INFORMATION THEORETIC ACTIVE LEARNING IN
UNSUPERVISED AND SUPERVISED PROBLEMS

A Dissertation Presented
by
Jamshid Sourati
to
The Department of Electrical and Computer Engineering
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in
Electrical Engineering

Northeastern University
Boston, Massachusetts
November 2016
Contents

List of Figures vi
List of Tables ix
Abstract of the Dissertation xi
Acknowledgments xiv

1 Introduction 1
1.1 General Learning Framework 2
1.2 Learning Costs 6
1.3 Measuring Performance 7
1.4 Two Different Querying Strategies 8
1.5 Information Theoretic Objectives 9
  1.5.1 Entropy 10
  1.5.2 Mutual Information (MI) 11
  1.5.3 Fisher information (FI) 12
1.6 Overview of Methods and Our Contribution 13
  1.6.1 Clustering and Image Segmentation 13
  1.6.2 Classification 15
  1.6.3 Our Contributions 17

2 Accelerated Interactive Image Segmentation 20
  2.1 Spectral Clustering and Affinity Propagation 20
## 2.1.1 Affinity Propagation

- 21

## 2.1.2 Computational Speed-up Techniques

- 22

## 2.2 Clusters Probabilistic Modeling

- 24

## 2.3 Edgewise Active Learning (EAL)

- 25
  - 2.3.1 Choosing the Uncertain Point
  - 27
  - 2.3.2 Choosing the Confident Points
  - 27

## 2.4 Implementation Details

- 28

## 2.5 Experimental Results

- 29
  - 2.5.1 Constant Softness Parameter
  - 31
  - 2.5.2 Dynamic Softness Parameter
  - 32
  - 2.5.3 Active Learning Evaluation
  - 35

## 3 MI-based Classification Active Learning

39

### 3.1 Classification Active Learning

- 39

### 3.2 Mutual Information vs. Entropy

- 41

### 3.3 Evaluating Mutual Information

- 42

### 3.4 Randomized vs. Deterministic Submodular Optimizations

- 46
  - 3.4.1 Submodular Set Functions
  - 46
  - 3.4.2 Submodular Maximization
  - 48

### 3.5 Total Complexity Reduction

- 50

### 3.6 Further Speed-up

- 52

### 3.7 Results

- 53
  - 3.7.1 Cardiotocography
  - 53
  - 3.7.2 MNIST
  - 53
  - 3.7.3 Bach Choral Harmony
  - 53
  - 3.7.4 Experimental Settings
  - 54
  - 3.7.5 Numerical Results
  - 55

## 4 FI-based Classification Active Learning

62

### 4.1 Theoretical Analysis

- 62
  - 4.1.1 The Framework and Assumptions
  - 63
C.2 Updating the Gradient Vector .................................................. 117
C.3 Updating the Inverse Hessian Matrix ...................................... 118

D Statistical Background ............................................................... 120
D.1 Convergence of Random Variables ........................................... 120
D.2 Parameter Estimation .............................................................. 122

E Proof of Second-order Multivariate Delta Method ......................... 124
E.1 Stochastic Order Notations ...................................................... 124
E.2 Second-order Statistical Taylor Expansion ................................. 127
E.3 Second-order Multivariate Delta Method .................................. 129

F Proof of Lemma 4.7 ...................................................................... 131

G Proof of Theorem 4.8 ................................................................. 134
List of Figures

1.1 Diagram of different parts of an active learning algorithm .................................................. 3

2.1 Illustrating probabilistic concepts of our approach: (a) three Gaussian components with different means and equal covariances used to generate the points that are displayed over contours of their marginal distribution. The points in each desired (gold-standard) cluster are shown in a different color and shapes; (b,c) the resulting groups after doing an unconstrained spectral clustering lying on contours of their corresponding posterior distribution; (d) the resulting groups shown together over contours of the posterior entropy $h$, with the encircled point as the one with the largest $h$; (e) the resulting groups over contours of the density weighted entropy $\phi$, with the encircled point as the one with maximum $\phi$ (i.e. $x_{\pi_0}$); (f) the resulting groups over contours of the inverse-density weighted entropy $\psi$, with the encircled points having the minimum $\psi$ and shortest distance to $x_{\pi_0}$ in each cluster (i.e. $x_{\pi_1}$ and $x_{\pi_2}$). .............................................................................................................................................. 26

2.2 Results of running the algorithm with constant $\epsilon$: (a) average and std RI versus different number of constraints; (b) the accuracy image: RI of segmenting individual images versus the iteration index ........................................................................................................ 31
2.3 Illustration of a potential reason for fluctuating results. Blue and red pixels form two clusters and the thick dark blue boundary separates the correct segments. (a) The segmentation result at iteration $t$ which is close to the ground-truth together with the constraints $(Q_{t+1}, Y_{Q_{t+1}})$ including a must- (Solid edge) and cannot-link (dashed line). Observe that the central point $x_0$ is an object pixel with features very similar to background. (b) The results after applying $(Q_{t+1}, Y_{Q_{t+1}})$. Notice that many background pixels have been pushed into the object cluster and the accuracy is reduced $\text{RI}(t) > \text{RI}(t + 1)$.

2.4 (a) Average and (b) standard deviation RI of trying different values for $m$ in equation (2.13) ($m = 10^{-4}, 10^{-3}, 10^{-2}$ and $10^{-1}$); average RI obtained by running the dynamic algorithm with $m = 10^{-2}$ over (c) Cars and (d) People categories of INRIA image data set.

2.5 RI measurements for segmentation of individual images while using dynamic softness with a linear schedule and $m = 10^{-2}$.

2.6 Segmentation results on four individual images: rows 1 and 4 indicate the original gray-value images and their RI plot. The blue lines show the selected iterations whose results are shown. Rows 2 and 5 show the static results, and the dynamic results are rows 3 and 6.

2.7 Comparing EAL ran with static affinity propagation ($\epsilon = 10^{-5}$) with three different query selection scenarios described as cases ((i)), ((ii)), ((iii)) in section 2.5.3, all of which ran with the sampling factor $\delta = 0.2$.

2.8 (a,b) the average and std RI of EAL and Wang and Davidson’s algorithm on a smaller dataset obtained by using $\delta = 0.5$; (c) the segmentation result of Wang and Davidson’s algorithm [7] on the old woman image shown in Figure 2.6(i) picked from an iteration after convergence. Observe that it has converged to an unbalanced segmentation with low accuracy; (d) Average running time of Wang and Davidson’s constrained spectral clustering [7] and our algorithm ran with constant $\epsilon = 10^{-5}$.

3.1 The experimental results of different querying approaches for sequential active learning ($k = 1$).
3.2 The experimental results of different querying approaches for batch active learning 
\((k = 5)\). ................................................................. 59
3.3 The experimental results of different querying approaches for batch active learning 
\((k = 10)\). ................................................................. 60
3.4 The experimental results of different querying approaches for batch active learning 
\((k = 20)\). ................................................................. 61
4.1 Different FIR-based querying algorithms with their objectives and assumptions over the data distributions. ................................................................. 92
4.2 Average and standard variation of accuracies on synthetic data set. .................. 93
4.3 Average and standard variation of accuracies on Bach Choral harmony data set. ... 93
4.4 Summary of the results of evaluating the querying algorithms over different pairs of the digits in MNIST data set for different batch sizes. ......................... 94
4.5 The frequency of rejecting the null hypothesis (in percentage) with confidence level 0.95 of our \(t\)-tests is shown in different querying iteration when using \(k = 30\). The percentage is reported across all combination of pairs of digits and randomly initialized experiments: 36 combinations and 10 initial labeled sets, leading to 360 total experiments. ................................................................. 94
4.6 The results of evaluating the querying algorithms over different pairs of the digits in MNIST data set for \(k = 30\). Figures 4.6(a), 4.6(b) and 4.6(c) show the average \(p\)-values for each pair of digits, averaged over all the iterations and randomly initialized experiments. ................................................................. 95
# List of Tables

1.1 Summarizing dichotomizations of the active learning methods discussed .................. 10

4.1 Reviewed FIR-based active learning algorithms for discriminative classifiers .............. 77

4.2 Computational complexity of different querying algorithms (recall that $n = |X|$, $k = |Q|$, $\Omega \subseteq \mathbb{R}^d$ and $c = \#$ classes) .................................................... 84
# List of Algorithms

1.1 Pseudocode of iterative active learning; the prediction step is not shown ........ 4  
2.1 Learning iterations of constrained spectral clustering with affinity propagation ... 24  
2.2 Edgewise Active Learning (EAL) ......................................................... 28  
3.1 Classification Active Learning ............................................................ 40  
3.2 The deterministic approach for solving cost-constrained querying optimization in (1.6) 49  
3.3 The randomized approach for solving cost-constrained querying optimization in (1.6) 51  
4.1 [1] ........................................................................................................... 78  
4.2 [2] ........................................................................................................... 79  
4.3 [3] ........................................................................................................... 80  
4.4 [4, 5] .................................................................................................... 81  
4.5 [6] ........................................................................................................... 83  
4.6 Our proposed FIR-based practical algorithm ............................................. 88  

B.1 Orthogonal iteration subroutine used in Spectral Clustering (SC) to estimate eigenvectors associated with the two largest eigenvalues of the Laplacian matrix. ............... 114
Abstract of the Dissertation

INFORMATION THEORETIC ACTIVE LEARNING IN UNSUPERVISED AND SUPERVISED PROBLEMS

by

Jamshid Sourati
Doctor of Philosophy in Electrical Engineering
Northeastern University, November 2016
Dr. Deniz Erdogmus, Dr. Jennifer G. Dy Advisers

In machine learning, the machine is a learning agent that aims to build a mapping function from a given set of objects to a specific domain of interest. In many applications such as in supervised classification or constrained clustering where the output domain includes class and cluster labels, respectively, learning this function requires having access to a labeled data set. Assuming that the labels are noiseless, this data set includes data samples for which the gold standard or true mapping has been evaluated, and therefore can be a useful guide towards estimating the true function. In other words, the labeled data set let the machine observe a sample of reality from which the machine will learn and extend this knowledge to unobserved cases. There are two remarkable points regarding this scenario: (1) different choices of labeled samples lead to different estimations of the mapping, and (2) observing the reality is always costly.

Depending on how the labeled data set is constructed, machine learning algorithms are categorized into two groups: passive learners receive the labels passively and play no roles in building the labeled data set, whereas active learners take part in choosing what part of the reality to observe. The goal in active learning is to query those labels that lead to a good estimation of the gold standard
mapping with the least cost. This problem can be formulated as an optimization problem provided that we define scoring functions to measure how good we might learn from observing certain parts of the reality after paying certain costs. Thus, one needs to define performance metrics and learning cost functions to do active learning.

In this thesis, we focus on active learning for two specific learning problems: classification and constrained clustering. For both problems, we consider solving the constrained optimization of maximizing the performance metrics when the cost of labeling (observation) is fixed. However, directly optimizing performance metrics is intractable in practice, and therefore we use information theoretic functions as surrogate objectives. Under certain assumptions, the learning cost is also shown to be proportional to the number of labels to be observed.

First, we consider the problem of constrained binary clustering, where a given set of samples are to be segmented into several groups, constrained that some samples fall into the same clusters and some are put in different groups. That is, the constraints are assumed to be provided in form of must-links and cannot-links between pairs of samples. We use spectral clustering with affinity propagation as an existing constrained clustering model, and develop an Edgewise Active Learning (EAL) algorithm to query pairwise relationships between certain samples. In this algorithm, the goal is to decrease uncertainty of the model, meaning that the surrogate objective is the model’s entropy. For optimizing this surrogate, we start from an initial clustering of the data and compute confidence of the model regarding cluster assignment of each sample. We then select the sample with lowest confidence and query its relationship with samples of highest confidence in each of the two clusters based on the available grouping.

Next, we switch to classification problem where the data samples are to be classified into given set of classes. In classification, entropy has the shortcoming of querying redundant information and hence sub-optimal. We develop active learning methods based on two more sophisticated objectives. Our algorithms are for cases where a pool of unlabeled samples is available and the queries to be labeled will be selected from that pool. One of our proposed algorithms is based on maximizing mutual information (MI) between the observed labels and the unlabeled samples, and the other is based on minimizing a scalar function of Fisher information matrix of the query distribution. We give a rigorous theoretical analysis of using the so-called Fisher information ratio whose optimization as the surrogate objective can be asymptotically viewed as a bound optimization.
of a common performance metric.
Acknowledgments

Thanks to my advisors and collaborators, I truly enjoyed learning many new things during my PhD years. This was a fruitful learning process helping me not only in technical topics, but also in organizing my thoughts, guiding me how to approach any given problem and how to present my ideas. I firstly joined Professor Deniz Erdogmus’s research group, Cognitive Systems Laboratory (CSL), at Northeastern University and soon after started working with Professors Jennifer Dy and Dana Brooks too on my first PhD project in a larger group called B-Spiral. Having three adviser’s in a single project can be challenging and helpful at the same time. Each of them was a limitless supply of ideas helping me with certain aspects of the project. I appreciate all their encouragements and advices in dealing with difficulties that I had to deal with during that project. Their suggestions to take few courses from department of mathematics turned out to be extremely useful in giving me strong tools for formulating our problems rigorously and better understand the challenges.

My second and third projects started almost simultaneously after Professor Murat Akcakaya joined our group. He played an enormous role in formalizing the problems we had to tackle with in those projects, understanding the existing solutions and then preparing our own framework. He was a great help for me with his firm and rich background in statistics, his sharpness in understanding and quick analysis of the problems that we kept encountering, and above all his extraordinary nice attitudes in our teamwork.

I also appreciate all my kind friends in and outside B-Spiral research laboratory, and my supportive family who always provided an absolutely comfortable environment for me to seek my interests and grow in them. I give my gratitude and love to my mother’s spirit who had always been my source of energy with her patience and kindness, and to my father who taught me how to fight persistently with difficulties to reach my goals.
Chapter 1

Introduction

The machine, as a learning agency, is usually considered as a student that is learning from the set of materials given to it by an external source (oracle). This scenario can be thought of as an education system, where the role of active learning as a way of involving “students in [something] more than listening” through “higher-order thinking” has become significant since few decades ago [8]. This is as opposed to the more traditional passive learning where students are passive receptors of the information and course materials. Although there are different ways of active learning in class, consider the act of asking questions from the teacher as an example of actively involving in the process of learning. The question itself can be viewed as an instance selected from (arbitrarily large) pool of all possible questions. Clearly, one cannot ask all possible questions due to the limited time. Hence, it is not only important to take the action of questioning, it also matters what question to ask. Good questions clearly are those to whom the teacher’s answers help the student understand the topic in the shortest time.

In machine learning, each learning agency (learner machine) can be viewed as a student that is learning a mapping function that maps a given data sample represented by a feature vector to a value in the response space. The learner is looking to select a reasonably “simple” mapping that fits into a set of learning materials given by an external source (oracle). These materials include either experiments to measure or raw data samples to annotate (label) by an external source (oracle). Here, the experiments or raw samples can be viewed as possible questions to be asked from the oracle, and as asking each question in a class has time cost, querying each unit of learning materials can be costly in terms of time or even money. In this thesis, we only consider selecting raw data samples to query their individual labels or their pairwise relationship from the user.

Similar to education, a passive machine learner has no control over the learning materials
that it receives, whereas an active one is permitted to query certain types of information from the oracle [9]. From this point of view, active learning is the process of coupled querying-learning strategies, where querying takes part in preparation of the learning materials and learning is the selection of the best mapping function. The problem in the first part of active learning, that is querying, is what materials to query to obtain the most accurate mapping with the least cost.

In this thesis, we consider two learning problems, clustering and classification and develop different active learning algorithms for each of them. We first consider an interactive clustering method as a grouping problem that can use user-provided information to make the clustering more accurate. We propose an algorithm that enables the clustering model to select certain data samples and ask the oracle if each pair of them are in the same cluster or not. These samples are chosen in a way that observing their pairwise relationship is expected to decrease the entropy or uncertainty of the model. Then, we describe two different active learning methods for classification problem where given objects are to be classified to certain classes. These models aim to choose the most informative learning materials based on Mutual information and Fisher information criteria. We analyze each model separately and show that one of them can be asymptotically related to the log-likelihood loss function.

1.1 General Learning Framework

In order to simplify the mapping selection, we assume that \( f \) is a parametric function. For clustering and classification problems, where the response space is finite and discrete, we can break the learning process into two parts: obtaining a continuous and parametric scoring function associated with each member of the response space (each label), and then using the scores generated by these functions to predict the label of a given sample.

According to our discussion so far, an active learning algorithm into three main parts as is shown in Figure 1.1(a):

(i) *(Preparing learning materials)* The input to this block is a set of raw unlabeled samples in form of a set of feature vectors, or a feature set, denoted by \( \mathcal{X} \). If \( \mathcal{X} \) is finite, it is denoted by \( X \) and is called the *unlabeled pool*. The step of preparing learning materials includes any process to prepare materials required for inference and prediction. If no external information is needed (e.g. in unsupervised models) only the input \( \mathcal{X} \) passes through and \( \mathcal{L} = \mathcal{X} \). Otherwise, this block can be viewed as the concatenation of the following two steps (see Figure 1.1(b)):
CHAPTER 1. INTRODUCTION

(a) Components of an active learning and their costs

(b) The sub-blocks regarding preparation of learning materials

Figure 1.1: Diagram of different parts of an active learning algorithm

(a) (Querying) In this step, the machine chooses or constructs a set of queries, denoted by $Q$, using samples inside the input feature set. The resulting query set $Q$ could be a subset of $\mathcal{X}$, $\mathcal{X} \times \mathcal{X}$ or any other forms of $\mathcal{X}$. Labels of the selected queries are requested from the external source of knowledge (the oracle). If $\mathcal{X}$ is finite, the resulting learning algorithm is called a pool-based active learning. Otherwise, querying from an infinite or uncountable space leads to a synthetic active learning.

(b) (Labeling) This step is performed outside the machine by an oracle, who annotates all the members in $Q$ and collect them in the label set $Y_Q$. The annotated queries are usually called labeled samples. The queried feature vectors together with their labels form the training data set denoted by $\mathcal{L} = (Q, Y_Q)^1$.

(ii) (Inference) Estimating parameter of the intermediate parametric scoring function based on the training data set formed in the previous step, and a given loss function.

(iii) (Prediction) Making decisions regarding class label of a given test sample $x_{\text{test}}$, which should have the same dimensionality as the labeled samples. In clustering (unsupervised) problems $x_{\text{test}}$ are samples from unlabeled parts of $\mathcal{X}$, whereas in classification (supervised) model, the test is not necessarily inside the training support, i.e. we may possibly have $x_{\text{test}} \notin \mathcal{X}$.

1For semi-supervised models the training data set is $\mathcal{L} = \mathcal{X} \cup (Q, Y_Q)$
CHAPTER 1. INTRODUCTION

Algorithm 1.1: Pseudocode of iterative active learning; the prediction step is not shown

Inputs: The feature set $\mathcal{X}$, initial learning materials $\mathcal{L}^{(0)}$

Outputs: Estimate of the learning model’s parameter $\hat{\theta}^{(t)}$, the predictions

```pseudocode
/* Initializing the iterations */
1 $\hat{\theta}^{(0)}$ ← initial parameter estimate based on $\mathcal{X}$ or $\mathcal{L}^{(0)}$
/* Learning iterations: */
2 for $t = 0, 1, 2, \ldots$ do
    /* Preparing Learning Materials */
3     $Q$ ← $\mathcal{A}(\mathcal{X}, \hat{\theta}^{(t)})$
4     $Y_{Q}$ ← labels of $Q$
5     $\mathcal{L}^{(t+1)}$ ← $(Q, Y_{Q})$
    /* Updating the parameter */
6     $\theta^{(t+1)}$ ← estimate of $\theta$ given $\mathcal{L}^{(t+1)}$ or $\bigcup_{i=1}^{t+1} \mathcal{L}^{(i)}$
    /* Predictions */
7     $\hat{y}_{\text{test}}$ ← label prediction for $x_{\text{test}}$ for all test samples
    /* Optional: if avoiding re-labeling */
8     $\mathcal{X}$ ← $\mathcal{X} - Q$
```

These steps are usually done iteratively, and at each (active) learning iteration the first block (preparation of the labeled data) uses the parameter estimate $\hat{\theta}$ available from the previous iteration. These steps are shown more formally in Algorithm 1.1 with $t$ as the iteration index. The module $\mathcal{A}(\mathcal{X}, \hat{\theta}^{(t)})$ is the function corresponding to preparation of learning materials, which includes the querying and labeling processes (see step (i) above). Initial estimation of the parameter in line 1 is needed to fuel the first-time running of the query selection $\mathcal{A}(\mathcal{X}, \hat{\theta}^{(t)})$ when $t = 0$ and it can be done differently in various applications. Moreover, line 8 is for cases when re-labeling the already labeled queries is to be avoided.

Example 1.1. Spectral clustering [10] is a learning model for clustering a given data set (see Section 2.1). In this model, $\mathcal{X}$ is the data set that has to be clustered (hence finite and denoted by $X$). In original form of clustering, there is no extra information other than the feature set and therefore there are no querying and labeling process are required. Hence, $\mathcal{L}$ would be the same as the unlabeled samples in $X$. Similarity matrix of the samples could be viewed as the model parameter of spectral clustering. The test samples $x_{\text{test}}$ are the same as the unlabeled samples in $X$, and the

---

2In case of spectral clustering, there are several other possible choices as for model parameter, such as Laplacian matrix or its first few eigenvectors. However, discarding uncertainties involved in numerical eigendecomposition processes, all
CHAPTER 1. INTRODUCTION

predictions \(\hat{y}_{test}\) is the cluster to which \(x_{test}\) is assigned.

Constrained clustering models fit better into the diagram in Figure 1.1. They interact with an external oracle to get constraints over the cluster assignments of the samples. The constraints are usually given in form of pairwise links, the query set \(Q\) includes pairs of unlabeled samples \((x_i, x_j)\) where \(x_i, x_j \in \mathcal{X}, i \neq j\) (hence \(Q \subseteq X \times X\)) and the labels \(Y_Q\) contains binary labels for the link to specify must-links (\(x_i\) and \(x_j\) in the same cluster) and cannot-links (\(x_i\) and \(x_j\) in different clusters). The initial parameter estimate \(\hat{\theta}^{(0)}\) for active clustering can simply be provided by running the unconstrained clustering over \(X\) and hence we can skip labeling samples in the beginning (that is \(L^{(0)}\) be the same as \(X\)).

Example 1.2. Logistic regression [11] is a popular discriminative classifier (see Section C.1). In this model, \(\mathcal{X}\) is the unlabeled data available in the training step (usually finite and therefore denoted by \(X\)), the queries \(Q\) contain a set of selected unlabeled samples (that is \(Q \subseteq \mathcal{X}\)) for whom the class labels would be requested. Hence, \(L\) would be a set of pairs of queried samples and their class labels which is given by the oracle. The parameter estimate \(\hat{\theta}\) would be maximum likelihood estimate (MLE) of the classes’ parameters which appear in the posterior models. Finally, in this application the test samples \(x_{test}\) are usually coming from a different distribution than the training distribution, and \(\hat{y}_{test}\) is the predicted class label for \(x_{test}\). Here, the initial parameter estimate \(\hat{\theta}^{(0)}\) can be provided through a given initial labeled data set \(L^{(0)}\).

In passive learning the querying step (step (a)) is missing, hence it includes labeling of a given \(Q\) that is followed by the training the model \(M\). However, in active learning the algorithm should first choose the queries \(Q\), and then proceed to passive learning. Generally speaking, the queries are chosen such that a high-performance algorithm can be trained using a small cost. We discuss about these two components in more details in the next two sections of this chapter. Here, we give a brief introduction to present our formulation.

Performing each step of Figure 1.1 requires paying some costs, which are in form of time duration, financial expenses or both. Let us denote the total cost as \(C_{\text{total}}(Q, M)\), where it depends on the query set and also the classification model. Furthermore, let \(\text{Perf}(M, L)\) be a metric evaluating performance of model \(M\) that is trained with labeled materials \(L\). Then, as is mentioned above, the

these variables are equivalent. Here, we chose the similarity matrix to be the parameter, since it is more compatible with the constrained clustering framework that we use in Chapter 2.
CHAPTER 1. INTRODUCTION

queries should be selected according to the following optimization:

\[
\arg \min_Q C_{\text{total}}(Q, \mathcal{M}) - \text{Perf}(\mathcal{M}, \mathcal{L})
\]  

(1.1)

Recall that \(\mathcal{L}\) includes \(Q\) together with labels of its members. Optimizing (1.1) where both costs and performance metric are involved is usually intractable, hence in practice usually one is kept bounded (or fixed) and optimization is done with respect to the other one. We call the resulting methods cost-constrained (when cost is kept constant) and performance-constrained (when the performance is kept constant) querying methods:

Cost-constrained: \[
\arg \min_Q \text{Perf}(\mathcal{M}, \mathcal{L})
\]  

such that \(C_{\text{total}}(Q, \mathcal{M}) \leq \text{const.}\)  

(1.2a)

Performance-constrained: \[
\arg \min_Q C_{\text{total}}(Q, \mathcal{M})
\]  

such that \(\text{Perf}(\mathcal{M}, \mathcal{L}) \geq \text{const.}\)  

(1.2b)

The constraints could also be in form of equalities. As we explain in the section 1.3, we focus on the cost-constrained methods in this thesis. Let us now explain in more details about the costs and the performance metrics.

1.2 Learning Costs

The total cost \(C_{\text{total}}(Q, \mathcal{M})\) includes two different parts: internal costs, \(C_{\text{int}}(Q, \mathcal{M})\), referring to the time duration that the machine needs to handle the computational complexities of selecting the queries and then processing the labeled data \(\mathcal{L}\) using the model \(\mathcal{M}\); and external costs, \(C_{\text{ext}}(Q)\), including time and financial expenses that have to be paid in order to access to labeled materials provided by the external oracle for queries \(Q\).

As is clear from Figure 1.1, only the labeling step contains external costs. Suppose \(c(x)\) is the external cost of labeling a single query \(x \in Q\). We make the following three assumptions:

- The external cost us uniformly distributed over all the queries that can be possibly chosen, that is \(c(x) = c \ \forall x \in Q\), where \(c > 0\) is a constant.
CHAPTER 1. INTRODUCTION

- The external cost is additive, that is the cost of labeling multiple queries is equal to the sum of individual costs: 
  \[ C_{\text{ext}}(Q) = \sum_{x \in Q} c(x) \]

- The external cost dominate the total internal costs: 
  \[ C_{\text{ext}}(Q) \gg C_{\text{int}}(Q, M) \]

Then, based on these assumptions, the total cost of constructing the algorithm can be approximated as:

\[
C_{\text{total}}(Q, M) = C_{\text{ext}}(Q) + C_{\text{int}}(Q, M) \approx C_{\text{ext}}(Q) \\
= \sum_{x \in Q} c(x) = \sum_{x \in Q} c = c \cdot |Q| 
\]

(1.3)

Hence, under the assumptions we mentioned above, the total learning cost is equivalent to the size of the query set. This implies that in performance-constrained querying (1.2b), among the query sets that lead to a fixed performance metric, we should choose the smallest one. Therefore, even if \( \mathcal{X} \) is finite and labeling all its members satisfies the performance inequality in (1.2b), it is not desirable to take \( Q = \mathcal{X} \) unless there is no smaller subset satisfying the inequality.

1.3 Measuring Performance

From the two different optimization problems in (1.2), two questions arise that should be clarified in each active learning model:

(1) How do we define a performance metric for an algorithm?

(2) In performance-constrained query in (1.2b), how does one be sure that a given set \( Q \) leads to a model with the desired performance?

Defining metrics for measuring performance of a learning algorithm is well-studied for different types of problems [12, 13]. We can categorize the metric into inference- and prediction-based metric. In the former, the evaluation is done by computing how close the parameter estimate \( \hat{\theta} \) is to the true parameter \( \theta_0 \) of the model, and in the former the evaluation is based on checking how accurate \( \hat{y}_{\text{test}} \) is with respect to the true class label \( y_{\text{test}} \). Mean-square-error (MSE) and accuracy, defined as below, are examples of inference- and prediction-based metrics, respectively:

- **MSE**:
  \[
  \text{MSE} = \mathbb{E} \left[ \| \hat{\theta} - \theta_0 \|^2 \right] 
  \]
  (1.4a)

- **Accuracy**:
  \[
  \text{Accuracy} = \mathbb{E} \left[ \hat{y}_{\text{test}} = y_{\text{test}} \right] 
  \]
  (1.4b)
CHAPTER 1. INTRODUCTION

Question (2) is more crucial. Giving an exact answer to this question is only possible if we actually label the samples in $Q$, do the learning accordingly and then evaluate the performance metric of the resulting model. However, following this procedure for all possible query sets is highly inefficient and usually impossible in practice since the number of possible sets $Q$ is huge. The crucial point here is that we should “predict” the performance of the algorithm if we had received labels of $Q$. Such prediction should be performed before any labeling is done.

