeled data to create the training set as much as possible. The idea of active learning is enlarging the training set iteratively by selecting the most informative points with the learner's involvement at each iteration. The training starts with a small group of labeled points in active learning. The most critical points in the data pool are determined by the learner's participation in the selection process. These points are labeled by the oracle and added to the training set. The learner parameters are updated with the new training set, and this loop repeats until one gets an accurate parameter for the learner. Since the training set is formed with the most informative data points at each iteration, one can expect better performance with less labeled data.

There are many existing active learning strategies. The most used active learning strategies can be summarized as follows. The expected error reduction strategy selects the point that reduces the expected future error most on the unlabeled data set, see [25]. The variance reduction strategy is another active learning strategy. The objective of variance reduction-based methods is to select the point that reduces the learner's output variance. Another one is the expected model change strategy. Choosing the point that changes the learner's parameters the most is an active learning strategy called expected model change. In the query by committee strategy, the point that the committee of the learners with different hypotheses disagree most is selected, see [25]. One of the most used active learning strategies is uncertainty sampling. In uncertainty sampling, the strategy is to choose the data point that the learner is most uncertain about its label, see [14]. The existing approaches are discussed in detail in Section 2.4.

A universally superior active learning algorithm outperforming all other active learning algorithms has not been found yet. The performance of the existing methods varies according to the task, the data pool, and the initial labeled set. In some cases, even random sampling (i.e., forming a training dataset randomly without using any active learning method) yields better performance than the active learning methods. One critical problem of the existing active learning algorithms is not considering whether the selected points represent the data pool, see [27, 21]. An active learning algorithm might be an inadequate selection strategy when the representativeness of a point is disregarded because the algorithm may result in over-exploited and under-exploited regions in the data pool, see [27, 21]. Density-weighted active learning algorithms may address this issue by selecting both informative and representative queries to label. Density-weighted methods weight the active learning objective's output by considering if the query represents others. Thus density-weighted techniques select both informative and representative points. Our focus on this thesis will be adopting a new method for density-weighted active learning using information-theoretic quantities such as divergence and entropy.

1.2 Proposed Method

In this thesis, our objective is to propose a density-weighted active learning algorithm using a divergence as the representativeness measure. A divergence is a measure of distance between two probability distributions, and different types of divergences exist (see Chapter 4). We propose a novel active learning selection strategy named the Cauchy-Schwartz divergence-weighted entropy-based selection strategy. The proposed model assigns higher weights to the candidate queries that reduce the divergence between the estimated distribution of the labeled dataset and the estimated distribution of the data pool. The idea is that the points that reduce the divergence between the training set and the data pool are the most representative ones of the data pool in terms of distribution. We propose working with the Cauchy-Schwartz divergence, see Definition 4.1.4. In estimating probability density functions, we use kernel density estimation, a non-parametric way of estimating probability distribution from the data (see Section 4.2.1). We combine the Cauchy-Schwartz divergence with the uncertainty sampling, an active learning strategy based on selecting the points where the learner's uncertainty about its label is highest (see Section 2.4.5 and Section 3.1). We show the experimental improvement observed with our proposed method.

1.3 The Outline of the Thesis

The remaining part of the thesis consists of five chapters. The following clauses summarize the thesis structure.

- Chapter 2 introduces to the general active learning framework. Some of the popular existing active learning algorithms, different types of active learning, and some of the existing active learning strategies are also discussed.
- Chapter 3 focuses on the information-theoretic measures as active learning objective functions and specifically investigates entropy and mutual information as querying objectives.
- Chapter 4 discusses density-weighted active learning and some of the common measures for density-weighted methods. The proposed model is introduced. A novel Cauchy-Schwartz divergence weighted entropy-based model is presented. Experiment methodology, results, and comparison with other recent models were given at the end of the chapter.
- Chapter 5 concludes the thesis with a discussion of possible extensions and future work on the proposed model.

CHAPTER 2

ACTIVE LEARNING

2.1 Introduction to Active Learning

To train a learner for a supervised machine learning task, the training set must include the labels for the instances. However, most of the time, the collected dataset for a real-life task does not have labels. Labeling the data is an expensive process. An oracle, usually a human annotator, labels the unlabeled samples in the data pool before the training process to create a training dataset in supervised machine learning tasks. However, the annotation process is usually time-consuming and expensive for real-world cases such as speech recognition, information extraction, and classification, see [25].

Active learning aims decrease the annotation effort (i.e., the required number of labeled instances for the learner) using a selection algorithm for the training set's elements. The main goal of the active learning approaches is to find the most informative points in the unlabeled dataset. Active learning aims to reduce the number of labeled samples required to get a learner to perform similar to or better than a learner trained by using most of the samples in terms of accuracy and the generalization performance (i.e., accuracy on the unseen data), see [20]. Instead of initially forming the whole training dataset, the instances in the training set are selected iteratively by an algorithm with the learner's involvement in the active learning framework. Training starts with a small labeled dataset, and the number of labeled points grows in each iteration. Thus, the learner has a chance to address the most difficult points in the data pool in each iteration. In the end, active learning tries to get an accurate learner with the least cost by choosing the most informative data points from the unlabeled dataset in each iteration. The distinction between different active learning methods is in their approach to finding the informative points in the data pool. Except for the method of determining the informative points, the general framework is similar for all active learning methods.

Similar ideas are discussed in the context of education in literature, see [2]. We can give the following example as a real-life analogue to active learning. Consider a student who participates in class actively and asks questions to the teacher when he does not understand a topic and a student who listens passively. Assuming both students have comparable intelligence, if the first student selects his questions wisely, then this student might learn faster than the other student. In this context, active learning deals with finding the most critical questions. Assuming the students are the learners who need to be trained, the same logic applies to learning machines.

The active learning scenarios considered in the literature can be classified into three groups: query synthesis, stream-based selective sampling, and pool-based sampling, see [26]. In the query synthesis scenario, the learner can ask the label of any sample from the input space to the oracle, assuming that the input space is defined appropriately, and the active learning algorithm needs to choose the samples to be queried from input space, see [26]. In the stream-based selective sampling scenario, the learner samples a stream of samples at each time, and the active learning algorithm decides whether to discard or to label the samples, see [26]. In the pool-based sampling scenario, a large pool of samples is already acquired from the source, and the active learning algorithm determines the samples that will be labeled from the data pool, see [26]. The focus of this thesis will be on the pool-based sampling scenario.

2.2 Pool-Based Active Learning Framework

In the pool-based sampling scenario, we have a large data pool consisting of the data points, and only a few data points are labeled initially. The symbol Ω represents the set of all data points in the data pool. The dummy index ω represents a point from Ω (i.e., $\omega \in \Omega$). The data pool consists of labeled and unlabeled data points (i.e., $\Omega = \mathsf{L} \cup \mathsf{U}$). U and L are the sets of unlabeled and labeled points in the data pool Ω, respectively. Note that the labeled part of the data pool (i.e., L) is the training set for the learner. The set variable Q, a subset of U (i.e., $\mathsf{Q} \subset \mathsf{U}$), represents the selected points by the active learning algorithm. The data of a point ω is represented with x_ω, which is a d dimensional column vector. The label of a point ω is represented with y_ω, which is an element of a finite set for the classification problems (i.e., $y_\omega \in \{1, \dots, c\}$ $\forall \omega \in \Omega$). The label of an unlabeled point ω is represented with Y_ω, which is an integer value random variable. That is, y_ω is the realization of Y_ω. Since the active learning framework is iterative, we identify the iteration as a subscript to express the set variable's state at the end of iteration t. For example, L_t is the set of labeled points at the end of iteration t. To express multiple points' data or labels, the subscript of the data (i.e., x) and the label (i.e., y or Y) are allowed to be sets as well. For example, x_Q represents the data of the points in Q.

Assumption 2.2.1 *The points in the data pool are fixed initially, and all the functions have access to the data of the points in the data pool (i.e., x_Ω). Thus, it is always assumed that x_Ω is given to all functions in the remaining parts, even if a function is not explicitly depend on x_Ω.*

Each machine learning model has a parameter vector and a loss function. In the framework, θ is the learner's parameter vector, and the set of all possible parameter vectors is ϑ. The loss function of the machine learning model is denoted by $\ell(\mathsf{L}, Y_\mathsf{L}, \theta)$. The value of the loss function depends on the training set L, the labels of the points in the training set Y_L, and learner's parameters θ. The parameter vector that minimizes the loss function is determined in the training process. At the end of iteration t, the learner's parameter vector $\theta^{(t)}$ is given in (2.1). Note that the learner trained by $(x_{\mathsf{L}},\, Y_{\mathsf{L}})$ as the training set is denoted by $\theta_{(x_{\mathsf{L}}, Y_{\mathsf{L}})}$.

$$\theta^{(t)} = \operatorname*{argmin}_{\theta \in \vartheta} \ell(\mathsf{L}_t,\, Y_{\mathsf{L}_t}, \theta) \tag{2.1}$$

$$= \theta_{(x_{\mathsf{L}_t}, Y_{\mathsf{L}_t})} \tag{2.2}$$

The general active learning framework consists of querying, labeling, and inference steps. The framework's goal is to find queries that will be labeled and used in the learner's training. Since the output of the querying part depends on the learner's parameters, the framework iteratively selects queries from a large pool of unlabeled sets of points. Thus, the learner trained with the enlarged training set participates in the selection process of new points in each iteration. The block diagram of the general model is given in Figure 2.1. Note that t refers to the iteration t.

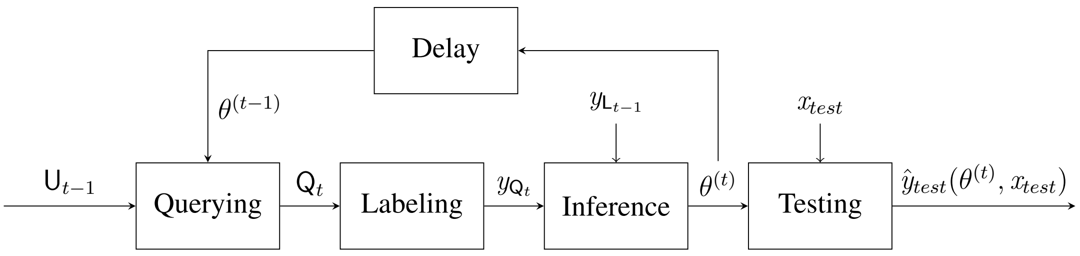

Figure 2.1: General Active Learning Framework

Q_t, U_t, and L_t are the set of points selected for labeling at iteration t, the set of all unlabeled points at the end of iteration t, and the set of all labeled points in the data pool at the end of iteration t, respectively. Thus,

$$\mathsf{U}_t = \Omega \setminus \mathsf{L}_t \tag{2.3}$$

$$= \mathsf{U}_{t-1} \setminus \mathsf{Q}_t. \tag{2.4}$$

2.2.1 Querying

Querying the data points from the unlabeled set is the part where active learning algorithms come into play. The algorithm selects k data points among the unlabeled dataset to send to the oracle. The selection strategy may depend on the learner's current state, the current state of the set of unlabeled points, or other possible indicators.

The selection algorithm aims to find the most informative points worth including in the learner's training. Note that k represents the size of the query set Q. If k is one, the framework is called sequential mode active learning, if k bigger than one, it is called batch mode active learning, see [25]. In this part, the question is how should we choose the query set Q. In the general framework, one needs to select a surrogate objective for his algorithm to approximate the increase in the model's performance when the labels of a candidate query data x_{Q} are learned, see [25, 30]. Then, the algorithm will select the query set that maximizes this surrogate objective among all candidate query sets. The surrogate objective function will be referred to as $\mathbf{f}(.)$.

In the querying process, the data of the query set x_{Q}, which maximizes the surrogate objective function $\mathbf{f}$, is found and handed over to the oracle. The surrogate objective function $\mathbf{f}$ is a real-valued function defined for all subsets of x_Ω (i.e., the surrogate objective directly defined on query set Q). The function $\mathbf{f}$ is chosen so as to be an estimate of the informativeness of the query for the learner. Since the objective function $\mathbf{f}$ is an approximation of the informativeness of the query set, it is assumed that the higher outputs of $\mathbf{f}$ yield more informative query sets, see [30]. For example, in the uncertainty sampling, $\mathbf{f}$ measures the learner's uncertainty for a query set. Since the informativeness of a point is considered to increase with the learner's uncertainty about that point, the query set that gives a larger output (i.e., uncertainty) is selected, see Section 2.4.5. Q_t represents the set of selected points by the active learning algorithm in iteration t. Since the labeled set determines the parameter vector, we used Θ to represent the parameters as a random variable. Θ is the random variable for the parameter vector θ, and the labeled set determines its value.

$$\mathsf{Q}_t = \underset{\mathsf{Q}\subset\mathsf{U}_{t-1}, |\mathsf{Q}|=k}{\operatorname{argmax}} \ \mathbf{f}\big(\mathsf{Q}|\mathsf{L}_{t-1}, Y_{\mathsf{L}_{t-1}}, \Theta^{(t-1)}\big) \tag{2.5}$$

The surrogate objective function $\mathbf{f}$ is specific to the active learning algorithm. Generally, the value of $\mathbf{f}$ depends on the state of the labeled/unlabeled points in the data pool L_{t-1}/U_{t-1}, the labels of the labeled points until the end of the previous iteration $Y_{\mathsf{L}_{t-1}}$, the learner's parameters in the previous iteration $\theta^{(t-1)}$, and the data of the points in Ω. To simplify the equations, we assumed that $\mathbf{f}$ has access to all of the data, labels of the labeled points, unlabeled points, and model parameters all the time. Thus, we use $\mathbf{f}(\mathsf{Q})$ instead of $\mathbf{f}\big(\mathsf{Q}|\mathsf{L}_{t-1}, Y_{\mathsf{L}_{t-1}}, \Theta^{(t-1)}\big)$ in the remaining parts.

2.2.2 Labeling

The oracle is the external source that annotates the unlabeled data for the learner's training. In the active learning framework, the queried data x_{Q} is given to the oracle, and the oracle annotates the data. Annotating the data means labeling the data (i.e., revealing the labels of the points).

$$y_{\mathsf{Q}} = Labeling(x_{\mathsf{Q}}) \tag{2.6}$$

The labeled indices by the oracle are added to the labeled set in each iteration. Note that the indices added to the labeled set at iteration t are equivalent to those queried at iteration t.

$$\mathsf{L}_t = \mathsf{L}_{t-1} \cup \mathsf{Q}_t \tag{2.7}$$

Note that the queried indices in iteration t are removed from the unlabeled set U_{t-1} at the end of the labeling process.

$$\mathsf{U}_t = \mathsf{U}_{t-1} \setminus \mathsf{Q}_t \tag{2.8}$$

In this part, it is assumed that the labels given by the oracle are correct; see Assumption 2.2.2.

Assumption 2.2.2 *The labels generated by the oracle are the same as the actual labels (i.e., the annotation of the oracle is always correct).*

2.2.3 Inference

Note that according to Assumption 2.2.3, the dependence of the label on data is determined by the learner's parameters and the parametric model $\mathsf{p}_\theta(y|\, x)$ is known when θ is given.

Assumption 2.2.3 *Given the parameter θ, which is an element of ϑ, the components of label vector Y_{L} are independent and conditional distribution of each label Y_ω is determined by the value of X_ω as follows*

$$\mathbf{P}[Y_{\mathsf{L}} = y_{\mathsf{L}}|\, \theta] = \prod\nolimits_{\omega \in \mathsf{L}} \mathsf{p}_\theta(y_\omega|\, x_\omega)\,. \qquad \forall \mathsf{L} \subset \Omega, \theta \in \vartheta. \tag{2.9}$$

Newly labeled data is added to the training set in each iteration, so the learner's parameter vector θ needs to be updated using the enlarged training set (i.e., $(x_{\mathsf{L}_t}, y_{\mathsf{L}_t})$). To find θ, the parameter vector that yields smallest loss function is determined in this step. Some of the most used loss functions provided below. Note that $\hat{y}_\omega(\theta, x_\omega)$ represents the output of the learner (i.e., label prediction) for a given data point and the given learner parameters.

$$\ell(\mathsf{L}, Y_{\mathsf{L}}, \theta) = \sum_{\omega \in \mathsf{L}} (Y_\omega - \hat{y}_\omega(\theta, x_\omega))^2 \qquad \textbf{Mean Squared Error} \tag{2.10}$$

$$\ell(\mathsf{L}, Y_{\mathsf{L}}, \theta) = \sum_{\omega \in \mathsf{L}} |Y_\omega - \hat{y}_\omega(\theta, x_\omega)| \qquad \textbf{Mean Absolute Error} \tag{2.11}$$

$$\ell(\mathsf{L}, Y_{\mathsf{L}}, \theta) = \sum_{\omega \in \mathsf{L}} -\log \mathsf{p}_\theta(Y_\omega | x_\omega) \qquad \textbf{Log Loss (Cross Entropy)} \tag{2.12}$$

For example, suppose the learner uses negative log-likelihood as a loss function (2.12) in training. In that case, the learner's parameters at iteration t are the parameters that minimize the loss function at iteration t (2.13).

$$\begin{aligned} \theta^{(t)} &= \operatorname*{argmin}_{\theta \in \vartheta} \ell(\mathsf{L}_t, Y_{\mathsf{L}_t}, \theta) & (2.13) \\ &= \operatorname*{argmin}_{\theta \in \vartheta} \sum_{\omega \in \mathsf{L}_t} -\log \mathsf{p}_\theta(y_\omega | x_\omega) & (2.14) \\ &= \operatorname*{argmax}_{\theta \in \vartheta} \log \mathbf{P}[Y_{\mathsf{L}} = y_{\mathsf{L}} | \theta] & (2.15) \\ &= \theta_{(x_{\mathsf{L}_t}, Y_{\mathsf{L}_t})} & (2.16) \end{aligned}$$

Note that minimizing the negative log-likelihood is equivalent to maximum likelihood (2.14). However, the loss function and the training process are specific to the machine learning model.