Unfortunately, the performance metrics are hard to predict and surrogate objectives are used to approximately answer question (2). Similar to performance metrics, there are inference-based and prediction-based surrogates. They “predict” how good a given set of queries can effect a quality measure related to inference or prediction steps, respectively. We will discuss different types of surrogates in detail in the next sections. One should keep in mind that some of these surrogates do not have any direct relationship with the target performance metric. That is bounding (or fixing) them to a specific value is not necessarily equivalent to the constraint of having a bounded (or fixed) performance metric. Because of this ambiguity, instead we fix the size of the query set $|Q|$ to $k \in \mathbb{Z}_+$ (a positive integer) and select the queries $x$ such that the resulting $Q$ is expected to give the best surrogate objective (cost-constrained method).

1.4 Two Different Querying Strategies

Our descriptions here are given based on the assumption that $Q \subseteq \mathcal{X}$, but they can be extended to other cases too (e.g. pairwise queries where $Q \subseteq \mathcal{X} \times \mathcal{X}$). Note that although the feature space $\mathcal{X}$ can be infinite, the query set $Q \subseteq \mathcal{X}$ is always a finite subset. Suppose we fix size of the query set to a positive integer $0 < k < \infty$ (equality cost-constrained method). Then, the problem is to construct a query set $Q$ such that $|Q| = k$ and it gives out the best surrogate objective. If $k = 1$, the resulting framework is called sequential active learning, and if $k > 1$, we get batch active learning.

There are two general groups of querying algorithms in terms of selecting the query members $q$:

- **Deterministic querying**: In this family of methods, the surrogate objective is defined directly over the subsets of $\mathcal{X}$. Meaning that it is a scoring function assigning a real-valued score to each subset:

$$f : 2^{\mathcal{X}} \rightarrow \mathbb{R}. \quad (1.5)$$

Suppose that the surrogate is designed such that higher scores imply better query sets, then the
cost-constrained method is formalized as:
\[
\arg \max_{Q \subseteq \mathcal{X}, |Q|=k} f(Q)
\]  
(1.6)

This optimization, in its exact solution, ranks subsets of \( k \) elements in terms of their surrogate values and aggressively chooses first one. That is, the rest of the subset does not have any chance of being selected. Note that such exact solution is not possible when \( \mathcal{X} \) is infinite, and even in the easier case of pool-based active learning, it is NP-hard.

- **Probabilistic querying:** Here, the surrogate is not defined directly over the subsets. Alternatively, it is a function over all probability distributions in \( \mathcal{X} \), assigning a score to each distribution. In order to select the queries, the surrogate is optimized with respect to the probability distribution and the queries are then sampled from the resulting distribution, which we refer to as training or proposal distribution. Suppose \( F \) is the surrogate and \( q \) is a probability measure over \( \mathcal{X} \), then the probabilistic active learning is formalized as
\[
q^* = \arg \min_q F(q)
\]  
(1.7a)

such that
\[
\int_{\mathcal{X}} dq(x) = 1,
q(x) \geq 0 \quad \forall x \in \mathcal{X}
\]

\[Q = \{x_1, \ldots, x_k\} \quad \text{where} \quad x_i \sim q^*(x)
\]  
(1.7b)

This optimization is less aggressive than (1.6), since even after optimizing with respect to the distribution \( q \), different subsets have a chance of being selected. Therefore, there is relatively more exploration and less exploitation in probabilistic querying [1]. Furthermore, in contrast to the deterministic approach, such probabilistic methods can handle the synthetic active learning too. It is also much more efficient for pool-based scenario where it can be done through a continuous optimization.

### 1.5 Information Theoretic Objectives

Different surrogates have been proposed to be used as querying objectives. We will review them in each application separately in the following sections. Here, we give a brief introduction
CHAPTER 1. INTRODUCTION

Table 1.1: Summarizing dichotomizations of the active learning methods discussed

<table>
<thead>
<tr>
<th>Dichotomization</th>
<th>Differentiation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost-constrained</td>
<td>Using either (1.2a) or (1.2b) as for the optimization querying</td>
</tr>
<tr>
<td>Pool-based</td>
<td>Nature of $\mathcal{X}$</td>
</tr>
<tr>
<td>Inference-based</td>
<td>The surrogate is related to either inference or prediction steps of learning</td>
</tr>
<tr>
<td>Sequential</td>
<td>Size of the query set $</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Type of the optimization: either (1.6) or (1.7)</td>
</tr>
</tbody>
</table>

of one of the most popular surrogates, that is information theoretic objectives. These objectives view the parameter estimate (in inference step) or the predicted label (in prediction step) as random variables, and score query subsets based on the amount of information they provide about those random variables. Hence, in all of them we need to have a probabilistic model for the inference or prediction steps.

In this section, we consider only one iteration of a pool-based active learning. That is, we are given a finite pool of samples $\mathcal{X}$ and we need to select a subset of queries $Q$. Recall that in cost-constrained learning, we fix size of $Q$ by a positive integer $k$. Three of information theoretic objectives include entropy, mutual information, and Fisher information:

### 1.5.1 Entropy

Entropy is a criterion that evaluate the uncertainty about the value of a given random variables. As one of the simplest querying objectives, a query subset $Q$ is to be chosen whose labels are expected to lower the entropy of the target random variables (either class labels or estimated parameter). In other words, this surrogate objective tends to minimize the uncertainty in the model with respect to the query subset to be labeled in the next learning iteration. Hence, the resulting querying method is also called uncertainty sampling [14].

Let $A$ be a random variable with possible outcomes $\Omega$, then its entropy $H(A)$ is\(^3\)

$$H(A) = - \sum_{A \in \Omega} P(A) \log P(A).$$ (1.8)

Now consider labels of the samples in $\mathcal{X}$ as random variables. That is, for any given $x \in \mathcal{X}$, its label $y$ is a random variable with posterior distribution $P(y| x)$. Then, one objective in learning

\(^3\)The definition of entropy is more general than this. Indeed, any function with certain properties can be called an entropy function. The definition given here is called Shannon entropy.
iteration is to lower the uncertainty or entropy of the labels. According to this objective, a query set \( Q \) should be selected whose labels result in the lower entropy of the remaining labels. Therefore, the querying objective would be the entropy of the remaining labels \( Y_{\mathcal{X}-Q} \) conditioned on the query labels \( Y_Q \):

\[
\arg\min_{Q \subseteq \mathcal{X}, |Q|=k} H(Y_{\mathcal{X}-Q}|Y_Q)
\]  

(1.9)

where \( Y_Q \) is still a random variable since we still haven’t done the labeling and hence haven’t observed \( Y_Q \). Conditional entropy is defined as:

\[
H(Y_{\mathcal{X}-Q}|Y_Q) = \sum_J H(Y_{\mathcal{X}-Q} | Y_Q = J)
\]  

(1.10)

where \( J \) is an assignment to the query labels \( Y_Q \). Equation (1.10) implies that computing conditional entropy requires training the model using all possible assignments \( J \)'s and then computing the resulting conditional entropy. This can be impossible since possible values of \( J \) can be very large. But, we can use the identity below to simplify the optimization. For any subset \( Q \subseteq \mathcal{X} \), we can write

\[
H(Y_{\mathcal{X}}) = H(Y_Q) + H(Y_{\mathcal{X}-Q}|Y_Q).
\]  

(1.11)

Since the left-hand-side of this identity is independent of \( Q \), minimizing the second term with respect to \( Q \) is equivalent to maximizing the first term:

\[
\arg\max_{Q \subseteq \mathcal{X}, |Q|=k} H(Y_Q)
\]  

(1.12)

In (1.12) there is no need to train the model and we can use the un-conditional probabilistic model to estimate the entropy. We will see in the next chapters that under certain conditions that happen very frequently in learning models, the optimization (1.12) leads to sub-optimal selection of queries. More specifically, it can lead to querying samples whose labels give redundant information.

1.5.2 Mutual Information (MI)

Mutual information between two random variables evaluates the amount of information one gives with respect to the other. Suppose \( A \) and \( B \) are two random variables. Then, MI between \( A \) and \( B \) is defined in terms of entropy functions as below:

\[
MI(A, B) = H(A) - H(A|B) = H(B) - H(B|A)
\]  

(1.13)
CHAPTER 1. INTRODUCTION

Notice that MI between the random variables is equal to the reduction in uncertainty of one of them if we observe the other one. Hence, MI can be used as an information theoretic objective to choose a query subset that contains the most amount of information about the target random variables. Similar to 1.5.1, consider predicted labels as the target variables. Then, in a cost-constrained pool-based scenario, the MI-based querying objective can be formalized as

$$\arg \max_{Q \subseteq \mathcal{X}, |Q| = k} MI(Y_Q, Y_{\mathcal{X} \setminus Q})$$

(1.14)

Using MI as the querying objective has nicer characteristics than entropy, especially in classification active learning. Specifically, it does not have the issue of redundancy, however, optimizing this objective is much harder to optimize.

1.5.3 Fisher information (FI)

Fisher information measures the amount of information that a given observation has about the parameter of its underlying parametric distribution. Suppose that the pair of a feature vector and its label, \((x, y)\), is seen as random variable with an underlying parametric joint distribution \(\mathbb{P}(x, y|\theta)\). Then, Fisher information of such a distribution is defined as:

$$I(\theta) = \mathbb{E} \left[ \nabla_\theta \log \mathbb{P}(x, y|\theta) \nabla_\theta^\top \log \mathbb{P}(x, y|\theta) \right].$$

(1.15)

Since FI is used only in conjunction with the parameter of the underlying distribution, it is always used as an inference-based surrogate objective and may be optimized to give a query subset which contains the most amount of information about the parameter of the underlying data distribution. This objective does not have redundancy issue of the entropy objective, and is also easier to optimize than MI. However, there are two points to be addressed:

(1) FI as is defined in (1.15) does not depend on the query set and it cannot be optimized with respect to \(Q\) as in deterministic querying (1.6). However, one should note that it does depend on the training distribution \(q(x)\) through the expectation which is taken with respect to the joint \(\mathbb{P}(x, y|\theta) = q(x)\mathbb{P}(y|\theta, x)\). Hence, probabilistic querying in (1.7) is more appropriate for this objective.

(2) Note that the definition in (1.15) leads to a matrix and therefore cannot be used as an optimization objective directly. We will see in the next chapters that there are scalar functions of
CHAPTER 1. INTRODUCTION

$I(\theta)$ that are theoretically justified to be used as querying objectives. But for simplicity, here we assume that the distribution parameter $\theta$ is a scalar and therefore we get a scalar Fisher information $I(\theta)$.

Then, using the two points mentioned above, we can write the FI-based probabilistic querying optimization as:

$$\arg \max_q I(\theta)$$

(1.16)

$$Q = \{x_1, ..., x_k\} \quad \text{where} \quad x_i \sim q(x), \quad i = 1, ..., k$$

Note that the joint distribution used above is also shown by $P_q(x, y|\theta)$ to emphasize its marginal distribution is the proposal $q$. Accordingly, the resulting Fisher information in (1.15) is denoted by $I_q(\theta)$. Similarly, one can define the test joint joint distribution $P_o = p(x)P(y|\theta, x)$ which yields the test Fisher information $I_o(\theta)$. In the test distribution, $p(x)$ is the test marginal, that is $x_{\text{test}} \sim p(x)$. We show in next chapters that a scalar function of test and training FI’s can be proved to be related to an inference-based performance metric.

Next we review different active learning work that has been done based on information theoretic objectives for two learning problems: clustering and classification.

1.6 Overview of Methods and Our Contribution

This thesis provides algorithms for doing active learning in clustering and classification models based on information theoretic objectives. In active clustering, we use entropy (model uncertainty) to select pairwise queries. And in classification, we use more sophisticated objectives, mutual information and Fisher information, to query class label of individual samples from the user. In this section, we review the existing active learning models for these problems and describe our contributions in more details.

1.6.1 Clustering and Image Segmentation

Data clustering is defined as the segmentation of a heterogeneous population into a number of more homogeneous subgroups [15]. Developing a generic unsupervised segmentation algorithm that can be accurately applied to all, or even many, types of images is not straightforward. This is mainly because automatic image segmentation is known to be an ill-posed problem [16] in the
CHAPTER 1. INTRODUCTION

sense that data can be segmented differently depending on hard-to-specify high level goals. Here, we only consider image segmentation as a common application of clustering, but our method can be applied to any other clustering problem. Though it can be more general, we concentrate on background-foreground separation of the images. The ill-posedness of image segmentation can be easily illustrated by imagining a street image, where a user might desire to segment cars or buildings as foreground depending on the underlying application: urban traffic or architecture, respectively.

Hence one promising approach is to leverage both automatic and manual segmentation in a user-interactive fashion that is tunable to the users target segmentation outcome with as few interactions as possible. In typical examples of these methods, a semiautomatic segmentation algorithm is constructed over a classical fully-automatic core method, where the users feedback is viewed as a set of constraints on the core.

Among interactive image segmentation algorithms, the user constraints are put either on background-foreground assignment of individual pixels or pairwise relationships between them. The former suggests requesting labels for a subset of individual pixels (individual constraints) [17, 18, 19, 20, 21, 22, 23, 24, 25] and is usually used with supervised cores, while the latter is based on specifying whether a pair of pixels should be in the same or different clusters (pairwise constraints) [26, 27, 28] and is more compatible with unsupervised methods1. There are also methods encoding both types of constraints in their formulation [29]. Here, since we approach the segmentation problem from an unsupervised point of view, we work with pairwise labeling.

Semi-automatic algorithms inherit some basic features from their core methods. One principal distinction among segmentation methods is whether they use features extracted from boundaries or regions. A boundary-based core method uses the constraints over the boundary locations [17, 30] to relocate them while a region-based core exploits constraints on the region assignments [31, 18, 32, 33] to re-characterize and re-group the pixels. Boundary-based interactive techniques are easy to run with the user in a loop [34]; however, the information provided by boundary constraints is local and not easy to propagate. Methods introduced by Zhang and Ji [35] and Lu and Carreira-Perpiñán [26] are two examples of region-based methods that interact with the user iteratively. In the former a probabilistic framework was built for semi-supervised segmentation, whereas the latter presented a constrained spectral clustering method. The major bottleneck of Lu and Carreira-Perpiñán’s work is the expensive computational cost, which makes its application to large images impractical. This difficulty is mainly due to the eigen-decomposition required by its spectral core method.

So far we have discussed different methods of incorporating constraints into an image
CHAPTER 1. INTRODUCTION

segmentation core. However, there are fewer algorithms that actively choose the queries to get the constraints from the oracle. Computing the uncertainty about the segment assignment of individual pixels in a given segmentation is common in active semi-automatic segmentation techniques: as two examples, Zhang and Ji [35] ranked image entities based on their impact on the uncertainty of a constructed Bayesian Network model; and Top et. al [36] selected the slice with the most uncertainty to be labeled for segmenting 3D images interactively. Almost all reported segmentation methods equipped with active learning, request user labels for individual pixels. Here, since we focus on pairwise constraints, we ask about the comparative assignments of pairs of samples.

1.6.2 Classification

A classifier is a mapping from the feature space to the class labels. It assigns each feature vector to a class. In supervised learning, this mapping is learned from a set of exemplary labeled samples, that is from a set of feature/label pairs. However, obtaining expert/human/manual labeling is expensive. Instead of randomly selecting samples for manual annotation (the standard setting), the goal of active learning is to intelligently select samples for annotation that enables efficiently learning an accurate classifier with as few labels as possible.

In classification, an active learning setting typically starts with an initial model (trained from a few labeled samples), then more samples are selected for label querying. Most of the classical studies in active learning use sequential querying [37, 38, 39, 40, 41, 42]. However, oftentimes querying a batch of samples is more efficient when the experts can label multiple samples in one step, since they can label more queries at each step without the need to wait for the retraining process.

Performing active learning in batch mode introduces new challenges. Since we need to select a set of queries, one should also make sure that the samples are non-redundant to maximize the amount of information that they provide. Another related challenge is that selecting a subset of samples based on a given objective function defined over sets can easily lead to intractable or to NP-hard optimization problems.

Recently, different batch mode active learning algorithms have been developed. Besides the querying strategy, active learning methods also differ based on the criterion they optimize for selecting queries. Some select samples that reduce model uncertainty [43, 44, 45]. Holub et. al [44] do this directly using the joint entropy function; Brinker [43] does this indirectly based on the distance of samples to the classifier’s boundary; and Chen and Krause [45] also do this indirectly based on the volume of version space. Most of these methods need to employ heuristics to introduce diversity.
CHAPTER 1. INTRODUCTION

among the queries. Choosing a subset of samples that maximizes the expected model change [46] is another type of batch selection strategy which works directly with classifier performance, but is usually tied to a particular classification model and is not general. On the other hand, Azimi et. al [47] develop a framework which constructs a batch mode variant of any given querying policy, such that it performs close to the sequential scenario. Their algorithm is based on the assumption that any sequential querying outperforms its batch mode correspondent. There are also studies that select the queries based on the amount of information they carry with respect to the underlying data distribution. For example, Hoi et. al [4] use the Fisher information ratio, which is specifically designed for a logistic regression model; and Guo [48] and Li and Guo [49] utilize mutual information (MI) between the input feature values of the queries and the remaining samples, where they used Gaussian Process distribution to model the joint probability over the instances.

MI between the labels has been employed in sequential active learning settings [50]; however, to date, we are not aware of any work that optimizes for this objective directly in batch mode. This is due to two main hurdles: (1) difficulty in calculating the MI between non-singleton subsets of labels and (2) its NP-hard optimization problem. NP-hardness of discrete objectives are usually tackled with by means of greedy approaches [4, 51, 52, 45]. However, whereas such greedy selection is efficient for a number of entropy-based objectives, it is not guaranteed to maximize MI.

In another point of view, the majority of classification active learning methods are prediction-based. Most of the inference-based querying algorithms in classification aim to choose queries that maximize the expected change in the objective of the inference step [53, 54] or Fisher information criterion [4, 3, 5, 6]. On the other hand, the wide range of studies in prediction-based active learning includes objectives of more variety; for instance the prediction error probability\(^4\) [9, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]; variance of the predictions [67, 68, 69]; uncertainty of the learner with respect to the unknown labels that may be evaluated by the entropy function [70], mutual information [71, 72, 73, 74], margin of the samples with respect to the trained hyperplanar discriminant function [75, 76] or Fisher information criterion [4, 3, 5].

Querying algorithms that are based on Fisher information (FI) use a scalar function of the Fisher information matrices computed for parametric models of training and test marginals. In the classification context, this scalar is sometimes called *Fisher information ratio (FIR)* [3] and its usage is motivated by older attempts in optimal experiment design for statistical regression methods [77, 78, 79, 1].

\[^4\text{Prediction error probability is indeed the frequentist risk function of 0/1 loss, and is also known as the generalization error.}\]
CHAPTER 1. INTRODUCTION

Among the existing FIR-based classification querying methods, only the very first one proposed by [2] approached the FIR objective from a parameter inference point of view. Using a maximum likelihood estimator (MLE), they claimed (with the proof skipped) that FIR is asymptotically equal to the expectation of the log-likelihood ratio with respect to both test and training samples (see sub-problem (i)). Later on, [4] and [5], inspired by [2], used FIR in connection with a logistic regression classifier with the motivation of decreasing the labels’ uncertainty and hence the prediction error. [3] employed this objective with the same motivation, but using a different approximation and optimization technique. More recently, [6] showed that even finite-sample FIR is closely related to the expected log-likelihood ratio of an MLE-based classifier. However, their results are derived under a different and rather restricting set of conditions and assumptions: they focused on the finite-sample case where the test marginal is a uniform PMF and the proposal marginal is a general PMF (to be determined) over a finite pool of unlabeled samples. Moreover, they assumed that the conditional Fisher information matrix is assumed to be independent of the class labels. Here, in a framework similar to [2] but with a more expanded and different derivation, we discuss a novel theoretical result based on which FIR is related to an MLE-based inference step for a large number of training data. More specifically, under certain regularity conditions required for consistency of MLE and in the absence of model mis-specification, and with no restricting assumptions on the form of test or training marginals, we show that FIR can be viewed as an upper bound for the expected variance of the asymptotic distribution of the log-likelihood ratio. Inspired by [6], we also show that under certain extra conditions, this relationship holds even in finite-sample case.

1.6.3 Our Contributions

1.6.3.1 Clustering with Edgewise Active Learning

We propose an entropy-based active learning for spectral clustering. As is described in Example 1.1, we select the queries in form of pairs of unlabeled samples. In Section 1.5.1, we discussed how to optimize entropy to query individual samples. In order to generalize that procedure to pairwise selection, we select the sample with highest uncertainty in a crowded region (high density) and query its relationship with another point of low uncertainty and reasonably high density. Querying the relationship means asking if the two samples should fall in the same cluster, or equivalently are must-linked. In order to avoid being biased to the cluster of the high-confidence point, we also query the edge between the uncertain point and a high-confidence point in the other cluster. Note that since we do not know the true underlying clustering, the clustering result based
on the current knowledge is used to select the confident points from each cluster. Hence, at each iteration, we select three points (one uncertain sample $x_{\pi_0}$ and two confident samples $x_{\pi_1}$ and $x_{\pi_2}$) and query the edges $q_{\pi_0\pi_1} = (x_{\pi_0}, x_{\pi_1})$ and $q_{\pi_0\pi_2} = (x_{\pi_0}, x_{\pi_2})$. We call this algorithm **Edgewise Active Learning (EAL)**.

We apply EAL to image segmentation problem, where we build our core method on a constrained spectral clustering core method using affinity propagation \[26\], where user feedback is used to learn new pairwise similarities. The central idea relies on the assumption of a Gaussian process prior over the vector containing the segmentation assignments. We use this algorithm iteratively, alternating between updating the segmentation and requesting pairwise constraints from the user, until the user is satisfied with the result (or in effect, the iterations converge). We scale up the original algorithm from the synthetic small data sets in \[26\] to handle large images by reducing the computational burden of each iteration using novelty selection \[80\] and an iterative numerical eigendecomposition \[81\].

We experimentally show that it converges reasonably quickly to the target segments even though the input image is represented only by naive pixel-wise features. Similar to other stochastic algorithms observing partial labels coming from the user at each iteration does not guarantee monotonically improving performance; however when it is likely to see performance fluctuations along the iterations, the algorithm is expected to climb up to a reasonable result at the convergence. We analyzed stability through the softness parameter of affinity propagation and found that a well-chosen dynamic schedule performed best in terms of convergence rate.

### 1.6.3.2 Classification Active Learning with MI and FI

Recall that in supervised application, active learning involves selecting individual samples to query their class labels, as opposed to querying the edges between pairs of points. In this work, we focus on two information theoretical criteria used in classification active learning algorithms: mutual information (MI) and Fisher information (FI). We develop a querying framework based on MI between the labels, as opposed to some previous work such as \[48\] and \[49\] which work with MI between the input feature values. Due to the involving combinatorial nature of MI between discrete class labels, it is harder to make this framework practical. The main hurdles, as described above, are expensive computations in its evaluation and maximization. We resolve these hurdles by means of: (1) utilizing pessimistic and optimistic approximations of MI; and (2) submodular optimization with guaranteed tight bounds. Both of these methods can be done efficiently in greedy fashion. Indeed,
CHAPTER 1. INTRODUCTION

their iterations are performed in tie with each other in a single loop.

Furthermore, regarding FI-based active learning, we give a novel theoretical analysis of the FI-based scalar querying objective, that is Fisher information ratio (FIR), and compare the existing FI-based querying algorithms according to this unified framework. We then proposed our own practical querying algorithm that optimizes FIR to choose queries from an unlabeled pool. Specifically, our theoretical derivations show that FIR is asymptotically an upper-bound of the expected variance of the log-likelihood loss function of any MLE-based discriminative classifier. This result is different than those obtained by [2] and [6]. We also discuss that finite-sample approximation of FIR can still be viewed as an upper-bound under certain circumstances. Finally, our practical FIR-based querying methods is formalized in form of a semi-definite programming which can be solved efficiently using available optimization packages.
Chapter 2

Accelerated Interactive Image Segmentation

Algorithms for fully automatic segmentation of images are often not sufficiently generic with suitable accuracy, and fully manual segmentation is not practical in many settings. There is a need for semi-automatic algorithms which are capable of interacting with the user and taking into account the collected feedback. Typically such methods have simply incorporated user feedback directly. Here, we employ active learning of optimal queries to guide user interaction. Our work is based on constrained spectral clustering that iteratively incorporates user feedback by propagating it through the calculated affinities [26]. The original framework does not scale well to large data sets, and hence is not straightforward to apply to interactive image segmentation. In order to address this issue, we adopt advanced numerical methods for eigen-decomposition implemented over a subsampling scheme. Our key innovation, however, is an active learning strategy that chooses pairwise queries to present to the user in order to increase the rate of learning from the feedback. Performance evaluation is carried out on the Berkeley segmentation and Graz-02 image data sets, confirming that convergence to high accuracy levels is realizable in relatively few iterations.

Let us first give a very brief background of spectral clustering and affinity propagation.

2.1 Spectral Clustering and Affinity Propagation

Let us represent each image as a finite set of feature vectors $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ where $x_i$ collects the features of the $i$'th pixel. In order to construct a graph over the image, we use a positive definite kernel $k(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$ for measuring similarity between pairs of data
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

points. The Gaussian kernel is a common function of this type:

\[ k(x_i, x_j) = \exp \left( -\frac{1}{2} (x_i - x_j)^T \Sigma^{-1} (x_i - x_j) \right), \tag{2.1} \]

where \( \Sigma \succeq 0 \) is the covariance matrix. Here the kernel is assumed to have uncorrelated variables yielding a diagonal matrix \( \Sigma \) with possibly different diagonal terms. We collect the calculated similarities leading to a positive definite affinity matrix \( K \) such that \( k_{ij} = k(x_i, x_j) \). As is explained in Example 1.1, here we consider \( K \) as the model parameter. This matrix also induces an undirected graph \( G = (X, K) \) over the set of pixels \( X \) and with edge weights defined by equation (2.1).

The binary graph-cut problem is how to bipartition \( X \) into two balanced groups \( C = \{C_1, C_2\} \), such that the sum of the similarities between nodes in different groups is minimized. One way of solving this NP-complete problem is by relaxing the discreteness and defining a continuous cluster assignment vector \( f \) that can be estimated by eigen-decomposition of the Laplacian matrix

\[ L := D^{-\frac{1}{2}}KD^{-\frac{1}{2}} \]

where \( D \) is the degree matrix whose \( i \)-th diagonal term is \( d_{ii} = \sum_j k_{ij} \). In order to discretize this solution, \( k \)-means clustering is usually used over the rows of the partially estimated eigenvector matrix [82].

2.1.1 Affinity Propagation

In the method proposed by Lu and Carreira-Perpiñán [26], pairwise constraints are propagated through pairwise affinities by assuming that \( f \) is a realization of a Gaussian process:

\[ p(f|X) = p(f|K) = \mathcal{N}(0, K) \tag{2.2} \]

where \( \mathcal{N} \) denotes a Gaussian distribution and the covariance, \( K \), is the similarity matrix of the data. Each constraint in this model is applied to a pair of samples, such as \( x_i \) and \( x_j \), denote by the edge \( q_{ij} = (x_i, x_j) \). This edge can get two possible labels from the oracle: must-link \( (y_{ij} = 1) \) implying that the pair should fall in the same cluster or cannot-link \( (y_{ij} = -1) \), ensuring that the points are in the same or different clusters respectively. Then the learning material \( L \) in our learning diagram is \( X \cup \{q_{ij}, y_{ij}\} \). In the affinity propagation technique [26], each constraint pair \( (q_{ij}, y_{ij}) \) is modeled probabilistically such that the corresponding continuous cluster assignments, \( f_i \) and \( f_j \), tend to have

---

1 In this paper, the terms affinity and similarity are used interchangeably.

2 More specifically, \( L \), as defined above, is the normalized symmetric Laplacian matrix. According to Luxburg [10] there are other forms of Laplacian as well.
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

the same or different signs if \( y_{ij} = 1 \) or \( y_{ij} = -1 \), respectively. In order to impose these conditions, the likelihood of a given constraint pair is model as the following:

\[
(q_{ij}, y_{ij}) \sim \mathcal{N}(f_i - y_{ij}f_j, \epsilon^2)
\]  

(2.3)

where the variance \( \epsilon^2 \) specify softness of our belief about the user’s constraints \(^3\). Using (2.2) as the prior and (2.3) as the likelihood function of the constraints, the posterior distribution over \( f \) given a set of labeled samples \( L = X \cup (Q, Y_Q) \), where \( Q = \bigcup_{(i,j)} q_{ij} \) and \( Y_Q = \bigcup_{(i,j)} y_{ij} \) with the pairs \((q_{ij}, y_{ij})\) independent from each other, would also be a Gaussian:

\[
p(f|L) = p(f|X, Q, Y_Q) \propto p(f|K) \cdot p(Q, Y_Q|f)
\]

(2.4)

\[
= p(f) \cdot \prod_{(q_{ij}, y_{ij}) \in Q} p(q_{ij}, y_{ij}|f)
\]

\[
= \exp \left( -\frac{f^T K^{-1} f}{2} \right) \prod_{y_{ij} \in Y_Q} \exp \left( -\left[ f_i - y_{ij} f_j \right]^2 \right)
\]

\[
= \exp \left( -\frac{1}{2} f^T \left[ K^{-1} + M(Q, Y_Q; \epsilon) \right] f \right),
\]

where \( M(Q, Y_Q; \epsilon) \) is a sparse matrix. Then the new affinity matrix would be the covariance of the obtained posterior: \( \overline{K} = [K^{-1} + M(Q, Y_Q; \epsilon)]^{-1} \) (for details see the original work \([26]\) and Appendix A). It can be verified easily that \( \overline{K} \) is also positive definite and therefore invertible.