2.2.4 Testing

In the testing part, the accuracy of updated learner $\theta^{(t)}$ is tested on the test set (x_{test}, y_{test}) to evaluate the learner's performance. Note that the test set is already provided in the beginning to verify the learner's performance, and it consists of the data-label pairs (x_{test}, y_{test}). The test set is not included in the data pool and is only used for testing purposes. For the testing, the data of the test set is given to the learner, and the learner's predictions for the labels of the test data are collected. The test set's data is given to the learner, and the learner's predictions for the test data are collected in the testing part. The learner's label predictions are shown below for a single data point and test set. Note that the prediction of each point in the test set is the label with the largest probability according to the parametric model given in Assumption 2.2.3.

$$\hat{y}_\omega(\theta^{(t)}, x_\omega) = \underset{\hat{y}}{\operatorname{argmax}}\, \mathsf{p}_{\theta^{(t)}}(\hat{y}|\, x_\omega) \tag{2.17}$$

$$\hat{y}_{test}(\theta^{(t)}, x_{test}) = \underset{\hat{y}_{test}}{\operatorname{argmax}} \prod_{\omega \in test} \mathsf{p}_{\theta^{(t)}}(\hat{y}_\omega|\, x_\omega) \tag{2.18}$$

The accuracy (i.e., the ratio of correct classifications) is found by comparing $\hat{y}_{test}$ and y_{test}. The process can be ended if the user is satisfied with the learner's performance (i.e., if the accuracy is higher than a predefined threshold). Otherwise, the active labeling process iteratively continues from the querying step.

2.2.5 General Algorithm

Algorithm 1 summarizes the whole process mentioned in previous subsections as an algorithm.

Algorithm 1 Pseudo Code of General Active Learning Algorithm

```
//initial model parameters
$\theta^{(0)} \leftarrow$ estimated model with $\mathsf{L}_{(0)}$
//initialize iteration variable
$t = 0$
while true do
  //increment iteration variable
  $t{+}{+}$
  //select queries that maximizes f
  $\mathsf{Q} \leftarrow \text{argmax}_{\mathsf{Q} \subset \mathsf{U}^{t-1}, |\mathsf{Q}|=k}\, \mathbf{f}\big(\mathsf{Q}\big)$
  //hand over queried data points to oracle
  $Y_{\mathsf{Q}} \leftarrow Labeling(x_{\mathsf{Q}})$
  //update labeled and unlabeled sets
  $\mathsf{L}_t = \mathsf{L}_{t-1} \cup \mathsf{Q}$
  $\mathsf{U}_t \leftarrow \mathsf{U}_{t-1} \setminus \mathsf{Q}$
  //update model parameters
  $\theta^{(t)} = \text{argmin}_{\theta \in \vartheta}\, \ell(\mathsf{L}_t,\, Y_{\mathsf{L}_t}, \theta)$
  //predict the labels of test data
  $\hat{y}_{test}(\theta^{(t)}, x_{test}) = \text{argmax}_{\hat{y}_{test}} \prod_{\omega \in test} \mathsf{p}_{\theta^{(t)}}(\hat{y}_\omega |\, x_\omega)$
  //check the accuracy of the learner
  if $Acc(y_{test}, \hat{y}_{test}) > \tau$ then
    return $\theta^{(t)}$
  end if
end while
```

2.3 Batch Mode Active Learning

The objective of active learning is to choose a subset Q of size k (i.e., $|\mathsf{Q}| = k$) from a large data pool. If k is greater than one, the framework is called batch mode active learning. Instead of labeling one by one, selecting multiple instances for labeling in each iteration would be more efficient for large scale projects. However, there are two main challenges in batch mode active learning.

1. $\mathtt{f}(\mathsf{Q})$ directly takes the candidate query set as input, not the individual points. If the size of Q (i.e., k) is larger than one, the number of candidate subsets can be too many compared to the sequential mode of active learning. For example, suppose the size of the unlabeled set (i.e., $|\mathsf{U}|$) is a thousand, and the size of Q (i.e., k) is ten. In that case, there are trillions of possible subset assignments for Q (i.e., $\binom{1000}{10} \approx 263 \times 10^{21}$). If k were one (i.e., sequential mode), the number of the possible subsets would be a thousand. Note that one must calculate the objective function for all candidate Q sets. This means that for batch mode, one must make approximately 263×10^{19} times more calculations than the sequential mode in this example. The difference would be much bigger for the larger unlabeled sets and k. For most active learning strategies, calculating the objective function requires too many calculations, especially if calculating the objective function involves retraining the learner, see [30]. Therefore, direct use of the batch mode active learning is not applicable in terms of computational complexity for the complex active learning strategies.

2. The points in Q should be non-redundant (i.e., the objective function $\mathtt{f}(\mathsf{Q})$ should be maximized jointly for the elements of Q), see [30]. Sampling more than one point is not reasonable if the samples contain similar information. For example, assume you have an objective function $\mathtt{f}$ that depends on the learner's parameters θ, and k is five. If you select five points at once without updating the model parameters, there is a strong possibility that these five points might be similar so that four of those points might be redundant.

2.3.1 Greedy Algorithm for Batch Mode Querying

In batch mode active learning, selecting the k individual best points is not necessarily optimal. The objective function $\mathbf{f}(\mathsf{Q})$ should be maximized jointly (i.e., not individually) for the elements of Q to get the optimal solution. However, the computational complexity of joint maximization is too much. There are $\binom{|\mathsf{U}|}{k}$ different candidate Q sets for batch mode active learning in each iteration. In other words, one needs to calculate the objective function $\binom{|\mathsf{U}|}{k}$ times to find Q that maximizes $\mathbf{f}(\mathsf{Q})$ by searching all possible candidates. Therefore, when k and $|\mathsf{U}|$ are high, this binomial problem is not applicable computationally in each iteration, especially for retraining-based active learning methods. The goal of the greedy approach is to get an approximate solution in a reasonable number of steps instead of finding the optimum solution that requires too many calculations.

Note that the definition of the submodular function is given in Definition 2.3.1. Submodularity is one of the conditions to apply greedy methods provided in this section.

Definition 2.3.1 *If the set function* $\mathbf{f} : 2^{\Omega} \to \mathbb{R}$ *is submodular, it satisfies one of the following conditions, see [23, 13].*

- $\forall A, B \subset \Omega$, *satisfy* $\mathbf{f}(A) + \mathbf{f}(B) \geq \mathbf{f}(A \cup B) + \mathbf{f}(A \cap B)$
- $\forall A \subset B \subset \Omega$ *and* $\omega \in \Omega - B$, *satisfy* $\mathbf{f}(A \cup \{\omega\}) - \mathbf{f}(A) \geq \mathbf{f}(B \cup \{\omega\}) - \mathbf{f}(B)$

We will discuss two different greedy algorithms. Algorithm 2 and Theorem 2.3.1 are for monotone, non-negative, and submodular objective maximization problems. Algorithm 3 and Theorem 2.3.2 are for non-negative and submodular objective maximization problems that are not necessarily monotone. Note that Q^* is used for the optimal solution (i.e., the optimization in (2.19)). Q_1^* is used for the output of Algorithm 2. Q_2^* is used for the output of Algorithm 3 in the remaining part.

$$\mathsf{Q}^* = \underset{\mathsf{Q} \subset \mathsf{U}, |\mathsf{Q}| = k}{\operatorname{argmax}} \; \mathbf{f}(\mathsf{Q}) \tag{2.19}$$

Theorem 2.3.1 *If the objective function is non-negative, sub-modular, and monotone (non-decreasing), one can use the* $1 - \frac{1}{e}$ $(\%63)$ *greedy approximation given in* (2.20)*, where* Q_1^* *is the query set found with Algorithm 2.*

$$\mathbf{f}(\mathsf{Q}_1^*) \geq \left[1 - \left(\frac{k-1}{k}\right)^k\right] \mathbf{f}(\mathsf{Q}^*) \geq \left(1 - \frac{1}{e}\right) \mathbf{f}(\mathsf{Q}^*) \tag{2.20}$$

Algorithm 2 Greedy algorithm for monotone, non-negative and submodular maximization [31, p. 49],[16, p. 276].

```
1: $\mathsf{Q}_1^* \leftarrow \emptyset$
2: for $i = 1 : k$ do
3:     $\omega \leftarrow \text{argmax}_{\omega \subset \mathsf{U},} \mathbf{f}(\mathsf{Q}_1^* \cup \{\omega\})$
4:     $\mathsf{Q}_1^* \leftarrow \mathsf{Q}_1^* \cup \{\omega\}$
5:     $\mathsf{U} \leftarrow \mathsf{U} \setminus \{\omega\}$
6: end for
7:
8: return $\mathsf{Q}_1^*, \mathsf{U}$
```

A proof of Theorem 2.3.1 may be found in [16, p. 268], [3, p. 7-8], or [13, p. 6-7].

Theorem 2.3.2 *If the objective function is sub-modular and non-negative, one can use the* $\frac{1}{e}$ $(\%37)$ *greedy approximation given in* (2.21)*, where* Q_2^* *is the query set found with Algorithm 3.*

$$\mathbf{E}[\mathbf{f}(\mathsf{Q}_2^*)] \geq \left[1 - \frac{1}{k}\right]^{k-1} \mathbf{f}(\mathsf{Q}) \geq \left(\frac{1}{e}\right) \mathbf{f}(\mathsf{Q}) \tag{2.21}$$

Algorithm 3 Greedy algorithm for non-negative and submodular maximization [31, p. 51], [3, p. 7].

```
1: $\mathsf{Q}_2^* \leftarrow \emptyset$
2: **for** $i = 1 : k$ **do**
3:     $M \leftarrow \operatorname{argmax}_{M \subset \mathsf{U}, |M|=k} \sum_{\omega \in M} \mathbf{f}(\mathsf{Q}_2^* \cup \{\omega\})$
4:     $\omega \leftarrow$ Uniformly selected random point in $M$
5:     $\mathsf{Q}_2^* \leftarrow \mathsf{Q}_2^* \cup \{\omega\}$
6:     $\mathsf{U} \leftarrow \mathsf{U} \setminus \{\omega\}$
7: **end for**
8:
9: **return** $\mathsf{Q}_2^*, \mathsf{U}$
```

A proof of Theorem 2.3.2 may be found in [3, p. 8-9].

Thus, using the given greedy algorithms, one can approach an approximate solution in $k|\mathsf{U}|$ steps instead of getting the best direct result in $\binom{|\mathsf{U}|}{k}$ steps. However, if the objective function $\mathbf{f}$ is non-negative, non-decreasing, and submodular, the output of the greedy algorithm guarantees $\%63$ of the optimal solution. Similarly, if $\mathbf{f}$ is non-negative and submodular, the mean of the greedy algorithm only guarantees $\%37$ of the optimal solution.

2.4 Existing Active Querying Strategies

This section will briefly explain some of the most used active learning strategies. A good survey of the existing active learning strategies can be found in [25].

2.4.1 Expected Error Reduction

The expected error reduction approach aims to query the point that reduces the expected future error most on the unlabeled data set U, see [25]. Although querying the data point, which reduces the expected future error most, is a reasonably good idea, the computational complexity of this method is very high. The querying objective of this strategy is given in (2.22). Note that up to this point, the loss function is used to find the error on the training set with the purpose of determining the learner parameters that minimize the loss on the training set. However, in this strategy, the loss function is used to find the expected future loss in the unlabeled set with the purpose of selecting the point that minimizes the expected future loss on the unlabeled set.

$$\omega^* = \operatorname*{argmin}_{\omega \in \mathsf{U}} \mathbf{E}\big[\ell(\mathsf{U} \setminus \{\omega\},\, Y_{\mathsf{U}\setminus\{\omega\}};\theta_{(x_{\mathsf{L}}\cup\{x_\omega\},\, Y_{\mathsf{L}}\cup\{Y_\omega\})})\big|\, \mathsf{L} \cup \{\omega\},\, Y_{\mathsf{L}}\big] \tag{2.22}$$

Since the actual labels of the candidate data points are unknown in the querying part, we need to train the model for each possible label assignment to calculate this expectation. The open form of the expectation in (2.22) is given in (2.23).

$$\omega^* = \operatorname*{argmin}_{\omega \in \mathsf{U}} \sum_{\tilde{y} \in J} \mathsf{p}_{\theta_{(x_{\mathsf{L}},\, Y_{\mathsf{L}})}}(\tilde{y}|\, x_\omega) \tag{2.23}$$
$$\sum_{\tilde{\omega} \in \mathsf{U}\setminus\{\omega\}} \sum_{y' \in J} \mathsf{p}_{\theta_{(x_{\mathsf{L}}\cup\{x_\omega\},(Y_{\mathsf{L}}\cup\{\tilde{y}\})}}(y'|\, x_{\tilde{\omega}})\, \ell(\{\tilde{\omega}\},\, y';\theta_{(x_{\mathsf{L}}\cup\{x_\omega\},(Y_{\mathsf{L}}\cup\{\tilde{y}\}))})$$

Note that J represents the set of all labels. However, retraining the learner for each possible candidate set and each potential label assignment is computationally expensive. Therefore, calculating the exact value of this expectation might not be practical for some tasks.

2.4.2 Variance Reduction

Another active learning strategy is building a selection strategy to reduce the learner's output variance. Minimizing the learner's output variance might reduce the generalization error indirectly, see [25]. This fact can be shown without explicitly calculating the expected generalization error reduction.

Remark 2.4.1 *This section considers the discussion on the framework given in [7]. In this framework, instead of the parametric model shown in Assumption 2.2.3, it is assumed that there is an unknown joint probability measure* $\mathbf{P}$ *determining the probabilities of all events of interest and the joint distribution of all random variables of interest, including the joint distribution of* $(X,\, Y)$.

A typical learning problem considered in this part; y is predicted from x, where $(x,\, y)$ obeys the unknown joint probability distribution $\mathbf{P}$, see [7]. The training set D consist of N input/output pairs (i.e., $(x_1, y_1), \ldots, (x_N, y_N)$). $f(x; D)$ is the constructed predictor using the training set and depends on the training set D. Under these conditions, the mean squared error of the predictor is

$$\mathbf{E}\big[(y - f(x;D))^2 \big|\, x, D\big]\,, \tag{2.24}$$

where $\mathbf{E}[\cdot|\, x, D]$ corresponds to conditional expectation w.r.t the probability measure $\mathbf{P}$ given x, D, see Remark 2.4.1.

$$\begin{aligned}
\mathbf{E}\big[(y - f(x;D))^2 \big|\, x, D\big] =& \mathbf{E}\big[((y - \mathbf{E}[y|\, x]) + (\mathbf{E}[y|\, x] - f(x;D)))^2 \big|\, x, D\big] && (2.25)\\
=& \mathbf{E}\big[(y - \mathbf{E}[y|\, x])^2 \big|\, x, D\big] + (\mathbf{E}[y|\, x] - f(x;D))^2 && (2.26)\\
& + 2\mathbf{E}[(y - \mathbf{E}[y|\, x])|\, x, D]\,(\mathbf{E}[y|\, x] - f(x;D)) \\
=& \underbrace{\mathbf{E}\big[(y - \mathbf{E}[y|\, x])^2 \big|\, x, D\big]}_{\text{noise}} + (\mathbf{E}[y|\, x] - f(x;D))^2 && (2.27)
\end{aligned}$$

Note that $\mathbf{E}[(y - \mathbf{E}[y|\, x])^2|\, x, D]$ is the variance of y given x and does not depend on the predictor or the training set, i.e.,

$$\mathbf{E}\big[(y - \mathbf{E}[y|\, x])^2 \big|\, x, D\big] = \mathbf{E}\big[(y - \mathbf{E}[y|\, x])^2 \big|\, x\big]\,. \tag{2.28}$$

Therefore $\mathbf{E}[(y - \mathbf{E}[y|\, x])^2|\, x, D]$ is called as noise that comes from the data, see [7]. We proceed with analyzing $(\mathbf{E}[y|\, x] - f(x; D))^2$.

$$\begin{aligned}
&\mathbf{E}\big[(\mathbf{E}[y|\, x] - f(x;D))^2 \big|\, x\big] \\
&\quad = \mathbf{E}\big[(\mathbf{E}[y|\, x] - \mathbf{E}[f(x;D)|\, x] + \mathbf{E}[f(x;D)|\, x] - f(x;D))^2 \big|\, x\big] \\
&\quad = \mathbf{E}\big[(\mathbf{E}[y|\, x] - \mathbf{E}[f(x;D)|\, x])^2 \big|\, x\big] + \mathbf{E}\big[(\mathbf{E}[f(x;D)|\, x] - f(x;D))^2 \big|\, x\big] \\
&\qquad + 2\mathbf{E}[(\mathbf{E}[y|\, x] - \mathbf{E}[f(x;D)|\, x])\,(\mathbf{E}[f(x;D)|\, x] - f(x;D))|\, x] \\
&\quad = \mathbf{E}\big[(\mathbf{E}[y|\, x] - \mathbf{E}[f(x;D)|\, x])^2 \big|\, x\big] + \mathbf{E}\big[(\mathbf{E}[f(x;D)|\, x] - f(x;D))^2 \big|\, x\big] \\
&\qquad + 2\mathbf{E}[\mathbf{E}[y|\, x] - \mathbf{E}[f(x;D)|\, x]|\, x]\,\mathbf{E}[\mathbf{E}[f(x;D)|\, x] - f(x;D)|\, x] \\
&\quad = \underbrace{\mathbf{E}\big[(\mathbf{E}[y|\, x] - \mathbf{E}[f(x;D)|\, x])^2 \big|\, x\big]}_{\text{bias}} + \underbrace{\mathbf{E}\big[(\mathbf{E}[f(x;D)|\, x] - f(x;D))^2 \big|\, x\big]}_{\text{variance}}
\end{aligned}$$

The first term is the model's bias and it depends on the type of learner and the data. If, on average, the estimated learner is different from $\mathbf{E}[Y|\, X]$, the learner model is biased, see [7]. The second term is the learner's output variance. Therefore, we conclude that the variance and the bias affects the estimation errors. Thus, minimizing variance reduces generalization error when the model type and data are fixed. The objective function of variance reduction-based methods is selected from the objectives that reduce the learner's output variance. Fisher information-based active learning is one of the variance reduction-based methods, and its applications can be seen in [32]. [25] and [7] can be checked for more details on this strategy.

2.4.3 Expected Model Change

Another active learning strategy is selecting the point that changes the learner's parameters the most. The insight is that the point that affects the parameters of the learner most is the most informative in the unlabeled set. In theory, the expected model change strategy can be applied to all learners that use gradient-based training, see [25]. The gradient of the loss function determines the change in the learner parameters in gradient-based training.