After computing \( \overline{K} \), the new degree matrix \( \overline{D} \) and consequently the updated Laplacian \( \overline{L} \) can also be obtained and used to update the clustering result.

2.1.2 Computational Speed-up Techniques

Spectral clustering is not practical to apply to large datasets, and in particular to typical digital images. This is mainly due to the costly eigen-decomposition of the Laplacian matrix. As mentioned before, there are several ways of combating this problem. In this work, we joined two approaches: (1) subsampling the data and (2) efficient numerical eigen-decomposition of the Laplacian matrix. These two methods are described in the following two sub-sections:

\(^3\)In the original affinity propagation algorithm different softness parameters are considered for must- and cannot-link constraints, but here we assumed they are the same and equal to \( \epsilon^2 \)

22
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

2.1.2.1 Sub-sampling

We simply reduce the size of the similarity matrix by subsampling the data, while trying to preserve as much information as possible. Novelty selection, introduced by Paiva and Tasdizen [80], is a greedy sampling approach aiming to select samples with low redundancy.

Suppose $\tilde{X} = \{\tilde{x}_i\}_{i=1}^n$ is the feature set of all the pixels in the image. We start from an empty subsample set, add the first point $\tilde{x}_1$ and go through the remaining data points one by one. Each time, the minimum Euclidean distance between a candidate point and the already selected samples is computed and compared with a pre-specified threshold $\delta$. The point will be added to the sample set only if the minimum distance is larger than $\delta$, otherwise its nearest neighbor among the selected samples will be stored. Therefore the denser a region is, the more samples are generated around that area. Clearly for homogeneous regions the algorithm is equivalent to uniform sampling. For the rest of the paper, we denote $X$ as the set of selected samples using this technique and also assume that $|X| = n$. Note that this technique has the computational complexity of $O(n^2)$.

Our segmentation algorithm runs on $X$ to get a sparse set of labels $C$ followed by generalization to the full set of labels $\tilde{C}$ by means of Nearest Neighbors (NN).

2.1.2.2 Numerical Eigen-decomposition

Numerical approaches are widely used to scale large eigen-decomposition problems. Orthogonal iteration\textsuperscript{4} is a generalization of the well-known power method to estimating multiple eigenvectors associated with the largest eigenvalues iteratively [83]. Specifically, since we work with two clusters, the goal is to estimate an orthonormal basis for the dominant 2-dimensional invariant subspace of the Laplacian $L$.

The algorithm is initialized with two orthonormal vectors. The following iterations can be summarized in two steps: the result of the last iteration is left-multiplied by $L^q$ where $q \geq 1$ is an integer parameter chosen according to the structure of $L$. Then we orthonormalize the resulting vectors using a Gram-Schmidt process. Iterations of this procedure have been shown to converge to the true dominant eigenvectors if the second and third eigenvalues are not equal.

The learning iterations in Algorithm 1.1 can be rewritten specifically for constrained spectral clustering with affinity propagation and speed-up techniques as in Algorithm 2.1. Note that the optional query set subtraction step (line 8 in Algorithm 1.1) is ignored. Also we only consider constrained samples of the current learning iteration (memoryless constraining) since in practice

\textsuperscript{4}also known as subspace or staircase iteration.
**Algorithm 2.1:** Learning iterations of constrained spectral clustering with affinity propagation

**Inputs:** Pixel-wise features $\mathcal{X} = \widetilde{X}$, Kernel support-width $\sigma^2$, novelty selection threshold $\delta$

**Outputs:** Constrained full segmentation result $\widetilde{C}_t$

```plaintext
/* Novelty selection */
1 $X \leftarrow \text{NS}(\widetilde{X}, \delta)$ (see Section 2.1.2.1)
/* Initializing the model parameter */
2 $K_0 \leftarrow \text{affinities of } X \text{ computed by (2.1)}$
3 $C_0 \leftarrow \text{SC}(K_0)$
/* Generalizing to the whole image */
4 $\widetilde{C}_0 \leftarrow \text{NN}(C_0)$
5 $t \leftarrow 0$
/* Starting the querying iterations */
6 for $t = 0, 1, 2, \ldots$ do
   /* Querying */
7 $L(t) = (Q, Y_Q) \leftarrow \mathcal{A}(X, C_t)$
/* Updating the affinity matrix and spectral clustering */
8 $K_{t+1} \leftarrow \left[ K_t^{-1} + M(Q, Y_Q) \right]^{-1}$
9 $C_{t+1} \leftarrow \text{SC}(K_{t+1})$
/* Generalization to the whole image */
10 $\widetilde{C}_{t+1} \leftarrow \text{NN}(C_{t+1})$
11 $t \leftarrow t + 1$
```

enlarging $Q$ with all the previous constrained samples made the results worse. Furthermore, the module $\mathcal{A}$ is a given procedure for preparing learning materials including the pairwise constraints. In the remaining sections of this chapter, we propose an entropy-based querying method in the following sections. The goal, as is explained in Section 1.5.1, is to reduce the uncertainty of the model as much as possible by labeling pairwise queries in each learning iteration.

### 2.2 Clusters Probabilistic Modeling

In order to compute the uncertainty with respect to a given sample $x \in X$, we need to estimate the posterior probability that the sample belongs to a certain cluster given its features. Note that the posterior probability is computed based on a given clustering result $C = \{C_1, C_2\}$:

$$
P(x \in C_i | x) = \frac{p(x | x \in C_i) P(x \in C_i)}{p(x)}, \ i = 1, 2. \quad (2.5)
$$
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

The likelihood function \( p(x|x \in C_i) \) for each cluster \( i = 1, 2 \) can be estimated using kernel density estimation (KDE):

\[
\ell_{C_1}(x) \triangleq p(x|x \in C_1) = \frac{1}{|C_1|} \sum_{\xi \in C_1} k(x, \xi),
\]

\[
\ell_{C_2}(x) \triangleq p(x|x \in C_2) = \frac{1}{|C_2|} \sum_{\xi \in C_2} k(x, \xi).
\]

(2.6a)

(2.6b)

where \( k \) is the same kernel function as in (2.1). The class priors can also be computed empirically (\( P(x \in C_i) = P(C_i) = |C_i|/n, i = 1, 2 \)), which results in the following class posterior probabilities:

\[
P(x \in C_i|x) = \frac{\ell_{C_i}(x)P(C_i)}{p(x)} = \frac{\sum_{\xi \in C_i} k(x, \xi)}{np(x)},
\]

(2.7)

Note that in equations (2.5) and (2.7), \( p(x) = \sum_{i=1}^2 \ell_{C_i}(x)P(C_i) \), the evidence, is just a normalization constant. Figure 2.1 gives a simple 2-dimensional example used to illustrate the probabilistic concepts introduced here and also later in section 2.3. In the first row, Figure 2.1(a) shows synthetic data generated by a mixture of three Gaussian components. The samples are colored according to the true clusters and displayed on top of the marginal density contours. Figures 2.1(b) and 2.1(c) show the two clusters resulting from an unconstrained spectral clustering, each of which is shown over its own posterior distribution contours computed by (2.7).

Finally the posterior entropy can be obtained using Shannon’s definition:

\[
h(x) = -\sum_{i=1}^2 P(x \in C_i|x) \log P(x \in C_i|x).
\]

(2.8)

2.3 Edgewise Active Learning (EAL)

Now, we are ready to explain our entropy-based strategy for choosing pairwise queries. As described in section 1.6.3, in each querying iteration of EAL, given a current clustering result \( C = \{C_1, C_2\} \), we choose an uncertain sample \( x_{\pi_0} \) to be linked with two confident samples \( x_{\pi_3} \) and \( x_{\pi_2} \) from each of the two clusters. In order to have sufficient amount of propagation of the constraints among the other samples, the selected samples have to be selected from regions with high marginal density \( p(x) \).
Figure 2.1: Illustrating probabilistic concepts of our approach: (a) three Gaussian components with different means and equal covariances used to generate the points that are displayed over contours of their marginal distribution. The points in each desired (gold-standard) cluster are shown in a different color and shapes; (b,c) the resulting groups after doing an unconstrained spectral clustering lying on contours of their corresponding posterior distribution; (d) the resulting groups shown together over contours of the posterior entropy $h$, with the encircled point as the one with the largest $h$; (e) the resulting groups over contours of the density weighted entropy $\phi$, with the encircled point as the one with maximum $\phi$ (i.e. $x_{\pi_0}$); (f) the resulting groups over contours of the inverse-density weighted entropy $\psi$, with the encircled points having the minimum $\psi$ and shortest distance to $x_{\pi_0}$ in each cluster (i.e. $x_{\pi_1}$ and $x_{\pi_2}$).

Incorporating the marginal density is mainly due to the problem that entropy-based querying is not robust to low-evidence points (outliers), which does not lead to maximal affinity propagation. This fact is illustrated in figure 2.1(d) where the result of unconstrained spectral clustering is shown together with the contours of its posterior entropy. As can be seen, the entropy is independent of the evidence and is large everywhere close to the boundary between the clusters. Indeed, in this specific example, maximum entropy occurred in a low-evidence region.
2.3.1 Choosing the Uncertain Point

One solution to avoid selecting the outliers, when choosing the uncertain point $q_0$, is to weight the entropies by marginal distributions to get a *density-weighted* scoring metric [14]:

$$
\phi(x) = h(x)p(x)
$$

where $h(x)$ is defined in equation (2.8). The uncertain point $q_0$ is selected by maximizing $\phi(x)$, which selects points with both high uncertainty and high evidence, and thus ignores outliers. Figure 2.1(e) shows contours of $\phi$ for our example, where the sample with the largest value is located in a crowded region close to the boundary.

2.3.2 Choosing the Confident Points

Similarly, using entropy by itself to find the high confidence end-points would suffer from outlier vulnerability (observe in Figure 2.1(d) that minimum entropy happens in low-density tails of each cluster). Again a weighting technique is used to make the metric more robust, but since here we encounter a minimization problem, we define an *inverse-density-weighted* entropy:

$$
\psi(x) = \frac{h(x)}{p(x)}
$$

where $h(x)$ is defined in equation (2.8). The confident points $q_1$ and $q_2$ selected with respect to $q_0$ displayed in Figure 2.1(e).

Finally, the edges $q_{\pi_0 \pi_1} = \{x_{\pi_0}, x_{\pi_1}\}$ and $q_{\pi_0 \pi_2} = \{x_{\pi_0}, x_{\pi_2}\}$ can be viewed as edges of an incomplete triangle. Label of the third edge $q_{\pi_1 \pi_2} = \{x_{\pi_1}, x_{\pi_2}\}$ can be easily inferred in a binary clustering (lines 8 through 13). By adding this additional label to the collection of constraints, the current iteration of query generation finishes. The generated pairwise queries $Q = \{q_{\pi_0 \pi_1}, q_{\pi_0 \pi_2}, q_{\pi_1 \pi_2}\}$ are then labeled and used to update the similarity matrix by affinity.
Algorithm 2.2: Edgewise Active Learning (EAL)

Inputs: Sampled features \( X \), softness parameters \( \epsilon \)
Outputs: The learning materials \( L \)

/* Choosing the most uncertain point in a crowded region */
1 \( x_{\pi_0} \leftarrow \arg\min_{x \in X} \phi(x) \)
/* Selecting the confident points */
2 for \( j = 1 \rightarrow 2 \) do
3 \( \gamma_j \leftarrow \) a threshold as a function of \( \Psi_j = \{\psi(x)|x \in C_j\} \)
4 \( X_{\text{thr}} \leftarrow \{x \in C_j - \{x_{\pi_0}\}|\psi(x) \leq \gamma_j\} \)
5 \( x_{\pi_j} \leftarrow \arg\min_{x \in X_{\text{thr}}} \|x - x_{\pi_0}\|^2 \)
6 \( q_{\pi_0\pi_j} \leftarrow (x_{\pi_0}, x_{\pi_j}) \)
/* Requesting labels from the oracle */
7 \( Y_Q \leftarrow \{y_{\pi_0\pi_1}, y_{\pi_0\pi_2}\} \) from the oracle
/* Auto-labeling the third edge using the received labels */
8 \( q_{\pi_1\pi_2} \leftarrow (x_{\pi_1}, x_{\pi_2}) \)
9 if \( y_{\pi_0\pi_1} = y_{\pi_0\pi_2} \) then
10 \( y_{\pi_1\pi_2} \leftarrow 1 \)
ext else
11 \( y_{\pi_1\pi_2} \leftarrow -1 \)
12 \( Q \leftarrow \{q_{\pi_0\pi_1}, q_{\pi_0\pi_2}, q_{\pi_1\pi_2}\} \)
13 \( Y_Q \leftarrow \{y_{\pi_0\pi_1}, y_{\pi_0\pi_2}, y_{\pi_1\pi_2}\} \)
14 \( L \leftarrow X \cup (Q, Y_Q) \)

propagation. Therefore, size of the query set is always equal to three in all the learning iterations \( (k = |Q| = 3) \). Pseudo-code of our proposed pairwise querying is shown in Algorithm 2.2.

2.4 Implementation Details

Some practical points regarding the affinity update process are discussed here. To analyze this process theoretically, we consider only one constrained pair \( q_{ij} \) and its label \( y_{ij} \) for simplicity. We rewrite the update equation in a linear form (using the matrix inversion lemma and after some simplifications) so that \( \overline{K} = KT(y_{ij}) \). The transformation matrix \( T(\omega_{ij}) \) can be described column-wise as:

\[
T(y_{ij})e_t = e_t - \omega_t(y_{ij}), \quad t = 1, \ldots, n \tag{2.11}
\]
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

where

\[ \omega_t(y_{ij}) = \begin{cases} 
\frac{k_{it} - k_{jt}}{k_{ii} + k_{jj} - 2k_{ij} + \epsilon^2} (e_i - e_j), & y_{ij} = 1 \\
\frac{k_{it} + k_{jt}}{k_{ii} + k_{jj} + 2k_{ij} + \epsilon^2} (e_i + e_j), & y_{ij} = -1 
\end{cases} \]

Equation (2.11) implies that each column of the updated matrix is a modification of the corresponding column in the original affinity matrix. This modification is minimal for regions far away from \(x_i\) and \(x_j\). Also observe that as \(\epsilon\) grows, \(\omega_t\) and hence the impact of the constraints, decreases.

The update shown in (2.11) suffers from the fact that there is no upper bound on the value of the modifications. As a result they may violate two key properties of the Gaussian assumption on the affinities in \(K\): non-negativity of the elements, and the requirement that self-similarities should be equal to unity and greater than or equal to cross-similarities. In Lu and Carreira-Perpiñán’s work \[26\], a heuristic was introduced for shrinking all negative updated elements to zero. Here, we also imposed a heuristic for projecting values larger than one to unity. Specifically, we limit each element of the updated matrix \(\tilde{k}_{ij}\) between zero and one as follows:

\[ g(\tilde{k}_{ij}) = \begin{cases} 
\tilde{k}_{ij}, & \text{if } 0 \leq \tilde{k}_{ij} \leq 1 \\
1, & \text{if } 1 < \tilde{k}_{ij} \text{ or } i = j \\
0, & \text{if } \tilde{k}_{ij} < 0 
\end{cases} \] (2.12)

Another observation is that if both end-points of a constraint \(\omega_{ij}\) are outliers, then the updates would be too small (\(k_{it}, k_{jt} \ll 1, \forall t \in \{1, ..., n\} - \{i, j\}\)). Even if only one end-point is an outlier, the amount of propagation is limited. This is the reason why in our active learning strategy we avoided choosing outliers.

2.5 Experimental Results

In order to test the performance of our algorithm, we first applied it to the Berkeley segmentation dataset \[84\]. We selected 55 images from both training and testing categories based on the existence of a meaningful object in each selected image that was not trivial to cluster using our core method. The performance metric that we use here to evaluate the clustering quality with respect
to a given gold-standard segmentation is the Rand Index, defined as:

$$\text{RI} = \frac{(TP + TN)}{\binom{n}{2}},$$

with TP and TN true positive and true negative pixels with respect to the targeted object. Note that although the tasks of constraining and clustering are performed over the sub-sampled pixels, the accuracy is measured on the entire image.

To avoid subjectivity from a particular user and for ease of testing, we ran the evaluation reported here in a fully automated mode. In particular we substituted a perfect oracle for the user, represented by the program itself, which answered queries using the ground truth segmentation. The while-loop in the main algorithm (Algorithm 2.2) was thus replaced by a for-loop whose length was fixed to a sufficiently large number (here 500). After processing all 55 images we computed the average and standard deviation of RI across the images as a function of iteration index. With the computational acceleration described earlier, the algorithm ran quickly in MATLAB on a standard desktop with 12 GB of memory with no further optimization beyond the eigen-decomposition method described above. Computed time per iteration varied from 0.05 to 4 seconds with an average of 0.8 seconds and a standard deviation of 0.65 seconds.

To minimize the effect of particular choice of features, we used a very simple pixel-wise features, namely the spatial coordinates and the luminance intensities ($d = 3$), all normalized to unit variance.

Throughout this section we used the following parameter values chosen empirically: the threshold $\delta$ in the novelty selection algorithm was set to 0.2; the diagonal terms in $\Sigma$ were set to 0.25 for both spatial features (normalized $x$- and $y$-coordinates) and to 0.5 for intensity; kernel values smaller than $5 \times 10^{-2}$ were shrunk to zero; and $\gamma_j = \min \Psi_j, j = 1, 2$ in EAL. See Appendix B for details regarding the orthogonal iteration and the default values used there.

A nice property of our constrained spectral clustering algorithm is that it allows incorporation of the confidence of the labeler regarding the constraints they provide through the softness parameter $\varepsilon$ (see equations 2.3). In real-world scenarios, we expect that during later stages of active learning, the labeler will be queried with more and more difficult edge queries and hence would have less confidence in their provided constraints. Mimicking such scenario, we introduce a dynamic parameter $\varepsilon(t)$ where the labeler’s confidence diminishes with iteration index $t$. In this section, we present experimental results for two settings: (1) constant softness parameter, where we assume that
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

![Image](image.png)

Figure 2.2: Results of running the algorithm with constant ε: (a) average and std RI versus different number of constraints; (b) the accuracy image: RI of segmenting individual images versus the iteration index

the labeler/oracle confidence in their constraint information is constant, and (2) dynamic softness parameter, where we assume that the labeler’s confidence diminish with iteration.

### 2.5.1 Constant Softness Parameter

The softness parameter $\epsilon$ is fixed to a small constant, $10^{-5}$. The mean and standard deviation of clustering accuracy measured for all iterations are shown in Figure 2.2(a). We recall that each iteration of the algorithm queries two edges, so that 500 iterations implies that $2 \times 500 = 1000$ edges have been labeled in total. The average RI starts from about 55% for the unconstrained segmentation and at its peak (around $2 \times 200$ labeled queries) achieves about 90% accuracy.

It can be observed that after getting about $2 \times 250$ constraints the average accuracy gradually starts decreasing. At the same time, the standard deviation also increased. This means that continued imposition of even correct constraints may cause accuracy reduction. We are observing a decrease in performance as labels are increased because at this point the learning algorithm is over-fitting the constraints. An accuracy image is shown in Figure 2.2(b), which displays a matrix whose rows and columns correspond to RI of individual images and iteration index respectively. A dark region on the left part of this figure means that the unconstrained segmentation gives a poor result. Ideally the figure should uniformly get brighter as we move towards the right. In contrast, although for most of the images (rows) the accuracy is much higher than that at the start, as we scan to the right we see many black spots and dark regions confirming that the aforementioned decrease in the average accuracy is mainly due to sharp decreases in the accuracy of individual results after getting close
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

Figure 2.3: Illustration of a potential reason for fluctuating results. Blue and red pixels form two clusters and the thick dark blue boundary separates the correct segments. (a) The segmentation result at iteration $t$ which is close to the ground-truth together with the constraints $(Q_{t+1}, Y_{Q_{t+1}})$ including a must- (Solid edge) and cannot-link (dashed line). Observe that the central point $x_{\pi_0}$ is an object pixel with features very similar to background. (b) The results after applying $(Q_{t+1}, Y_{Q_{t+1}})$. Notice that many background pixels have been pushed into the object cluster and the accuracy is reduced $\text{RI}(t) > \text{RI}(t + 1)$.

to a perfect segmentation. Figure 2.3 illustrates how over-fitting can occur. The transition between the objects in natural images usually takes place smoothly. Consequently some object pixels near the boundary can have intensities similar to nearby background pixels as shown in Figure 2.3. At the same time, such pixels are the likeliest ones to be selected by EAL as the most uncertain point, especially when the current segmentation result is nearly perfect. Putting constraints in these cases can be misleading because it can easily push many background pixels into the segmented object. Note that in real settings, because active learning segmentation is interactive, the user will stop the iteration once almost perfect segmentation is met, which would thus avoid the over-fitting issues that usually occur in later iterations.

2.5.2 Dynamic Softness Parameter

In real interactive settings, the user can also incorporate their uncertainty in their labels through the softness parameter $\epsilon$. Because, we are simulating the user labeling automatically, we make the softness parameter $\epsilon$ dynamic so that it increases (reflecting diminishing confidence) according to some schedule. This implies the algorithm will put less weight on the constraints as the
Figure 2.4: (a) Average and (b) standard deviation RI of trying different values for $m$ in equation (2.13) ($m = 10^{-4}, 10^{-3}, 10^{-2}$ and $10^{-1}$); average RI obtained by running the dynamic algorithm with $m = 10^{-2}$ over (c) Cars and (d) People categories of INRIA image data set.

iterations continue. Here we use a linear schedule for the softness parameter of the algorithm ($t = 0$ corresponds to the unconstrained initialization):

$$
\epsilon(t) = mt + \epsilon_0, \quad m > 0
$$

(2.13)

where $\epsilon_0$ is the minimum softness. We tried such linear schedules on the Berkeley dataset with $\epsilon_0 = 10^{-5}$ and different values for the slope $m$. According to Figure 2.4(a), which shows the average
accuracies, using $m = 10^{-4}$ yielded practically the same result as $m = 0$. Increasing $m$ to $10^{-3}$
reduced the decrease in the average accuracy to some degree while setting it equal to $10^{-2}$ eliminated
the decrease along with the fluctuations. Further increase in $m$ caused the algorithm to become
saturated prematurely before reaching an acceptably high accuracy. Figure 2.4(b) indicates that
the value $m = 10^{-2}$ outperformed others in terms of RI standard deviation too. An accuracy map
of individual images in the dynamic case is shown for $m = 10^{-2}$ in Figure 2.5, confirming the
stabilizing effect of using a dynamic parameter. We observed that incorporating a dynamic schedule
makes our constrained spectral clustering with active learning robust to over-fitting.

In order to investigate the generalization of the described dynamic algorithm, we ran it with
the same set of parameter values mentioned above and with the best schedule understood from the
results coming from the Berkeley data set (that is $m = 10^{-2}$) on the Graz-02 image dataset [85, 86],
categories of Cars and People. We chose 80 images from each category and applied our dynamic
algorithm on them for 500 iterations. The resulting average and standard deviation RI are shown in
Figures 2.4(c) and 2.4(d) implying that the results are very similar to what we got from the Berkeley
data set.

Figure 2.6 shows the results on four individual images. The first and fourth rows show
the gray-scale original images and the RI accuracy plots. The dashed lines in the plots show the
iterations from which the results have been selected to be displayed in the other rows: the second
and fifth rows containing the static results, and the third and sixth rows showing the dynamic results
(where a linear schedule with $m = 10^{-2}$ was used). The selected iterations for all, except the child
image are the same. They were chosen to show the initial, pre-convergence and post-convergence
cases. In the first image, they were chosen such that the second phase coincides with a valley and the
third happened before a permanent break in the static case.

Since the feature set used here is not descriptive enough, the initial unconstrained segmen-
tations are far from the desired result, but, as expected, constraining them increases the accuracy.
We observe that there are sharp valleys in the accuracy plots, especially in static cases. The valleys
usually occur after convergence. In real settings, users would have terminated the iterations before
they occur because they would have been already satisfied with the segmentation (see the plots in
2.6(d), 2.6(j), 2.6(l)). But some valleys occur in earlier iterations (see figure 2.6(b)). The algorithm
was able to recover to high accuracy from these values most of the time.

In Figure 2.6 we can see that the boundary of clusters are not smooth and sometimes small
isolated components appeared. These are the effects of generalization of the cluster assignments
from a sparse set of labels to the full set that contains all the pixels. By decreasing the sampling
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

Figure 2.5: RI measurements for segmentation of individual images while using dynamic softness with a linear schedule and \( m = 10^{-2} \).

threshold \( \delta \) we may reduce this sampling effect at the expense of higher computational complexity.

2.5.3 Active Learning Evaluation

We evaluated our active learning strategy against four different scenarios explained below. The first three applies active learning strategies based on three simulated users (cases ((i)), ((ii)), and ((iii))) and the segmentation is learned via our constrained spectral clustering approach. The fourth one is an alternative active learning strategy and constrained clustering proposed by Wang and Davidson [7] (case ((iv))).

(i) Random query selection: the query points \( q_0, q_1 \) and \( q_2 \) are chosen randomly from the samples.

(ii) Object-oriented query selection: we simulate a smarter user who looks at the structure of the target object. Specifically, the user starts by putting cannot-links between interior and exterior points that are located close to the object’s border, and gradually moves to points further from the borders. Interior and exterior points themselves are must-linked separately.

(iii) Boundary-oriented query selection: We simulate another user behaving similar to our active learning strategy. The central point \( q_0 \) is chosen randomly as a pixel on the current segmentation boundary and the other end-points, \( q_1 \) and \( q_2 \), are selected randomly from within a ball of a specific radius around \( q_0 \).

(iv) Wang and Davidson’s method [7]: Their active learning strategy selects the best edge to query that minimizes the expected error. To perform constrained spectral clustering, they add must-links and cannot-links as additional constraints to the spectral clustering optimization
Figure 2.6: Segmentation results on four individual images: rows 1 and 4 indicate the original gray-value images and their RI plot. The blue lines show the selected iterations whose results are shown. Rows 2 and 5 show the static results, and the dynamic results are rows 3 and 6.
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

formulation. All the parameters in this algorithm here are set to the recommended values given in the original work [7].

Cases ((i)), ((ii)), and ((iii)) are compared in Figure 2.7 against EAL in both static and dynamic modes. It shows that random querying increased the accuracy more slowly compared to our method, EAL, and also did not achieve the same accuracy. The object-oriented strategy improved the accuracy as rapidly as EAL during the first iterations, but totally failed before convergence. Note that for the images with small target objects, the number of possible constraints with this strategy was less than $2 \times 500$. Also querying based on ((iii)) produced very slow improvement in terms of the average RI.

The active learning strategy in ((iv)) uses a different base constrained clustering algorithm. The numerical acceleration technique described in 2.1.2.2 cannot be directly applied to their method and therefore it is not computationally cheap enough to be applied to the same data used in the last section (obtained using $\delta = 0.2$). Therefore we increased the sampling threshold to $\delta = 0.5$ in order to reduce the number of samples and used it for comparison. The results are illustrated in Figures 2.8(a) and 2.8(b). Note that in order to make Wang and Davidson’s algorithm comparable to ours we queried two pair-wise edges at each of its iterations. While our algorithm behaved similar to the case when $\delta$ was 0.2, the alternative active spectral clustering method increased the accuracy in early iterations but then failed to consider efficiently further constraints. This method was designed for general data and not ready for scaling to large data, such as images, unlike our method, EAL. Furthermore, [7] optimizes for expected error. Note that in images, typically there are more background pixels than foreground. We noticed empirically that the results of their algorithm tend to converge to unbalanced segmentations (see the individual segmentation example in Figure 2.8(c) for the *old woman* image shown in Figure 2.6(i)). Correctly classifying background tends to reduce the error. We investigated and found that their active learning strategy tends to select edge queries in the background region, rather than querying foreground or boundary information, leading to poorer segmentation performance. Our proposed active learning strategy, on the other hand, optimizes for uncertainty. This tends to pick points on image boundaries, which is important for image segmentation.

Since the cases ((i)), ((ii)) and ((iii)) all use our constrained clustering approach, their average running time for individual iterations are similar, removing the need for any comparison. Figure 2.8(d) shows that the average running time for case ((iv)) is almost twice as large as our algorithm in all iterations. This illustrates the speed improvement effect of our fast numerical
CHAPTER 2. ACCELERATED INTERACTIVE IMAGE SEGMENTATION

Figure 2.7: Comparing EAL ran with static affinity propagation ($\epsilon = 10^{-5}$) with three different query selection scenarios described as cases ((i)), ((ii)), ((iii)) in section 2.5.3, all of which ran with the sampling factor $\delta = 0.2$.