Note that in the gradient descent-based training, the parameters are updated according to the negative gradient of the loss of the training set, and α is the learning rate.

$$\theta^{(t)} := \theta^{(t-1)} - \alpha \nabla \ell(\{\omega\}, \{\tilde{y}\}, \theta_{(x_\mathsf{L}, Y_\mathsf{L})}) \qquad \forall \omega \in \mathsf{L} \tag{2.29}$$

The gradient of the loss function is the vector that consists of the partial derivatives of $\ell(\{\omega\}, \{\tilde{y}\}, \theta)$ with respect to the elements of the parameter vector. Assuming $\theta = [\theta_1, \ldots \theta_m]^T$, the gradient of the loss function is

$$\nabla \ell(\{\omega\}, \{\tilde{y}\}, \theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \ell(\{\omega\}, \{\tilde{y}\}, \theta) \\ \vdots \\ \frac{\partial}{\partial \theta_m} \ell(\{\omega\}, \{\tilde{y}\}, \theta) \end{bmatrix} \tag{2.30}$$

The expectation of the gradient of the loss function when a candidate point is added to the labeled set is the way of measuring the change in the learner in this strategy, see [25]. The reason for calculating the expected change is that the label of the candidate point is unknown. That's why the expected change in the parameters is calculated by considering all labels for any candidate. The objective function for the expected gradient change strategy is given as follows.

$$x_\omega = \underset{x_\omega \in \mathsf{U}}{\operatorname{argmax}} \sum_{\tilde{y} \in J} \mathsf{p}_{\theta_{(x_\mathsf{L}, Y_\mathsf{L})}}(\tilde{y} | \, x_\omega) \, ||\nabla \ell(\mathsf{L} \cup \{\omega\}, \, Y_\mathsf{L} \cup \{\tilde{y}\}, \theta_{(x_\mathsf{L}, Y_\mathsf{L})})|| \tag{2.31}$$

$||\cdot||$ corresponds to the Euclidean norm of each resulting gradient vector, see [25]. Note that since the learner is trained with the labeled set L in each iteration (i.e., $||\nabla\ell(\mathsf{L}, Y_{\mathsf{L}}, \theta_{(x_{\mathsf{L}}, Y_{\mathsf{L}})})|| \approx 0$), one may use the following approximation.

$$||\nabla\ell(\mathsf{L} \cup \{\omega\}, Y_{\mathsf{L}} \cup \{\tilde{y}\}, \theta_{(x_{\mathsf{L}}, Y_{\mathsf{L}})})|| \approx ||\nabla\ell(\{\omega\}, \{\tilde{y}\}, \theta_{(x_{\mathsf{L}}, Y_{\mathsf{L}})})|| \tag{2.32}$$

2.4.4 Query by Committee

The strategy of the query by committee method is based on disagreement. We may construct different hypotheses for the same task. As an example, think linearly separable binary classification problem like in Figure 2.2. One can find many linear lines that separate one class from another.

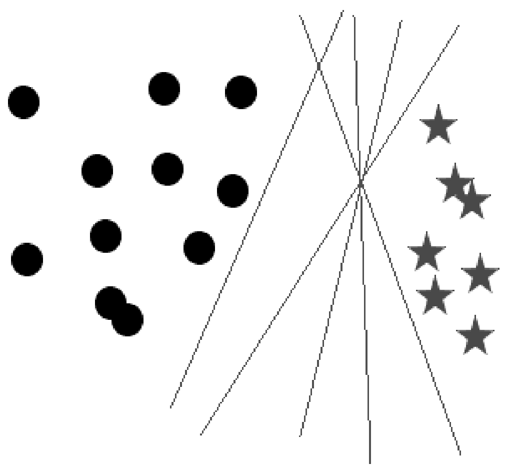

Figure 2.2: Linear Binary Classification Problem

The idea is instead of having a single hypothesis, having competing hypotheses (i.e., more than one). The strategy is to query the data point that competing hypotheses have more disagreement on, see [25].

2.4.5 Uncertainty Sampling

Uncertainty sampling is one of the most used and fundamental algorithms for active learning. The strategy is to choose the data point that the learner is most uncertain about its label, see [14]. There are several ways already proposed to measure the uncertainty of the model according to the data point, such as Shannon's entropy, margin, and least confident, see [25]. These are summarized in the following items.

- Uncertainty sampling with Shannon's Entropy:

$$\underset{\omega \in \mathsf{U}}{\operatorname{argmax}} - \sum_{\tilde{y} \in J} \mathsf{p}_\theta(\tilde{y} | \, x_\omega) \log \mathsf{p}_\theta(\tilde{y} | \, x_\omega) \tag{2.33}$$

- Margin method:

$$\underset{\omega \in \mathsf{U}}{\operatorname{argmin}} \, \mathsf{p}_\theta(\hat{y}_\omega | \, x_\omega) \tag{2.34}$$

- Least confident method:

$$\underset{\omega \in \mathsf{U}}{\operatorname{argmin}} \, \mathsf{p}_\theta(\hat{y}_\omega | \, x_\omega) - \mathsf{p}_\theta(\hat{y}_{\omega_2} | \, x_\omega) \tag{2.35}$$

  where $\hat{y}_\omega$ and $\hat{y}_{\omega_2}$ are the first and second highest probable labels according to model θ respectively

Although uncertainty sampling is proposed for the probabilistic models, there are also some applications of uncertainty sampling on non-probabilistic models, such as for the support vector machine (SVM), nearest neighbour classifier, and random forest classifier. The distance of a point to the decision line is considered an uncertainty measure in SVM, see [33, 25]. For the nearest neighbour classifier, the ratio of the neighbours' votes about the label of a point represents the posterior label probability, see [25]. For the random forest classifier, the class probability is the mean predicted class probabilities of the trees in the forest, see [17]. The next chapter will investigate uncertainty sampling with information-theoretic quantities in detail (i.e., Shannon entropy and mutual information).

CHAPTER 3

INFORMATION THEORETIC OBJECTIVES IN ACTIVE LEARNING

In uncertainty sampling based active learning strategies, using information-theoretic quantities such as the entropy and the mutual information is possible and reasonable. The entropy and the mutual information are suitable to use as uncertainty measures. This chapter will investigate the entropy and the mutual information as active learning querying objectives.

3.1 Entropy Based Approach

Shannon's entropy of a random variable, introduced by C. Shannon in 1948, can be interpreted as the amount of information, surprise, or uncertainty in the random variable, see [28]. Entropy is defined for a random variable.

Definition 3.1.1 *Let $b_1, ..., b_n$ be the possible values of the random variable B and $\mathbf{P}[B = b_i]$ be the probability that the random variable B takes the value b_i. Then, the entropy of B is*

$$\begin{aligned} H\left(B\right) &:= -\sum_{i=1}^{n} \mathbf{P}[B = b_i] \log \mathbf{P}[B = b_i]\,, && (3.1) \\ &= -\sum_{i=1}^{n} \mathbf{E}\big[\mathbb{1}_{\{B=b_i\}}\big] \log \mathbf{E}\big[\mathbb{1}_{\{B=b_i\}}\big]\,. && (3.2) \end{aligned}$$

Let A be another random variable. The conditional entropy of the random variable B given the random variable A is

$$H\left(B\middle|\, A\right) := -\sum_{i=1}^{n} \mathbf{E}\big[\mathbb{1}_{\{B=b_i\}}\big|\, A\big] \log \mathbf{E}\big[\mathbb{1}_{\{B=b_i\}}\big|\, A\big]\,. \qquad (3.3)$$

Remark 3.1.1 *Note that the usual definition of the conditional entropy is different than ours. In [4], for the case when A is a discrete random variable with possible values $a_1, ..., a_m$, the conditional entropy of B given A is defined as*

$$H\left(B|\, A\right) := -\sum_{i=1}^{m} \mathbb{P}(A = a_i) H\left(B|\, A = a_i\right), \tag{3.4}$$

where

$$H\left(B|\, A = a\right) := -\sum_{i=1}^{n} \mathbb{P}(B = b_i | A = a) \log \mathbb{P}(B = b_i | A = a). \tag{3.5}$$

The difference is that the conditional entropy of B given A is a non-negative real number according to this definition, whereas the conditional entropy of B given A is a non-negative random variable whose value is determined by the value of A according to our definition. In particular, the conditional entropy defined in (3.4) *is equal to the expected value of the conditional entropy we have defined in* (3.3).

The mutual information is another information-theoretic quantity that is the information gained for a random variable by observing another random variable. The mutual information can be expressed in terms of entropy.

Definition 3.1.2 *Let A and B are random variables. The mutual information between the random variables A and B is*

$$I\left(A; B\right) := H\left(A\right) - \mathbf{E}[H\left(A|\, B\right)]\,. \tag{3.6}$$

where $H(A)$ and $H(A|\,B)$ are the entropy of A and the conditional entropy of A given B, respectively, see Definition 3.1.1. The mutual information between the random variables A and B conditioned on the random variable C is

$$I\left(A; B|\, C\right) := H\left(A|\, C\right) - \mathbf{E}[H\left(A|\, B, C\right)|\, C]\,. \tag{3.7}$$

In the entropy-based active learning, we aim to use the learner's uncertainty about the label of a point. To write the learner's uncertainty about the label of a point when data of a point and the learner's parameters are given, we use the parametric model given in (2.9) of Assumption 2.2.3, i.e.,

$$\mathbf{P}[Y_\omega = y_\omega|\, \Theta] = \mathsf{p}_\Theta(y_\omega|\, x_\omega)\,. \tag{3.8}$$

Remark 3.1.2 *In Assumption 2.2.1, it is assumed that all functions have access to the data of all points in the data pool (i.e., x_Ω). Therefore, each element of the equations in this part is considered conditioned on x_Ω.*

Remark 3.1.3 *Given Θ, the distribution of the label Y_Q is determined solely by x_Q, and each label in Y_Q is independent by Assumption 2.2.3.*

Thus, the conditional entropy of Y_ω given Θ is

$$H\left(Y_\omega \middle| \Theta\right) = \sum_{y_\omega \in \{1,\ldots,c\}} -\mathsf{p}_\Theta(y_\omega | x_\omega) \log \mathsf{p}_\Theta(y_\omega | x_\omega)\,. \tag{3.9}$$

This can also be interpreted as the learner's uncertainty about the label of the point ω (i.e., the learner's entropy about Y_ω).

The parameter vector of the learner is estimated using the current labeled set L (i.e., the training set). Assuming negative log-likelihood is used as the loss function (i.e., $\ell(\mathsf{L}, Y_\mathsf{L}, \theta) = \sum_{\omega\in\mathsf{L}} -\log \mathsf{p}_\theta(y_\omega | x_\omega)$), the estimation of the parameters is given in (3.10) for a basic machine learning model.

$$\theta_{(x_\mathsf{L}, Y_\mathsf{L})} = \operatorname*{argmax}_{\theta\in\vartheta} \sum_{\omega\in\mathsf{L}} \log \mathsf{p}_\theta(y_\omega | x_\omega) \tag{3.10}$$

In uncertainty sampling, the purpose is to reduce the learner's total uncertainty (i.e., entropy) for the points' labels in the data pool (Ω). When we have a labeled set L, the learner's total entropy for Y_Ω is given in (3.13) by using the assumption that Y_ω only depends on x_ω when Θ is given, see Assumption 2.2.3.

$$\begin{aligned} H\left(Y_\Omega \middle| Y_\mathsf{L}, \Theta\right) &= H\left(Y_{\Omega\setminus\mathsf{L}} \middle| Y_\mathsf{L}, \Theta\right) && \text{by the definition of the conditional entropy} \quad (3.11)\\ &= H\left(Y_\mathsf{U} \middle| Y_\mathsf{L}, \Theta\right) && \text{by } \mathsf{U} = \Omega \setminus \mathsf{L} \quad (3.12)\\ &= \sum_{\omega\in\mathsf{U}} H\left(Y_\omega \middle| \Theta\right) && \text{by Assumption 2.2.3} \quad (3.13) \end{aligned}$$

According to (3.13), the total conditional entropy consists of the sum of the individual points' conditional entropies with respect to the learner. Therefore, it is reasonable to select the points with the highest individual entropy according to the learner.

To show this strategy from another perspective, let's use the mutual information as our objective. The information-theoretic approach using mutual information is maximizing the information gained about $Y_{\mathsf{U}\setminus\mathsf{Q}}$ by observing Y_Q. This corresponds to the mutual information between $Y_{\mathsf{U}\backslash\mathsf{Q}}$ and Y_{Q} given Y_{L} and Θ. The objective of this strategy is given in (3.15). This objective can also be interpreted as maximizing the reduction of uncertainty on the remaining unlabeled set when the labels of the query set Q are revealed, see [31]. In terms of entropy, the explicit form of mutual information between $Y_{\mathsf{U}\backslash\mathsf{Q}}$ and Y_{Q} is

$$I\left(Y_{\mathsf{U}\backslash\mathsf{Q}}; Y_{\mathsf{Q}} \middle| Y_{\mathsf{L}}, \Theta\right) = H\left(Y_{\mathsf{U}} \middle| Y_{\mathsf{L}}, \Theta\right) - \mathbf{E}[H\left(Y_{\mathsf{U}} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \Theta\right) | Y_{\mathsf{L}}, \Theta] . \tag{3.14}$$

Thus, the maximization of this mutual information w.r.t Q is equivalent to the minimization of the second term of (3.14) w.r.t Q.

$$\underset{\mathsf{Q}\subset\mathsf{U},|\mathsf{Q}|=k}{\operatorname{argmax}}\ I\left(Y_{\mathsf{U}\backslash\mathsf{Q}}; Y_{\mathsf{Q}} \middle| Y_{\mathsf{L}}, \Theta\right) = \underset{\mathsf{Q}\subset\mathsf{U},|\mathsf{Q}|=k}{\operatorname{argmin}}\ \mathbf{E}\left[H\left(Y_{\mathsf{U}\backslash\mathsf{Q}} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \theta\right) \middle| Y_{\mathsf{L}}, \Theta\right] \tag{3.15}$$

The actual value of Y_{Q} is unknown at this point, and the value of $H\left(Y_{\mathsf{U}\backslash\mathsf{Q}} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \Theta\right)$ depends on the value of Y_{Q}. Therefore this expectation requires calculation for each possible value of Y_{L}, see [22].

$$\mathbf{E}\left[H\left(Y_{\mathsf{U}\backslash\mathsf{Q}} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \Theta\right) \middle| Y_{\mathsf{L}}, \Theta\right] = \sum_{\tilde{y_{\mathsf{Q}}}\in J} \mathsf{p}_{\Theta}(\tilde{y_{\mathsf{Q}}}| x_{\mathsf{Q}})\, H\left(Y_{\mathsf{U}\backslash\mathsf{Q}} \middle| Y_{\mathsf{Q}} = \tilde{y_{\mathsf{Q}}}, Y_{\mathsf{L}}, \Theta\right) \tag{3.16}$$

To simplify the objective, the total conditional entropy of the unlabeled points' labels in (3.12) can be written in terms of this expectation. Therefore, we will use the following equation to change the objective.

$$H\left(Y_{\mathsf{U}} \middle| Y_{\mathsf{L}}, \Theta\right) = \mathbf{E}\left[H\left(Y_{\mathsf{U}\backslash\mathsf{Q}} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \Theta\right) \middle| Y_{\mathsf{L}}, \Theta\right] + H\left(Y_{\mathsf{Q}} \middle| Y_{\mathsf{L}}, \Theta\right) \tag{3.17}$$

Since the entropy on the left-hand side of (3.17) is fixed and does not depend on Q, we can maximize $H\left(Y_{\mathsf{Q}} \middle| Y_{\mathsf{L}}, \Theta\right)$ to minimize $\mathbf{E}\left[H\left(Y_{\mathsf{U}\backslash\mathsf{Q}} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \Theta\right) \middle| Y_{\mathsf{L}}, \Theta\right]$ by the chain rule of entropy given in (3.17). Then, the final simplified querying objective for entropy-based active learning is provided in (3.19).

$$\begin{aligned} \mathsf{Q} &= \underset{\mathsf{Q}\subset\mathsf{U},|\mathsf{Q}|=k}{\operatorname{argmax}}\ H\left(Y_{\mathsf{Q}} \middle| Y_{\mathsf{L}}, \Theta\right) && \text{(3.18)} \\ &= \underset{\mathsf{Q}\subset\mathsf{U},|\mathsf{Q}|=k}{\operatorname{argmax}} \sum_{\omega\in\mathsf{Q}} H\left(Y_{\omega} \middle| \Theta\right) \qquad \text{by Assumption 2.2.3} && \text{(3.19)} \end{aligned}$$

Therefore, we minimize $\mathbf{E}[H\left(Y_{\Omega} \middle| Y_{\mathsf{Q}}, Y_{\mathsf{L}}, \Theta\right) | Y_{\mathsf{L}}, \Theta]$ by choosing the points with the largest $H\left(Y_{\omega} \middle| \Theta\right)$ values. As a result, for the entropy-based active learning strategy, the objective function $\mathbf{f}$ in the context of the framework we describe in Chapter 2 is selecting the points that have the highest entropy (i.e., (3.20)).

$$\mathbf{f}(\mathsf{Q}) = H\left(Y_{\mathsf{Q}} \middle| Y_{\mathsf{L}}, \Theta \right) \tag{3.20}$$

Remark 3.1.4 *Note that in* (3.19)*, we showed that the joint conditional entropy of* Y_{Q} *is equivalent to the sum of the individual conditional entropies. Therefore, there is no need to use the greedy approaches mentioned in Section 2 for this entropy-based strategy.*

Note that with this method, we choose the points with the highest entropy according to the current state of the learner machine. When k is greater than one, a possible problem is redundant information caused by the fact that the top points selected by this method might be from a similar region in the sample space. Therefore, selecting multiple points at once might result in choosing the points that contain overlapping information, see [30]. Another problem is that the strategy only relies on the learner's state, which depends on the labeled set. Therefore, if the estimation of the model is not good enough or the initial labeled set is not well chosen, this strategy might not perform well, see [21]. There is also a possibility to select extraordinary points (i.e., the points with a very low probability), see [27]. Based on these problematic scenarios, in Chapter 4, we proposed a novel active learning strategy on top of the entropy sampling.

There are also some different ideas. [8] proposes to calculate the entropy in an optimistic way as active learning strategy. They only consider the label assignment that gives minimum entropy while calculating this objective, and they do not take into account the other possible label assignments.

$$\underset{j \in J}{\operatorname{argmin}}\, H\left(Y_{\mathsf{U}\setminus\mathsf{Q}} \middle| Y_{\mathsf{Q}} = j,\, Y_{\mathsf{L}}, \Theta \right) \tag{3.21}$$

They argue that considering all possible labels might add great deal of uncertainty to selection strategy and could make selection strategy ineffective, see [8]. One may use this approach to apply a greedy approach to choose label while iteratively expanding the query set with greedy approach, see [31].

CHAPTER 4

DENSITY WEIGHTED ACTIVE LEARNING

When the querying process only depends on a utility function that measures the information of the query set Q, the result is highly dependent on the learner and the labeled set L. For example, assume the initial labeled set is not well-selected; thus, it is not representative enough of the actual data source. The learner trained with this labeled set would probably perform poorly and not give the accurate results. Since the learner is a part of the active learning selection strategy, the points selected will be based on the distorted learner. Ultimately, this might result in not selecting the actual informative points by affecting the selection strategy negatively. An illustrative example given in [21] is reproduced in Figure 4.1.

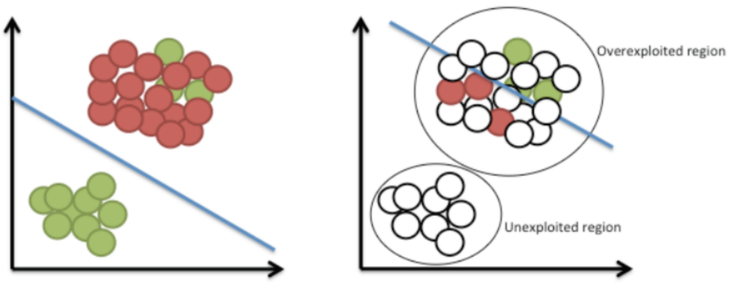

Figure 4.1: A Problematic Scenario for Uncertainty Sampling [21]

In Figure 4.1, on the left, the actual labels of the data pool are shown with the colored points, and the decision boundary of the learner trained with the colored data is represented with a bar. On the right, the initial labeled set is given with colored points, the corresponding decision boundary of the learner is provided with a line, and uncolored points represent the unlabeled set. Note that the more the points become closer to the decision boundary more the learner becomes confident about its prediction. Therefore the most uncertain points are the point closest to the decision boundary. In this example, the uncertainty sampling will be stuck in the over-exploited region, and the resulting decision boundary will be inaccurate.

This case shows that the uncertainty sampling highly depends on the initial labeled set and may fail in such cases. Therefore, the utility function of the active learning strategy may be supported with additional representativeness weight to take into account representatives of the data pool in the selection strategy.

Another possible problem is sampling the instance which is not representative of other instances, see [27]. To illustrate this problem for the uncertainty sampling, [27] discusses the case in Figure 4.2. Note that the line in Figure 4.2 is a decision boundary, and labeled data points are in the form of squares and triangles according to their label, while unlabeled data points are in the form of a circle. If one uses only the uncertainty sampling, that will choose point A since it is on the decision boundary, although the representativeness of point A is very low. However, point B would be a much better selection instead of point A because it is close to the decision boundary and a much more representative point for others than point A.

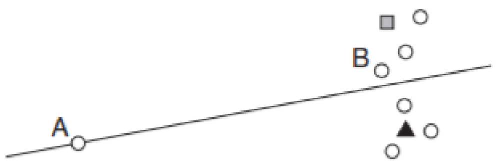

Figure 4.2: A Possible Problematic Scenario for Uncertainty Sampling [27]

Therefore, an active learning strategy without any representativeness measure support may be a poor selection strategy, see the discussions in [21, 25, 27]. To deal with this kind of problem, the objective function of the active learning strategy may be supported with representativeness weight also to consider the representativeness of the instance. The density-weighted active learning surrogate objective function is described in [27] to address this problem. The introduced surrogate objective function is called the information density (ID) method and is given in (4.1).

$$\mathbf{f}_{(ID)}(\omega^*) = \mathbf{f}(\omega^*) \times \left(\frac{1}{|\mathsf{U}|} \sum_{\omega \in \mathsf{U}} \mathbf{sim}(x_\omega, \omega^*) \right)^\beta \tag{4.1}$$

We may adapt this into batch mode querying as follows to make it applicable for any size of the query set Q.

$$\mathbf{f}_{(ID)}(\mathsf{Q}) = \mathbf{f}(\mathsf{Q}) \times \left(\frac{1}{|\mathsf{U}||\mathsf{Q}|} \sum_{\omega \in \mathsf{U}} \sum_{\omega^* \in \mathsf{Q}} \mathbf{sim}(x_\omega, x_{\omega^*}) \right)^\beta \tag{4.2}$$

This method proposes to weight the informativeness measure of candidate set Q (i.e., $\mathbf{f}(\mathsf{Q})$), with its similarity to the whole unlabeled set of points U where β is the term for adjusting the importance of representatives measure in selection criteria. Cosine and Gaussian similarity functions are given as possible similarity functions in [24]. Cosine similarity function is the same formula with cosine angle formula between two non zero vectors.

$$\mathbf{cos}(x_\omega, x_{\omega^*}) = \frac{x_\omega \cdot x_{\omega^*}}{||x_\omega|| \times ||x_{\omega^*}||} \tag{4.3}$$

Gaussian similarity can be calculated as follows where the α is the variance parameter for the Gaussian shape. Note that for simplicity, [24] uses the same variance for all features but each feature ($z_{i,\omega}$) has a different variance in reality.

$$\mathbf{gauss}(x_\omega, x_{\omega^*}) = \exp\left(-\sum_{i=1}^{d} \frac{(z_{i,\omega} - z_{i,\omega^*})^2}{\alpha^2} \right) \tag{4.4}$$

4.1 Proposed Density Weighted Method

Instead of measuring the point-wise similarity between the candidate query set Q and the unlabeled set U, we propose to measure the distance between the probability density function estimated from the data pool ($\hat{p}_\Omega$) and the probability density function estimated from the labeled set L ($\hat{q}_\mathsf{L}$). The idea is to give higher weights to the query sets that reduce the distance between the estimated distributions more. We propose to use divergence to measure the distance between the estimated probability density functions. The formal definition of the proposed strategy is given in Definition 4.1.1.

Definition 4.1.1 *Let set Ω consist of all data points $x_1, ..., x_\Omega$ and set L consists of labeled data points. Assume that the probability density function estimated from the data in Ω is $\hat{p}_\Omega$ and the probability density function estimated from the data in $\mathsf{L} \cup \mathsf{Q}$ is $\hat{p}_{\mathsf{L}\cup\mathsf{Q}}$. Then, the proposed querying model is*

$$\mathbf{f}_{(div)}(\mathsf{Q}) = \mathbf{f}(\mathsf{Q}) \times \exp\Bigg(- \mathbf{dist}(\hat{p}_\Omega, \hat{p}_{\mathsf{L}\cup\mathsf{Q}})\beta\Bigg) \tag{4.5}$$

where β is the importance given the density difference and $\mathbf{dist}$ *is divergence that measures the distance between two different probability density functions.*

4.1.1 Divergence

A divergence is a function that takes two probability distributions as inputs and measures the statistical distance between these two probability distributions. A divergence is always non-negative and equal to zero if and only if the two probability distributions are identical. The divergence increases if the difference between the probability distributions increases. Some of the most used divergences are the Kullback–Leibler divergence, the Renyi divergence, and the Cauchy-Schwarz divergence.

Definition 4.1.2 *The Kullback–Leibler divergence between two probability density functions,* $\mathbf{p}$ *and* $\mathbf{q}$ *, is defined as*

$$D_{KL}\left(\mathbf{p} \| \, \mathbf{q}\right) = \int \mathbf{p}(x)\Bigg(\log \frac{\mathbf{p}(x)}{\mathbf{q}(x)}\Bigg) dx \tag{4.6}$$

Definition 4.1.3 *The order α Renyi divergence between two probability density functions,* $\mathbf{p}$ *and* $\mathbf{p}$ *, is defined as*

$$D_\alpha(\mathbf{p} \| \, \mathbf{q}) = \begin{cases} \frac{1}{\alpha-1} \log \int \mathbf{p}(x)\Bigg(\frac{\mathbf{p}(x)}{\mathbf{q}(x)}\Bigg)^{\alpha-1} dx & \textit{if}\, \alpha \neq 1 \\ \int \mathbf{p}(x)\Bigg(\log \frac{\mathbf{p}(x)}{\mathbf{q}(x)}\Bigg) dx & \textit{if}\, \alpha = 1 \end{cases} \tag{4.7}$$

The Cauchy-Schwarz divergence is a divergence based on the Cauchy-Schwarz inequality, see [10].

Definition 4.1.4 *The Cauchy-Schwarz divergence between two probability density functions,* $\mathbf{p}$ *and* $\mathbf{q}$ *, is defined as*

$$D_{CS}\left(\mathbf{p} \| \, \mathbf{q}\right) = -\log \frac{\int \mathbf{p}(x)\mathbf{q}(x)dx}{\sqrt{\int \mathbf{p}(x)^2 dx \int \mathbf{q}(x)^2 dx}} \tag{4.8}$$

4.2 Density Estimation

To use the divergence, first, we need to estimate the probability density function from the labeled set (i.e., $\hat{p}_{\mathrm{L}}$) and the probability density function from the data pool (i.e., $\hat{p}_{\Omega}$). We will use the kernel density estimation method to estimate the probability density functions. We use the kernel functions and continuous divergence measures because we want to extend the distributions of the observed points to the universe so that the points' distance to each other matters while calculating the distance between the distributions.

4.2.1 Kernel Density Estimation

The kernel density estimation, also known as the Parzen windowing, is a non-parametric way of density estimation. It empirically estimates the probability density function of a random variable by using each observation in the set of observations. The estimated probability density function by the kernel density estimation is given in (4.9) for a given set of data $\{x_1, ..., x_N\}$.

$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K_h(x - x_i) \tag{4.9}$$

Note that $K_h(x - x_i)$ represents a kernel function that takes two input data points, and h is the scale factor. A larger h results in a smoother probability density function. A valid kernel function must be symmetrical and satisfy the following three properties where x is defined on $\mathbb{R}$.

$$\lim_{x - x_i \to \infty} K_h(x - x_i) = \lim_{x - x_i \to -\infty} K_h(x - x_i) = 0 \tag{4.10}$$

$$0 \leq K_h(x - x_i) \leq \infty \tag{4.11}$$

$$\int_{-\infty}^{\infty} K_h(x, x_i)dx = 1 \tag{4.12}$$

Note that these properties need to be extended for the multivariate case. If x is a d dimensional (i.e., $x \in \mathbb{R}^d$ and $x = [x_1, x_2, \ldots, x_d]$), the kernel function needs to satisfy

$$\int_{x_1=-\infty}^{x_1=\infty} \int_{x_2=-\infty}^{x_2=\infty} \cdots \int_{x_d=-\infty}^{x_d=\infty} K_h(x, x_i)dx_1 dx_2 \ldots dx_d = 1. \tag{4.13}$$

Some of the frequently used kernel functions are given below.

- Gaussian Kernel Function:

$$K_h(x - x_0) = \frac{1}{h\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-x_0}{h})^2} \tag{4.14}$$

- Exponential Kernel Function

$$K_h(x - x_0) = \frac{1}{2h} e^{-\frac{|x-x_0|}{h}} \tag{4.15}$$

- Multivariate Gaussian Kernel Function

$$K_\Sigma(x - x_0) = \frac{1}{\sqrt{2\pi}|\Sigma|} \exp\left[-\frac{1}{2}(x - x_0)^T \Sigma^{-1}(x - x_0) \right] \tag{4.16}$$

Note that for the multivariate Gaussian kernel function, we used Σ instead of h since it takes a covariance matrix as a scaling parameter. The logic behind the kernel density estimation is closely related to histograms. The estimated probability density function is just the sum of the kernel functions centered on the observations (i.e., a sum of the kernel functions located on the observed data points).

Definition 4.2.1 *Let A be a population with an unknown probability density function $f(a)$, and N data points are sampled from A (i.e., $\{a_1, , a_n\}$). The kernel density estimation of $f(a)$ (i.e., $\hat{f}(a)$) using the observed data points is given below, where K_h is a kernel function, and h is a scaling parameter.*

$$\hat{f}(a) = \frac{1}{|N|} \sum_{a_i \in A} K_h(a, a_i) \tag{4.17}$$

We use $\hat{p}_\Omega$ for the probability density function estimated from the data pool and $\hat{p}_\mathsf{L}$ for the probability density function estimated from the labeled set. The followings are the kernel density estimations for $\hat{p}_\Omega$ and $\hat{p}_\mathsf{L}$ with a kernel function $K_h(x - x_\omega)$.

$$\hat{p}_\Omega(x) = \frac{1}{|\Omega|}\sum_{\omega\in\Omega} K_h(x - x_\omega) \tag{4.18}$$

$$\hat{p}_\mathsf{L}(x) = \frac{1}{|\mathsf{L}|}\sum_{\omega\in\mathsf{L}} K_h(x - x_\omega) \tag{4.19}$$

If the kernel function is a multivariate Gaussian kernel function, $\hat{p}_\Omega$ and $\hat{p}_\mathsf{L}$ become as follows. Note that the covariance matrices, Σ_1 and Σ_2, are the square matrix of order d, where d is the dimension of the data.

$$\hat{p}_\Omega(x) = \frac{1}{|\Omega|}\sum_{\omega\in\Omega} \frac{1}{\sqrt{2\pi}|\Sigma_1|}\exp\left[-\frac{1}{2}(x - x_\omega)^T\Sigma_1^{-1}(x - x_\omega)\right] \tag{4.20}$$

$$\hat{p}_\mathsf{L}(x) = \frac{1}{|\mathsf{L}|}\sum_{\omega\in\mathsf{L}} \frac{1}{\sqrt{2\pi}|\Sigma_2|}\exp\left[-\frac{1}{2}(x - x_\omega)^T\Sigma_2^{-1}(x - x_\omega)\right] \tag{4.21}$$

4.2.2 Covariance Matrix Selection for the Gaussian Kernel

To get a good and meaningful estimation, the covariance matrices need to be selected carefully according to data properties. For covariance selection of the Gaussian kernels, many resources use a strategy called Silverman's rule of thumb, see [18, 29]. It scales the standard deviation of the data, which is directly calculated on data with a factor that depends on the dataset's size and data dimensions, see [29].

Definition 4.2.2 ([29]) *Silverman's rule of thumb to select scale parameter for the Gaussian kernel density estimation is*

$$\sigma_{silverman} := \sigma_x\left(\frac{4}{N(d+2)}\right)^{\frac{1}{d+4}}, \tag{4.22}$$

where N is the size of the dataset and d *is the dimensions of data and σ_x is the standart deviation calculated from the dataset, see [29, p. 86].*

According to Silverman's rule, the scaled covariance matrices for $\hat{p}_\mathsf{L}$ and $\hat{p}_\Omega$ are,

$$\Sigma_1 = \Sigma_x\left(\frac{4}{|\Omega|(d+2)}\right)^{\frac{2}{d+4}}, \tag{4.23}$$

$$\Sigma_2 = \Sigma_x\left(\frac{4}{|\mathsf{L}^*|(d+2)}\right)^{\frac{2}{d+4}}, \tag{4.24}$$

where Σ_x is the covariance matrix estimated from the data pool and $|\mathsf{L}^*|$ is the expected size for the labeled set after the active learning process ends, (i.e., expected number of iterations + initial size of labeled points) and can be selected approximately. For example, if the initial labeled set is 25 and we plan to add 300 more points into the labeled set, then $|\mathsf{L}^*|$ becomes 325.

4.3 Divergence Estimation

After $\hat{p}_\Omega(x)$ and $\hat{p}_\mathsf{L}(x)$ are defined by the kernel density estimation method, we need to discuss the estimation method of the divergences given in Section 4.1.1. The point of the discussion for this section is to find easily calculable divergence among all divergences when we use kernel density estimation to estimate the probability density functions.

4.3.1 Kullback-Leibler Divergence Estimator

The Kullback–Leibler divergence is a widely used divergence; see Definition 4.1.2. The estimation of the Kullback–Leibler divergence between the probability density function estimated from the data pool ($\hat{p}_\Omega$) and the probability density function estimated from the labeled set ($\hat{p}_\mathsf{L}$) is given as follows.

$$D_{KL}\left(\hat{p}_\Omega \| \, \hat{p}_\mathsf{L}\right) = \int \hat{p}_\Omega(x) \left(\log \frac{\hat{p}_\Omega(x)}{\hat{p}_\mathsf{L}(x)} \right) dx \tag{4.25}$$

$$= \int \left(\frac{1}{|\Omega|} \sum_{\omega^* \in \Omega} K_h(x, x_{\omega^*}) \log \frac{\frac{1}{|\Omega|} \sum_{\omega^{**} \in \Omega} K_h(x, x_{\omega^{**}})}{\frac{1}{|\mathsf{L}|} \sum_{\omega^{***} \in \mathsf{L}} K_h(x, x_{\omega^{***}})} \right) dx \tag{4.26}$$

No further simplification is possible without an approximation. Note that this estimator is also discussed in [18]. In [18], they replace the integral with the empirical mean; however, there is no proof or evidence that expectation can be approximated with an empirical sum for our case.

4.3.2 Rényi Divergence Estimator

The order α Rényi divergence is one of the most commonly used divergences in information theory; see Definition 4.1.3. The estimation of the order α Rényi divergence between the probability density function estimated from the data pool ($\hat{p}_\Omega$) and the probability density function estimated from the labeled set ($\hat{p}_{\mathsf{L}}$) is given as follows.

$$D_\alpha(\hat{p}_\Omega \| \hat{p}_{\mathsf{L}}) = \frac{1}{\alpha-1} \log \int \hat{p}_\Omega(x) \left(\frac{\hat{p}_\Omega(x)}{\hat{p}_{\mathsf{L}}(x)} \right)^{\alpha-1} dx \tag{4.27}$$

$$= \frac{1}{\alpha-1} \log \int \left(\frac{1}{|\Omega|} \sum\nolimits_{\omega^* \in \Omega} K_h(x - x_{\omega^*}) \right)^{\alpha} \left(\frac{1}{|\mathsf{L}|} \sum\nolimits_{\omega^{**} \in \mathsf{L}} K_h(x - x_{\omega^{**}}) \right)^{1-\alpha} dx \tag{4.28}$$

Similarly, no further simplification is possible. Calculating the result of this integral is not applicable in practice, especially when we have high dimensional input.

4.3.3 Cauchy-Schwarz Divergence Estimator

The definition of the Cauchy-Schwarz divergence is given in Definition 4.1.4. The Cauchy-Schwarz divergence estimator has a special case when we use the Gaussian kernel function in the kernel density estimation. The Gaussian kernel function is given in (4.29).

$$\varphi(x, \Sigma) = \frac{1}{\sqrt{2\pi \det(\Sigma)}} \exp\left(-\frac{1}{2} x^T \Sigma^{-1} x \right) \tag{4.29}$$

Recall that the sum of two independent Gaussian random variables is also a Gaussian random variable. In other words, the convolution of two Gaussian probability density functions is also a Gaussian probability density function.

$$\int \varphi\left(x - x_i, \hat{\Sigma}\right) \varphi\left(x_j - x, \tilde{\Sigma}\right) dx = \varphi\left(x_i - x_j, \hat{\Sigma} + \tilde{\Sigma}\right) \tag{4.30}$$

Note that the Gaussian probability density function is a symmetric function, therefore the below equation is also valid.

$$\int \varphi\left(x - x_i, \hat{\Sigma}\right) \varphi\left(x - x_j, \tilde{\Sigma}\right) dx = \varphi\left(x_i - x_j, \hat{\Sigma} + \tilde{\Sigma}\right) \tag{4.31}$$

Using (4.31), in estimating the Cauchy Schwartz divergence, we may calculate the divergence efficiently without any approximation as follows, see [10].