Figure 2.8: (a,b) the average and std RI of EAL and Wang and Davidson’s algorithm on a smaller dataset obtained by using $\delta = 0.5$; (c) the segmentation result of Wang and Davidson’s algorithm [7] on the old woman image shown in Figure 2.6(i) picked from an iteration after convergence. Observe that it has converged to an unbalanced segmentation with low accuracy; (d) Average running time of Wang and Davidson’s constrained spectral clustering [7] and our algorithm ran with constant $\epsilon = 10^{-5}$ eigen-decomposition.
Chapter 3

MI-based Classification Active Learning

MI-based approaches in classification active learning rely on maximizing the mutual information between class labels of the query random variables and that of the remaining unlabeled samples. There are two hurdles in implementation of such approaches: (1) difficulty in calculating the MI between non-singleton subsets of labels and (2) its NP-hard optimization problem. In this chapter, we introduce a general framework for MI-based deterministic batch active learning formulated in (1.6). That is, we consider mutual information between the class labels as an objective defined directly over the query set and therefore can be optimized to select the queries. Our framework makes MI-based querying tractable by a combination of greedy approximation of MI (pessimistically and optimistically), and employing submodular maximization. This framework can be applied with any discriminative classifier.

3.1 Classification Active Learning

Let $X$ denote our finite feature set (with $|X| = n < \infty$) where its members $x \in X$ are $d$-dimensional feature vectors representing the data samples. Also, let $Y_X$ be their respective class labels where each $y \in Y_X$ represents the numerical category varying between 1 to $c$, with $1 < c < \infty$ the number of classes. The class labels here are considered as random variables unless they are observed after the labeling process. We focus on supervised algorithms here, hence the learning materials include only the labeled samples $\mathcal{L} = (Q, Y_Q)$. In this chapter and the next one, we assume that a discriminative classifier is being used for modeling the posterior distribution of class labels given the data samples with a parametric distribution $\mathbb{P}(y|\theta, x)$, parametrised by $\theta \in \Omega$. In these models, the labels are assumed to be independent from each other given the model parameter. For an
Algorithm 3.1: Classification Active Learning

Inputs: The input unlabeled pool $X$, the initial training set $L^{(0)}$, the batch size $k$

Outputs: The parameter estimate $\hat{\theta}^{(t)}$, the predictions

1. $\hat{\theta}^{(0)} \leftarrow$ MLE using $L^{(0)}$ as in (3.1)
2. for $t = 0, 1, 2, \ldots$ do
   /* Preparing Learning Materials */
   3. $Q \leftarrow A \left( X, \hat{\theta}^{(t)}, k \right)$
   4. $Y_Q \leftarrow$ labels of $Q$
   5. $L^{(t+1)} \leftarrow (Q, Y_Q)$
   /* Update the parameter estimate */
   6. $\hat{\theta}^{(t+1)} \leftarrow$ MLE using $\bigcup_{i=1}^{t+1} L^{(i)}$ as in (3.1)
   /* Predictions */
   7. $\hat{y}_{\text{test}} \leftarrow \arg\max_y \mathbb{P} \left( y \mid \hat{\theta}^{(t+1)}, x_{\text{test}} \right)$ for all test samples
   /* remove the already queried samples from the pool to avoid relabeling them */
   8. $X \leftarrow X - Q$

initial labeled set $L^{(0)} = (Q_0, Y_{Q_0})$ (where $Q_0 \subseteq X$) maximum likelihood estimation (MLE) of the model parameter is used to do the inference:

$$\hat{\theta}^{(1)} = \arg\max_{\theta \in \Omega} \sum_{(x, y) \in L^{(0)}} \log \mathbb{P}(y \mid \theta, x). \tag{3.1}$$

In the $t$'th iteration of a learning, $\theta^{(t)}$ can be similarly estimated based on $L^{(t)}$. The estimated parameter could then be used to do the querying for iteration $t + 1$. The pseudocode of the learning iterations for a classification active learning is shown in 3.1. The initial training data, $L^{(0)}$ is given to the algorithm, say by randomly querying an initial subset of samples. Note that size of the query set $k$ in each learning iteration (batch size) is given as an input to the learning algorithm and more specifically, to the querying modeul $A$, as opposed to the clustering active learning where $k = |Q|$ was fixed to three. Furthermore, in each learning iteration we use all the labeled samples to update estimate of the model parameter, and we remove the already queried samples from the pool not to query them again (avoiding relabeling).
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

3.2 Mutual Information vs. Entropy

Here, first we explain the main shortcoming of the entropy querying objective (see (1.8)) and discuss how MI can resolve that issue. In the remaining sections of this chapter, we focus on the $t$’th iteration with labeled data set $\mathcal{L}_{\text{total}} = \bigcup_{i=1}^{t} \mathcal{L}^{(i)}$ and parameter estimate $\hat{\theta} = \hat{\theta}^{(t)}$ being available from the previous iterations. Recall that the entropy objective, denoted by $f_{H}$, measures the uncertainty about the class labels of a candidate query set $Q$:

$$f_{H}(Q) = H(Y_{Q}|X, \mathcal{L}_{\text{total}}) = H(Y_{Q}|X, \hat{\theta}),$$

where the second equality is due to using a point estimation (MLE), which implies

$$\mathbb{P}(Y_{Q}|X, \mathcal{L}_{\text{total}}) = \int \mathbb{P}(Y_{Q}\mid \theta, X) \mathbb{P}(\theta \mid X, \mathcal{L}_{\text{total}}) \, d\theta = \int \mathbb{P}(Y_{Q}\mid \theta, X) \cdot \delta(\theta - \hat{\theta}) \, d\theta = \mathbb{P}(Y_{Q}\mid \hat{\theta}, X).$$

Now, using independence of the class labels given the parameter, $f_{H}$ can be written as a summation of individual entropies:

$$f_{H}(Q) = \sum_{y \in Y_{Q}} H(y\mid \hat{\theta}, X)$$

and the entropy-based querying optimization becomes

$$\arg \max_{Q \subseteq X, |Q|=k} \sum_{y \in Y_{Q}} H(y\mid \hat{\theta}, X)$$

(3.5)

Hence, choosing a query set with size $k$ and highest joint entropy (3.2) becomes equivalent to choosing $k$ queries with highest individual entropies in (3.5). In such query selection, interaction between the chosen samples are not taken into account and therefore if $k > 1$ we may choose queries with similar characteristics that give redundant information. That is to say, while information gain of each individual queries is high, there might be large overlapping information between them such that we can remove one of them without significant information loss (hence smaller query set with almost the same amount of information).

Geometrically, redundancy of the entropy-based querying can be explained by high entropy samples are those that are located close to the current class boundaries. Now suppose two of such
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

high-entropy queries are very close to each other in the feature space. Hence knowing the class label of one of them reduces the uncertainty regarding the other and vice versa, and it is not efficient to request labels for both of these queries.

Mutual information does not break into individual terms because of the conditional entropy involved in its formulation:

\[
\begin{align*}
 f_{MI}(Q) &= MI(Y_Q, Y_{X-Q}|X, \mathcal{L}^{(t)}) = H(Y_Q|X, \mathcal{L}^{(t)}) - H(Y_Q|X, \mathcal{L}^{(t)}, Y_{X-Q}) \\
 &= H(Y_{X-Q}|X, \mathcal{L}^{(t)}),
\end{align*}
\]

(3.6)

where the first term is the same as \( f_H(Q) \). Having the conditionality on \( Y_{X-Q} \) in the second term prevents the elements in \( Y_Q \) from becoming independent and considers them jointly, which in turn incorporates the interaction among the queries. This is done in the expense of increasing the computational complexity. Maximizing \( f_{MI}(Q) \) for \( k > 1 \) is NP-hard (the optimization hurdle). Relaxing combinatorial optimizations into continuous spaces is a common technique to make the computations tractable [48], however these methods still involve a final discretization step that often includes using heuristics. In the following two sections, we introduce our strategies to overcome the practical hurdles in MI-based active learning algorithms by introducing (1) pessimistic/optimistic approximations of MI; and (2) submodular maximization algorithms that allow us to perform the computations within the discrete domain.

3.3 Evaluating Mutual Information

In this section, we address the hurdle of evaluating MI between non-singleton subset of labels. Since MI is commutative, (3.6) can also be written as below

\[
\begin{align*}
 f_{MI}(Q) &= MI(Y_{X-Q}, Y_Q|X, \mathcal{L}^{(t)}) = H(Y_{X-Q}|X, \mathcal{L}^{(t)}) - H(Y_{X-Q}|X, \mathcal{L}^{(t)}, Y_Q) \\
 &= H(Y_{X-Q}|X, \mathcal{L}^{(t)}),
\end{align*}
\]

(3.7)

We prefer this equation, since usually we have \( |Y_Q| = k \ll |Y_X| \) and thus it leads to a more computationally efficient problem. Note that the first term in the right hand side of (3.7) can be evaluated similar to equation (3.4). The major difficulty we need to handle in (3.7) is the computation of the second term, which requires considering all possible label assignments to \( Y_Q \). To make this computationally tractable, we propose to use a greedy strategy based on two variants: pessimistic and optimistic approximations of MI. To see this, we focus on the second term where \( Y_Q \) is not observed yet nevertheless the entropy is conditioned upon it. Such conditional entropy is defined in terms of
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

the expectation with respect of the unobserved conditioned random variables:

\[ H(Y_{X-Q}|X, \mathcal{L}^{(t)}, Y_Q) = \sum_{J \in \{1, \ldots, c\}^k} \mathbb{P}(Y_Q = J|X, \mathcal{L}^{(t)}) \cdot H(Y_{X-Q}|X, \mathcal{L}^{(t)}, Y_A = J) \]  

(3.8)

where \( \{1, \ldots, c\}^{|A|} \) is the set of all possible class label assignments to the samples in \( Q \). For example, if \( Q \) has three samples (\( k = |Q| = 3 \)) and \( c = 2 \), then this set would be equal to

\[ \{\{1,1,1\}, \{2,1,1\}, \{1,2,1\}, \{2,2,1\}, \{1,2,2\}, \{2,1,2\}, \{1,1,2\}\} \].

For each fixed label permutation \( J \), the classifier should be retrained after adding the new labels \( Y_Q = J \) to the already observed labels in \( \mathcal{L}^{(t)} \) to compute the conditional entropy \( H(Y_{X-Q}|X, \mathcal{L}^{(t)}, Y_Q = J) \). It is also evident from the example above that the number of possible assignments \( J \) to \( Y_Q \) is \( c^k \). Therefore, the number of necessary classifier updates grows exponentially with \( k \). This is computationally very expensive and makes equation (3.8) impractical. Alternatively, we can replace the expectation in (3.8) with a minimization/maximization to get a pessimistic/optimistic approximation of MI. Such a replacement enables us to employ efficient greedy approaches to estimate \( f_{MI} \) in a conservative/aggressive manner. The greedy approach that we use here is compatible with the iterative nature of the optimization Algorithms 3.2 and 3.3 (described in section 3.4). In the remainder of this section, we focus on the pessimistic approximation. Similar equations can be derived for the optimistic case. The first step is replacing the weighted summation in (3.8) by a maximization:

\[ f_{MI}^{\text{pess}}(Q) = H(Y_{X-Q}|X, \mathcal{L}^{(t)}) - \max_{J \in \{1, \ldots, c\}^k} H(Y_{X-Q}|X, \mathcal{L}^{(t)}, Y_Q = J) \]  

(3.9)

Note that \( f_{MI}^{\text{pess}}(A) \) is always less than or equal to \( f_{MI} \). Equation (3.9) still needs the computation of the conditional entropy for all possible assignments \( J \). However, it enables us to use greedy approaches to approximate \( f_{MI}^{\text{pess}}(A) \) for any candidate query set \( A \subseteq \mathcal{U} \), as described below.

For now assume that \( Q \) is fixed to a set of size \( k \) that can be shown element-wise by \( Q = \{u_1, \ldots, u_k\} \). Define \( Q_i = \{u_1, \ldots, u_i\} \) for any \( i \leq k \) (hence \( Q_k = Q \)). In the first iteration we can evaluate (3.9) simply for the singleton \( f_{MI}^{\text{pess}}(\{u_1\}) \) and store \( \hat{y}_{u_1} \), the assignment to \( y_{u_1} \) which
maximizes the conditional entropy in (3.9):

\[
\begin{align*}
    f_M^{\text{pess}}(\{u_1\}) &= H(Y_{X-\{u_1\}}|X, \mathcal{L}^{(t)}) - \max_{j \in \{1, \ldots, c\}} H(Y_{X-\{u_1\}}|X, \mathcal{L}^{(t)}, y_{u_1} = j) \\
    &= H(Y_{X-\{u_1\}}|X, \hat{\theta}) - H(Y_{X-\{u_1\}}|X, \mathcal{L}^{(t)}, y_{u_1} = \hat{y}_{u_1}), \\
\end{align*}
\]

(3.10)

where we used (3.3) to substitute the first term with \(H(Y_{X-\{u_1\}}|\hat{\theta})\) and \(\hat{y}_{u_1} \in \{1, \ldots, c\}\) is defined as the label maximizing \(H(Y_{X-Q}|X, \mathcal{L}^{(t)}, y_{u_1})\). Note that computing \(\hat{y}_{u_1}\) requires \(c\) times of retraining the classifier for all possible assignment \(j \in \{1, \ldots, c\}\) to \(y_{u_1}\). In practice, the retraining process can be very time-consuming. Here, instead of retraining the classifier from scratch, we leverage the current estimate of the classifier’s parameter vector and take one quasi-Newton step to update this estimate:

\[
\hat{\theta}_1 := \hat{\theta} - H^{-1}_1 \cdot g_1.
\]

(3.11)

where \(g_1\) and \(H_1\) are the gradient vector and Hessian matrix of the log-likelihood function of our classifier given the labels in \(\mathcal{L}^{(t)}\) and the newly added one \(\{y_{u_1} = \hat{y}_{u_1}\}\). Then we can use the following approximation

\[
H(Y_{X-\{u_1\}}|X, \mathcal{L}^{(t)}, y_{u_1} = \hat{y}_{u_1}) \approx H(Y_{X-\{u_1\}}|X, \hat{\theta}_1).
\]

(3.12)

In Appendix C we derive the update equation in case a multinomial logistic regression is used as the discriminative classifier. Specifically, we will see that \(g_1\) and \(H_1^{-1}\) can be obtained efficiently from \(g_0\) and \(H_0^{-1}\) which denote the gradient and Hessian matrix associated with the current labeled data in \(\mathcal{L}^{(t)}\).

If \(k = 1\), we are done. Otherwise, to move from iteration \(i - 1\) to \(i\) (\(1 < i \leq k\)), \(f_M^{\text{pess}}(Q_{i-1} \cup \{u_i\})\) will be approximated from the previous iterations:

\[
f_M^{\text{pess}}(Q_i) \approx H \left( Y_{X-Q_i} \big| \hat{\theta} \right) - \max_{j \in \{1, \ldots, c\}} H \left( Y_{X-Q_i} \big| X, \mathcal{L}^{(t)}, \hat{Y}_{Q_{i-1}}, y_{u_i} = j \right),
\]

(3.13)

where \(\hat{Y}_{Q_{i-1}} = \{\hat{y}_{u_1}, \ldots, \hat{y}_{u_{i-1}}\}\) are the assignments maximizing the conditional entropy that are stored from the previous iterations, such that the \(z\)-th element \(\hat{y}_{u_z}\) is the assignment stored for \(u_z = A_z - A_{z-1}(1 \leq z \leq i)\). Note that (3.13) is an approximation of the pessimistic MI, as is defined by (3.9), however, in order to keep the notations simple we use the same notation \(f_M^{\text{pess}}\) for both. Moreover, similar to (3.10) there are \(c\) time of classifier updates involved in the computation.
of (3.13). To complete iteration \(i\), we make \(\hat{y}_{u_i}\) equal to the assignment to \(y_{u_i}\) that maximizes the second term in (3.13) and add it to \(\hat{Y}_{A_{i-1}}\) to form \(\hat{Y}_{Q_i}\).

As in the first iteration, the conditional entropy term in (3.13) is estimated by using the set of parameters obtained from the quasi-Newton step:

\[
H \left( Y_{X \leftarrow Q_i} \mid X, \mathcal{L}^{(t)}, \hat{Y}_{Q_{i-1}}, y_{u_i} = j \right) \approx H \left( Y_{U \leftarrow A_i} \mid X, \hat{\theta}_i \right) \tag{3.14}
\]

where

\[
\hat{\theta}_i = \hat{\theta}_{i-1} - H^{-1}_i \cdot g_i. \tag{3.15}
\]

Considering equations (3.10) and (3.13) as the greedy steps of approximating \(f_{MI}\), we see that the number of necessary classifier updates are \(c \cdot k\), since there are \(k\) iterations each of which requires \(c\) times of retraining the classifier. Thus, the computational complexity reduced from the exponential cost in the exact formulation (3.8) to the linear cost in the greedy approximation.

Similar to (3.13), for the \(i\)'th iteration of optimistic approximation (not to be confused with learning iterations), we will have:

\[
f_{opt}^{MI}(Q_i) = H \left( Y_{X \leftarrow Q_i} \mid \hat{\theta} \right) - \min_{j \in \{1, \ldots, c\}} H \left( Y_{X \leftarrow Q_i} \mid X, \mathcal{L}^{(t)}, \hat{Y}_{Q_{i-1}}, y_{u_i} = j \right), \tag{3.16}
\]

where \(\hat{Y}_{Q_{i-1}} = \{\hat{y}_{u_1}, \ldots, \hat{y}_{u_{i-1}}\}\) is the set of class assignments minimizing the conditional entropy that are stored from the previous iterations. Clearly, the reduction of the computational complexity remains the same in the optimistic formulation.

Let us emphasize that, from the definitions of \(f_{pess}^{MI}\) and \(f_{opt}^{MI}\), we always have the following inequality

\[
f_{pess}^{MI}(A) \leq f_{MI}(A) \leq f_{opt}^{MI}(A), \forall A \subseteq \mathcal{U}. \tag{3.17}
\]

The first (or second) inequality turns to equality, if the results of averaging in conditional entropy in (3.8) is equal to maximization (or minimization) involved in the approximations. This is equivalent to saying that the posterior probability \(\mathbb{P}(Y_A \mid X, Y_A^*)\) is a degenerative distribution concentrated at the assignment \(Y_A = J\) that maximizes (or minimizes) the conditional entropy. Furthermore, if the posterior is a uniform distribution, giving the same posterior probability to all possible assignments \(J \in \{1, \ldots, c\}^{\lvert A \rvert}\), then the averaging, minimization and maximization lead to the same numerical result and therefore we get \(f_{pess}^{MI} = f_{opt}^{MI} = f_{MI}\).

In theory, the value of MI between any two random variables is non-negative. However,
because of the approximations made in computing the pessimistic or optimistic evaluations of MI, it is possible to get negative values depending on the distribution of the data. Therefore, after going through all the elements of $\mathcal{A}$ in evaluating $f_{\text{MI}}^{\text{pess}}$ (or $f_{\text{MI}}^{\text{opt}}$), we take the maximum between the approximations of $f_{\text{MI}}^{\text{pess}}(\mathcal{A})$ (or $f_{\text{MI}}^{\text{opt}}(\mathcal{A})$) and zero to ensure its non-negativity.

### 3.4 Randomized vs. Deterministic Submodular Optimizations

In this section, we begin by reviewing the basic definitions regarding submodular set functions, and see that both $f_{\text{MI}}$ and $f_{H}$ satisfy submodularity condition. We then present two methods for submodular maximization: a deterministic and a randomized approach. The latter is applicable to submodular and monotone set functions such as $f_{H}$. But $f_{\text{MI}}$ is not monotone in general, hence we present the randomized approach for this objective.

#### 3.4.1 Submodular Set Functions

**Definition 3.1.** A set function $f : 2^X \rightarrow \mathbb{R}$ is said to be submodular if

$$f(Q) + f(Z) \geq f(Q \cup Z) + f(Q \cap Z), \quad \forall Q, Z \subseteq X$$

(3.18)

We call $f$ supermodular if the inequality in (3.18) is reversed. In many occasions, it is easier to use an equivalent definition, which uses the notion of discrete derivative defined as:

$$\rho_f(Q, u) := f(Q \cup \{u\}) - f(Q), \quad \forall Q \subseteq X, \ u \in X$$

(3.19)

**Proposition 3.2.** Let $f : 2^X \rightarrow \mathbb{R}$ be a set function. $f$ is submodular if and only if we have

$$\rho_f(Q, u) \geq \rho_f(Z, u), \quad \forall Q \subseteq Z \subseteq X, \ u \in X - Z$$

(3.20)

Before showing submodularity of MI, we first prove it for the entropy objective.

**Lemma 3.3.** The set function $f_{H}(Q) = H(Y_Q|X, \mathcal{L})$ is submodular for any given learning material $\mathcal{L}$.

**Proof.** Let us first write the discrete derivative $\rho_{f_{H}}(Q, u)$ for a general subset $Q \subseteq X$ and an element
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

\[ u \in X: \]

\[
\rho_{f_H}(Q) = f_H(Q \cup \{u\}) - f_H(Q) = H(Y_{Q \cup \{u\}}|X, \mathcal{L}) - H(Y_Q|X, \mathcal{L})
\]
\[
= H(Y_Q|X) + H(y_u|X, \mathcal{L}, Y_Q) - H(Y_Q|X, \mathcal{L})
\]
\[
= H(y_u|X, \mathcal{L}, Y_Q),
\]

where by our index definitions we have \( Y_{Q \cup \{u\}} = Y_Q \cup y_u \) (note that \( y_u \) is the same as \( Y_{\{u\}} \)). In the third line of (3.21) we used the joint entropy identity \( H(Q, Z) = H(Q) + H(Z|Q) \forall Z, Q \subseteq X \).

Let us now check (3.20) for \( f_H \). Let \( Q \subseteq Z \subseteq X \) and \( u \in X - Z \), then:

\[
\rho_{f_H}(Q, u) - \rho_{f_H}(Z, u) = H(y_u|X, \mathcal{L}, Y_Q) - H(y_u|X, \mathcal{L}, Y_Z)
\]
\[
= H(y_u|X, \mathcal{L}, Y_Q) - H(y_u|X, \mathcal{L}, Y_Q, Y_{Z-Q})
\]
\[
= MI(y_u, Y_{Z-Q}|X, \mathcal{L}, Y_Q) \geq 0.
\]

The positivity of mutual information (MI) can be easily proven using Jenson inequality. Hence, from Proposition 3.2, \( f_H \) is submodular.

Note that Lemma 3.3 is a very general statement and does not assume using a point estimation in (3.3) or label independence given the parameter in (3.4).

**Theorem 3.4.** The set functions \( f_{MI} \), defined in (3.6) or equivalently as in (3.7), is submodular.

**Proof.** As is defined in (3.6), for any given learning material \( \mathcal{L} \), we can write \( f_{MI}(Q) \) as

\[
f_{MI}(Q) = f_H(Q) - H(Y_Q|X, \mathcal{L}, Y_{X-Q}).
\]

The first term is already shown to be submodular in Lemma 3.3. It remains to show that \( g(Q) := H(Y_Q|X, \mathcal{L}, Y_{X-Q}) \), the second term with the opposite sign, is supermodular (meaning that \( -g(Q) \) is submodular). Once again, first we write the discrete derivative of \( g \) for a subset \( Q \) and element
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

\( u \in X: \)

\[
\rho_g(Q) = g(Q \cup \{u\}) - g(Q) \tag{3.24}
\]

\[
= H(Y_{Q \cup \{y\}}|X, \mathcal{L}, Y_{X - Q \cup \{u\}}) - H(Y_Q|X, \mathcal{L}, Y_{X - Q})
\]

\[
= H(y_u|X, \mathcal{L}, Y_{X - Q \cup \{u\}}) + H(Y_Q|X, \mathcal{L}, Y_{X - Q \cup \{u\}}, y_u) - H(Y_Q|X, \mathcal{L}, Y_{X - Q})
\]

\[
= H(y_u|X, \mathcal{L}, Y_{X - Q \cup \{u\}})
\]

And now we check the different between the derivatives for any \( Q \) and \( Z \) such that \( Q \subseteq Z \subseteq X \) and element \( u \in X - Z \):

\[
\rho_g(Q) - \rho_g(Z) = H(y_u|X, \mathcal{L}, Y_{X - Q \cup \{u\}}) - H(y_u|X, \mathcal{L}, Y_{X - Z \cup \{u\}}) \tag{3.25}
\]

\[
= H(y_u|X, \mathcal{L}, Y_{X - Z \cup \{u\}} \cup Y_{Z - Q}) - H(y_u|X, \mathcal{L}, Y_{X - Z \cup \{u\}})
\]

\[
= -MI(y_u, Y_{Z - Q}||X, \mathcal{L}, Y_{X - Z \cup \{u\}}) \leq 0.
\]

where the negativity comes from positivity of mutual information. Hence, \( g(Q) \) is supermodular and consequently \( f_{MI}(Q) \) is submodular.

Another concept that we need in submodular optimization is \textit{monotonicity} of a set function:

\textbf{Definition 3.5.} The set function \( f : 2^X \rightarrow \mathbb{R} \) is said to be monotone (non-decreasing) if for every \( Q \subseteq Z \subseteq X \) we have \( f(Q) \leq f(Z) \).

It is easy to show if a set function has non-negative discrete derivative for every subset \( Q \subseteq X \) and element \( u \in X \), then it is non-decreasing. Note that based on 3.21, we know that \( \rho_{f_H}(Q, u) \geq 0 \) and entropy objective \( f_H(Q) \) is non-decreasing, however, \( f_{MI}(Q) \) is not monotone\(^1\).

3.4.2 Submodular Maximization

Although submodular functions can be minimized efficiently, they are NP-hard to maximize \cite{87}, and therefore we have to use approximate algorithms. Next, we briefly discuss the classical approximate submodular maximization method widely used in batch querying \cite{4, 51, 52, 47, 45}.

This greedy approach, we call \textit{deterministic}, is first proposed in the seminal work of \cite{88} (shown

\(^1\)To understand this it suffices to recall that \( f_{MI}(\emptyset) = f_{MI}(U) = 0 \).
Algorithm 3.2: The deterministic approach for solving cost-constrained querying optimization in (1.6)

**Inputs:** The objective set function $f$, the unlabeled pool $X$, the query batch size $k > 0$

**Outputs:** a query subset $Q \subseteq X$ of size $k$

```plaintext
/* Initializations */
1 $Q_0 \leftarrow \emptyset$
2 $X_0 \leftarrow U$
/* Starting the Iterations */
3 for $i = 1 \rightarrow k$ do
    /* Local maximization */
4     $u_i \leftarrow \arg\max_{u \in X_{i-1}} f(Q_{i-1} \cup \{u\})$
    /* Updating the Loop Variables */
5     $Q_i \leftarrow Q_{i-1} \cup \{u_i\}$
6     $X_i \leftarrow X - Q_i$
7 return $Q = Q_k$
```

in Algorithm 3.2) and its performance has been analyzed for nondecreasing and submodular set functions as follows:

**Theorem 3.6.** Let $f : 2^X \rightarrow \mathbb{R}$ be a submodular and nondecreasing set function as the querying objective with $f(\emptyset) = 0$ \(^2\), $Q$ be the output of Algorithm 3.2 and $Q^*$ be the optimal solution to the cost-constrained querying optimization (see (1.6)). Then we have:

$$f(Q) \geq \left[ 1 - \left( \frac{k-1}{k} \right)^k \right] f(Q^*) \geq \left( 1 - \frac{1}{e} \right) f(Q^*).$$

(3.26)

where $k$ is the size of $Q$ in the cardinality constraint.

The proof is given by [88] and [89]. [90] also showed that Algorithm 3.2 gives the optimal approximate solution to the cost-constrained querying optimization problem in (1.6) for nondecreasing functions such as $f_H$. As is mentioned before, $f_{MI}$ is not monotone in general and therefore Theorem 3.6 is not applicable.

Recently several algorithms have been proposed for approximate maximization of non-negative submodular set functions, which are not necessarily monotone. [91] made the first attempt towards this goal by proposing a $(2/5)$-approximation algorithm and also proving that $1/2$ is the

\(^2\)This can always be assumed since maximizing a general set function $f(Q)$ is equivalent to maximizing its adjusted version $g(Q) := f(Q) - f(\emptyset)$ which satisfies $g(\emptyset) = 0$. 

49
optimal approximation factor in this case. \[92\] could achieve this optimal bound in expectation by proposing a randomized iterative algorithm. However, these algorithms are designed for \textit{unconstrained} maximization problems. Later, \[93\] devised a \((1/e)\)-approximation randomized algorithm with cardinality constraint, which is more suitable for batch active learning. A pseudocode of this approach is shown in Algorithm 3.3 where instead of selecting the sample with maximum objective value at each iteration, the best \(k\) samples are identified (line 4) and one of them is chosen randomly (line 5). Such a randomized procedure provides a \((1/e)\)-approximation algorithm for maximizing a nonnegative submodular set function such as \(f_{MI}\):

\begin{equation}
\mathbb{E}[f(Q)] \geq \left(1 - \frac{1}{k}\right)^{k-1} f(Q^*) \geq \frac{1}{e} f(Q^*).
\end{equation}

The proof can be found in \[93\]. In order to be able to select \(k\) samples from \(X_i\) to form \(\mathcal{M}_i\) for all \(i\), it suffices to ensure that the smallest unlabeled set that we sample from \(X_{k-1}\) has enough members, i.e. \(k \leq |Q_{k-1}| = |X| - k + 1\) hence \(k \leq (|X| + 1)/2\). While this is a reasonable assumption in batch active learning, \[93\] relaxed it by introducing \textit{dummy} variables which have been assigned zero objective values.