Remark 4.3.1 *The idea of simplifying Cauchy-Schwarz divergence using Gaussian kernel density estimation is taken from [10].*

$$D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}}\right) = -\log \frac{\int \hat{p}_{\Omega}(x)\hat{p}_{\mathsf{L}}(x)dx}{\sqrt{\int \hat{p}_{\Omega}(x)^2 dx \int \hat{p}_{\mathsf{L}}(x)^2 dx}} \tag{4.32}$$

$$= -\log \frac{\int \frac{1}{|\Omega|} \sum\limits_{\omega_i \in \Omega} \varphi\left(x - x_{\omega_i}, \Sigma_1\right) \frac{1}{|\mathsf{L}|} \sum\limits_{\omega_j \in \mathsf{L}} \varphi\left(x - x_{\omega_j}, \Sigma_2\right) dx}{\sqrt{\int \left[\frac{1}{|\Omega|} \sum\limits_{\omega_k \in \Omega} \varphi\left(x - x_{\omega_k}, \Sigma_1\right)\right]^2 dx \int \left[\frac{1}{|\mathsf{L}|} \sum\limits_{\omega_l \in \mathsf{L}} \varphi\left(x - x_{\omega_l}, \Sigma_2\right)\right]^2 dx}} \tag{4.33}$$

$$= -\log \frac{\sum\limits_{\omega_i \in \Omega, \omega_j \in \mathsf{L}} \int \varphi\left(x - x_{\omega_i}, \Sigma_1\right) \varphi\left(x - x_{\omega_j}, \Sigma_2\right) dx}{\sqrt{\sum\limits_{\omega_k \in \Omega, \omega_r \in \Omega} \int \varphi\left(x - x_{\omega_k}, \Sigma_1\right) \varphi\left(x - x_{\omega_r}, \Sigma_1\right) dx \sum\limits_{\omega_l \in \mathsf{L}, \omega_h \in \mathsf{L}} \int \varphi\left(x - x_{\omega_l}, \Sigma_2\right) \varphi\left(x - x_{\omega_h}, \Sigma_2\right) dx}} \tag{4.34}$$

$$= -\log \frac{\sum\limits_{\omega_i \in \Omega, \omega_j \in \mathsf{L}} \varphi\left(x_{\omega_i} - x_{\omega_j}, \Sigma_1 + \Sigma_2\right)}{\sqrt{\sum\limits_{\omega_k \in \Omega, \omega_r \in \Omega} \varphi\left(x_{\omega_k} - x_{\omega_r}, 2\Sigma_1\right) \sum\limits_{\omega_l \in \mathsf{L}, \omega_h \in \mathsf{L}} \varphi\left(x_{\omega_l} - x_{\omega_h}, 2\Sigma_2\right)}} \tag{4.35}$$

4.4 Cauchy-Schwarz Divergence as Density Weight

In density-weighted active learning, we aim to add both informative and representative points into the labeled set. Therefore, minimizing the divergence between $\hat{p}_{\mathsf{L}}$ and $\hat{p}_{\Omega}$ might be a good idea to have a representative labeled set. When we select a single point ω^+ from U and add it to L, the kernel density estimation of $\hat{p}_{\Omega}$ does not change, but $\hat{p}_{\mathsf{L}}$ changes to $\hat{p}_{\mathsf{L}\cup\{\omega^+\}}$. The kernel density estimation of $\hat{p}_{\mathsf{L}\cup\{\omega^+\}}$ is given below.

$$\hat{p}_{\mathsf{L}\cup\{\omega^+\}}(x_\omega) = \frac{1}{|\mathsf{L}|+1} \sum_{\omega^* \in \mathsf{L}} K_h(x_\omega, x_{\omega^*}) + \frac{1}{|\mathsf{L}|+1} K_h(x_\omega, x_{\omega^+}) \tag{4.36}$$

$$= \frac{1}{|\mathsf{L}|+1} \sum_{\omega^* \in \mathsf{L}\cup\omega^+} K_h(x_\omega, x_{\omega^*}) \tag{4.37}$$

Note that in (4.35), when we add a point into L, we just need to add one more index into the sums that contain set L. Then, the Cauchy-Schwarz divergence estimator between the updated labeled set and the data pool (i.e., $D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^*\}}\right)$) becomes as follows for the Gaussian kernel function.

$$D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^*\}}\right) = \\ -\log \frac{\sum\limits_{\omega_i\in\Omega,\omega_j\in\mathsf{L}} \varphi\left(x_{\omega_i} - x_{\omega_j}, \Sigma_1 + \Sigma_2\right) + \sum\limits_{\omega_o\in\Omega} \varphi\left(x_{\omega_o} - x_{\omega^*}, \Sigma_1 + \Sigma_2\right)}{\sqrt{\left(\sum\limits_{\omega_k\in\Omega,\omega_r\in\Omega} \varphi\left(x_{\omega_k} - x_{\omega_r}, 2\Sigma_1\right)\right)\left(\sum\limits_{\omega_l\in\mathsf{L},\omega_h\in\mathsf{L}} \varphi\left(x_{\omega_l} - x_{\omega_h}, 2\Sigma_2\right) + 2\sum\limits_{\omega_t\in\mathsf{L}} \varphi\left(x_{\omega_t} - x_{\omega^*}, 2\Sigma_2\right) + \varphi\left(0, 2\Sigma_2\right)\right)}} \tag{4.38}$$

We aim is to minimize $D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^+\}}\right)$ according to ω^+ (i.e., according to the point which is considered to be added into the labeled set). For this task, there is no need to consider the terms that are not dependent on ω^+. Thus, minimization of $D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^+\}}\right)$ according to ω^+ needs to be performed. To simplify the below equations, we defined two functions in Definition 4.4.1 and Definition 4.4.2.

Definition 4.4.1 $\tilde{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L})$ *is a function that calculates the sum of the Gaussian kernel functions for all values of* $(x_{\omega_l} - x_{\omega_h})\forall l, h \in \mathsf{L}$ *with the covariance* Σ.

$$\tilde{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L}) = \sum_{\omega_l\in\mathsf{L},\omega_h\in\mathsf{L}} \varphi\left(x_{\omega_l} - x_{\omega_h}, \Sigma\right) \tag{4.39}$$

Definition 4.4.2 $\hat{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L}, \Omega)$ *is a function that calculates the sum of the Gaussian kernel functions for all values of* $(x_{\omega_i} - x_{\omega_j})\forall j \in \mathsf{L}, \forall i \ni \Omega$ *with the covariance* Σ.

$$\hat{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L}, \Omega) = \sum_{\omega_i\in\Omega,\omega_j\in\mathsf{L}} \varphi\left(x_{\omega_i} - x_{\omega_j}, \Sigma\right) \tag{4.40}$$

$\tilde{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L})$ and $\hat{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L}, \Omega)$ will be only used to simplify the following equations. At the end of the minimization, we will end with a function that can easily be expressed by the functions we defined; $\tilde{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L})$ and $\hat{\mathbf{w}}_{\mathbf{\Sigma}}(\mathsf{L}, \Omega)$. The explicit form of $D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^*\}}\right)$ in terms of the functions introduced in Definition 4.4.1 and 4.4.2 is given below.

$$\begin{aligned}
D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^*\}}\right) &= \left[-\log \frac{\sum\limits_{\omega_i\in\Omega,\omega_j\in\mathsf{L}\cup\{\omega^*\}} \varphi\left(x_{\omega_i}-x_{\omega_j},\Sigma_1+\Sigma_2\right)}{\sqrt{\left(\sum\limits_{\omega_k\in\Omega,\omega_r\in\Omega}\varphi\left(x_{\omega_k}-x_{\omega_r},2\Sigma_1\right)\right)\left(\sum\limits_{\omega_l\in\mathsf{L}\cup\{\omega^*\},\omega_h\in\mathsf{L}\cup\{\omega^*\}}\varphi\left(x_{\omega_l}-x_{\omega_h},2\Sigma_2\right)\right)}}\right] \\
&= \left[-\log \frac{\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\omega^*\},\Omega)}{\sqrt{\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_1}}(\Omega)\right)\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega^*\})\right)}}\right] \\
&= \left[\frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_1}}(\Omega)\right)+\frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega^*\})\right)-\log\left(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\{\omega^*\},\Omega)\right)\right]
\end{aligned} \tag{4.41}$$

The objective is to minimize $D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^*\}}\right)$ w.r.t the point ω^*. The minimization of (4.41) w.r.t the point ω^* is equivalent to the maximization of (4.42) w.r.t the point ω^*.

$$\begin{aligned}
&\underset{\omega^*\in\mathsf{U}}{\operatorname{argmin}} D_{CS}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}\cup\{\omega^*\}}\right) \\
&\quad= \underset{\omega^*\in\mathsf{U}}{\operatorname{argmin}}\left[\frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_1}}(\Omega)\right)+\frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega^*\})\right)-\log\left(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\{\omega\},\Omega)\right)\right] \\
&\quad= \underset{\omega^*\in\mathsf{U}}{\operatorname{argmin}}\left[\frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega^*\})\right)-\log\left(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\omega^*\},\Omega)\right)\right] \\
&\quad= \underset{\omega^*\in\mathsf{U}}{\operatorname{argmax}}\left[\log\left(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\omega^*\},\Omega)\right)-\frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega^*\})\right)\right]
\end{aligned} \tag{4.42}$$

This result shows that the more the candidate point ω^* gives higher output for equation (4.42), the more the Cauchy-Schwarz divergence between the labeled set and the data pool decreases. Therefore, (4.42) is reasonable to use as weight in density-weighted active learning.

4.4.1 Minimization Complexity of the Estimator

Note that we end up with two function in (4.42); $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L} \cup \{\omega^*\})$ and $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}, \Omega)$. The explicit forms of that these functions are given below.

$$\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}) = \sum_{\omega_l \in \mathsf{L}, \omega_h \in \mathsf{L}} \varphi\left(x_{\omega_l} - x_{\omega_h}, 2\Sigma_2\right) + 2 \sum_{\omega_t \in \mathsf{L}} \varphi\left(x_{\omega_t} - x_{\omega^*}, 2\Sigma_2\right) + \varphi\left(0, 2\Sigma_2\right) \tag{4.43}$$

$$\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}, \Omega) = \sum_{\omega_i \in \Omega, \omega_j \in \mathsf{L}} \varphi\left(x_{\omega_i} - x_{\omega_j}, \Sigma_1 + \Sigma_2\right) + \sum_{\omega_o \in \Omega} \varphi\left(x_{\omega_o} - x_{\omega^*}, \Sigma_1 + \Sigma_2\right) \tag{4.44}$$

Note that from each previous iteration, we already calculated the value of $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L})$ for that iteration. Therefore, we may calculate $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L} \cup \{\omega^*\})$ from $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L})$ in the next iteration.

$$\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}) - \tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}) = 2 \sum_{\omega_t \in \mathsf{L} \cup \{\omega^*\}} \varphi\left(x_{\omega_t} - x_{\omega^*}, 2\Sigma_2\right) - \varphi\left(0, 2\Sigma_2\right) \tag{4.45}$$

Since we already have $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L})$, to calculate $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L} \cup \{\omega^*\})$ for a single candidate ω^*, we just need to calculate the sum of $|\mathsf{L}|$ number of kernels as follows in each iteration.

$$2 \sum_{\omega_t \in \mathsf{L} \cup \{\omega^*\}} \varphi\left(x_{\omega_t} - x_{\omega^*}, 2\Sigma_2\right) \tag{4.46}$$

The same logic is also valid for $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}, \Omega)$ and $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}, \Omega)$.

$$\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}, \Omega) - \hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}, \Omega) = \sum_{\omega_o \in \Omega} \varphi\left(x_{\omega_o} - x_{\omega^*}, \Sigma_1 + \Sigma_2\right) \tag{4.47}$$

For a single candidate ω^*, we just need to calculate the sum of $|\Omega|$ number of kernels as follows in each iteration to calculate $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}, \Omega)$ from $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}, \Omega)$.

$$\sum_{\omega_o \in \Omega} \varphi\left(x_{\omega_o} - x_{\omega^*}, \Sigma_1 + \Sigma_2\right) \tag{4.48}$$

Assuming that we have calculated $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L})$ and $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}, \Omega)$ and stored in memory in the initial iteration. Then, complexity to calculate divergence weight for a single point ω^* is just $\mathcal{O}\big(|\Omega| + |\mathsf{L}|\big)$. Considering there are $|\mathsf{U}|$ number of candidate points in each iteration, the total complexity becomes $\mathcal{O}\big(|\mathsf{U}|(|\Omega| + |\mathsf{L}|)\big)$. Note that, at the end of each iteration (i.e., after selecting ω^* to add set L), we should update $\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L})$ and $\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}, \Omega)$ as follows.

$$\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}) \leftarrow \tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}) \tag{4.49}$$

$$\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}, \Omega) \leftarrow \hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L} \cup \{\omega^*\}, \Omega) \tag{4.50}$$

For further calculation efficiency, the values of the Gaussian kernel function with both covariance matrices can also be computed for all the points in the data pool initially and stored in the memory. Thus, the value of the kernel function might be restored from memory without calculating its value again and again in each iteration.

4.5 The Cauchy-Schwarz Divergence Weighted Querying

The proposed strategy is to select the point that maximizes the information-based objective function $\mathbf{f}(\mathsf{Q})$ and minimizes the estimated Cauchy-Schwarz divergence between the data pool and the labeled set (i.e., $D_{CS}\left(\hat{p}_\Omega \| \, \hat{p}_{\mathsf{L}\cup\mathsf{Q}}\right)$) at the same time. To reduce the divergence, we know that we need to maximize (4.42). For the following parts, we will rename the right-hand side of (4.42) as $\mathbf{weight}(\mathsf{Q})$.

Definition 4.5.1 $\mathbf{weight}(\mathsf{Q})$ *is the density weight of the query set* Q *calculated by the minimization of* $D_{CS}\left(\hat{p}_\Omega \| \, \hat{p}_{\mathsf{L}\cup\mathsf{Q}}\right)$ *with respect to* Q *in* (4.42) *and defined as*

$$\mathbf{weight}(\mathsf{Q}) = \left[\log\left(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\mathsf{Q}, \Omega)\right) - \frac{1}{2}\log\left(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\mathsf{Q})\right)\right] \tag{4.51}$$

where the functions $\tilde{\mathbf{w}}_{\mathbf{\Sigma}}$ *and* $\hat{\mathbf{w}}_{\mathbf{\Sigma}}$ *are defined in Definition 4.4.1 and 4.4.2.*

Note that since the weight is the sum of logarithms of the Gaussian kernel functions between the points. One possible problem is that the resulting weight can be negative or positive while minimizing the estimated divergence. Another problem is that the weight difference between candidate points might be very small. To address these problems, a filter such as an exponential filter or a min-max filter can be applied to $\mathbf{weight}(\mathsf{Q})$. We proposed two approaches as the Cauchy-Schwarz divergence weighted querying; the min-max filtered querying objective in Definition 4.5.2 and the exponential filtered querying objective in Definition 4.5.3.

Definition 4.5.2 $\mathbf{f}_{w_{mm}}(\mathsf{Q})$ *is the Cauchy-Schwarz divergence weighted min-max filtered objective function;*

$$\mathbf{f}_{w_{mm}}(\mathsf{Q}) = \left[\mathbf{f}(\mathsf{Q}) \times \bigg(\mathbf{weight}(\mathsf{Q})\bigg)_{scaled}\right] \tag{4.52}$$

where $\mathbf{f}(\mathsf{Q})$ *is the information based objective function such as 3.20 and*

$$\bigg(\mathbf{weight}(\mathsf{Q})\bigg)_{scaled} = \frac{\mathbf{weight}(\mathsf{Q}) - \operatorname{argmin}_{\mathsf{Q}^*\in\mathsf{U}}\big(\mathbf{weight}(\mathsf{Q}^*)\big)}{\operatorname{argmax}_{\mathsf{Q}^*\in\mathsf{U}}\big(\mathbf{weight}(\mathsf{Q}^*)\big) - \operatorname{argmin}_{\mathsf{Q}^*\in\mathsf{U}}\big(\mathbf{weight}(\mathsf{Q}^*)\big)} \tag{4.53}$$

Definition 4.5.3 *Let* $\mathbf{f}_{w_{exp}}(\mathsf{Q})$ *is the divergence weighted exponential filtered objective function and* $\mathbf{f}(\mathsf{Q})$ *is the information based objective function such as 3.20, then the proposed objective function is*

$$\mathbf{f}_{w_{exp}}(\mathsf{Q}) = \left[\mathbf{f}(\mathsf{Q}) \times \exp\left(\beta\bigg(\log\big(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\mathsf{Q},\Omega)\big) - \frac{1}{2}\log\big(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\mathsf{Q})\big)\bigg)\right)\right], \tag{4.54}$$

where β *is a hyper-parameter that represents the importance of the weight.*

4.5.1 Sequential Entropy-Based Cauchy-Schwarz Divergence Weighted Model

We applied the proposed density-weighted strategy to the entropy-based active learning strategy in sequential mode. Using Definition 4.5.2, the entropy-based min-max filtered Cauchy-Schwarz divergence weighted objective function is given in (4.55) for selecting a single point ω, where scaled subscript means the min-max scaling.

$$\mathbf{f}_{w_{mm}}(\{\omega\}) = \left[H\left(Y_\omega | x_\omega, \Theta\right) \times \left(\log\bigg(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\omega\},\Omega)\bigg) - \frac{1}{2}\log\bigg(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega\})\bigg)\right)_{scaled}\right] \tag{4.55}$$

The exponential filtered version based on Definition 4.5.3 is given in

$$\mathbf{f}_{w_{exp}}(\{\omega\}) = \left[H\left(Y_\omega | x_\omega, \Theta\right) \times \exp\left(\beta\bigg(\log\big(\hat{\mathbf{w}}_{\mathbf{\Sigma_1+\Sigma_2}}(\mathsf{L}\cup\{\omega\},\Omega)\big) - \frac{1}{2}\log\big(\tilde{\mathbf{w}}_{\mathbf{2\Sigma_2}}(\mathsf{L}\cup\{\omega\})\big)\bigg)\right)\right] \tag{4.56}$$

The framework for the proposed strategy is summarized in Algorthim 4.

Algorithm 4 Entropy Based Cauchy-Schwarz Divergence Weighted Querying

```
Find $\Sigma_1$ and $\Sigma_2$ from the data as given in (4.23) and (4.24)
Calculate and store $\varphi\left(x_{\omega_l} - x_{\omega_h}, 2\Sigma_2\right) \forall l, h \in \Omega$ in the memory
Calculate and store $\varphi\left(x_{\omega_l} - x_{\omega_h}, \Sigma_1 + \Sigma_2\right) \forall l, h \in \Omega$ in the memory
$\theta = \text{argmax}_{\theta \in \vartheta} \sum_{\omega \in \mathsf{L}} \log \mathsf{p}_\theta(y_\omega | \, x_\omega)$
while true do
    $\omega^* \leftarrow \text{argmax}_{\omega \in \mathsf{U}} \sum_{\omega \in \mathsf{U}} \mathbf{f}_{w_{exp}}(\{\omega\})$ or $\text{argmax}_{\omega \in \mathsf{U}} \sum_{\omega \in \mathsf{U}} \mathbf{f}_{w_{mm}}(\{\omega\})$
    $\mathsf{U} \leftarrow \mathsf{U} \setminus \{\omega^*\}$
    $\mathsf{L} \leftarrow \mathsf{L} \cup \{\omega^*\}$
    $\theta = \text{argmax}_{\theta \in \vartheta} \sum_{\omega \in \mathsf{L}} \log \mathsf{p}_\theta(y_\omega | \, x_\omega)$
    $\hat{y}_{test} \leftarrow \text{argmax}_{\hat{y}_{test}} \prod_{\omega \in test} \mathsf{p}_\theta(\hat{y}_\omega | \, x_\omega)$
    if $(Accuracy(y_{test}, \hat{y}_{test}) > \tau)$ then
        return $\theta$
    end if
end while
```

4.6 Experiments

The two versions of the proposed sequential entropy-based Cauchy-Schwarz divergence weighted strategies (i.e., (4.55), (4.56)) are tested on various datasets.

In the first part of the experiments, we referenced [12]. This paper is one of the latest active learning papers published at the Conference on Neural Information Processing Systems (NIPS) in 2017. They propose a novel active learning strategy called as learning from the data (LAL). LAL strategy is tested on various datasets for a classification task using a random forest classifier in [12]. The software of the proposed strategy in [12] was posted on GitHub for the random forest classifier. Therefore, we can compare our strategy with LAL on the same datasets using the same classifier with identical parameters. We selected [12] to compare with our model because the software of the paper is available, the paper is published in NIPS which is a reputable conference, and it is one of the latest papers published in this area. The more detail on the strategy of LAL is given in Section 4.6.1.3. In addition to the LAL strategy, we also compared our strategy with entropy sampling and random sampling. Random sampling selects the points that will be added to the labeled set randomly without using any active learning strategy.

In the second part of the experiments, we test our strategy on the real data sets used in [30] to see more of our approach's performance on real data. In [30], the proposed uncertainty sampling selects the point that most reduces the remaining unlabeled set. According to the current model, they measure the unlabeled set's uncertainty without the candidate point. Then, they retrain their model for all possible label assignments of the candidate point. The reduction in the entropy of the remaining unlabeled set is checked according to the initially calculated entropy by using the label assignment that yields the parameter vector with the minimum entropy for the remaining unlabeled set (i.e., they use the optimistic approach mentioned on (3.21)). The candidate that maximizes this reduction is selected. Note that this strategy includes the retraining of the learner for each possible candidate for all label assortments. Therefore, the computational complexity of the strategy is very high. Since the software of [30] is not published, we only compared the result of our strategy with random and entropy sampling. However, in the experiments, we used the same parameters and configurations mentioned on [30]. You can see their results in [30]. In the second part of the experiments, logistic regression is used as a classifier.

Note that the software of our strategy can be reachable from the following GitHub repository: `https://github.com/senmenes/THE-CAUCHY-SCHWARZ-DIVERGENCE-AND-ENTROPY-BASED-DENSITY-WEIGHTED-ACTIVE-LEARNING`

4.6.1 The Classifiers Used in The Experiments

4.6.1.1 The Random Forest Classifier

The random forest is a machine learning technique that can be used for classification tasks or regression tasks. A random forest classifier is just an ensemble of multiple decision trees. Decision trees tend to overfit their training sets, and the random forests are a way of averaging numerous deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance, see [9]. In the training of each decision tree, different training sets are observed from the original training set by the bagging method. In our framework, θ is the parameters of all decision trees in the random forest. For the classification tasks, the output of the random forest classifier is the class selected by most trees, see [9]. In this technique, the posterior probability of class labels given feature vector x is the ratio of the votes of the decision trees in the random forest.

4.6.1.2 Multinomial Logistic Regression

Multinomial logistic regression is a classification method for multi-class problems. For c number of classes, there are $c-1$ different parameter vectors. Therefore, for this classifier, θ in our framework is contains $c-1$ parameter vectors (i.e., $\theta = [\theta_1^T, ..., \theta_{c-1}^T]^T$ where $\theta_i = [\alpha_i, \beta_i^T]^T$). Note that α represents bias and β represents weight vector. Then the posterior probability of class labels given feature vector x is;

$$\mathsf{p}_\theta(y|\, x) = \frac{e^{\alpha_y + \beta_y^T X_\omega}}{1 + \sum_{t=1}^{c-1} e^{\alpha_t + \beta_t^T x}}, \qquad y \in \{1, \ldots, c-1\} \tag{4.57}$$

$$\mathsf{p}_\theta(y|\, x) = \frac{1}{1 + \sum_{t=1}^{c-1} e^{\alpha_t + \beta_t^T x}}, \qquad y = c \tag{4.58}$$

The objective of multinomial logistic regression is to find θ that gives maximum likelihood estimation for the true labels of the point in set L, see [30].

$$\theta = \underset{\theta \in \vartheta}{\operatorname{argmax}} \sum_{\omega \in \mathsf{L}} \log \mathsf{p}_\theta(y_\omega|\, x_\omega) \tag{4.59}$$

4.6.1.3 LAL Strategy

The idea of the LAL strategy is to train a regressor to predict the reduction in error depending on the learner's state using simple 2D synthetic data or domain-specific data, see [12]. Then, the active learning objective is to select the point predicted to reduce the error most by this pre-trained regressor. The insight is that the similar states of the classifier yield similar behavior while annotating the similar samples, see [12].

The regressor is trained before the active learning process and can be specific to the type of classier used. The features of the regressor's input for the random forest classifier are given as follows in [12]; the average and the variance of the prediction probabilities of the trees in the forest for the candidate point, the average depth of the trees in the forest, the ratio of positive points in the labeled set, average error calculated on out-of-bag samples of the trees in the forest (i.e., out-of-bag score or error), the variance of the trees' feature importances and the size of the labeled set.

They propose two different strategies as LAL-Independent and LAL-iterative. The only difference between the two strategies is the training part of the LAL regressor. In the LAL-Independent, they randomly select bags from the data to train their LAL regressor from the synthetic data. In the LAL-Iterative strategy, they iteratively select the training samples using the current regressor.

4.6.2 Datasets and Experiment Results

In the following subsections, the results of the experiments and the details of used data sets are given and discussed. Note that the results labeled as "*Div&Ent for* $\beta = 1$", "*Div&Ent for* $\beta = 10$" and "*Div&Ent for* $\beta = 100$" correspond to our exponential filtered strategy given in (4.56). "*Div&Ent MinMax*" is the results for our min-max filtered strategy given in (4.55). "Lal-Rand" and "Lal-Iter" correspond to LAL-Independent and LAL-Iterative strategies mentioned on Section 4.6.1.3, respectively. Random sampling and entropy sampling are labeled as it is.

4.6.3 Checkerboard Data Sets Experiments

Checkerboard data sets are synthetic data sets used in the experiments of [12]. There are three synthetic Checkerboard data sets; Checkerboard 2×2, Checkerboard 4×4, and Checkerboard Rotated. These are known as XOR-like datasets and are given in Figure 4.3. It is reported in [1] that the active learning algorithms struggles with XOR-like datasets such as in Figure 4.3, see [12]. Note that the data point's color represents the data point's label. Since there are two classes, the task is a binary classification.

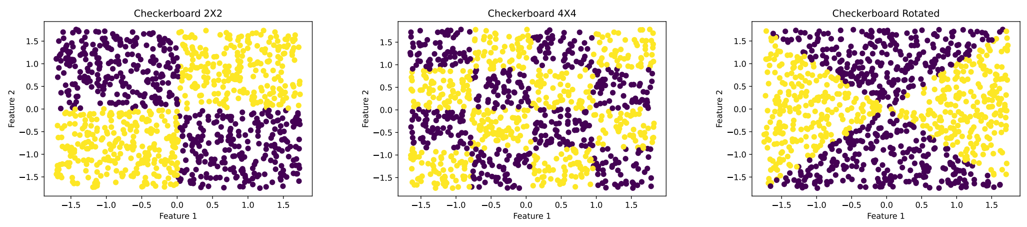

Figure 4.3: Synthetic XOR Datasets

We used the same methodology in the experiments with [12]. Random forest is used as the classifier. The initial labeled set is set to 2, one randomly selected point from each class. The tests are repeated 20 times. The average results of the 20 trials are presented in the following figures.

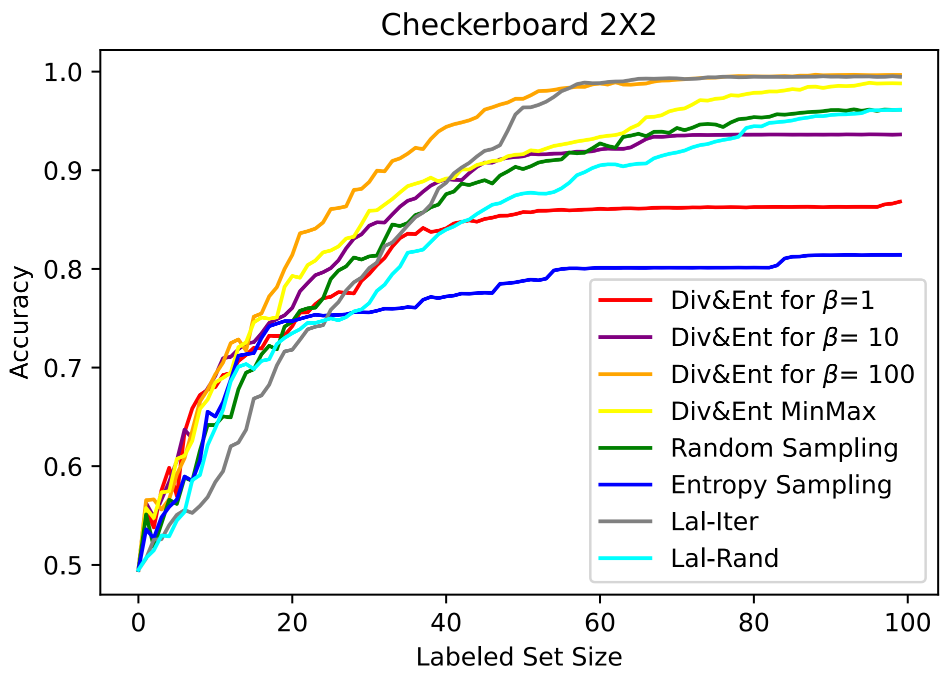

Figure 4.4: The Experiment Results for Checkerboard 2X2

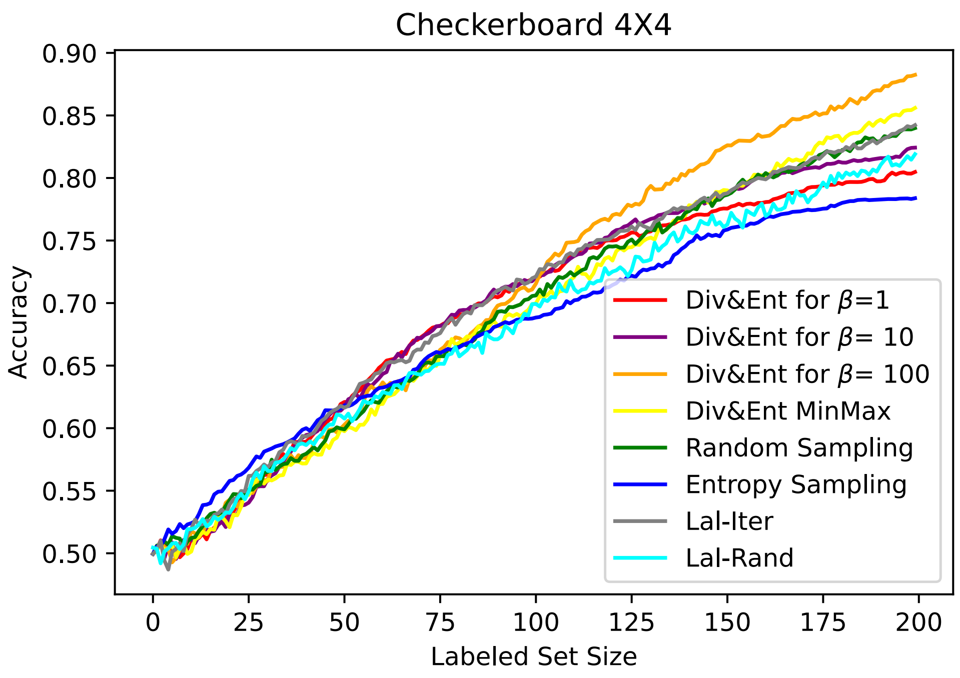

Figure 4.5: The Experiment Results for Checkerboard 4X4

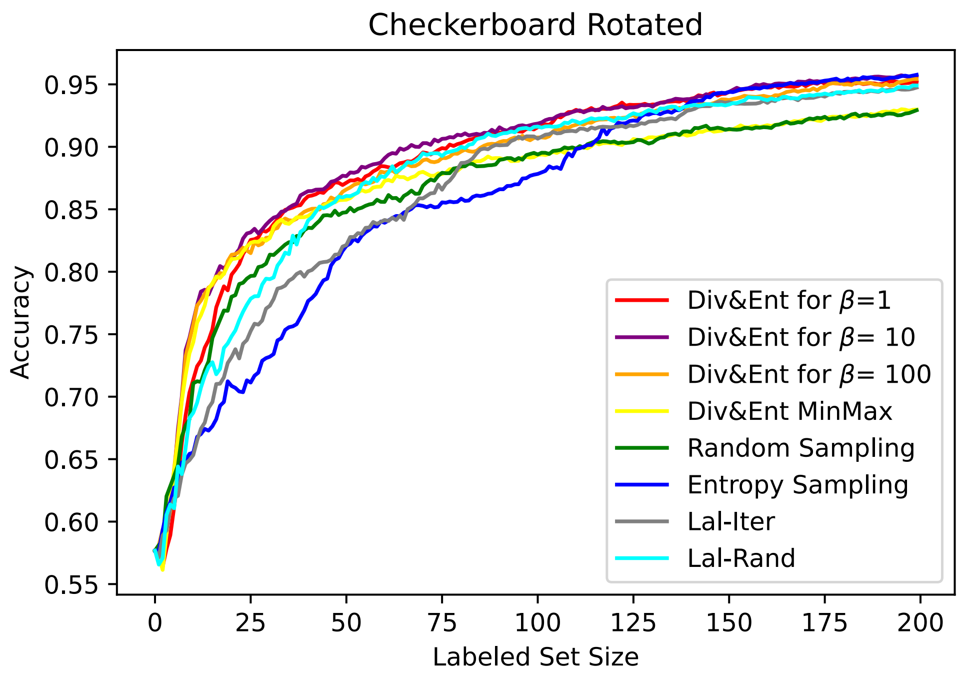

Figure 4.6: The Experiment Results for Checkerboard Rotated

The results show that for the high values of β, the strategy in (4.56) shows superior performance than the other strategies. Especially, our strategy converges to the optimal accuracy rate much faster than other strategies. Note that our aim, in the beginning, was to prevent entropy sampling from failing. For Checkerboard 2×2, entropy sampling has the worst performance. Even random sampling has better performance. However, our entropy-based strategy outperforms the others and reaches optimal accuracy faster. This shows that the proposed model accomplishes the goal.

Remark 4.6.1 *Note that entropy sampling is equivalent to our strategy when β equals to 0.*

Remark 4.6.2 *Note that to keep the experimental setup same as in reference papers, the tests were repeated the same number of times as the experiments done in the reference papers. Increasing the number of trials would give much more general and accurate results since the initial labeled set is selected randomly in each trial.*

4.6.4 Stratium-Mini Data Set Experiments

The stratium-mini dataset is a real-life data set. Stratium dataset is a 3D Electron Microscopy stack of rat neural tissue, and the task is to detect and segment mitochondria, see [12, 15, 11]. The stratium-mini dataset consists of 2000 data points, and each data point consists of 272 features. This dataset is also used in the paper of LAL, see [12]. We used the same methodology in the experiments with [12]. Random forest is used as the classifier. The initial labeled set is set to 2; one randomly selected point from each class. The tests are repeated 20 times. The average results of the 20 tests are presented in Figure 4.7.

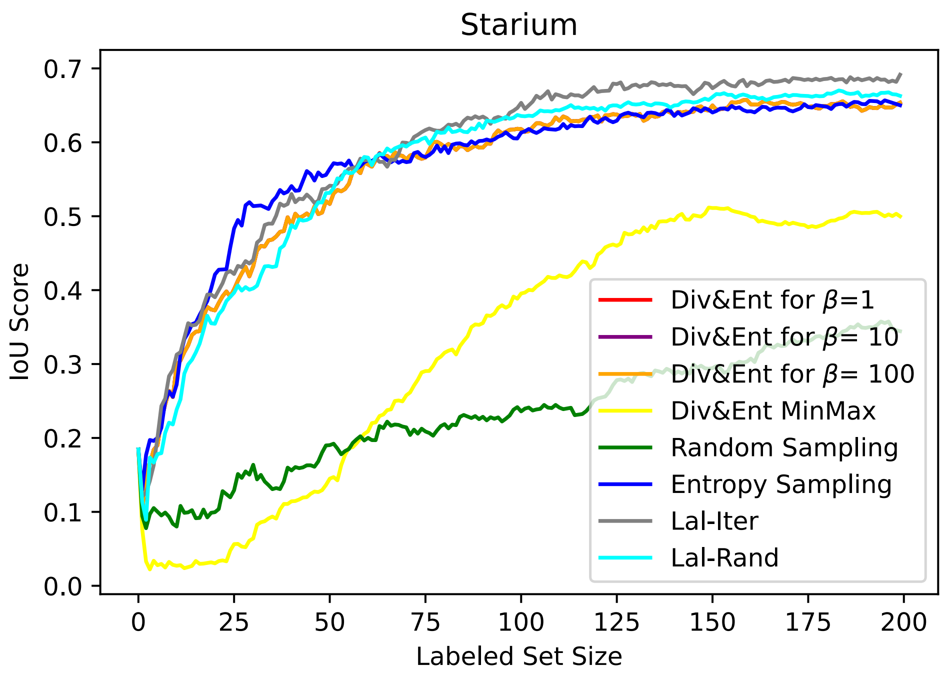

Figure 4.7: The Experiment Results for Stratium-Mini

Note that since the number of positive instances in the data set is low, in [12], instead of accuracy, intersection over union (IoU) metric is used to assess the performance. IoU metric is the number of true positives over the sum of true positives, false positives, and false negatives, see [6]. This time, the performance of our strategy is lower than the LAL strategy. One possible reason might be the poor estimation of the covariance matrix. Note that the number of dimensions of the data is 272. Compared the all other tests in this section, the most significant difference in this data set has high dimensional data. Although the number of dimensions is high, the number of samples is 2000, which is low considering the dimensions. Estimating the covariance matrix for high-dimensional data is a much more challenging task.