Observe that although the assumptions in Theorem 3.7 are weaker than those in Theorem 3.6, the bound shown in (3.27) is also looser than that in (3.26). However, interestingly, it is proven that inequality (3.26) will still hold for Algorithm 3.3 if the monotonicity of \(f\) is satisfied (see Theorem 3.1. in \[93\]). Thus, the randomized Algorithm 3.3 is expected to be performing similar to Algorithm 3.2 for monotone functions.

Note that in both algorithms, the variables \(u_i\) in iteration \(i\), is determined by deterministic or stochastic maximization of \(f(Q_{i-1} \cup \{u\})\). Fortunately, such maximization needs only computations in the form of (3.13) or (3.16). Finally, one can observe that Algorithms 3.2 and 3.3 are equivalent for sequential querying \((k = 1)\).

### 3.5 Total Complexity Reduction

We measure the complexity of a given querying algorithm in terms of the required number of classifier updates. This makes our analysis general and independent of the updating procedure,


**Algorithm 3.3:** The randomized approach for solving cost-constrained querying optimization in (1.6)

**Inputs:** The objective set function \( f \), the unlabeled pool \( X \), the query batch size \( k > 0 \)

**Outputs:** a query subset \( Q \subseteq X \) of size \( k \)

```
/* Initializations */
1 Q₀ ← ∅
2 X₀ ← \( U \)
/* Starting the Iterations */
3 for \( i = 1 \rightarrow k \) do
   /* Selecting \( k \) points with highest \( f \) values */
4 \( \mathcal{M}_i \leftarrow \arg \max_{\mathcal{M} \subseteq X_{i-1}} \sum_{u \in \mathcal{M}} f(Q_{i-1} \cup \{u\}) \)
   /* Random Selection */
5 \( u_i \leftarrow \text{RANDOM}(\mathcal{M}_i) \)
   /* Updating the Loop Variables */
6 \( Q_i \leftarrow Q_{i-1} \cup \{u_i\} \)
7 \( X_i \leftarrow X - Q_i \)
8 return \( Q = Q_k \)
```

which can be done in several possible ways. As we discussed in the last section, we chose to perform a single step of quasi Newton in (3.11) but alternatively one can use full training or any other numerical parameter update.

Consider the following optimization problems:

\[
\arg \max_{Q \subseteq X, |Q|=k} f_{\text{MI}}(Q) \quad \text{(3.28a)}
\]

\[
\text{greedy } \arg \max_{Q \subseteq X, |Q|=k} \tilde{f}_{\text{MI}}(A) \quad \text{(3.28b)}
\]

where “greedy \( \arg \max \)” denotes the greedy maximization operator that uses Algorithm 3.2 or 3.3 to maximize the objective, and \( \tilde{f}_{\text{MI}} \) is either \( f^\text{pess}_{\text{MI}} \) or \( f^\text{opt}_{\text{MI}} \). Note that (3.28a) formulates the global maximization of the exact MI function and (3.28b) shows the optimization in our framework, that is a greedy maximization of the pessimistic/optimistic MI approximations. In the following remark, we compare the complexity of solving the two optimizations in (3.28) in terms of the number of classifier updates required for obtaining the solutions.

**Remark 3.8.** For a fixed \( k \), the number of necessary classifier updates for solving (3.28a) increases
with order $k$, whereas for (3.28b) it changes linearly.

Proof. As is explained in section 3.3, the number of classifier updates for computing $f_{M_{I}}(Q)$ without any approximations, is $c^{k}$. Moreover, in order to find the global maximizer of MI, $f_{M_{I}}$ needs to be evaluated at all subsets of $Q$ with size $k$. Recall that $|X|$ is denoted by $n$. There are $\binom{n}{k} = O(n^{k})$ of such subsets. Hence, the total number of classifier update required for global maximization $f_{M_{I}}$ is of order $O((n \cdot c)^{k})$.

Now, regarding (3.28b), recall from section 3.3 that if $g_{i-1}$ and $H_{i-1}$ are stored from the previous iteration, computing $\tilde{f}_{M_{I}}(Q_{i-1} \cup \{u_{i}\})$ needs only $c$ classifier updates. However, despite the evaluation problem in section (3.3), in computing line (4) of Algorithms 3.2 and 3.3, the next sample to add, that is $u_{i}$, is not given. In order to obtain $u_{i}$, $\tilde{f}_{M_{I}}$ is to be evaluated at all the remaining samples in $X_{i-1}$. Since, $|X_{i-1}| = n - i + 1$, the number of necessary classifier updates in the $i$'th iteration is $c \cdot (n - i + 1)$. Both algorithms run $k$ iterations that results the following total number of classifier updates:

$$cn + c(n - 1) + \ldots + c(n - k + 1) = ck\left(n - \frac{k + 1}{2}\right) = O(ckn).$$

\[\square\]

3.6 Further Speed-up

Even after approximating $f_{M_{I}}$ using the pessimistic or optimistic formulations, MI-based algorithms can be significantly slow for large data sets. In order to further scale up our algorithm, induced by [94], we first selects a subset of the unlabeled samples by only choosing the most $\beta$ uncertain samples (where $\beta \in \mathbb{Z}^+$). We then ran our MI-based algorithm over such filtered data. More formally, the input set of unlabeled indices to Algorithms 3.2 and 3.3 will be

$$X_f = \arg \max_{U \subseteq X, |U| = \beta} \sum_{u \in U} f_{H}(\{u\}). \quad (3.29)$$

It is evident that for $\beta = |X|$ the filtered data will be equal to the original unlabeled pool $X_f = X$. From now on, we add the adjective filtered to any querying algorithm that is preceded by reduction of the unlabeled data into the samples with high uncertainty as described above.
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

3.7 Results

In this section, we show our experimental results over several data sets on three different fields: medicine (3.7.1), image processing (3.7.2) and music harmony analysis (3.7.3). We ran our MI-based querying algorithms against entropy-based and random active learning benchmarks. In the following section, first we describe the data sets that have been used, then we explaining the experimental settings and present the numerical results.

3.7.1 Cardiotocography

This data set is downloaded from UCI repository [95] and contains 2126 fetal cardiotocograms each of which is represented by a 21-dimensional feature vector. The data is categorized into three classes based on the fetal states: normal, suspect and pathological (therefore \( c = 3 \)). All the data samples are first projected into a 15-dimensional PCA subspace obtained from the unlabeled pool. The initial labeled data set chosen randomly in the beginning of each experiment, consists of 75 samples (25 samples per class). We will refer to this data set as “Cardio” in the following sections.

3.7.2 MNIST

This is an image database of handwritten digits 1 to 9 [96]. Here, we only use images for digits 1 to 4, hence \( c = 4 \). The data set, consisting of \( 20 \times 20 \) images, is already divided into a testing/training partitions. In our experiments, these partitions are fixed as given, but each time the initial labeled data sets are randomly chosen from the training partition. The raw 400-dimensional feature vectors are projected into 10-dimensional PCA subspace constructed based on the training partition. After choosing only images of digits from 1 to 4, the size of the testing and training partitions are 4130 and 4159, respectively. We also set the size of our initial labeled data set \( \mathcal{L}(0) \) to 200 (50 samples per class).

3.7.3 Bach Choral Harmony

The other data set that we used for evaluating performance of the algorithms contains pitch information of time events of 60 chorales by Johann Sebastian Bach [97]. Each event is represented by pitch-wise and meter information and is assigned a chord class label. We selected the events associated with the five most frequent chords in the data set: D-major, G-major, C-major, F-major and A-major (hence \( c = 5 \)); resulting a set of 2221 samples. Discarding pitch class of the bass
notes and the metric information, we used the binary indicators of the pitch classes corresponding to equal-tempered 12 notes of the chromatic scale. This leads to a set of 12-dimensional binary feature vectors that are projected into 8-dimensional PCA subspace obtained based on the training data at each experiment. We will refer to this data set simply as “Bach” in the remaining sections.

3.7.4 Experimental Settings

From the previous sections, we have two methods of evaluating MI and two optimization techniques, leading to four different ways of doing MI-based querying, in all of which we used the filtered pool of unlabeled samples \( X_f \) that is obtained with \( \beta = 100 \). Throughout this section, we distinguish different approaches involved in our experimental settings using the labels listed below:

- **Pess-MI-Det**: Pessimistic MI \( f_{MI}^{\text{pess}} \) with deterministic optimization (Algorithm 3.2);
- **Pess-MI-Rand**: Pessimistic MI \( f_{MI}^{\text{pess}} \) with randomized optimization (Algorithm 3.3);
- **Opt-MI-Det**: Optimistic MI \( f_{MI}^{\text{opt}} \) with deterministic optimization (Algorithm 3.2);
- **Opt-MI-Rand**: Optimistic MI \( f_{MI}^{\text{opt}} \) with randomized optimization (Algorithm 3.3);
- **entropy**: Entropy objective \( f_H \) with deterministic optimization (Algorithm 3.2);
- **random**: Random querying.

In sequential querying, where deterministic and randomized optimization algorithms are equivalent, we use **Pess-MI** and **Opt-MI** to refer to the MI-based objectives without mentioning the optimization type.

In running the querying experiments over all the data sets, we used a linear logistic regression as the core classifier. In case that the data under consideration is not already divided into testing/training partitions, we randomly generate such partitions in each experiment with fixed ratio of 3/7 (testing size to training size). The initial training data set \( \mathcal{L}^{(0)} \) is randomly selected from the training partition and the rest of the training samples are considered as the unlabeled pool \( X \) from which the queries are to be selected in each querying iteration. Moreover, in each experiment we first reduce the dimensionality of the data using PCA over the unlabeled pool.

In the experiments, we iteratively select the query batches of either sizes \( k = 1, 5, 10 \) or 20, add the selected queries together with their class labels to the labeled data set and re-calculate the parameters of the classifier, which in turn, leads to an updated accuracy value based on the testing
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

partition. For each value of $k$, we repeated running the experiments for 25 times, each time with a different random selection of testing/training partitions and a different initial labeled set $\mathcal{L}^{(0)}$. Hence, in total, we get 25 accuracy curves for each value of $k$. Ideally, we want an active learning algorithm whose accuracy curve increases as fast as possible, i.e. obtaining a more accurate classifier with labeling fewer number of query batches.

In order to present the performance of the listed algorithms, we calculate the average and standard deviation (STD) of the 25 accuracy curves for each algorithm. Furthermore, for pairwise comparison between the MI-based approaches and the benchmarks for a fixed value of $k$, we perform two-sample one-tail T-tests over the accuracy values of the competing algorithms. Note that such hypothesis test can and should be done over the accuracy values calculated after each querying iteration $t$ separately. Here, the assumption is that the accuracy levels generated from the 25 querying experiments at iteration $t$ are independent from each other. Let $\eta_t$ denote the random variable presenting the accuracy of the updated classifier after $t$ times of running an MI-based querying algorithm and $\eta'_t$ be a similar random variable for a competing non-MI-based method. Then, we consider the null and alternative hypotheses to be:

$$H_0 : \mu(\eta_t) \leq \mu(\eta'_t)$$
$$H_1 : \mu(\eta_t) > \mu(\eta'_t)$$  \hspace{1cm} (3.30)

where $\mu(\eta_t)$ and $\mu(\eta'_t)$ are the mean of the random variables $\eta_t$ and $\eta'_t$. We perform such T-test for comparing all modes of MI-based objectives (see the list above) versus the entropy-based and random querying algorithms. Rejecting the null hypothesis implies that the new accuracy in the $t$-th iteration of an MI-based querying is not less than or equal to the case when we use a querying objective other than the approximating variants of $f_{MI}$. In other words, obtaining a smaller $p$-value for a T-test described above, means that with a higher probability the accuracy of the updated classifier is larger when using the MI-based approach for querying.

3.7.5 Numerical Results

The numerical results of running sequential active learning with different querying algorithms are shown in Figure 3.1 and the results of batch active learning with different batch sizes are shown in Figures 3.2 (for $k = 5$), 3.3 (for $k = 10$) and 3.4 (for $k = 20$). The figures show the average
accuracy curves (first row of each figure), the standard deviation of the accuracy curves (second rows), and the resulting $p$-values of the hypothesis tests for comparison between the MI-based approaches and entropy-based (third rows) or random querying (fourth rows).

As it is mentioned before, the deterministic and randomized optimization algorithms described in section 3.4 are equivalent in sequential querying. Therefore, we have only two variants of MI-based querying in Figure 3.1. This figure shows that for two data sets (MNIST and Bach) the MI-based approaches perform similar or sometimes even worse than the entropy-based benchmark. This can be explained by noting that the main shortcoming of using the entropy objective $f_h$, that is redundancy among the queries, is meaningful only when we have multiple samples in the batch, that is $k > 1$. However, they mostly outperform random querying. Another observation is that using optimistic approximation of MI gave better results both in terms of average accuracy and the hypotheses $p$-values. Recall that the optimistic approximation of MI tries to minimize the entropy over the class labels of the remaining samples in each iteration, while the pessimistic approximation uses maximization of this entropy. Hence, our conjecture for this observation is that the optimistic approach does more aggressive exploitation in comparison with the pessimistic variant, in the sense of choosing the queries from the set of samples that lead to a lower classifier entropy.

For batch mode active learning, the MI-based approaches generally show better performance. That is, their accuracy curves grow more rapidly than the benchmarks. When comparing against the entropy-based approach, for data sets Cardio and Bach, we observe that the $p$-values are small in the beginning iterations. Whereas for MNIST data set, low $p$-values are mostly seen in the middle or late iterations. Hence, the probability that MI-based variants outperform the entropy is high in early querying iterations, before the labeled training set becomes large, for the two former data sets, and in later iterations for MNIST. This behavior can also be seen from the plots of the average accuracy.

Regarding the comparison against the random benchmark, we see from the $p$-value plots more conspicuously that MI-based approaches generally outperform random with high confidence soon after the early iterations. However, the plots for Bach, show that this confidence decrease in late iterations, which is mainly due to the growth of the accuracies of random to the same level as the MI-based curves.

Whereas in sequential active learning the optimistic approach did a better job in comparison with the pessimistic variant, the difference between these two variants shrinks as the size of query batch increases (and so does the number of required approximation iterations). Our conjecture is that accumulation of the approximation error makes the performance of optimistic and pessimistic MI-
based querying methods closer to each other, though still better than the benchmarks. Additionally, there is no significant difference between using deterministic or randomized optimization algorithms, which might be because of local monotonicity of the approximations $f_{MI}$. Also note that although there are large distinctions between MI-based variants when their $p$-values are large, we ignore those parts as uninformative regions, since they just imply that the probability of MI-based approaches outperforming the benchmarks is not high.
$k = 1$:

<table>
<thead>
<tr>
<th>Cardio</th>
<th>MNIST</th>
<th>Bach</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
<td><img src="image3" alt="Graph" /></td>
</tr>
<tr>
<td><img src="image4" alt="Graph" /></td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
</tr>
<tr>
<td><img src="image7" alt="Graph" /></td>
<td><img src="image8" alt="Graph" /></td>
<td><img src="image9" alt="Graph" /></td>
</tr>
<tr>
<td><img src="image10" alt="Graph" /></td>
<td><img src="image11" alt="Graph" /></td>
<td><img src="image12" alt="Graph" /></td>
</tr>
</tbody>
</table>

Figure 3.1: The experimental results of different querying approaches for sequential active learning ($k = 1$).
$k = 5$:

<table>
<thead>
<tr>
<th>Chapter 3. MI-Based Classification Active Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 5$:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cardio</th>
<th>MNIST</th>
<th>Bach</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Cardio_graph.png" alt="Graph" /></td>
<td><img src="MNIST_graph.png" alt="Graph" /></td>
<td><img src="Bach_graph.png" alt="Graph" /></td>
</tr>
<tr>
<td><img src="Cardio_STD.png" alt="Graph" /></td>
<td><img src="MNIST_STD.png" alt="Graph" /></td>
<td><img src="Bach_STD.png" alt="Graph" /></td>
</tr>
<tr>
<td><img src="Cardio_p_value.png" alt="Graph" /></td>
<td><img src="MNIST_p_value.png" alt="Graph" /></td>
<td><img src="Bach_p_value.png" alt="Graph" /></td>
</tr>
<tr>
<td><img src="Cardio_random.png" alt="Graph" /></td>
<td><img src="MNIST_random.png" alt="Graph" /></td>
<td><img src="Bach_random.png" alt="Graph" /></td>
</tr>
<tr>
<td><img src="Cardio_entropy.png" alt="Graph" /></td>
<td><img src="MNIST_entropy.png" alt="Graph" /></td>
<td><img src="Bach_entropy.png" alt="Graph" /></td>
</tr>
</tbody>
</table>

Figure 3.2: The experimental results of different querying approaches for batch active learning ($k = 5$).
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

\[ k = 10: \]

<table>
<thead>
<tr>
<th>Cardio</th>
<th>MNIST</th>
<th>Bach</th>
</tr>
</thead>
</table>

![Graphs showing experimental results](image)

Figure 3.3: The experimental results of different querying approaches for batch active learning \((k = 10)\).
CHAPTER 3. MI-BASED CLASSIFICATION ACTIVE LEARNING

$k = 20$:

<table>
<thead>
<tr>
<th>Cardio</th>
<th>MNIST</th>
<th>Bach</th>
</tr>
</thead>
</table>

- **Figure 3.4:** The experimental results of different querying approaches for batch active learning ($k = 20$).
Chapter 4

FI-based Classification Active Learning

In Section 1.5.3, we explained how we can do querying in a univariate learning model by maximizing the Fisher information criterion. Here, we discuss the general case of multivariate models by first explaining that the so-called Fisher information ratio (FIR) scalar function can be used as the querying objective of a probabilistic active learning (see (1.7)), and then giving a practical querying algorithm based on that. We also unify several existing FI-based classification active learning models using our framework.

4.1 Theoretical Analysis

Recall that training (proposal) distribution \( q(x) \) is the distribution where the queries are sampled from, and the test distribution \( p(x) \) is underlying distribution of test samples \( x_{\text{test}} \). Note that we are not applying any specific parametric form on these marginals. However, we assume that the conditional distribution of class labels \( y \) given the features \( x \) follow a parametric form as \( P(y|\theta,x) \) either in training or test joint distributions. That is:

\[
\begin{align*}
\text{training:} & \quad P_q(x,y|\theta) = q(x) \cdot P(y|\theta_0,x) \quad (4.1a) \\
\text{test:} & \quad P_o(x,y|\theta) = p(x) \cdot P(y|\theta_0,x) \quad (4.1b)
\end{align*}
\]

where \( \theta_0 \) is the true parameter of the conditional distribution in both training and test distributions. The marginals \( q(x) \) and \( p(x) \) are called training (proposal, or query) and test marginals, respectively.
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

Then, the Fisher information ratio (FIR) is defined as the following:

\[ \text{tr} \left( I_q(\theta_0)^{-1} I_o(\theta_0) \right) . \]  

where \( \text{tr}[\cdot] \) is the trace function, and \( I_q(\theta) \) and \( I_o(\theta) \) are Fisher information matrices for training and test joint distributions, respectively, as is defined in (1.15). Both of these matrices are assumed to be positive definite and therefore invertible. Note that the dependence of FIR on the query distribution \( q \) makes it a potential surrogate objective for probabilistic querying as is described in (1.7). Our theoretical contribution in this chapter is showing that FIR upper-bounds inference-based performance metric, that is log-likelihood loss function, which justifies FIR as a valid inference-based surrogate objective in probabilistic active learning framework. We then explain how our result is different than other theoretical results regarding FIR.

We give the theoretical analysis based on the original formulation of FIR in (4.2), which cannot be used directly in practice objective since it contains \( \theta_0 \) and \( I_0 \) that are not known. However, after proving FIR as the upper-bound, we show that a finite-sample approximation of this objective is also an upper-bound under circumstances. Such approximation is the objective that is used in practical active learning algorithms.

4.1.1 The Framework and Assumptions

The classification framework that we will be working in this chapter is similar to that in Chapter 3. Let the \( X \) denote the feature set (not necessarily finite) and each of its members \( x \in X \) can be assigned a class label \( y \in \{1, \ldots, c\} \). In order to present the statistical results more easily, we slightly change our notations in this section. For now, let us assume that there is no learning material available from the previous iteration since FIR, as is defined in (4.2), does not depend on learning materials given \( \theta_0 \) and \( I_0 \). Furthermore, focusing on a fixed learning iteration \( t \) in Algorithm 3.1, we discard putting the superscript \( (t) \) on the variables and denote the parameter estimate by \( \hat{\theta}_k \) and the learning material that is used to obtain that estimate by \( L_k = (Q, Y_Q) \) (where \( |Q| = k \)). This representation is to emphasize dependence of the variables to the number of labeled samples, which is essential in establishing convergence of various sequences indexed by the size of \( Q \).

Viewing the pair \( (x, y) \) as random variables, we assume that they are distributed according to a parametric joint distribution as is shown in (4.1), with the parameter space denoted by \( \Omega \subseteq \mathbb{R}^d \). Using a finite set of observed pairs as the training data, \( \mathcal{L}_k := \{(x_1, y_1), \ldots, (x_k, y_k)\} \) (with \( 0 <
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

\( k < \infty \), \( \hat{\theta}_k \) can be obtained by means of MLE and class labels of the unseen test sample \( x_{\text{test}} \) are predicted by maximizing \( P(y|x, \theta) \) with respect to \( y \).

Throughout this chapter, we make the following assumptions regarding the oracle, our classification model and the underlying data distributions:

(A0). The dependence of the joint distribution to the parameter \( \theta \) comes only from the class-conditional distribution and the marginal distribution does not depend on \( \theta \) (see (4.1)). Joint distributions with such parameter dependence is sometimes called type-II models, as opposed to type-I models which have parameter dependence in both class conditionals and marginal [2].

(A1). (Identifiability): Any joint distribution model of the form (4.1) are identifiable for different parameters. Meaning that for every distinct parameter vectors \( \theta_1 \) and \( \theta_2 \) in \( \Omega \), \( P(x,y|\theta_1) \) and \( P(x,y|\theta_2) \) are also distinct. That is

\[
\forall \theta_1 \neq \theta_2 \in \Omega \ \exists A \subseteq \mathcal{X} \times \{1, \ldots, c\} \quad \text{s.t.} \quad P(A|\theta_1) \neq P(A|\theta_2).
\]

(A2). The joint distribution has common support for all \( \theta \in \Omega \).

(A3). (Model Faithfulness): For any \( x \in \mathcal{X} \), we have access to an oracle that generates a label \( y \) according to the conditional \( P(y|\theta_0, x) \). That is, the posterior parametric model matches the oracle distribution. We call \( \theta_0 \) the true model parameter.

(A4). (Training joint): The set of observations in \( \mathcal{L}_k := \{(x_1, y_1), \ldots, (x_k, y_k)\} \) are drawn independently from the training joint distribution \( P_q \) in (4.1a).

(A5). (Test joint): The unseen test pairs \( (x_{\text{test}}, y_{\text{test}}) \) are distributed according to the test joint distribution \( P_\circ \) in (4.1b).

(A6). (Differentiability): The log-conditional \( \log P(y|\theta, x) \) is of class \( C^1(\Omega) \) as a function of \( \theta \) and for all \((x, y) \in \mathcal{X} \times \{1, \ldots, c\} \).

(A7). The parameter space \( \Omega \) is compact and there exists an open ball around the true parameter of the model \( \theta_0 \in \Omega \).

\[1\] We say that a function \( f : X \to Y \) is of \( C^p(X) \), for an integer \( p > 0 \), if its derivatives up to \( p \)-th order exist and are continuous at all points of \( X \).
(A8). (Invertibility): The Fisher information matrices of the joint distributions are positive definite and therefore invertible for all \( \theta \in \Omega \), and for any type of marginal that is used under assumption (A0).

Regarding assumptions (A4) and (A5), note that the training and test marginals are not necessarily equal. The test marginal is usually not known beforehand and \( q \) cannot be set equal to \( p \) in practice. Such lack of knowledge is what [98] had called covariate shift in distribution. In the remaining sections of the report, we use subscripts \( \phi \) and \( q \) for the statistical operators that consider \( p(x) \) and \( q(x) \) as the marginal in the joint distribution, respectively. We explicitly mention \( x \) as the input argument in order to refer to marginal operators. For instance, \( E_q \) denotes the joint expectation with respect to \( q(x) \mathbb{P}(y | \theta, x) \), whereas \( E_{q(x)} \) denotes the marginal expectation with respect to \( q(x) \).

### 4.1.2 Background

Here, we provide a short review of maximum likelihood estimation (MLE) as our inference method, and briefly introduce Fisher information of a parametric distribution. These two basic concepts enable us to explain some of the key properties of MLE, upon which our further analysis of FIR objective relies. Note that our focus in this section is on sub-problem (ii) with the assumptions listed above.

#### 4.1.2.1 Maximum Likelihood Estimation (MLE)

In this section, we review MLE as the inference method of our learning diagram (Figure 1.1) in more details than the previous chapter. Given a training data set \( \mathcal{L}_k = (Q, Y_Q) = \{(x_1, y_1), \ldots, (x_k, y_k)\} \), a maximum likelihood estimate (MLE) is obtained by maximizing the log-likelihood function over all pairs inside \( \mathcal{L}_k \), with respect to the parameter \( \theta \):

\[
\hat{\theta}_k = \arg \max_{\theta} \log \mathbb{P}(\mathcal{L}_k | \theta), \tag{4.3}
\]

Under the assumptions (A0) and (A4), the optimization in (4.3) can be written as

\[
\hat{\theta}_k = \arg \max_{\theta} \sum_{i=1}^{k} \log \mathbb{P}(y_i | x_i, \theta), \tag{4.4}
\]

Equation (4.4) shows that MLE does not depend on the marginal when using type-II model. Hence, in our analysis we focus on the conditional log-likelihood as the classification objective, and simply
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

call it the log-likelihood function when viewed as a function of the parameter vector \( \theta \), for any given pair \((x, y) \in \mathcal{X} \times \{1, ..., c\}\):

\[
\ell(\theta; x, y) := \log P(y|x, \theta). \tag{4.5}
\]

Moreover, for any set of pairs independently generated from the joint distribution of the training data, such as \( \mathcal{L}_k \) mentioned in (A4), the log-likelihood function will be:

\[
\ell(\theta; \mathcal{L}_k) = \sum_{i=1}^{k} \ell(\theta; x_i, y_i) = \sum_{i=1}^{k} \log P(y_i | x_i, \theta). \tag{4.6}
\]

hence the MLE can be rewritten as

\[
\hat{\theta}_k = \arg \max_{\theta} \sum_{i=1}^{k} \ell(\theta; x_i, y_i). \tag{4.7}
\]

Doing this maximization usually involves the computation of the stationary points of the log-likelihood, which requires calculating \( \nabla_\theta \ell(\theta; \mathcal{L}_k) = \sum_{i=1}^{k} \nabla_\theta \ell(\theta; x_i, y_i) \) For models assumed in (A0), each of the derivations in the summation is equal to the score function defined as the gradient of the joint log-likelihood:

\[
\nabla_\theta \ell(\theta; x, y) = \nabla_\theta \log P(y|x, \theta) = \nabla_\theta \log P_q(x, y|\theta), \tag{4.8}
\]

Equation (4.8) implies that the score will be the same no matter whether we choose the training or test distribution as our marginal. Furthermore, under regularity conditions (A6), the score is always a zero-mean random variable\(^2\).

Finally, using MLE to estimate \( \hat{\theta}_k \), class label of a test sample \( x_{\text{test}} \) will be predicted in the prediction step as the class with the highest log-likelihood value:

\[
\hat{y}(x) = \arg \max_y \ell(\hat{\theta}_k; x, y). \tag{4.9}
\]

4.1.2.2 Fisher Information

We have already defined Fisher information matrix in (1.15). Here we give some other properties of this criterion. More detailed descriptions about this well-known criterion can be found in various textbooks, such as [99].

\(^2\)Score function is actually zero-mean even under weaker regularity conditions.
Recall that Fisher information of a parametric distribution is a measure of information that the samples generated from that distribution provide regarding the parameter. It owes part of its importance to the Cramér-Rao Theorem (see Appendix D.2, Theorem D.10), which guarantees a lower-bound for the covariance of the parameter estimators.

Fisher information is defined as the expected value of the outer-product of the score function with itself, evaluated at some $\theta$. In our classification context, using (1.15) we can write the training and test Fisher information matrices with respect to $P_q$ and $P_o$:

$$I_q(\theta) := \mathbb{E}_q \left[ \nabla_{\theta} \log P_q(x, y|\theta) \cdot \nabla_{\theta}^\top \log P_q(x, y|\theta) \right]$$  \hspace{1cm} (4.10a)

$$I_o(\theta) := \mathbb{E}_o \left[ \nabla_{\theta} \log P_o(x, y|\theta) \cdot \nabla_{\theta}^\top \log P_o(x, y|\theta) \right]$$  \hspace{1cm} (4.10b)

Here, we focus on $I_q$ to further explain Fisher information criterion. Our descriptions here can be directly generalized to $I_o$ as well. First, note that from equation (4.8) and that the score function is always zero-mean, one can reformulate the definition as:

$$I_q(\theta) = \mathbb{E}_q \left[ \nabla_{\theta} \ell(\theta; x, y) \cdot \nabla_{\theta}^\top \ell(\theta; x, y) \right] = \text{Cov}_q [\nabla_{\theta} \ell(\theta; x, y)]$$  \hspace{1cm} (4.11)

Under the differentiability conditions (A6), it is easy to show that we can also write the Fisher information in terms of the Hessian matrix of the log-likelihood:

$$I_q(\theta) = -\mathbb{E}_q \left[ \nabla_{\theta}^2 \ell(\theta; x, y) \right]$$  \hspace{1cm} (4.12)

Recall that the subscript $q$ in equations (4.11) and (4.12) indicates that the expectations are taken with respect to the joint distribution that uses $q(x)$ as the marginal, that is $P(x, y|\theta) = q(x)P(y|x, \theta)$. Expansion of the expectation in (4.11) results

$$I_q(\theta) = \mathbb{E}_q(x) \left[ \mathbb{E}_{y|x, \theta} \left[ \nabla_{\theta} \ell(\theta; x, y) \cdot \nabla_{\theta}^\top \ell(\theta; x, y) \right] \right]$$

$$= \int_{x \in X} q(x) \left[ \sum_{y=1}^c P(y|x, \theta) \cdot \nabla_{\theta} \ell(\theta; x, y) \cdot \nabla_{\theta}^\top \ell(\theta; x, y) \right] dx$$  \hspace{1cm} (4.13)
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

4.1.2.3 Some Properties of MLE

In this section, we formalize some of the key properties of MLE, which make this estimator popular in various fields. They are also very useful in the theoretical analysis of FIR, provided in the next section. More detailed descriptions of these properties, together with the proofs that are skipped here, can be found in different sources, such as [100] and [99].

Note that a full understanding of the properties described in this section requires the knowledge of different modes of statistical convergence, specifically, convergence in probability (\( P \rightarrow \)), and convergence in law (\( L \rightarrow \)). A brief overview of these concepts are given in Appendix D.

**Theorem 4.1** ([99], Theorem 5.1). If the assumptions (A0) to (A7) hold, then there exists a sequence of solutions \( \left\{ \hat{\theta}_k \right\}_{k=1}^{\infty} \) to \( \nabla_{\theta} \ell(\theta; L_k) = 0 \) that converges to the true parameter \( \theta_0 \) in probability.

Note that Theorem 4.1 does not imply that convergence holds for any sequence of MLEs. Hence, if there are multiple solutions to equation \( \nabla_{\theta} \ell(\theta; L_k) = 0 \) (the equation to solve for finding the stationary points) for every \( k \), it is not obvious which root to select as \( \hat{\theta}_k \) to sustain the convergence. Therefore, while consistency of the MLE is guaranteed for models with a unique root of the score function evaluated at \( L_k \), it is not trivial how to build a consistent sequence when multiple roots exist. Here, in order to remove this ambiguity, we assume that either the roots are unique, become asymptotically unique, or we have access to an external procedure guiding us to select the proper roots so that \( \hat{\theta}_k \xrightarrow{P} \theta_0 \). We will denote the selected roots the same as \( \hat{\theta}_k \) from now on.

**Theorem 4.2** ([99], Theorem 5.1). Let \( \hat{\theta}_k \) be the maximum likelihood estimator based on the training data set \( L_k \). If the assumptions (A0) to (A8) hold, then the MLE \( \hat{\theta}_k \) has a zero-mean normal asymptotic distribution with the covariance equal to the inverse Fisher information matrix, and with the convergence rate of \( 1/2 \):

\[
\sqrt{n}(\hat{\theta}_k - \theta_0) \xrightarrow{L} \mathcal{N}(0, I_q(\theta_0)^{-1}) \tag{4.14}
\]

Theorems 4.2 and Cramér-Rao bound (see Appendix D), together with the consistency assumption, i.e. \( \hat{\theta}_k \xrightarrow{P} \theta_0 \), imply that MLE is an asymptotically efficient estimator with the efficiency equal to the training Fisher information. One can rewrite (4.14) as

\[
\sqrt{n} \cdot I_q(\theta_0)^{1/2}(\hat{\theta}_k - \theta_0) \xrightarrow{L} \mathcal{N}(0, I_d) \tag{4.15}
\]
In the following corollary, we see that if we substitute $I_q(\theta_0)$ with $I_q(\hat{\theta}_k)$, the new sequence still converges to a normal distribution:

**Corollary 4.3** ([100], Theorem 9.18). *Under the assumptions of Theorem 4.2, we get*

$$\sqrt{n} \cdot I_q(\hat{\theta}_k)^{1/2}(\hat{\theta}_k - \theta_0) \xrightarrow{L} N(0, I_d)$$

(4.16)

### 4.1.3 Establishing FIR as an Upper-bound

In this section, we give our main theoretical analysis to relate FIR to the asymptotic distribution of the parameter log-likelihood ratio. Using the established relationship, we then show that FIR can be viewed as an asymptotic upper-bound of the expected variance of the loss function.

Recall that the estimated parameter $\hat{\theta}_k$ is obtained from a given proposal distribution $q(x)$. The log-likelihood ratio function, at a given pair $(x, y)$, is defined as:

$$\ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y).$$

(4.17)

This ratio can be viewed as an example of the classification loss function whose expectation with respect to the test joint distribution of $x$ and $y$, results in the discrepancy between the true conditional $\mathbb{P}(y|\theta_0, x)$ and MLE conditional $\mathbb{P}(y|\hat{\theta}_k, x)$ [101]. Here, we analyze this measure asymptotically as $(n \to \infty)$. Primarily, note that based on continuity of the log-likelihood function (A6) and consistency of MLE (Theorem 4.1), equation (4.17) converges in probability to zero for any $(x, y)$.

Furthermore, equation (4.17) is dependent on both the true marginal $p(x)$ (through the test pairs, where it should be evaluated) and the proposal marginal $q(x)$ (through the MLE $\hat{\theta}_k$). In the classification context, [2] claimed that the expected value of this ratio with respect to both marginals converges to $\text{tr}[I_q(\theta_0)^{-1} I_q(\theta_0)]$. In the scalar case, $\text{tr}[I_q(\theta_0)^{-1} I_q(\theta_0)]$ is equal to the ratio of the Fisher information of the true and proposal distributions, the reason why it is sometimes referred to as the Fisher information ratio [3]. This objective have been widely studied in linear and non-linear regression problems [77, 78, 101, 79, 1]. However, it is not as fully analyzed in classification.

[2] and many papers following them [4, 3, 5], used this function as an *asymptotic* objective in active learning to be optimized with respect to the proposal $q$. Here, we show that this objective can also be viewed as an *upper bound* for the expected variance of the asymptotic distribution of (4.17).

First, we investigate the asymptotic distribution of the log-likelihood ratio in two different cases:
Theorem 4.4. If the assumptions (A0) to (A8) hold, then, at any given \((x, y) \in \mathcal{X} \times \{1, \ldots, c\}\):

(I) In case \(\nabla_{\theta} \ell(\theta_0; x, y) \neq 0\), the log-likelihood ratio follows an asymptotic normality with convergence rate equal to \(1/2\). More specifically

\[
\sqrt{n} \cdot \left( \ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y) \right) \xrightarrow{L} \mathcal{N} \left( 0, \text{tr} \left[ \nabla_{\theta} \ell(\theta_0; x, y) \cdot \nabla_{\theta}^\top \ell(\theta_0; x, y) \cdot \mathbf{I}_q(\theta_0)^{-1} \right] \right). \tag{4.18}
\]

(II) In case \(\nabla_{\theta} \ell(\theta_0; x, y) = 0\) and \(\nabla_{\theta}^2 \ell(\theta_0; x, y)\) is non-singular, the asymptotic distribution of the log-likelihood ratio is a mixture of first-order Chi-square distributions, and the convergence rate is one. More specifically:

\[
n \cdot \left( \ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y) \right) \xrightarrow{L} \sum_{i=1}^{d} \lambda_i \cdot \chi_i^2 \tag{4.19}
\]

where \(\lambda_i\)’s are eigenvalues of \(\mathbf{I}_q(\theta_0)^{-1/2} \nabla_{\theta}^2 \ell(\theta_0; x, y) \mathbf{I}_q(\theta_0)^{-1/2}\).

Proof. Due to assumptions (A0) to (A7), Theorem 4.2 holds and therefore we have \(\sqrt{n} \cdot (\hat{\theta}_k - \theta_0) \xrightarrow{L} \mathcal{N}(0, \mathbf{I}_q(\theta_0)^{-1})\). The rest of the proof is based on the Delta method in the two modes described in Appendix D (Theorems D.7 and D.8):

(I) \(\nabla_{\theta} \ell(\theta_0; x, y) \neq 0\):

Since the expected log-likelihood function, evaluated at a given pair \((x, y)\), is assumed to be continuously differentiable (A6) and that \(\nabla_{\theta} \ell(\theta_0; x, y) \neq 0\), we can apply Theorem D.7 to \(\ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y)\) to write:

\[
\sqrt{n} \cdot \left( \ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y) \right) \xrightarrow{L} \mathcal{N} \left( 0, \nabla_{\theta}^\top \ell(\theta_0; x, y) \cdot \mathbf{I}_q(\theta_0)^{-1} \cdot \nabla_{\theta} \ell(\theta_0; x, y) \right), \tag{4.20}
\]

where the scalar variance can also be written in a trace format.

(II) \(\nabla_{\theta} \ell(\theta_0; x, y) = 0\) and \(\nabla_{\theta}^2 \ell(\theta_0; x, y)\) non-singular:

In this case, the conditions in Theorem D.8 are satisfied (with \(\Sigma = \mathbf{I}_q(\theta_0)^{-1}\) and \(g(\theta) = \ell(\theta; x, y)\)), and therefore we can directly write (4.19) from equations (D.6).

\(\square\)

Theorem 4.4 regards the log-likelihood ratio (4.17) evaluated at any arbitrary pair \((x, y)\). Note that if we consider the training pairs in \(\mathcal{L}_k\), which are used to obtain \(\hat{\theta}_k\), it is known that the
The equality holds when the set of pairs

\[ \ell(\hat{\theta}_k; L_k) - \ell(\theta_0; L_k) \xrightarrow{L} \frac{1}{2} \chi^2_1 \]  

(4.21)

Theorem 4.4 implies that variance of the asymptotic distribution of the log-likelihood ratio in case (I) is \( \text{tr} [\nabla_{\theta} \ell(\theta_0; x, y) \cdot \nabla_{\theta}^T \ell(\theta_0; x, y) \cdot I_q(\theta_0)^{-1}] \), whereas in case (II), from Theorem D.8 (see Appendix D), the variance is \( \frac{1}{2} \|I_q(\theta_0)^{-1/2} \nabla_{\theta} \ell(\theta_0; x, y) I_q(\theta_0)^{-1/2}\|_F^2 \). Therefore, it is evident that the variance of the log-likelihood ratio at any \((x, y)\) is reciprocally dependent on the training Fisher information. From this point of view, one can set the training distribution such that it leads to a Fisher information that minimizes this variance. Unless the parameter and hence the Fisher information is univariate, it is not clear what objective to optimize with respect to \( q \) such that the resulting Fisher information minimizes the variance. In the next theorem, we show that the Fisher information ratio, \( \text{tr} \left[I_q(\theta_0)^{-1} I_0(\theta_0)\right] \), is a reasonable candidate objective to minimize in order to get a training distribution \( q \) for the multivariate case:

**Theorem 4.5.** If the assumptions (A0) to (A8) hold, then:

\[
\mathbb{E}_q \left[ \text{Var}_q \left( \lim_{k \to \infty} \sqrt{k} \cdot [\ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y)] \right) \right] \leq \text{tr} \left[I_q(\theta_0)^{-1} I_0(\theta_0)\right]. 
\]  

(4.22)

The equality holds when the set of pairs \((x, y)\) where we have zero score function at \( \theta_0 \), i.e. \( \nabla_{\theta} \ell(\theta_0; x, y) = 0 \), has measure zero under the test joint distribution \( \mathbb{P}(x, y | \theta_0) \) in \( X \times \{1, \ldots, c\} \).

**Proof.** Note that, from Theorem 4.4, when \( \nabla_{\theta} \ell(\theta_0; x, y) = 0 \) the convergence rate of the log-likelihood ratio is one and therefore it is of \( O_p \left( \frac{1}{\sqrt{k}} \right) \). Thus, in this case we have \( \sqrt{k} \cdot [\ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y)] = O_p \left( \frac{1}{\sqrt{k}} \right) \) and it converges to zero in probability (and in law). Now, define the region \( R_0 \subseteq X \times \{1, \ldots, c\} \) by

\[
R_0 := \{(x, y)| \nabla_{\theta} \ell(\theta_0; x, y) = 0\}
\]  

(4.23)

Variance of the asymptotic distribution of \( \sqrt{k} \cdot [\ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y)] \), considering both cases \( \nabla_{\theta} \ell(\theta_0; x, y) = 0 \) (with probability \( \mathbb{P}(R_0 | \theta_0) \)) and \( \nabla_{\theta} \ell(\theta_0; x, y) \neq 0 \) (with probability \( 1 - \mathbb{P}(R_0 | \theta_0) \)), can be written as:

\[
\text{Var} \left( \lim_{k \to \infty} \sqrt{k} \cdot [\ell(\hat{\theta}_k; x, y) - \ell(\theta_0; x, y)] \right) \\
= [1 - \mathbb{P}(R_0 | \theta_0)] \cdot \text{tr} [\nabla_{\theta} \ell(\theta_0; x, y) \cdot \nabla_{\theta}^T \ell(\theta_0; x, y) \cdot I_q(\theta_0)^{-1}] + \mathbb{P}(R_0 | \theta_0) \cdot 0 \\
\leq \text{tr} [\nabla_{\theta} \ell(\theta_0; x, y) \cdot \nabla_{\theta}^T \ell(\theta_0; x, y) \cdot I_q(\theta_0)^{-1}] 
\]  

(4.24)
Taking the expectation of both sides with respect to the true joint, gives the inequality (4.22). If the set of pairs \((x, y)\) where \(\nabla_{\theta} \ell(\theta_0; x, y) = 0\) form a zero measure set under \(P(x, y | \theta_0)\), then \(P(R_0 | \theta_0) = 0\) and we get equality in (4.24) and hence an equality in (4.22).

Theorem 4.5 implies that minimizing the Fisher information ratio with respect to \(q\), is indeed the upper-bound minimization of the expected variance of the asymptotic distribution of the log-likelihood ratio.

### 4.2 Fisher Information Ratio in Practice

In this section, we explain how inequality (4.22) can be utilized in practice as an objective function for active learning. The left-hand-side of this inequality is an inference-based performance metric but optimizing this metric is intractable and FIR-based methods approximate it by its upper-bound minimization. As FIR is a function of the query distribution \(q\) (not the query set \(Q\)), querying can be done in a probabilistic way as is described in (1.7):

\[
q^* = \arg\min_q \text{tr}[I_q(\theta_0)^{-1} I_\circ(\theta_0)]
\]  
\begin{equation}
\text{such that} \quad \int_{\mathcal{X}} dq(x) = 1,
\end{equation}
\begin{equation}
q(x) \geq 0 \quad \forall x \in \mathcal{X}
\end{equation}
\begin{equation}
Q = \{x_1, \ldots, x_k\} \sim q^*(x)
\end{equation}

As mentioned, \(q^*\) is an upper-bound minimization of the expected asymptotic loss variance. Moreover, there are a number of unknown variables involved in FIR, such as \(I_\circ\) and \(\theta_0\). In practice, estimations of these unknown variables are used in the optimization process for active learning. Therefore, although the derivations in the previous section (Theorem 4.5) are made based on one querying of infinitely many samples, in practice a finite-sample approximation of the objective is used in the iterative Algorithm 3.1. As the number of iterations increases, the parameter estimate gets more accurate and so does the approximate FIR objective. In the this section, we show that under certain assumptions the optimization with respect to the proposal distribution in each iteration is yet another upper-bound minimization similar to (4.22). More specifically, Remark 4.6 in Section 4.2.1 shows that although the proposal distribution is optimized separately in each iteration of an FIR-based
active learning algorithm, minimizing the approximate FIR at each iteration is still an upper-bound minimization of the original cost function (i.e. left-hand-side of (4.22)).

As opposed to the previous section, here we assume that a set of learning materials is available from the previous section which enables us to have an estimate of the parameter. We denote this estimate by $\hat{\theta}_{k'}$ where $k'$ is the size of the samples labeled by the current learning iteration. We use the available parameter estimate to approximate FIR objective and select queries to be labeled for the next iteration.

In what follows, we show that this approximation can still be viewed as an upper-bound for the log-likelihood loss function. Moreover, we discuss further techniques that simplify using FIR as a practical querying objective. The main difficulties consist of (1) having unknown variables in the objective, such as the test marginal, $p(x)$, and the true parameter, $\theta_0$, and (2) lack of closed form for Fisher information matrices for most cases. In the next two sections, we review different hacks and solutions that have been proposed to resolve these issues.

### 4.2.1 Replacing $\theta_0$ by $\hat{\theta}_{k'}$

Since $\theta_0$ is not known, the simplest idea is to replace it by the current parameter estimate, that is $\hat{\theta}_{k'}$ [1, 3, 4, 5, 6]. Clearly, as the algorithm keeps running the iterations ($k'$ increases), the approximate objective (which contains $\hat{\theta}_{k'}$ instead of $\theta_0$) gets closer to the original objective. This is due to the regularity and invertibility conditions assumed for the log-likelihood function and Fisher information matrices, respectively. Moreover, [6] analyzed how this approximation effects the querying performance in finite-sample case.

Their analysis is done only for pool-based active learning, and when the test marginal $p(x)$ is a uniform distribution $\mathcal{U}(x)$ over the pool $X$. It is also assumed that the Hessian $\frac{\partial^2 l(\theta; x, y)}{\partial \theta^2}$ is independent of the class labels $y$, and therefore can be viewed as the conditional Fisher information $I(\theta, x)$ (that is $I(\theta) = \mathbb{E}_{p(x)}[I(\theta, x)]$). Furthermore, there assumed to exist four positive constants $L_1, L_2, L_3, L_4 \geq 0$ such that the following four inequalities hold for all $x \in X$, $y \in \{1, \ldots, c\}$ and
\[ \nabla \ell(\theta_0; x, y) \mathbf{I}_o(\theta_0)^{-1} \nabla \ell(\theta_0; x, y) \leq L_1 \] (4.26)

\[ \| \mathbf{I}_o(\theta_0)^{-1/2} \mathbf{I}(\theta_0, x) \mathbf{I}_o(\theta_0)^{-1/2} \| \leq L_2 \]

\[ \| \mathbf{I}_o(\theta_0)^{-1/2} (\mathbf{I}(\theta', x) - \mathbf{I}(\theta'', x)) \mathbf{I}_o(\theta_0)^{-1/2} \| \leq L_3 (\theta' - \theta'')^\top \mathbf{I}_o(\theta_0) (\theta' - \theta'') \]

\[ -L_4 \| \theta - \theta_0 \|_2 \mathbf{I}(\theta_0, x) \leq \mathbf{I}(\theta, x) - \mathbf{I}(\theta_0, x) \leq L_4 \| \theta - \theta_0 \|_2 \mathbf{I}(\theta_0, x) \]

where \( \theta' \) and \( \theta'' \) are any two parameters in a fixed neighborhood of \( \theta_0 \). Then, provided that \( k' \) is large enough, the following remark can be shown regarding the relationship between the FIRs computed at \( \theta_0 \) and an estimate \( \hat{\theta}_{k'} \):

**Remark 4.6.** Let the assumptions ((A0)) to ((A8)) and those in (4.26) hold. Moreover, assume that the Hessian is independent of the class labels. If \( k' \) is large enough, then the following inequality holds for any \( \beta \geq 10 \) with high probability:

\[ \text{tr} \left[ \mathbf{I}_q(\theta_0)^{-1} \mathbf{I}_o(\theta_0) \right] \leq \frac{\beta + 1}{\beta - 1} \cdot \text{tr} \left[ \mathbf{I}_q(\hat{\theta}_{k'})^{-1} \mathbf{I}_o(\hat{\theta}_{k'}) \right] \] (4.27)

The minimum value for \( k' \) that is necessary for having this inequality with probability \( 1 - \delta \), increases quadratically with \( \beta \) and reciprocally with \( \delta \) [6, Lemma 2].

**Proof.** It is shown in the proof of Lemma 2 in [102] that under assumptions mentioned in the statement, the following inequalities hold with probability \( 1 - \delta \):

\[ \frac{\beta - 1}{\beta} \mathbf{I}(x, \theta_0) \leq \mathbf{I}(x, \hat{\theta}_{k'}) \leq \frac{\beta + 1}{\beta} \mathbf{I}(x, \theta_0). \] (4.28)

Taking expectation with respect to \( p(x) \) and \( q(x) \) result:

\[ \frac{\beta - 1}{\beta} \mathbf{I}_p(\theta_0) \leq \mathbf{I}_p(\hat{\theta}_{k'}) \leq \frac{\beta + 1}{\beta} \mathbf{I}_p(\theta_0) \] (4.29a)

\[ \frac{\beta - 1}{\beta} \mathbf{I}_q(\theta_0) \leq \mathbf{I}_q(\hat{\theta}_{k'}) \leq \frac{\beta + 1}{\beta} \mathbf{I}_q(\theta_0) \] (4.29b)

Since \( \mathbf{I}_q(\theta_0) \) and \( \mathbf{I}_q(\hat{\theta}_{k'}) \) are assumed to be positive definite, we can write (4.29b) in terms of inverted matrices\(^3\):

\[ \frac{\beta}{\beta + 1} \mathbf{I}_q^{-1}(\theta_0) \leq \mathbf{I}_q^{-1}(\hat{\theta}_{k'}) \leq \frac{\beta}{\beta - 1} \mathbf{I}_q^{-1}(\theta_0) \] (4.30)

\( ^3 \)For any two positive definite matrices \( A \) and \( B \), we have that \( A \geq B \Rightarrow A^{-1} \leq B^{-1} \).
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

Now considering the first inequalities of (4.29a) and (4.30), multiplying both sides and taking the trace result (4.27).

Inequality (4.27) implies that minimizing $\text{tr}\left[I_q(\hat{\theta}_{k'})^{-1} I_\phi(\hat{\theta}_{k'})\right]$ (or an approximation of it) with respect to $q$ in each iteration of FIR-based querying algorithms, namely through the operation $A(X, \hat{\theta}_{k'}, k)$ (line 3 of Algorithm 3.1), is equivalent to upper bound minimization of the original cost function, i.e. left-hand-side of (4.22).

4.2.2 Monte-Carlo Approximation

Computation of Fisher information matrices is intractable unless when the marginal distributions are very simple or when they are restricted to be PMFs over finite number of samples. The latter is widely used in pool-based active learning, when the samples in the pool are assumed to be generated from $p(x)$. In such cases, one can simply utilize a Monte-Carlo approximation to compute $I_\phi(\theta)$. More specifically, denote the set of observed instances in the pool by $X$. Then the test Fisher information at any $\theta \in \Omega$ can be approximated by

$$I_\phi(\theta) \approx \tilde{I}(\theta; X) := \frac{1}{|X|} \sum_{x \in X} \sum_{y=1}^c \mathbb{P}(y|x, \theta) \nabla_\theta \ell(\theta; x, y) \nabla^\top_\theta \ell(\theta; x, y) + \delta \cdot \mathbb{I}_d$$

(4.31)

where $\delta$ is a small positive number and the weighted identity matrix is added to ensure positive definiteness. It is important to remark that when using equation (4.31), we are actually utilizing some of the test samples in the training process, hence we cannot use those in $X$ in order to evaluate the performance of the trained classifier.

Similarly, $I_q(\hat{\theta}_{k'})$ can be estimated based on a candidate query set $Q$. Let $Q$ be the set of samples drawn independently from $q(x)$. Then we can have the approximation $I_q(\hat{\theta}_{k'}) \approx \tilde{I}(\hat{\theta}_{k'}; Q)$. Putting everything together, the best query set $Q \subseteq X$ is chosen to be the one that minimizes the approximate FIR querying objective:

$$\text{tr}\left[\tilde{I}(\hat{\theta}_{k'}; Q)^{-1} \tilde{I}(\hat{\theta}_{k'}; X)\right].$$

(4.32)

In contrast to FIR in (4.2), this objective is directly written in terms of $Q$, and therefore the queries can be deterministically determined as in deterministic querying in (1.6).
4.2.3 Bound Optimization

There are other types of approximation methods occurring in the optimization side. These methods are able to remove part of the unknown variables by doing upper-bound minimization or lower-bound maximization. Recall that in probabilistic active learning, the querying objective is to be optimized with respect to $q$ (or $Q$ in deterministic querying). In a very simple example, when $d = 1$, note that the $I_q(\theta)$ is a constant scalar in (4.25a) and hence can be ignored. Hence, in the scalar case, we can simply focus on maximizing the training Fisher information. In the multivariate case, though it is not clear what measure of $I_q(\hat{\theta}_k)$ to optimize, one may choose the objective to be $|I_q(\hat{\theta}_k)|$ (where $|\cdot|$ is the determinant function), or $\text{tr}[I_q(\hat{\theta}_k)]$. The latter is worth paying more attention due to the following inequality [103]:

$$
\text{tr}
\begin{bmatrix} I_q(\hat{\theta}_k')^{-1} I_\phi(\hat{\theta}_k') \end{bmatrix}
\leq
\text{tr}
\begin{bmatrix} I_q(\hat{\theta}_k')^{-1} \end{bmatrix}
\cdot
\text{tr}
\begin{bmatrix} I_\phi(\hat{\theta}_k') \end{bmatrix}.
$$

(4.33)

Since $\text{tr}[I_\phi(\hat{\theta}_k')]$ is a constant with respect to $q$, minimizing the right-hand-side of inequality (4.22) can itself be approximated by another upper-bound minimization:

$$
\arg\min_q \text{tr}
\begin{bmatrix} I_q(\hat{\theta}_k')^{-1} \end{bmatrix}.
$$

(4.34)

This helps removing the dependence of the objective to the test distribution. A lower bound can also be established for the FIR. Using the inequality between arithmetic and geometric means of the eigenvalues of $I_q(\hat{\theta}_k')^{-1} I_\phi(\hat{\theta}_n')$, one can see that $d \cdot |I_q(\hat{\theta}_k')^{-1}| \cdot |I_\phi(\hat{\theta}_n')| \leq \text{tr}
\begin{bmatrix} I_q(\hat{\theta}_k')^{-1} I_\phi(\hat{\theta}_k') \end{bmatrix}$. Hence, when minimizing the upper-bound by minimizing the trace of $I_q(\hat{\theta}_k')^{-1}$, one should be careful about the determinant of this matrix as a term influencing the lower-bound of the objective.

In practice, of course, the minimization in (4.34) can be difficult due to matrix inversion. Thus, sometimes it is further approximated by

$$
\arg\max_q \text{tr}[I_q(\hat{\theta}_k')].
$$

(4.35)

Hence, algorithms that aim to maximize $\text{tr}[I_q(\hat{\theta}_k')]$, indeed introduce three layers of objective approximations through equations (4.33) to (4.35). As discussed before, the dependence of the objectives in all the layers (in either (4.34) or (4.35)), can be removed by replacing it with the current

---

4 Similar to D-optimality in Optimal Experiment Design [77].
5 Similar to A-optimality in Optimal Experiment Design [77].
Table 4.1: Reviewed FIR-based active learning algorithms for discriminative classifiers

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1]</td>
<td>(4.35)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>[2]</td>
<td>(4.35)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[3]</td>
<td>(4.32)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[4, 5]</td>
<td>(4.32)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>[6]</td>
<td>(4.27)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

estimate $\theta_{k'}$.

4.3 Some Existing Algorithms

In this section, we discuss several existing algorithms for implementing the query selection task based on minimization of FIR. Each of these algorithms is a different realization of the querying method $\mathcal{A}(X, \theta_{k'}, k)$ in Algorithm 3.1. They include both deterministic and probabilistic methods. The former group optimizes the Monte-Carlo approximation of FIR in (4.32) with respect to $Q$, whereas the latter group solves one of the optimizations in (4.34) or (4.35). We will analyze these algorithms below in context of our unifying framework presented in Sections 4.1 and 4.2, sorted according to date of their publication.

Table 4.1 lists the algorithms that we reviewed here, together with a summary of their properties and the approximate objective that they optimize for querying. Note that among these algorithms, the one by [6] makes extra assumptions as is described in Section 4.2.1. In addition to these listed properties, all these algorithms are clearly cost-constrained and inference-based querying methods.

4.3.1 Algorithm by [1]

This probabilistic querying algorithm is the classification version of the parametric active learning proposed by [1] for regression problem. The assumption is that the proposal belongs to a parametric family and is of the form $q(x; \alpha)$, where $\alpha$ is the parameter vector of the family. Thus, optimizing with respect to the proposal distribution reduces to the optimization with respect to the parameter $\alpha$. Because of parametric representation of the query distribution, this algorithm is more suitable for synthetic active learning.