4.6.4.1 Cardiotocography Data Sets Experiments

Cardiotocography dataset is available in UCI Machine Learning Repository, see [5]. The Cardiotocography dataset consists of 2126 fetal cardiotocograms that have 21 features. Each data point is labeled as "normal" or "suspect" or "pathological". We used the same setting with [30] for this dataset as follows. The data's dimensions are reduced to 15 using principal component analysis(PCA). The initial labeled set size is assigned as 75 samples: 25 samples for each class. 30% of the data is separated randomly for each test set. Logistic regression is used as the classifier. The tests are repeated 25 times. The average results of the 25 trials are presented in Figure 4.8.

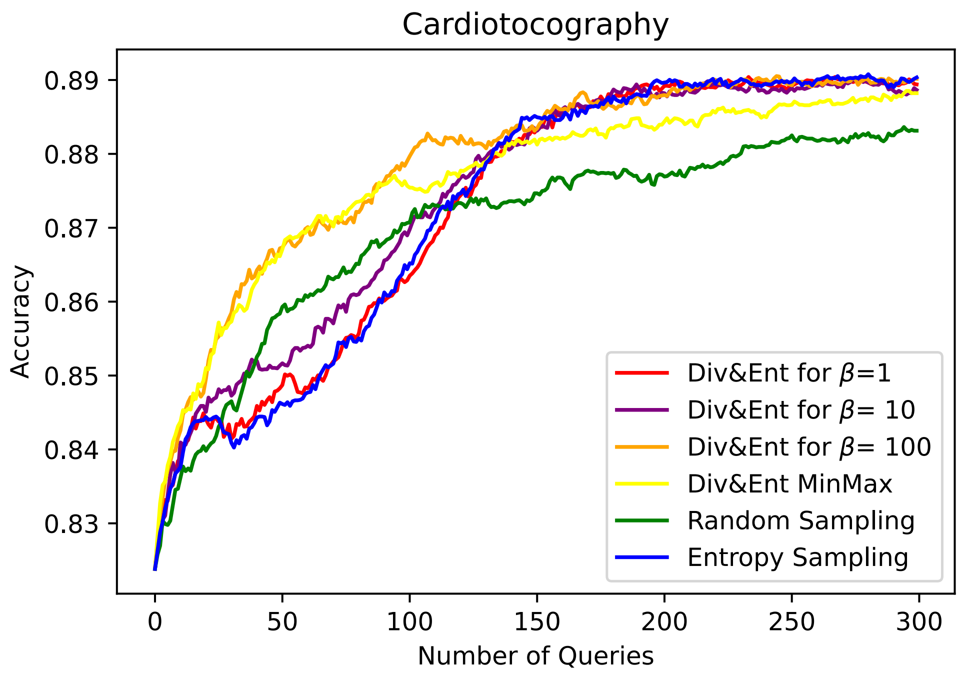

Figure 4.8: Cardiotocography Experiment Result

The results show that our exponential filtered strategy converges to the optimal accuracy rate much faster than others. Note that in the beginning of the experiment, although the entropy sampling have poor performance, our strategy shows remarkably better performance.

Since the software of [30] is not published, we could not produce the exact results of [30] for comparison. The trends of entropy sampling and random sampling are similar. However, the accuracy of the initial labeled set is quite different. The reason is that the test set is randomly selected for this dataset, and we do not know the exact settings for the logistic regression used by [30]. You can see their results in the following figure. Pess-MI and Opt-MI are the strategies introduced in [30].

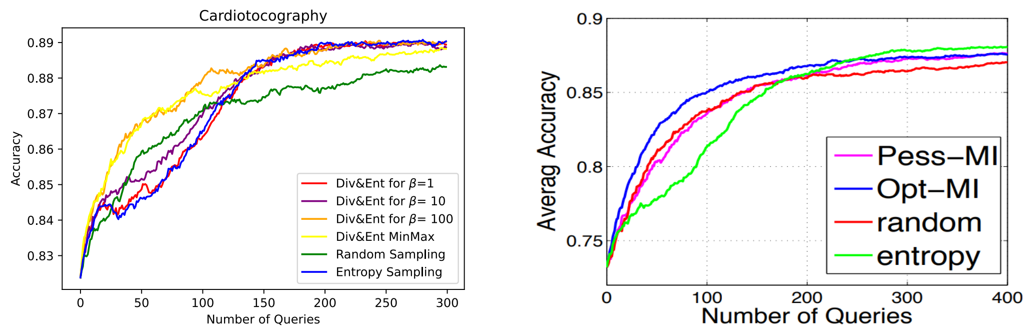

Figure 4.9: Cardiotocography Experiment Results' Comparison (Left: Our Results, Right: Result of[30])

The results are also shown in the same graph to see the difference better. You can find the results of [30] and our results in the following figure.

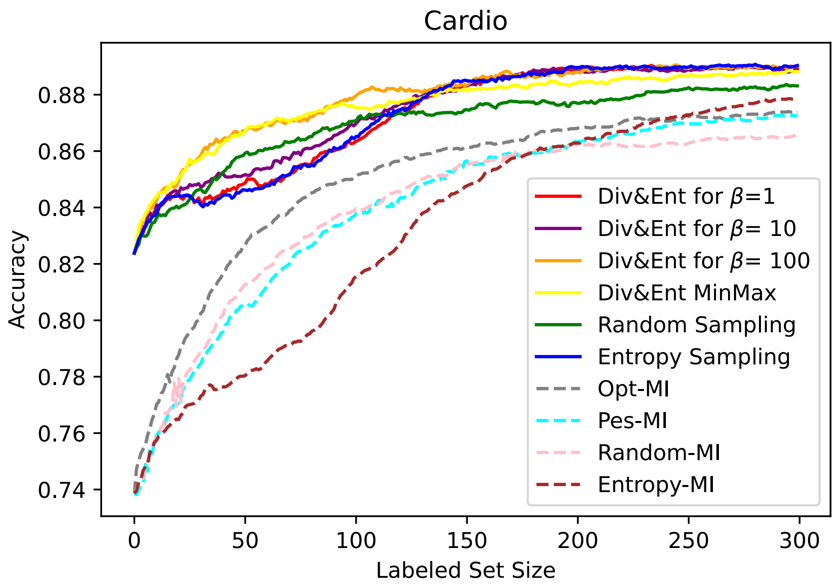

Figure 4.10: Cardiotocography Harmony Experiment Results' Comparison (Dashed lines are the results of [30])

4.6.4.2 Bach Choral Harmony Data Sets Experiments

Bach Choral Harmony is another data set used in [30]. Bach Choral Harmony contains pitch class information of time events of 60 chorales by Johann Sebastian Bach, see [19]. Each data point is represented by 12 features. The features are reduced to 8 using PCA as [30] did. In this dataset, we only used the points that belong to one of the following labels; D-major, G-major, C-major, F-major, and A-major as used in [30]. The size of the dataset is 2221. 30% of the data is separated randomly for each test set. Logistic regression is used as the classifier. The tests are repeated 25 times. The initial labeled set size is assigned as 25 samples; with 5 samples from each class. The average results of the 25 trials are presented in Figure 4.12.

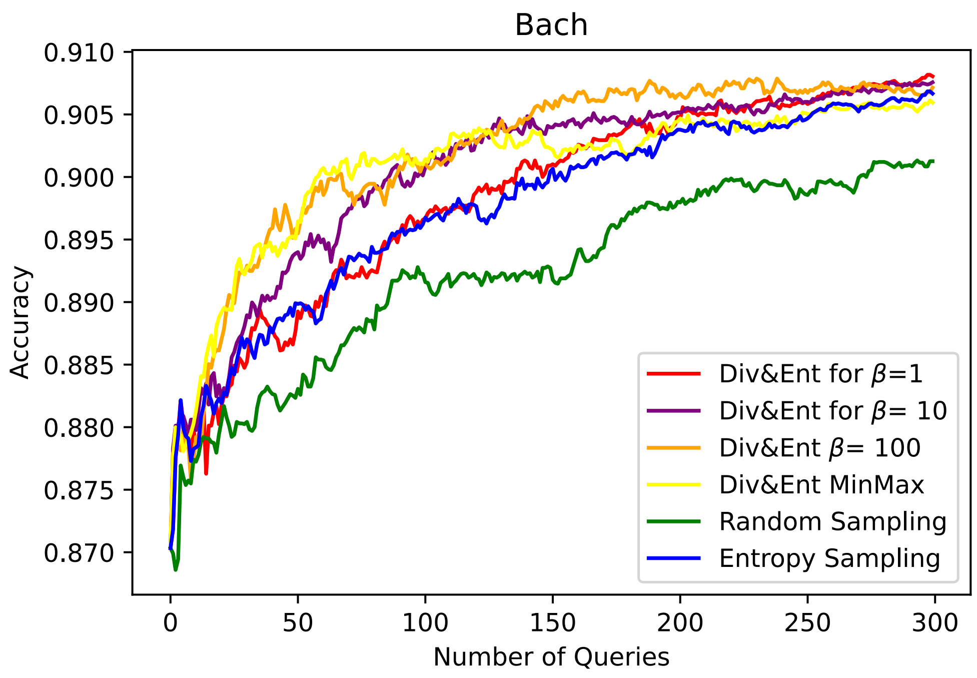

Figure 4.11: Bach Choral Harmony Experiment Result

The results show that our exponential filtered strategy converges to the optimal accuracy rate much faster than others. Since the software of [30] is not published, we could not produce the exact results of [30] for comparison. The trends of entropy sampling and random sampling are similar. However, the accuracy of the initial labeled set is quite different. The reason is that the test set is randomly selected for this dataset, and we do not know the exact settings for the logistic regression used by [30]. You can see their results in the following figure. Pess-MI and Opt-MI are the strategies introduced in [30].

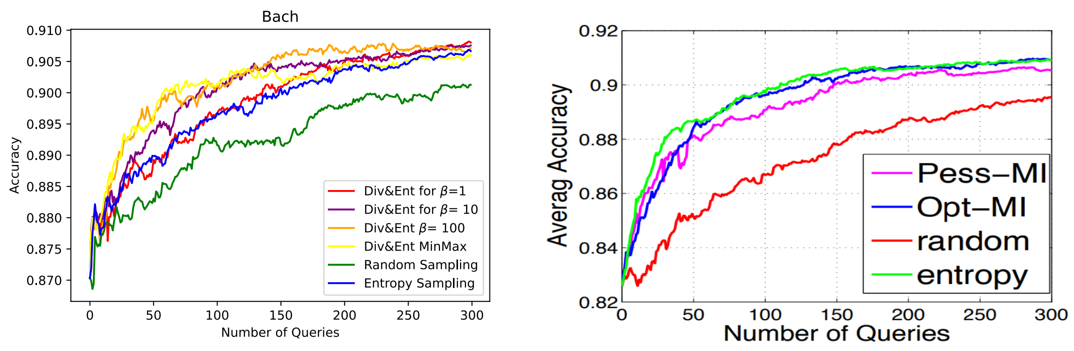

Figure 4.12: Bach Choral Harmony Experiment Results' Comparison (Left: Our Results, Right: Result of[30])

The results are also shown in the same graph to see the difference better. You can find the results of [30] and our results in the following figure.

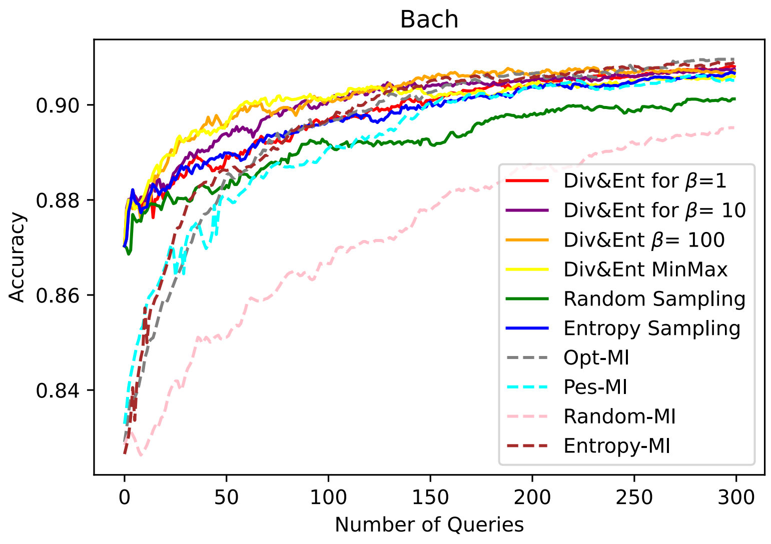

Figure 4.13: Bach Choral Harmony Experiment Results' Comparison (Dashed lines are the results of [30])

4.6.5 Extended Beta Parameter Tests

To investigate the effect of the parameter β, we extended our experiments on CheckerBoard4x4 and CheckerBoard2x2. Experiment results are presented in the following figures. The maximum observed accuracy value in the experiments according to β parameter is presented in the following table. The performance of the strategy decreases after β value reaches 130. The overflow was encountered in the calculation of the weights when β is greater than equals to 130.

Parameter	**CheckerBoard 2x2**	**CheckerBoard 4x4**
β	Max Observed Accuracy	Max Observed Accuracy
1	0.8543	0.794
10	0.935	0.82155
30	0.9967	0.85165
50	0.9971	0.8686
70	0.9922	0.86815
90	0.9964	0.86875
100	0.99625	0.87665
110	0.99605	0.8801
120	0.9967	0.89405
130	0.9646	0.83455

Table 4.1: Maximum Observed Accuracies in Extended Beta Parameter Tests

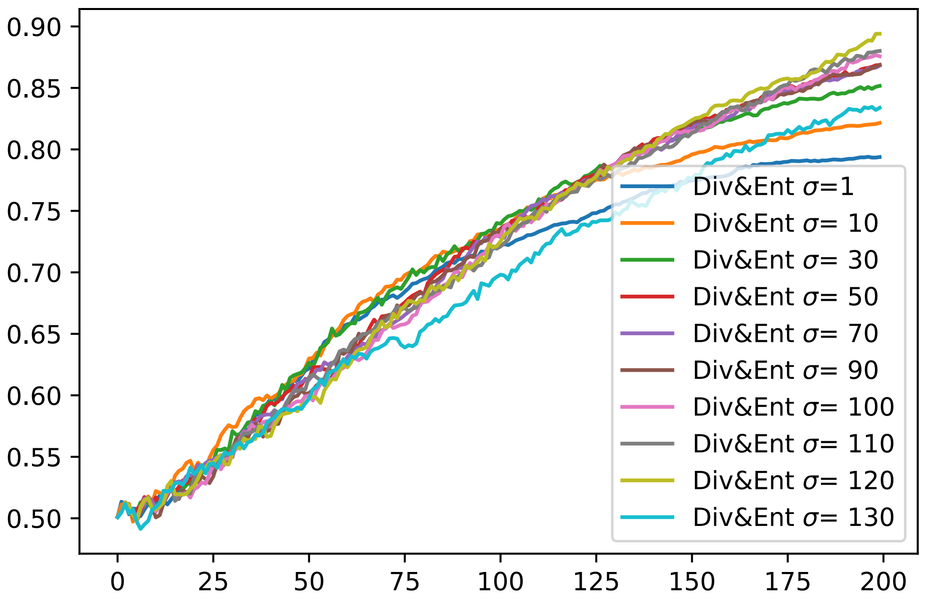

Figure 4.14: Extended Beta Parameter Tests on CheckerBoard4x4

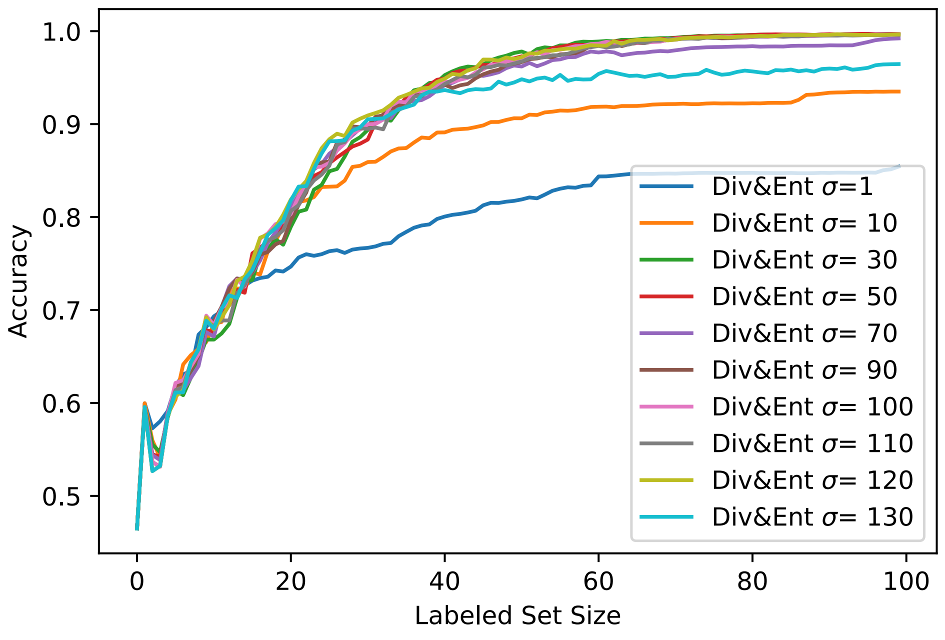

Figure 4.15: Extended Beta Parameter Tests on CheckerBoard2x2

4.6.6 An Idea with Kullback-Leibler Divergence

Although we could not show or prove (4.62) is a good way of estimating $D_{KL}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}}\right)$, we also tried using the Kullback-Leibler divergence as a density weight to the entropy-based strategy.

$$D_{KL}\left(\hat{p}_{\Omega} \| \hat{p}_{\mathsf{L}}\right) = \int_{-\infty}^{+\infty} \hat{p}_{\Omega}(x) \left(\log \frac{\hat{p}_{\Omega}(x)}{\hat{p}_{\mathsf{L}}(x)} \right) dx \tag{4.60}$$

$$= E_{\hat{p}_{\Omega}} \left[\log \left(\frac{\hat{p}_{\Omega}(x)}{\hat{p}_{\mathsf{L}}(x)} \right) \right] \tag{4.61}$$

$$\approx \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \left(\log \frac{\hat{p}_{\Omega}(x_{\omega})}{\hat{p}_{\mathsf{L}}(x_{\omega})} \right) \tag{4.62}$$

Basically, we used the left-hand side of (4.63) as density weight with this idea.

$$\underset{\omega^* \in \mathsf{U}}{\operatorname{argmin}} \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \left(\log \frac{\hat{p}_{\Omega}(x_{\omega})}{\hat{p}_{\mathsf{L} \cup \{\omega^*\}}(x_{\omega})} \right) = \underset{\omega^* \in \mathsf{U}}{\operatorname{argmax}} \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \log \hat{p}_{\mathsf{L} \cup \{\omega^*\}}(x_{\omega}) \tag{4.63}$$

The best results with the Kullback-Leibler divergence-based strategy were observed when β is 10. The comparisons between the Cauchy-Schwarz divergence-based strategy and the Kullback-Leibler divergence-based strategy are presented in the following figures.