This algorithm optimizes the objective in (4.35) to obtain the parameter of the query distribution. From the definition of Fisher information in (4.13), we can simply rewrite optimization (4.35)
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

Algorithm 4.1: [1]

Inputs: The current parameter estimate $\hat{\theta}_{k'}$, the batch size $k$

Outputs: The query set $Q$

/* Parameter optimization as is described in (4.36) */
1 $\alpha \leftarrow \arg \max_{\alpha} E_{q(x;\alpha)} \left[ \sum_{y=1}^{c} \mathbb{P} \left( y \mid x, \hat{\theta}_{k'} \right) \nabla_{\theta} \ell(\hat{\theta}_{k'}; x, y) \nabla_{\theta} \ell(\hat{\theta}_{k'}; x, y) \right]$
/* Sampling from the parametric proposal */
2 $x_i \sim q(x; \alpha), i = 1, \ldots, k$
3 return $Q = \{x_1, \ldots, x_k\}$

as:

$$\arg \max_{\alpha} \text{tr} \left[ E_{q(x;\alpha)} \left[ E_{y|x,\theta} \left[ \nabla_{\theta} \ell(\hat{\theta}_{k'}; x, y) \cdot \nabla_{\theta} \ell(\hat{\theta}_{k'}; x, y) \right] \right] \right]$$ (4.36)

More specifically, the new parameter vector is obtained by maximizing the expected contribution of the queries $X_q$ generated from $q(x; \alpha)$ to this objective. The two-step procedure of generating queries from parametric query distribution is shown in Algorithm 4.1. This algorithm can be used in both sequential and batch modes by changing the number of samples drawn from $q(x; \alpha)$. As is mentioned, this algorithm is not pool-based although it could be constrained to do so with parameter $\alpha$ being the probability atoms of the probability mass function $q$ defined over the pool.

4.3.2 Algorithm by [2]

Similar to the parametric active learning method described above, [2] also developed a querying method based on the maximization in ((4.35)) but in a deterministic framework. Specifically, they proposed solving (4.35) under the simplifying assumption that the proposal distribution is a uniform distribution. They developed their algorithm specifically under using binary logistic regression classifier. Here, we discuss their formulation for a general discriminate framework.

For choosing a single query $x^*$, the uniformity assumption for $q$ leads to an optimization similar to 4.36 but without the expectation:

$$x^* = \arg \max_{x \in X} \sum_{y=1}^{c} \mathbb{P} \left( y \mid x, \hat{\theta}_{k'} \right) \nabla_{\theta} \ell(\hat{\theta}_{k'}; x, y) \cdot \nabla_{\theta} \ell(\hat{\theta}_{k'}; x, y)$$ (4.37)

The resulting single-step deterministic approach is shown in Algorithm 4.2, where the feature set
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

Algorithm 4.2: [2]

Inputs: The feature set $\mathcal{X}$, the current parameter estimate $\hat{\theta}_{k'}$

Outputs: The query set $Q$

1. $x^* \leftarrow \arg\max_{x \in \mathcal{X}} \sum_{y=1}^c \mathbb{P}(y|x, \hat{\theta}_{k'}) \nabla_\theta^\top \ell(\hat{\theta}_{k'}; x, y) \nabla_\theta \ell(\hat{\theta}_{k'}; x, y)$

2. return $Q = \{x^*\}$

$\mathcal{X}$ can be either finite (hence pool-based querying) or uncountable (hence synthetic querying). In case of finite (discrete) feature set, the optimization in (4.37) is just a sorting problem, and in case of uncountable (continuous) feature set it is a general optimization problem that may be convex or non-convex depending on the classification model.

4.3.3 Algorithm by [3]

Inspired by the method proposed by [2], [3] employed Fisher information ratio to develop a pool-based active learning, which can be used in either sequential or batch querying. The unlabeled samples in the pool are assumed to be drawn from the test marginal $p(x)$. Therefore, the Fisher information matrix of the test distribution can be approximated by Monte-Carlo simulation over the samples in $X$, meaning $\mathbf{I}(\hat{\theta}_{k'}; X)$. Moreover, since the queries in a given candidate query set $Q \subseteq X$ are similarly assumed to be draws from $q(x)$, Fisher information of the training distribution can also be approximated with Monte-Carlo simulation for any $\theta \in \Omega$:

$$
I_q(\theta) = \mathbf{I}(\hat{\theta}_{k'}; Q) = \frac{1}{k} \sum_{x \in Q} \sum_{y=1}^c \mathbb{P}(y|x, \theta) \nabla_\theta^\top \ell(\theta; x, y) \nabla_\theta \ell(\theta; x, y) + \delta \cdot \mathbb{I}.
$$

(4.38)

Thus, since we do have an approximation of both Fisher information matrices, the deterministic querying objective to minimize with respect to $Q$ is chosen to be in the form of (4.32) and the optimization would be

$$
\arg\min_{Q \subseteq X, |Q| = k} \text{tr} \left[ \mathbf{I}(\hat{\theta}_{k'}; Q)^{-1} \mathbf{I}(\hat{\theta}_{k'}; X) \right]
$$

(4.39)

For $k = |Q| > 1$, the optimization (4.39) is NP-hard and [3] approximated it using sequential forward query selection, that is iteratively adding samples with lowest objective to the query set. The steps are shown in Algorithm 4.3.
4.3.4 Algorithms by [4] and [5]

The algorithms proposed by [4] and [5] are very similar to the one developed by [3] described above, except that they use a more sophisticated optimization method for solving 4.39. Their method shown in Algorithm 4.4, is different from Algorithm 4.3 mainly in the way that it greedily chooses a query at each inner loop iteration of the querying. While Algorithm 4.3 exclusively considers the contribution of each \( x \in X \), ignoring the samples selected in the previous iterations (hence \( \hat{I} (\theta_{k'}; x) \) in line 3 of Algorithm 4.3), Algorithm 4.4 takes into account all the queries chosen so far (hence \( \hat{I} (\theta_{k'}; Q \cup \{x\}) \) in line 3 in Algorithm 4.4).

[4] and [5] showed that when using binary logistic regression classifier, the minimization in (4.39) is equivalent to maximizing a submodular set function with respect to the query set \( Q \). This allowed them to use the well-known iterative algorithm for submodular maximization explained in the last chapter (see Algorithm 3.2), which guarantees a tight lower-bound for maximization problems (see Theorem 3.6). Here, we show that such equivalent holds for any discriminative classifier.

The following lemma shows that (4.39) is approximately equivalent to maximizing a simplified set function, for any unlabeled sample pool \( X \):

**Lemma 4.7.** Let \( Q \) and \( X \) be two finite subsets of samples randomly generated from \( p(x) \) and its resample distribution \( q(x) \), respectively, and \( Q \subset X \). Also let the parameter \( \delta \geq 0 \) in (4.31) is a small constant. If assumptions (A0), (A4), (A6) and (A8) hold, then the following optimization problems are approximately equivalent for some function \( g_\theta : X \times \{1, \ldots, c\} \times X \rightarrow \mathbb{R}^+ \), \( d \)-dimensional non-zero
Algorithm 4.4: [4, 5]

**Inputs:** The unlabeled pool $X$, the current parameter estimate $\hat{\theta}_k$, the batch size $k$

**Outputs:** The query set $Q$

```plaintext
/* Initializing the query set */
1 $Q \leftarrow \emptyset$
/* The loop for greedy batch querying */
2 for $j = 1 \rightarrow k$ do
   /* Query optimization */
3 $x^* = \text{arg min}_{x \in X} \text{tr} \left[ \hat{I}(\hat{\theta}_k; Q \cup \{x\})^{-1} \hat{I}(\hat{\theta}_k; X) \right]$
   /* Add the selected query into the query set */
4 $Q \leftarrow Q \cup \{x^*\}$
   /* Remove the selected instance from the pool */
5 $X \leftarrow X - \{x^*\}$
6 return $Q$
```

vector $v_\theta$ depending on $x$ and $y$, and for all $\theta \in \Omega$:

$$
(i) \quad \text{arg min}_{Q \subseteq X, |Q|=k} \text{tr} \left[ \hat{I}(\theta; Q)^{-1} \hat{I}(\theta; X) \right]
$$

$$
(ii) \quad \text{arg max}_{Q \subseteq X, |Q|=k} \sum_{x \in X-Q} \sum_{y=1}^{c} \frac{-1}{\delta \cdot \|v_\theta(x, y)\|^{-2} + \sum_{x' \in Q} g_\theta(x, y, x')} 
$$

The approximation is more accurate for smaller $\delta$ and well-conditioned Monte-Carlo approximation of proposal Fisher information matrix.

The proof can be found in Appendix F.

**Theorem 4.8.** Suppose $f_\theta : 2^X \rightarrow \mathbb{R}$ is defined as:

$$
f_\theta(Q) = \sum_{x \in X-Q} \sum_{y=1}^{c} \frac{-1}{\delta \cdot \|v_\theta(x, y)\|^{-2} + \sum_{x' \in Q} g_\theta(x, y, x')}, \quad \forall Q \subseteq X
$$

with $v_\theta$ a $d$-dimensional vector depending on $x$ and $y$, and $g_\theta$ defined in (F.10). Then $f_\theta$ is a submodular and monotone (non-decreasing) set function for all $\theta \in \Omega$.

The proof is in Appendix G. The result above, together with Lemma 4.7, imply that the objective of (4.40b) is a monotonically increasing set function with respect to $Q$. In Algorithm 4.4, the inner loop (lines 2 to 5) implements the minimization in (4.39) greedily. We have seen above that
this set minimization is approximately equivalent to maximizing a submodular and monotone set maximization, which, in turn, is shown to be efficient.

4.3.5 Algorithm by [6]

This algorithm uses FIR for doing a probabilistic pool-based active learning. It has extra assumptions in comparison to our general framework, which are briefly explained in Section 4.2.1. Note that these assumptions are to be made as well as those listed in Section 4.1.1. In such settings, [6] gave a finite-sample theoretical analysis for FIR when applied to pool-based active learning.

More specifically, suppose \( p(x) \) is a uniform PMF and \( q(x) \) is a general PMF, both defined over the pool \( X \). Using the notations in (4.26), the training Fisher information can be written as

\[
I_q(\hat{\theta}_{k'}) = \sum_{x \in X} q(x) I(\hat{\theta}_{k'}, x). \tag{4.42}
\]

Minimizing the last term in (4.42) with respect to PMF \( \{q(x) | x \in X \} \) is equivalent to a semidefinite programming after introducing a set of auxiliary variables \( t_j, j = 1, \ldots, d \) and applying Schur complements [104]:

\[
\arg \min_{q(x), x \in X} \sum_{j=1}^{d} \sigma_j t_j \quad \text{such that} \quad \begin{bmatrix} t_j & u_j^\top \\ u_j & \sum_{x \in X} q(x) I(\hat{\theta}_{k'}, x) \end{bmatrix} \succeq 0,
\]

\[
\sum_{x \in X} q(x) = 1.
\]

The steps for this querying method is shown in Algorithm 4.5. Note that the solution to (4.43) is slightly modified by mixing it with the uniform distribution over the pool \( U_X(x) \). Such modification is mainly to establish their theoretical derivations. The mixing coefficient, \( 0 \leq \lambda \leq 1 \) reciprocally depends on the number of queries. More specifically, [6] made it equal to \( 1 - \frac{1}{k^{1/6}} \). That is, as the number of queries increases, \( \lambda \) shrinks and so does the modification. Furthermore, in their analysis,
they assumed that sampling from $\tilde{q}(x)$ (line 3 of Algorithm 4.5) is done with replacement. That is, label of a given sample might be queried multiple times.

4.4 Fisher Information Ratio vs. Mutual Information and entropy

Let us now give a brief comparison between the FIR querying objective and the two other objectives that are discussed in previous chapters, entropy $f_H$ and MI $f_{MI}$ (see Section 3.2). The goal of these two objectives is mainly to select a subset of queries which contain the largest amount of information about class labels of the unlabeled pool, hence they are prediction-based surrogate objectives and more suitable for pool active learning. On the other hand, FIR-based querying aims to propose a query distribution which results in the highest information regarding parameter of the underlying data distribution, hence it is an inference-based surrogate objective that is more appropriate for synthetic querying although it can also be used in pool-based applications.

As is explained in Chapter 3, entropy $f_H$ of class labels of unlabeled samples has the drawback of not considering the interaction between the samples due to independence of labels given the parameter (a common scenario in many discriminative classifiers). We discussed that such deficiency can lead to redundancy of information within the query set and therefore sub-optimal performance in batch active learning. Mutual information resolves this issue in the expense of adding higher computational complexities and therefore needs to be approximated multiple times to become tractable. Fisher information ratio, when being used in pool-based scenario, is not broken into multiple functions of individual samples and hence does not suffer from the issue of redundancy. Moreover, with careful formulation of the optimization and considering the proposal as a PMF over the pool, the querying optimization can be written as a convex optimization with respect to the atomic
Table 4.2: Computational complexity of different querying algorithms (recall that $n = |X|$, $k = |Q|$, $\Omega \subseteq \mathbb{R}^d$ and $c = \#$ classes)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy (3.5)</td>
<td>$O(ncd)$</td>
</tr>
<tr>
<td>Mutual Information (3.28a)</td>
<td>$O((nc)^k)$</td>
</tr>
<tr>
<td>Approximate Mutual Information (3.28b)</td>
<td>$O(n^k)$</td>
</tr>
<tr>
<td>Algorithm 4.2 (by [2])</td>
<td>$O(ncd)$</td>
</tr>
<tr>
<td>Algorithm 4.3 (by [3])</td>
<td>$O(nk(cd + d^3))$</td>
</tr>
<tr>
<td>Algorithm 4.4 (by [4, 5])</td>
<td>$O(nk(cd + cdk + d^3))$</td>
</tr>
<tr>
<td>Algorithm 4.5 (by [6])</td>
<td>$O(d^3n^2 + d^4n + d^5)$</td>
</tr>
</tbody>
</table>

probabilities of the proposal (as is explained in Section 4.3.5 and will be later discussed in Section ). Nevertheless, as we have seen in Section 4.2, due to presence of unknown variables in the original formulation of FIR, it still needs to be approximated to be used in practice.

Although entropy and MI functions are prediction-based objectives that aims to improve the performance in predicting class labels, there is no guarantee that having a low entropy objective, or high mutual information, gives an accurate label prediction. For example, the model may have a low entropy around a set of wrong labels and therefore while the entropy objective shows small uncertainty, the actual prediction performance of the model can be very poor. Whereas, we saw that FIR is shown to asymptotically upper-bound an inference-based performance metric (i.e. log-likelihood loss function) and therefore if we have small value for FIR, this performance metric (in asymptotic sense) will also be small. Although, such inequality gets looser when we approximate the FIR objective with finite-sample precision and further bound optimizations.

Computational complexity of an FIR-based querying depends on what type of approximations are being used. Table 4.2 summarizes computational complexity different algorithms shown in Section 4.3 and compares it with entropy and MI-based approaches. We denote size of the finite unlabeled pool by $n$. The algorithm by [1] is excluded from this table since it cannot be used in pool-based sampling. Also the complexity reported for mutual information is reported from Remark 3.8 in Section 3.5. Entropy-based and the algorithm by [2] have the lowest complexity, but in the expense of introducing redundancy into the batch of queries. Algorithms by [3], [4, 5] and [6] become very expensive when $d$ is large, whereas mutual information can easily get intractable for
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

selecting batches of higher size (large $k$). Observe that algorithm by [4, 5] is more expensive than [3]. Recall that despite similarities in appearance, the former guarantees tight bound for its greedy optimization, whereas the latter does not.

The complexity for the algorithm by [6] is computed assuming that a barrier method (following path) is used as its numerical optimization [105]. From Table 4.2, this algorithm is the only one whose complexity increases quadratically with size of the pool $n$, and therefore can get significantly slow for huge pools. Furthermore, it does not depend on $k$ since the optimization in (4.43) as its main source of computation, only depends on $n$ and $d$ (computing $I(\hat{\theta}_{k'}, x)$ is assumed to cost $O(1)$ for each $x \in X$ as it is taken to be independent of $y$).

4.5 Our Practical Querying Algorithm

In this section, we focus on a pool-based scenario and develop a general practical FI-based batch active learning. As the reviewed algorithms, we replace $\theta_0$ by $\hat{\theta}_{k'}$, which we showed can be justified by Remark 4.6. Also in order to remove the dependence on $P_0$ through $I_0$, we use an upper-bound minimization to remove instead of applying uniformity assumption to $p(x)$ as the existing algorithms. That is, we make use of inequality (4.33) to eliminate the dependence of our objective to $I_0(\hat{\theta})$, but we remove the second layer of approximation by solving the minimization (4.34) which is closer to the original optimization rather than the maximization (4.35). The most similar existing algorithm to our framework is the one proposed by [6] (Algorithm 4.5), however our method does not make restrictive uniformity assumptions over the test or proposal distribution or the form of Fisher information matrices (except positive definiteness). Our proposed algorithm is compared with the existing algorithms that have been reviewed in the previous section in Figure 4.1. The algorithms are categorized in synthetic and pool-based groups. In the former $X$ is not finite and possibly uncountable. In the synthetic category of Figure 4.1, $X$ is assumed to be the real axis, with the test and proposal marginals, $p(x)$ and $q(x)$, respectively, being any two arbitrary PDFs. This category includes two classic algorithms [1, 2] that do not make any restrictive assumptions on marginal distributions and differ in their optimizations. On the other hand, pool-based algorithms assume that $X$ is finite and therefore $q(x)$ is a PMF. The first sub-group of algorithms in this category (Algorithms 4.3 and 4.4) enforced an additional assumption that $q(x)$ is a uniform PMF over the candidate query set $Q$. This assumption makes the objective a set function of the query set $Q$ and thus these algorithms are examples of deterministic querying methods. Whereas the most recent existing algorithm in pool-based category relaxed this assumption and performed probabilistic querying by
optimizing probability masses in \( q(x) \). Although, by definition of pool-based algorithms, there is no restriction on the test marginal \( p(x) \), all these existing algorithms assumed that \( p(x) \) is a uniform PMF over the finite pool \( X \). As is shown in Figure 4.1, we advance the pool-based direction by relaxing the uniformity assumption over the test marginal and proposed a FIR-based querying algorithm where \( p(x) \) could be any PDF over the feature set \( \mathcal{X} \). Because of this relaxation, we cannot approximate \( I_\phi \) anymore and use bound optimization to remove it from the objective (similar to [11]).

Recall that based on our notation in this chapter, we are given a current parameter estimate \( \hat{\theta}_k \) and aim to choose a query set \( Q \subseteq X \) which minimizes Fisher information ratio (or through any other approximate optimization). To simplify the notations, we represent the unlabeled pool element-wise as \( X = \{ x_1, \ldots, x_n \} \). For each sample \( x_i \in X \) we define the average outer product of the log-likelihood derivative (score function) as

\[
A_i = \sum_{j=1}^c \mathbb{P}(y_i = j|\hat{\theta}_k; x_i) \cdot \nabla_\theta \ell(\hat{\theta}_k; x_i, j) \nabla_\theta^\top \ell(\hat{\theta}_k; x_i, j)
\] (4.44)

Then, the Fisher information of the proposal distribution defined in (4.10a), with a given PMF \( q(x) \) over \( X \), can simply be written as

\[
I_q(\hat{\theta}_k) = \sum_{i=1}^n q_i A_i,
\] (4.45)

where \( q_i = q(x_i) \). Hence the optimization in our querying strategy will be

\[
\arg\min_{q_1, \ldots, q_n} \text{tr} \left[ \left( \sum_{i=1}^n q_i A_i \right)^{-1} \right]
\] (4.46)

such that \( q_i \geq 0 \), \( i = 1, \ldots, n \)

\[
\sum_{i=1}^n q_i = 1
\]

### 4.5.1 Semidefinite Programming

Next, we show that the optimization in (4.46) can be reformulated as a semidefinite programming [104]. First, note that the objective can be expressed in terms of a matrix-vector multiplication as \( \sum_{j=1}^d e_j^\top \left( \sum_{i=1}^n q_i A_i \right)^{-1} e_j \), where \( e_j \) is a \( d \)-dimensional canonical vector with value 1 in the \( j \)-th component and zero everywhere else. Then, introducing \( d \) scalar auxiliary variables
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

\( t_1, \ldots, t_d \) yields the following reformulation of the optimization

\[
\arg\min_{t_1, \ldots, t_d, q_1, \ldots, q_n} \quad t_1 + \cdots + t_d
\]

such that

\[
\sum_{i=1}^{n} q_i = 1, \quad q_i \geq 0, \quad i = 1, \ldots, n,
\]

\[
\begin{align*}
\mathbf{e}_1^\top (\sum_i q_i \mathbf{A}_i)^{-1} \mathbf{e}_1 & \leq t_1 \\
\mathbf{e}_2^\top (\sum_i q_i \mathbf{A}_i)^{-1} \mathbf{e}_2 & \leq t_2 \\
& \vdots \\
\mathbf{e}_d^\top (\sum_i q_i \mathbf{A}_i)^{-1} \mathbf{e}_d & \leq t_d
\end{align*}
\]

Now, using Schur complement to transform the non-linear set of inequalities in (4.47) into a linear matrix inequality, we get the following optimization:

\[
\arg\min_{t_1, \ldots, t_d, q_1, \ldots, q_n} \quad t_1 + \cdots + t_d
\]

such that

\[
\sum_{i=1}^{n} q_i = 1,
\]

\[
\bigoplus_{j=1}^{d} \left[ \begin{array}{c|c}
\sum_i q_i \mathbf{A}_i & \mathbf{e}_j \\
\hline
\mathbf{e}_j^\top & t_j
\end{array} \right] \succeq 0,
\]

where \( \bigoplus \) is the direct sum of matrices such that the \( i^{\text{th}} \) matrix argument is the \( i^{\text{th}} \) block of the block diagonal constraint matrix. These blocks are associated with the non-linear constraints in (4.47).

Using barrier method (path-following) as an interior-point method for solving (4.47) [105], the computational complexity of each optimization iteration will be \( O(n^2 d^3 + nd^4 + d^5) \). Hence, solving the semidefinite programming of the form (4.48) can be slow for large \( d \) or \( n \). Hence, in order to reduce the computational complexity we apply uncertainty filtering, as is described in Section 3.6, before solving 4.48.

The pseudocode of our querying method is summarized in Algorithm 4.6. This algorithm, like all the other algorithms reviewed in the previous sections, is a proposed function for being used as the module \( \mathcal{A}(X, \hat{\theta}_k, k) \) in the general classification active learning Algorithm in 3.1 (with \( \hat{\theta}_k \) a different notation for \( \hat{\theta}^{(k)} \)). Our proposed algorithm takes an extra input argument \( 1 \leq \beta \leq n \).
Algorithm 4.6: Our proposed FIR-based practical algorithm

**Inputs:** The unlabeled pool $X$, the current parameter estimate $\theta_{k'}$, the batch size $k$,
the parameter for uncertainty sampling $\beta$

**Outputs:** The query set $Q$

```plaintext
/* Uncertainty sampling */
1 $X_f \leftarrow \arg \max_{U \subseteq X, |U| = \beta} \sum_{u \in U} f_H(\{u\})$
/* Forming the average outer product of score functions in (4.44) */
2 $A_i \leftarrow \mathbb{E}_{y|\theta_{k'},x_i} [\nabla_{\theta} \ell(\theta; x, y) \nabla^\top_{\theta} \ell(\theta; x, y)]$
/* Solving the SDP */
3 $q_i = q(x_i) \leftarrow \text{solution to (4.48) for all } x_i \in X_f$
/* Sampling from the obtained proposal distribution */
4 $x_i \sim q(x), i = 1, \ldots, k$
5 return $Q = \{x_1, \ldots, x_k\}$
```

which is the parameter of uncertainty filtering. Because of running this uncertainty sampling in the
beginning of the algorithm, the queries will be chosen from $X_f$ only ($Q \subseteq X_f$), that is support of
the proposal will be $X_f$ and not $X$. Furthermore, in line 4, due to sampling with replacement, we
might get repeated samples within the selected query set, in that case we simply remove the repetitions
before requesting their labels. Therefore, it is possible to get fewer than $k$ queries as the output of the
algorithm.

### 4.6 Experimental Results

We show the results of the proposed algorithm over synthetic and real data sets and compare
them with the results of random and entropy-based querying benchmarks, and also the pool-based
querying strategy developed by [4] (labeled by Hoi et al. in the figures). The latter algorithm
can be thought of as a generalized pool-based version of two other algorithms developed in [2]
and [3]. The other close algorithm by [1] is developed for synthetic scenarios and therefore is not
comparable. In all the experiments, we used a logistic regression classifier to model the posterior
$P(y|\theta, x)$ and also set the uncertainty selection set size as $\beta = 100$. We ran SDPT3 solver of the
CVX package [106, 107] to solve our semidefinite programming.
4.6.1 Synthetic Data

We generated 10 Monte-Carlo simulations of a 2-dimensional data set with two classes, where each class is sampled from a mixture of 10-component Gaussian distributions, and sampled 1000 samples from each class \((n = 2000)\). We constructed 10 random test/training partitions (with size ratio 3/7) and also randomly selected three samples from the training partition of each class as the initial labeled data set \(L^{(0)}\). Then, each querying algorithm is run over the 10 data partitions iteratively. At each iteration the selected batch of queries with their true labels are added into the labeled data set and the classification accuracy is computed over the test partition.

The average and standard deviation of the accuracies are shown in Figure 4.2. One can observe that for small sizes of batch, \(k = 1, 5\), our algorithm is close to entropy-based querying although with smaller variance. However, for larger query batches (\(k = 20\)) the performance of entropy-based querying decayed. This behaviour of entropy-based querying was expected, since the presence of redundancy among the entropy-based queries becomes more crucial in larger batches. Moreover, the algorithm developed by [4] shows a lower average accuracy than our proposed algorithm in case of all the batch sizes.

4.6.2 Bach Choral Harmony

We used the “Bach Choral Harmony Data Set” [97], where the class labels are the resonating chords assigned to given sequences with measured pitches (with 12-dimensional feature vectors). We selected the samples of the four most frequent chords in this data set: D-major, G-major, C-major and F-major; resulting a set of 1869 samples. We randomly partitioned this data for 25 times into test/training parts. The initial labeled data set for each partition is constructed by randomly selecting 30 samples from each chord class of training partition. In each experiment, we reduced the dimensionality of the features to \(d = 5\) using PCA over the training partition.

The results for running the algorithm with different batch sizes are shown in Figure 4.3. Again for small batches (\(k = 1\)) our algorithm is close to entropy-based querying. However, for all the other sizes our algorithm outperformed others. Also similar to the synthetic data, the algorithm by [4] is inferior to ours for all values of \(k\). This can be because of the uniform assumption for the marginal made by [4] that cannot efficiently capture the distribution of the queries.
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

4.6.3 MNIST

As the second real data, we used the MNIST database of handwritten digits [96]. This data set consists of 20 × 20 images of handwritten digits from 1 to 9, and with the test and training parts stored separately. Here, we considered all pairs of the digits (36 combinations), and evaluated the querying algorithms over the binary classification on each combination. For each combination, we constructed 10 initial labeled data sets each of which is a random selection of 10 samples from the training part.

Figure 4.4 summarizes our results over this data set for different iterations. Figures 4.4(a) to Figures 4.4(e) show the total average of accuracies for various batch sizes across all experiments for different pair combinations of digits (36 combinations, together with 10 different initialized labeled data set for each, made total 360 accuracy values for each iteration). Here, our algorithm outperformed random querying and the algorithm by [4] for all $k$ values. Similar to the previous data sets, we see that its performance is close to entropy-based querying for the small sizes of query batches ($k = 1$ and $k = 5$), but gets better for larger batches.

In order to statistically compare our proposed algorithm with the competing methods, we ran one-sided $t$-tests over the difference of the accuracy values of our algorithm and the rest at each iteration. The null hypothesis $H_0$ of the test is that the mean of the difference is less than or equal to zero (similar to (3.30)), that is our proposed algorithm worked no better than the competing method. Therefore, the smaller the test’s $p$-value is, the stronger the evidence we have for rejecting the null hypothesis or equivalently having a significant improvement in our method. Figure 4.4(f) shows the total average $p$-values over all the experiments for each batch size (that is across all the digit combinations, different initial labeled data sets and all the querying iterations). We observe that while the average $p$-values associated with Ours vs. random and Ours vs. Hoi et al. are almost constant and small for all the batch sizes, it is decreasing in case of Ours vs. entropy. This is in accordance with our observation in the average accuracy plots, implying that our querying method does not show significance improvement in comparison with entropy-based querying for small $k$ values, but shows significant improvement for larger sizes.

Figure 4.5 shows the frequency of null hypothesis rejection with confidence level of 0.95 for the largest batch size $k = 30$. Recall that the null hypothesis $H_0$ was the hypothesis that our algorithm does not outperform the competing method. Hence more frequent rejection of $H_0$ in a given querying iteration implies better performance of our algorithm in comparison with the competing method. Figure 4.5 indicates that the numbers of rejections for entropy-based benchmark
and the algorithm developed by [4] are higher during initial iterations and they decrease as the number of queries increases. On the other hand, the number of rejections for the tests against the random benchmark is lower in the initial iterations, but it increases with $k$. This is in accordance with Figure 4.4(e), where in early iterations the differences between the average accuracies of the proposed method and the entropy and Hoi et al methods are large (higher number of null hypothesis rejections), but they gradually converge to a close accuracy (lower number of rejections in late iterations). On the other hand, we observe a reverse pattern for the random benchmark.

Finally, the average $p$-values for each combination of digits for $k = 3$ are also shown in Figures 4.6(a) to 4.6(c). The $(i, j)$-th element of each lower-triangular matrix indicates the average $p$-value of the $t$-test that is done for the binary classification of digits $i$ and $j$ in MNIST data set. The averaging for each combination is done over all the querying iterations of the competing querying algorithms. As in the previous figures, smaller average $p$-values (darker cells) in Figure 4.6 means a more significant outperformance of our proposed algorithm in comparison with the competing methods.