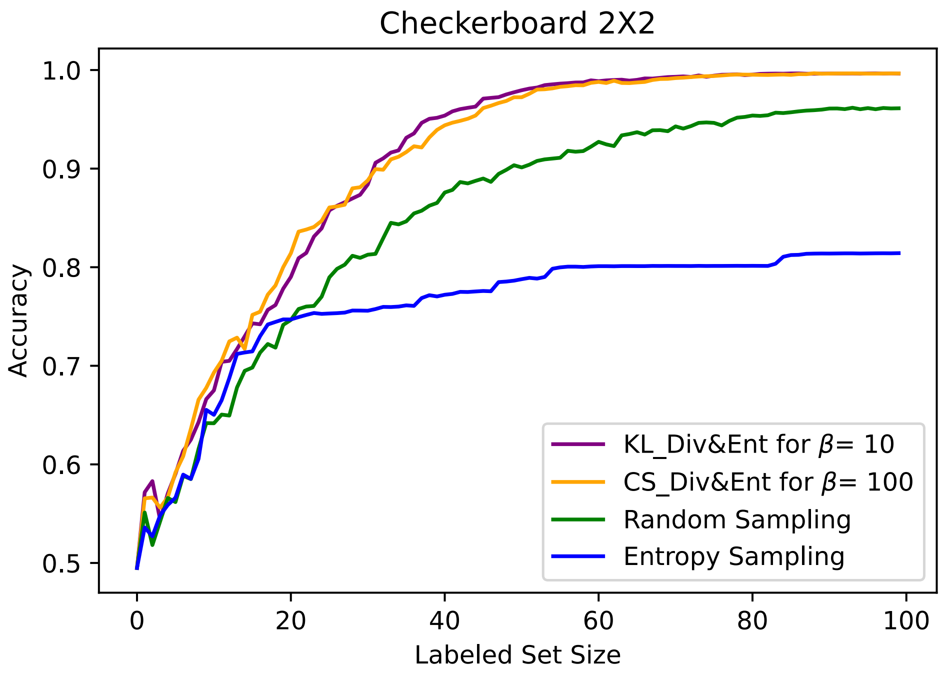

Figure 4.16: CheckerBoard 2x2 KL-CS Experiment Result

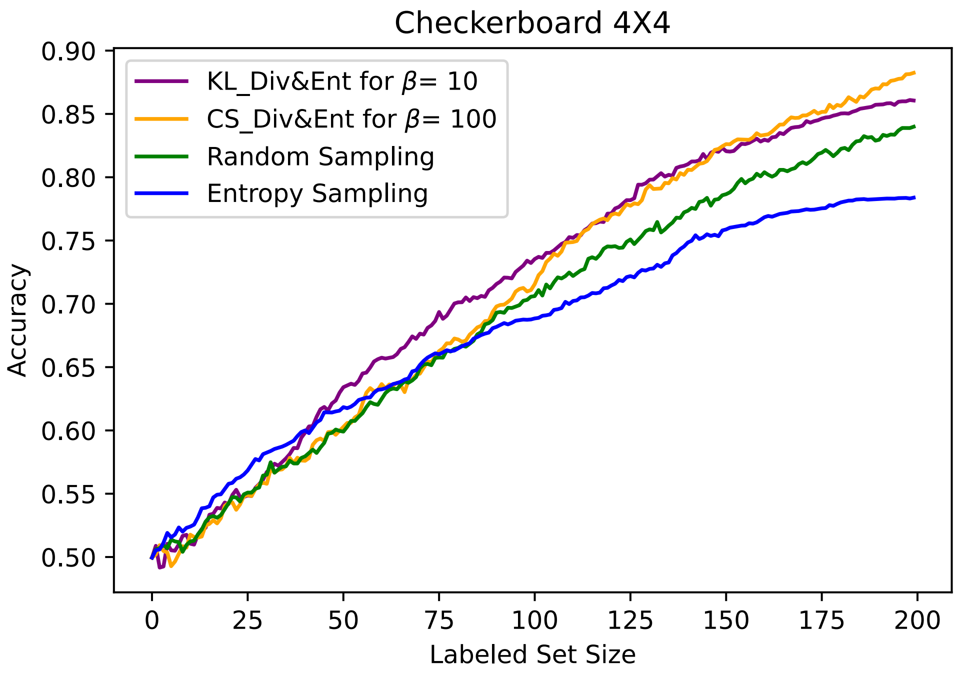

Figure 4.17: CheckerBoard 4x4 KL-CS Experiment Result

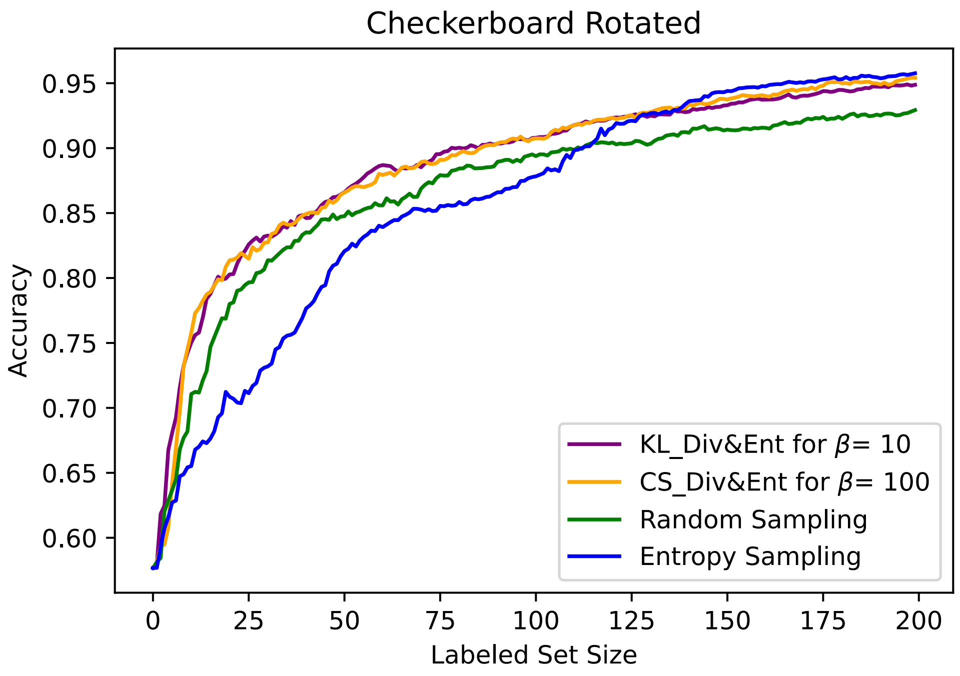

Figure 4.18: CheckerBoard Rotated KL-CS Experiment Result

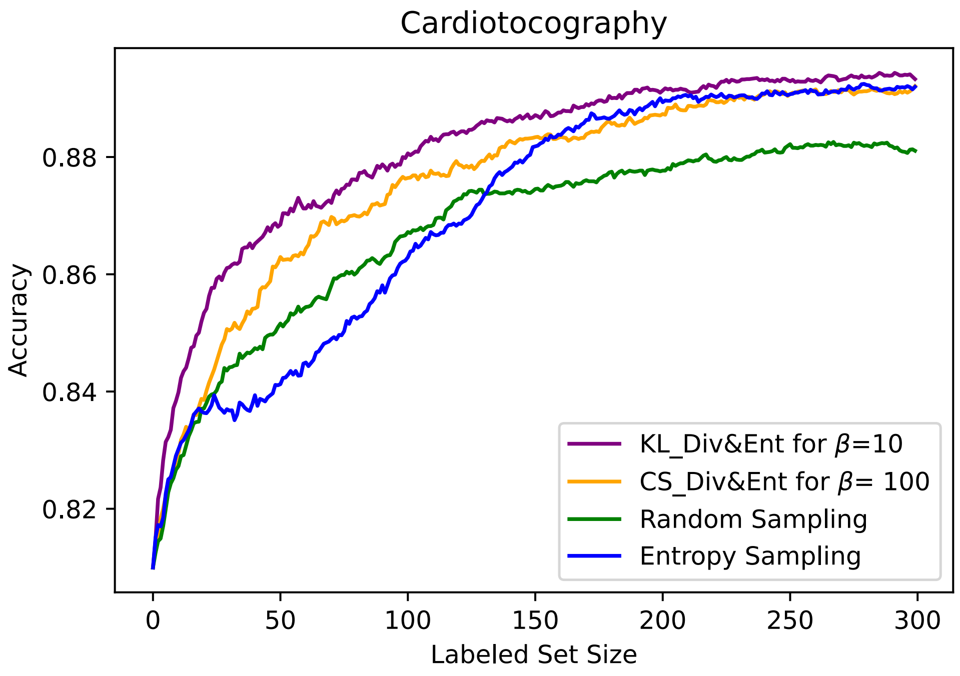

Figure 4.19: Cardio KL-CS Experiment Result

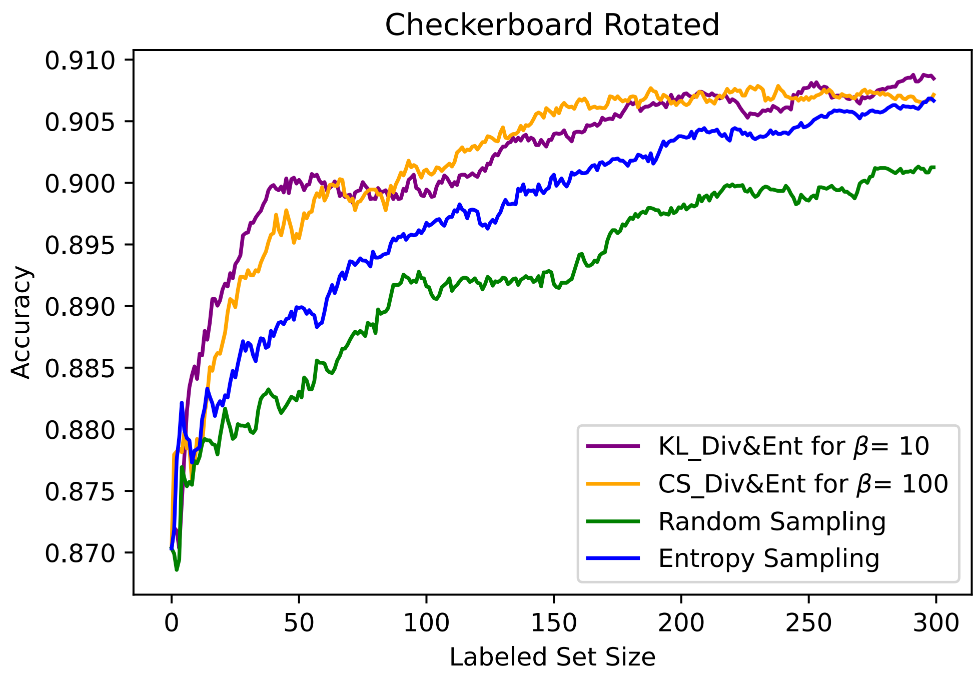

Figure 4.20: Bach Choral Harmony KL-CS Experiment Result

CHAPTER 5

CONCLUSION

In this thesis, the general active learning framework is discussed and reviewed. In the given framework, the most used active learning strategies are explained. The focus is on the active learning strategies that use information-theoretic measures such as entropy and mutual information. We show the theoretical reasoning behind uncertainty-based active learning in terms of information theory by examining the entropy and mutual information as active learning objectives.

We investigated the density-weighted active learning strategy to question whether divergence measure can be used as density-weight in this context. Our insight is that to get a representative labeled set in the end, the distance between the pdf estimated from the labeled set and the pdf estimated from the data pool should be reduced. Thus, we proposed to give higher weights to the candidate points that reduce the distance between pdf estimated from the labeled set and pdf estimated from the data pool. In the estimation of the pdfs, we used the kernel density estimation method using the Gaussian kernel function. As the covariance of the Gaussian kernel, we estimated the covariance matrix from the data and applied Silverman's strategy. Several divergences were analyzed to use our strategy; Kullback-Leibler Divergence, Renyi Divergence, and Cauchy-Schwarz Divergence. We only manage to simplify the calculation of the Cauchy-Schwarz divergence. To use the Cauchy-Schwarz divergence as density-weight, the simplified calculation of the Cauchy-Schwarz divergence is minimized concerning the candidate point. We found that the result of the minimization only requires sums of the Gaussian kernels. Therefore, the computational complexity of our strategy is low. We integrated the Cauchy-Schwarz divergence-based density-weight to the entropy-based active learning objective. We proposed two different versions of our strategy; exponential filtered density-weight and min-max filtered density-weight. While the exponential filtered version requires an additional importance parameter, the min-max filtered version does not require any parameter to decide.

The test results showed that the exponential filtered density-weight entropy sampling performs very well on most of the test data sets, mainly when the importance of the density-weight is high (i.e., $\beta = 100$). The most significant difference is that the learner that uses our strategy reaches the optimal accuracy rate much more faster than the other strategies.

Lastly, we tried to apply a similar approach to the Kullback-Leibler divergence in Section 4.6.6. Although we could not justify our approach with the Kullback-Leibler divergence analytically, in the experiments we made, we got similar results with the Cauchy-Schwarz divergence based approach. Despite the lack of analytical justification, the result of the Kullback-Leibler divergence based approach was slightly better than the Cauchy-Schwarz divergence based approach for the Cardiotocography dataset.

5.1 Future Work

It is possible to extend the work discussed in this thesis. Note that the proposed Cauchy-Schwarz divergence-based density-weight can be applied to any active learning strategy. It has no dependency on the learner or the active learning strategy. Any active learning strategy's output can be weighted with our Cauchy-Schwarz divergence-based density-weight. The future work of this thesis can be listed as follows.

- Investigating the performance of the Cauchy-Schwarz divergence based density-weight on the existing active learning strategies other than the entropy sampling
- Studying on how to find the optimal value of the importance terms in exponential filtered weight for a given task
- Applying the strategy to other possible divergence measures
- Searching better covariance matrix estimation techniques

- Finding an analytic justification for the Kullback-Leibler divergence based approach given in Section 4.6.6

REFERENCES

[1] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 12 2004.

[2] C. C. Bonwell and J. A. Eison. Active learning: Creating excitement in the classroom. 1991 ashe-eric higher education reports. 1991.

[3] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, page 1433–1452, USA, 2014. Society for Industrial and Applied Mathematics.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[5] D. Dua and C. Graff. UCI machine learning repository, 2017.

[6] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–308, September 2009. Printed version publication date: June 2010.

[7] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[8] Y. Guo and R. Greiner. Optimistic active learning using mutual information. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 823–829, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[10] R. Jenssen, J. C. Principe, D. Erdogmus, and T. Eltoft. The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, 343(6):614–629, 2006.

[11] K. Konyushkova, R. Sznitman, and P. Fua. Introducing geometry in active learning for image segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2974–2982, 2015.

[12] K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] A. Krause and D. Golovin. Submodular function maximization. In *Tractability*, 2014.

[14] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in Information Retrieval*, 29, 2001.

[15] A. Lucchi, Y. Li, K. Smith, and P. Fua. Structured image segmentation using kernelized features. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 400–413, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[16] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions - i. *Mathematical Programming*, 14:265–294, 12 1978.

[17] 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.

[18] J. C. Principe. *Information Theoretic Learning*. Renyi's Entropy and Kernel Perspectives. Springer, New York, 2010.

[19] D. P. Radicioni and R. Esposito. *BREVE: An HMPerceptron-Based Chord Recognition System*, pages 143–164. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[20] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. A survey of deep active learning. *ACM Comput. Surv.*, 54(9), 2021.

[21] C. E. Ribeiro de Mello. *Active Learning : an unbiased approach*. Theses, Ecole Centrale Paris ; Universidade federal do Rio de Janeiro, June 2013.

[22] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[23] A. Schrijver. Combinatorial optimization. 2003. Springer.

[24] B. Settles. *Curious Machines: Active Learning with Structured Instances*. Phd, University of Wisconsin-Madison, 2008.

[25] B. Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin-Madison, 2009.

[26] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool Publishers, 2012.

[27] B. Settles and M. Craven. *An Analysis of Active Learning Strategies for Sequence Labeling Tasks*. Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 1070–1079. Association for Computational Linguistics, 2008.

[28] C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

[29] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Renyi's Entropy and Kernel Perspectives. Chapman and Hall, London, 1986.

[30] J. Sourati. *Information Theoretic Active Learning in Unsupervised and Supervised Problems*. Phd, Northeastern University, Boston, Massachusetts, November 2016.

[31] J. Sourati, M. Akcakaya, J. G. Dy, T. K. Leen, and D. Erdogmus. Classification active learning based on mutual information. *Entropy*, 18(2), 2016.

[32] J. Sourati, M. Akcakaya, T. K. Leen, D. Erdogmus, and J. G. Dy. Asymptotic analysis of objectives based on fisher information in active learning. *J. Mach. Learn. Res.*, 18(1):1123–1163, 2017.

[33] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.