### Synthetic

\[ x_i \sim \int_X q(x) \, dx \] \[
\text{x}_{\text{test}} \sim \int_X p(x) \, dx
\]

arg min_q \, \text{tr} \left[ I_q(\theta_{k'})^{-1} I_q(\theta_{k'}) \right] \[ Algorithm 4.1 \, [1] \]

arg max_q \, \text{tr} \left[ I_q(\hat{\theta}_{k'}) \right] \[ Algorithm 4.2 \, [2] \]

### Pool-based

\[ x_i \sim \int_X q(x) \, dx \] \[
\text{x}_{\text{test}} \sim \int_X p(x) \, dx
\]

arg min_q \, \text{tr} \left[ I_{U(Q)}(\hat{\theta}_{k'})^{-1} I_{U(X)}(\hat{\theta}_{k'}) \right] \[ Algorithms 4.3, 4.4 \, [4, 42, 5] \]

\[ x_i \sim \int_X q(x) \, dx \] \[
\text{x}_{\text{test}} \sim \int_X p(x) \, dx
\]

arg min_q \, \text{tr} \left[ I_q(\hat{\theta}_{k'})^{-1} I_{U(X)}(\hat{\theta}_{k'}) \right] \[ Algorithm 4.5 \, [6] \]

\[ x_i \sim \int_X q(x) \, dx \] \[
\text{x}_{\text{test}} \sim \int_X p(x) \, dx
\]

arg min_q \, \text{tr} \left[ I_q(\hat{\theta}_{k'})^{-1} \right] \[ Algorithm 4.6 \, (Ours) \]

---

Figure 4.1: Different FIR-based querying algorithms with their objectives and assumptions over the data distributions.
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

Figure 4.2: Average and standard variation of accuracies on synthetic data set.

Figure 4.3: Average and standard variation of accuracies on Bach Choral Harmony data set.
CHAPTER 4. FI-BASED CLASSIFICATION ACTIVE LEARNING

Figure 4.4: Summary of the results of evaluating the querying algorithms over different pairs of the digits in MNIST data set for different batch sizes.

Figure 4.5: The frequency of rejecting the null hypothesis (in percentage) with confidence level 0.95 of our t-tests is shown in different querying iteration when using $k = 30$. The percentage is reported across all combination of pairs of digits and randomly initialized experiments: 36 combinations and 10 initial labeled sets, leading to 360 total experiments.
Figure 4.6: The results of evaluating the querying algorithms over different pairs of the digits in MNIST data set for $k = 30$. Figures 4.6(a), 4.6(b) and 4.6(c) show the average $p$-values for each pair of digits, averaged over all the iterations and randomly initialized experiments.
Chapter 5

Conclusion

To conclude this thesis, we first summarize our contribution in developing practical methods and deriving theoretical results in analyzing active learning algorithms that use information theoretic surrogates as the querying objectives. Then, we give recommendations as for future directions to enrich the field of active learning from different points of view including the choice of objectives, applications or optimization methods.

5.1 Contributions

If the queries to be labeled by an oracle is not selected intelligently enough, the number of necessary labels required for training a machine learning algorithm will be unnecessarily large. As opposed to passive learning where there is no control over building the labeled data set, in active learning the goal is to select those queries whose labeling has the lowest cost and at the same time leads to a high-performance algorithm. Using appropriate performance metrics and cost functions, one can then formalize this querying with an optimization problem. In this thesis, under certain assumptions, we showed that the learning cost is proportional to the size of the query set. Then, formalizing active learning as an iterative process, we solve a constrained optimization problem at each iteration which optimizes a surrogate objective given that size of the query set is fixed to a positive integer.

In the preceding chapters, we focused on two learning problems and used three information theoretic functions as surrogate objectives to be optimized in each learning iteration. First, in Chapter 2, we set entropy function as our surrogate objective to develop a Edgewise Active Learning (EAL) algorithm for constrained binary clustering. That is, we query pairwise relationship between
particular samples in an attempt to reduce the uncertainty about cluster assignment of the samples. This querying algorithm starts with an initial result (for example through running an unconstrained clustering). Then, it uses density-weighted entropy function to choose a sample that is located in a crowded region and has the most uncertain cluster assignment. We ask whether it should fall in the same cluster with two other points, each of which is the most confident sample in the given clustering result. We hope to reduce the uncertainty about the clustering of the first point, by coupling it with two confident samples, which are also in crowded regions. We intentionally tend to choose samples from populated regions, since it is faster to propagate information provided about them to other samples. The proposed active constrained clustering has been applied to image segmentation and showed that with certain speed-up techniques it scales up and can be used with large images.

We then switch to classification problems where discriminative classifiers are used to model the posterior distribution with a parametric function. We discussed that when doing batch active learning (that is selecting multiple queries) for such models, entropy function has a crucial drawback as the querying objective. Specifically, entropy of a set of class labels given the posterior’s parameter breaks into entropy of individual samples and hence choosing a set of queries with maximum joint entropy yields a set of samples with highest individual entropies. In other words, selecting multiple queries with entropy maximization might lead to choosing samples with overlapping or redundant information, and therefore it can be shrunk without significant information loss.

For classification problem, we use two more sophisticated information theoretic function, that is mutual information (MI) between class labels and Fisher information (FI). MI is very close to entropy function with an additional term that can take into account the interaction between samples, and hence resolve the redundancy issue. It does so in expense of computational complexity. Maximizing this objective is NP-hard, and even evaluating MI for non-singleton set of class labels can be intractable. In Chapter 3, we developed a practical active learning algorithm that approximates MI greedily and utilizes randomized and deterministic submodular maximization to approximate the original MI optimization. Our experimental results on several real data sets show that MI-based querying can improve the entropy-based querying, especially in case of batch active learning.

Finally, we analyzed classification active learning based on Fisher information in Chapter 4. Fisher information (FI) is a matrix for multivariate models and therefore cannot be optimized directly. Most of the previous FI-based querying algorithms have used a scalar function of FI called Fisher information ratio (FIR). Minimizing this function tends to select queries that are most informative with respect to the posterior’s parameter. Using this function as the objective does not suffer from redundancy issue, and it can be easy to optimize if formalized carefully. Focusing on the same
objective, we show a novel theoretical result on asymptotically relating FIR with log-likelihood loss function as a common performance metric for inference step. Our result is general and holds for synthetic as well as pool-based active learning. Furthermore, it is different than the previously proposed theoretical justifications of FIR and, if under certain circumstances, it still holds even after finite-sample approximation. Based on our framework, minimizing FIR is equivalent to upper-bound minimization of the expected variance of the log-likelihood loss function (in asymptotic sense). Using this framework, we unify the previous FI-based classification active learning algorithms. At the end, we proposed a new practical algorithm for querying by minimizing FIR with fewer approximations and assumptions in comparison with the existing algorithms. The experimental results over synthetic and real data sets showed outperformance of our proposed algorithms over entropy-based and some FI-based existing algorithms.

5.2 Recommendations

In this section, we recommend several research paths for further enriching the field of active learning in areas that have not been analyzed as much as others.

- In this thesis, we have discussed about developing active learning algorithms for two specific learning problems. There are other problems in machine learning that may need interaction with the oracle to improve their performance. One area that is not explored too much in terms of actively supervising it is anomaly detection, that is discovering samples in a given data set that does no agree with the majority of data in terms of the underlying distribution. This problem was traditionally an unsupervised problem, however, recently there have been attempts to make them semi-supervised \[108\]. Semi-supervised algorithms need labeled queries and therefore active learning can be applied to such problems. One future path in active learning could be switching to this problem and develop an active learning method for a (semi)supervised (semi)supervised anomaly detector. Of course, here we might need to change our surrogate objectives to better fit the goals in the application.

- In our querying algorithms for discriminative classifiers, we assumed that the label \(y\) that the oracle gives for a sample \(x\) is a draw from \(P(y|\theta_0, x)\), that is the same parametric posterior that the classifier is using with the true parameter. However, this assumption might not be realistic in many applications. Developing a querying framework where this assumption is relaxed or turned into a less strict one would be an interesting research path.
As we discussed, it is not straightforward to set the performance metrics as our objective to optimize in active learning. The main reason is that these metrics cannot be evaluated unless the learning algorithm is built (updated) after paying to actually label the queries\(^1\). That is, the problem can be viewed as maximizing an *expensive* objective. Bayesian optimization seems to be an appropriate tool for solving such problem \([109]\). Another research path that can significantly enrich the field of active learning is to utilize Bayesian optimization for solving active learning problem, that is maximizing the performance metric with small number of observations.

\[^1\text{Even after updating the algorithm, one needs to know the distribution of test samples in order to accurately evaluate the performance. However, for sake of simplicity, let's assume that once the model is updated, we can use tricks such as cross-validation to estimate the performance metric.}\]
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


Appendix A

Affinity Update

The basic derivations of the affinity matrix update can be found in [26]. Here we give a more detailed explanation of them.

For a constraint set $\Omega = \bigcup_{(i,j)} \omega_{ij}$, the non-zero elements of the sparse matrix $M = M(Q, Y_Q; \epsilon)$ appeared in (2.4), are as follows:

$$m_{ij} = m_{ji} = \pm \frac{1}{\epsilon^2}, \quad m_{ii} = \frac{1}{\epsilon^2} \sum_{j=1, j \neq i}^{n} m_{ij},$$

where the plus and minus signs in the off-diagonal terms are associated with cannot- and must-links respectively. This characterization implies that $M = \sum_{q_{ij} \in Q} M(q_{ij}, y_{ij})$. Therefore, the update equation for $L$ can be written as a sequential updates for the individual constraints:

$$K = \left( K^{-1} + M_1 + M_2 + ... + M_k \right)^{-1}$$

$$= \left( K_1^{-1} + M_2 + M_3 + ... + M_k \right)^{-1}$$

$$\vdots$$

$$= \left( K_{k-1}^{-1} + M_k \right)^{-1},$$

where we associate each individual constraint with an index between 1 and $k = |Q|$. Here, we focus on one of the individual updates for a labeled pair $(q_{ij}, y_{ij})$. From the structure of $M(q_{ij}, y_{ij})$, it is
clear that there exists a permutation matrix such as $P \in \{0, 1\}^{n \times n}$ that can make it block diagonal:

$$M_p := P^T M (\omega_{ij}) P = \begin{bmatrix} \Gamma & 0 \\ 0 & 0 \end{bmatrix},$$

where $\Gamma = \frac{1}{\xi^2} \left[ \begin{array}{cc} 1 & \pm 1 \\ \pm 1 & 1 \end{array} \right]$. Also define

$$K_p := P^T K P = \begin{bmatrix} 2 & n-2 \\ n-2 & \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}.$$

Now, we show that if the update equation is written in terms of $M_p$ and $K_p$, $K$ can be computed efficiently. To do this, first we define $\overline{K}_p$ and relate it to $K$:

$$\overline{K}_p := (K_p^{-1} + M_p)^{-1} = \left[ P^T (K_p^{-1} + M_p) P \right]^{-1} = P^T K P.$$

In this derivation, we used the orthogonality of permutation matrices ($P^T P = PP^T = I_n$). Using the Woodbury inversion identity, we can expand the equation for $\overline{K}_p$ as:

$$\overline{K}_p = K_p - K_p (I_n + M_p K_p)^{-1} M_p K_p. \quad (A.1)$$

Now it’s easy to verify the following key point regarding the inversion that appeared in the expanded format:

$$(I_n + M_p K_p)^{-1} = \begin{bmatrix} I_2 + \Gamma K_{11} & \Gamma K_{12} \\ 0_{(n-2) \times 2} & I_{n-2} \end{bmatrix}^{-1} = \begin{bmatrix} (I_2 + \Gamma K_{11})^{-1} & -(I_2 + \Gamma K_{11})^{-1} \Gamma K_{12} \\ 0_{(n-2) \times 2} & I_{n-2} \end{bmatrix}.$$
Plugging this result into equation (A.1) for computing $\mathbf{K}_p$ yields:

$$\mathbf{K}_p = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{12}^T & \mathbf{K}_{22} \end{bmatrix} \left( \mathbf{I}_2 + \mathbf{\Gamma K}_{11} \right)^{-1} \mathbf{\Gamma} \begin{bmatrix} \mathbf{K}_{11}^T & \mathbf{K}_{12}^T \end{bmatrix}.$$

The inversion here operates on a $2 \times 2$ matrix, and hence is straightforward to compute. Note that the condition number of this matrix for a given constraint $(q_{ij}, y_{ij})$ is proportional to $1 + 2 \frac{1 + k_{ij}}{\epsilon^2}$ (plus when $y_{ij} = -1$ and minus when $y_{ij} = 1$). So one should be careful not to make $\epsilon^2$ too small such that the machine precision cannot handle inverting this ill-conditioned matrix. Moreover, the left and right matrix products, plus the necessary permutations for getting back to $\mathbf{K}$ have the complexity of $O(n)$, therefore the whole update can be done with $O(n)$ operations.
Appendix B

Orthogonal Iteration

The eigendecomposition process is the main bottleneck in scaling up spectral clustering. Accurate estimation of all eigenvalues of $L$ is very time consuming ($O(n^3)$) and often not practical, especially when working with images. We employed the so-called orthogonal iteration method to numerically estimate the first two eigenvectors. This algorithm along with its extensions are discussed in many sources such as [83, 81]. Here we explain it in terms of our notations.

For any estimated eigenpair $(\hat{v}, \hat{\lambda})$ of a symmetric matrix $L$, where $\hat{\lambda}$ is computed using the Rayleigh quotient, i.e. $\hat{\lambda} = \hat{v}^T L \hat{v} / \hat{v}^T \hat{v}$, the residual vector is defined as:

$$r := L \hat{v} - \hat{\lambda} \hat{v}. \quad (B.1)$$

Provided that $\|\hat{v}\|_2 = 1$ the eigenvalue estimation error is upper-bounded as below [81]:

$$|\lambda - \hat{\lambda}| \leq \|r\|_2. \quad (B.2)$$

Therefore the 2-norm of the residual vector can be used as a stopping criterion in iterative eigen-decomposition methods.

Algorithm B.1 shows different steps in orthogonal iteration. Except for the initialization step (lines 1 through 3), each iteration (lines 4 to the end) consists of two phases: (1) updating the estimations and (2) computing the 2-norm of the residuals.

The algorithm is initialized by a random orthonormal matrix $\hat{V}_0 \in \mathbb{R}^{n \times 2}$. At each iteration $\tau = 0, ..., \tau_{\text{max}}$, in the first phase, either one or both columns of $\hat{V}_\tau$ are left-multiplied by the $q$'th power of $L$ followed by orthonormalization to get $\hat{V}_{\tau+1}$ (CO in lines 7 and 8 stands for column-
APPENDIX B. ORTHOGONAL ITERATION

Algorithm B.1: Orthogonal iteration subroutine used in Spectral Clustering (SC) to estimate eigenvectors associated with the two largest eigenvalues of the Laplacian matrix.

Subroutine: Orthogonal Iteration

Inputs: Input matrix \( L \in \mathbb{R}^{n \times n} \), initialized orthonormal eigenvectors as columns of \( \hat{V}_0 \), residual 2-norm threshold \( \rho \), maximum number of iterations \( \tau_{\text{max}} \), power of multiplicand \( q \in \mathbb{Z}^+ \)

Outputs: Estimated invariant subspace basis vectors as columns of \( \hat{V}_\tau \)

1. \( \hat{A}_0 \leftarrow \text{diag}(\hat{V}_0^T L \hat{V}_0) \)
2. \( \hat{R}_0 \leftarrow L \hat{V}_0 - \hat{A}_0 \hat{V}_0 \)
3. \( \bar{u}_0 \leftarrow \begin{bmatrix} 1 & \mathbb{I}_{\mathbb{R}^n} > \rho(\|r_{01}\|) \\ 1 & \mathbb{I}_{\mathbb{R}^n} > \rho(\|r_{02}\|) \end{bmatrix} \)
4. \( \tau \leftarrow 0 \)
5. while \((\bar{u}_\tau^T 1_2 > 0) \land (\tau \leq \tau_{\text{max}})\) do
6. if \( \bar{u}_\tau^T 1_2 = 2 \) then
7. \( \tilde{V}_{\tau+1} \leftarrow \text{CO}(L^q \hat{V}_\tau) \)
else
8. \( \tilde{V}_{\tau+1} \leftarrow \text{CO}\left(\begin{bmatrix} L^q \hat{V}_\tau u_\tau, \hat{V}_\tau(1_2 - u_\tau) \end{bmatrix}\right) \)
9. \( \hat{A}_{\tau+1} \leftarrow \text{diag}(\tilde{V}_{\tau+1}^T L \tilde{V}_{\tau+1}) \)
10. \( \hat{R}_{\tau+1} \leftarrow L \tilde{V}_{\tau+1} - \hat{A}_{\tau+1} \hat{V}_{\tau+1} \)
11. \( \bar{u}_{\tau+1} \leftarrow \begin{bmatrix} 1 & \mathbb{I}_{\mathbb{R}^n} > \rho(\|r_{\tau+11}\|) \\ 1 & \mathbb{I}_{\mathbb{R}^n} > \rho(\|r_{\tau+12}\|) \end{bmatrix} \)
12. \( \tau \leftarrow \tau + 1 \)

orthonormalization). In our implementation we empirically fix \( q \) to 100. In the second phase, the Rayleigh quotient for each column of \( \tilde{V}_{\tau+1} \) is computed (line 9) followed by calculating the residuals \( r_{\tau+11} \) and \( r_{\tau+12} \) as defined in (B.1) and forming the residual matrix \( \hat{R}_{\tau+1} \). Finally computing the 2-norm of the residuals specifies if any of the iterating vectors has converged, so that it can stay fixed. The initialization only consists of phase (2) applied over the initial matrix \( \hat{V}_0 \).

Let us start from the second phase (lines 1-3 and 9-11). First, it implements equation (B.1) and then creates an indicator vector \( \bar{u}_{\tau+1} \) of length two whose components indicate whether the 2-norm of each individual column of \( \hat{R}_{\tau+1} \) is greater than a pre-specified threshold \( \rho \) (fixed to \( 10^{-5} \) in our implementation). There are three possibilities for the element-wise summation of this vector:
APPENDIX B. ORTHOGONAL ITERATION

(A1). $u_{\tau+1}^T L q \hat{V}_{\tau+1} = 0$ meaning that none of the columns in $\hat{V}_{\tau+1}$ has converged and both of them will be updated during phase (1) of the next iteration (line 7, both columns of the estimation multiplied by $L^q$).

(A2). $u_{\tau+1}^T L q \hat{V}_{\tau+1} = 1$ meaning that only one of the vectors has not yet converged, thus the algorithm keeps iterating on that, fixing the other one (line 8: just one column of the estimation multiplied by $L^q$).

(A3). $u_{\tau+1}^T L q \hat{V}_{\tau+1} = 0$ meaning that both vectors have already converged, so that the loop can be terminated.

The convergence rate can be shown to be proportional to $|\lambda_2/\lambda_1|$ which is small when the data has two well-separated partitions. Moreover, the major operations of each iteration consist of the matrix-vector product and the orthonormalization process for which Gram-Schmidt algorithm is used. Since the input matrix is sparse, the product $L^q \hat{V}_{\tau+1}$ has the complexity of order $O(nq)$. Performing Gram-Schmidt process on the product result with size $n \times 2$ takes $O(n)$ operations [83]. Therefore complexity of the whole algorithm will be $O(\tau_{\max}.n(q + 1))$ which is linear with respect to $n$. 

115
Appendix C

Update Equation for Multinomial Logistic Regression

In this section, we show that the update equations in equations (3.11) and (3.15), can be efficiently computed for a multinomial logistic regression. More specifically, we see that the gradient $g_{n+t+1}$ and the inverse Hessian $H^{-1}_{n+t+1}$ in iteration $t+1$, can be efficiently computed from $g_{n+t}$ and $H^{-1}_{n+t}$ in iteration $t$.

C.1 Multinomial Logistic Regression as a Discriminative Classifier

In a classification problem, with $c > 1$ denoting the number of classes, a multinomial logistic regression models the posterior distribution of the class labels given the feature vectors $x$ and a given $(d+1)(c-1)$-dimensional parameter vector $\theta = [\theta_1^\top, ..., \theta_{c-1}^\top]^\top$, where $\theta_i = [\alpha_i, \beta_i^\top]^\top$ (for $1 \leq i \leq c-1$), as the following [110]:

$$\begin{align*}
\mathbb{P}(y = j | x, \theta) &= \frac{e^{\alpha_j + \beta_j^\top x}}{1 + \sum_{t=1}^{c-1} e^{\alpha_t + \beta_t^\top x}}, j = 1, ..., c-1 \\
\mathbb{P}(y = c | x, \theta) &= \frac{1}{1 + \sum_{t=1}^{c-1} e^{\alpha_t + \beta_t^\top x}}
\end{align*}$$ (C.1)

Now for an indexed feature-label pair $(x_i, y_i = j)$ define

$$\pi_{ij} := \mathbb{P}(y_i = j | x_i, \theta), j = 1, ..., c.$$ (C.2)
APPENDIX C. UPDATE EQUATION FOR MULTINOMIAL LOGISTIC REGRESSION

Note that \( \pi_{ij} \), and equivalently the distributions shown in (C.1), are the likelihood functions when viewed from the point of view of parameter vector \( \theta \). The objective to optimize in order to find the maximum likelihood estimation (MLE), denoted by \( \hat{\theta}_n \), given an i.i.d set of training samples \( \mathcal{L} = \{(x_1^*, y_1^*), \ldots, (x_n^*, y_n^*)\} \), is

\[
\ell(\theta; \mathcal{L}) := \sum_{i=1}^{n} \mathbb{P}(y^*_i | x_i, \theta) = \sum_{i=1}^{n} \sum_{j=1}^{c} \mathbb{1}(y^*_i = j) \log \pi_{ij}.
\]

(C.3)

that is

\[
\hat{\theta}_n = \arg \max_{\theta} \ell(\theta; \mathcal{L}).
\]

The subscript \( n \) in \( \hat{\theta}_n \) is to emphasize the sample size of the training data using which the MLE is obtained.

C.2 Updating the Gradient Vector

Here, we formulate the gradient vector in terms of the individual log-likelihood functions \( \pi_{ij} \) and the feature vectors \( x_i \), which readily enables us to derive an update equation for the gradient. The \( k \)-th partial gradient (for \( 1 \leq k \leq c - 1 \)) of the log-likelihood evaluated at \((x_i, y_i^* = j)\) is:

\[
\nabla_k \log \pi_{ij} := \frac{\partial \log \pi_{ij}}{\partial \theta_k} = \left[ \mathbb{1}(j = k) - \pi_{ik} \right] \cdot \begin{bmatrix} 1 \\ x_i \end{bmatrix}
\]

(C.4)

where \( \mathbb{1}(\cdot) \) is the indicator function. Since \( 1 \leq k \leq c - 1 \), we always get \( \mathbb{1}(j = k) = 0 \) when \( j = c \).

From (C.4), we can write the \( k \)-th partial gradient of the total log-likelihood function \( \ell(\theta; \mathcal{L}) \):

\[
\nabla_k \ell(\theta; \mathcal{L}) = \sum_{i=1}^{n} \sum_{j=1}^{c} \mathbb{1}(y_i^* = j) \nabla_k \log \pi_{ij} = \sum_{i=1}^{n} \left[ \mathbb{1}(y_i^* = k) - \pi_{ik} \right] \cdot \begin{bmatrix} 1 \\ x_i \end{bmatrix}
\]

(C.5)

The complete gradient vector of the log-likelihood function \( \ell(\theta; \mathcal{L}) \), denoted by \( g_n \), is obtained by
APPENDIX C. UPDATE EQUATION FOR MULTINOMIAL LOGISTIC REGRESSION

concatenating the partial gradient vectors. It can be written compactly as below:

\[
g_n = \left[ \begin{array}{c}
\nabla_1 \ell(\theta; L) \\
\vdots \\
\nabla_{c-1} \ell(\theta; L)
\end{array} \right] = \sum_{i=1}^{n} \left[ \begin{array}{c}
\mathbb{1}(y_i^* = 1) - \pi_{i1} \\
\vdots \\
\mathbb{1}(y_i^* = c - 1) - \pi_{i1,c-1}
\end{array} \right] \otimes \left[ \begin{array}{c}
x_i \\
\end{array} \right],
\]

where \(\otimes\) denotes the Kronecker product. Equation (C.6) implies that the gradient is additive and the update equation, after adding a pair \((x_{u1}, y_{u1})\) to the training set \(L\) is simply equal to

\[
g_{n+1} = g_n + \left[ \begin{array}{c}
\mathbb{1}(y_{u1} = 1) - \pi_{u1,1} \\
\vdots \\
\mathbb{1}(y_{u1} = c - 1) - \pi_{u1,c-1}
\end{array} \right] \otimes \left[ \begin{array}{c}
x_{u1} \\
\end{array} \right].
\]

Similarly, \(g_{n+t}\) in equation (3.15) can be obtained by adding a single product to the gradient vector calculated in the previous iteration:

\[
g_{n+t} = g_{n+t-1} + \left[ \begin{array}{c}
\mathbb{1}(y_{ut} = 1) - \pi_{ut,1} \\
\vdots \\
\mathbb{1}(y_{ut} = c - 1) - \pi_{ut,c-1}
\end{array} \right] \otimes \left[ \begin{array}{c}
x_{ut} \\
\end{array} \right].
\]

C.3 Updating the Inverse Hessian Matrix

Let us first focus on calculating the Hessian matrix using the training set \(L\). From (C.5) and after doing some algebra, the partial second derivative of the log-likelihood function with respect to \(\theta_k\) and \(\theta_k\) (for \(1 \leq k, s \leq c - 1\)) is

\[
\nabla^2_{ss} \ell(\theta; L) := \frac{\partial^2 \ell(\theta; L)}{\partial \theta_k \partial \theta_s} = \sum_{i=1}^{n} \pi_{ik}(\pi_{is} - 1) \cdot \left[ \begin{array}{c}
x_i \\
\end{array} \right] \left[ \begin{array}{c}
x_i^\top \\
\end{array} \right] (C.9)
\]
APPENDIX C. UPDATE EQUATION FOR MULTINOMIAL LOGISTIC REGRESSION

The partial second derivative in (C.9) forms the \((s, k)\)-th block of the total Hessian matrix \(H_n\), which can be written in a compact manner as below:

\[
H_n = \sum_{i=1}^{n} \pi_i (\pi_i - 1)^\top \otimes \begin{bmatrix} 1 \\ x_i \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ x_i \end{bmatrix} \tag{C.10}
\]

where \(\pi_i = [\pi_{i1}, \ldots, \pi_{ic-1}]^\top\) and \(1\) is a \((c - 1)\)-dimensional vector with all the elements equal to 1. Therefore, the Hessian matrix also has an additive formulation. The new Hessian matrix \(H_{n+1}\) after adding \((x_{u1}, y_{u1})\) to the training set is equal to

\[
H_{n+1} = H_n + \pi_{u1} (\pi_{u1} - 1)^\top \otimes \begin{bmatrix} 1 \\ x_{u1} \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ x_{u1} \end{bmatrix}
= H_n + R_1. \tag{C.11}
\]

Note that the update term \(R_1\) in (C.11) is the Kronecker product of two rank-one matrices, implying that \(H_{n+1}\) is a rank-one perturbation of \(H_n\). Therefore, we can use Sherman-Morrison inversion formula to write the inverse Hessian update equation:

\[
H_{n+1}^{-1} = H_n^{-1} + \frac{H_n^{-1}R_1H_n^{-1}}{1 + \text{tr}[R_1H_n^{-1}]}. \tag{C.12}
\]

Hence, we do not need to explicitly perform a matrix inversion in order to update the inverse Hessian matrix. Similarly, for obtaining \(H_{n+t}^{-1}\) from \(H_{n+t-1}^{-1}\) that is needed in equation (3.15), we have:

\[
H_{n+t}^{-1} = H_{n+t-1}^{-1} + \frac{H_{n+t-1}^{-1}R_tH_{n+t-1}^{-1}}{1 + \text{tr}[R_tH_{n+t-1}^{-1}]}, \tag{C.13}
\]

where

\[
R_t = \pi_{u_t} (\pi_{u_t} - 1)^\top \otimes \begin{bmatrix} 1 \\ x_{u_t} \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ x_{u_t} \end{bmatrix}
\]
Appendix D

Statistical Background

Asymptotic analysis plays an important role in statistics. It considers the extreme cases where the number of observations is increased with no bounds. In such scenarios, discussions on different notions of convergence of the sequence of random variables naturally arise. Generally speaking, there are three major types of stochastic convergence: convergence in probability, convergence in law (distribution) and convergence with high probability (almost surely). Here, we focus on the two former modes of convergence, discuss two fundamental results based on them and formalize our notations regarding parameter estimators. Further details of the following definitions and results can be found in any standard statistical textbook such as [99].

D.1 Convergence of Random Variables

Throughout this section, \( \{\theta_1, \theta_2, \ldots, \theta_n, \ldots\} \), denoted simply by \( \{\theta_n\} \), is a sequence of multivariate random variables lying in \( \Omega \subseteq \mathbb{R}^d \). Also suppose that \( \theta_0 \) is a constant vector and \( \tilde{\theta} \) is another random variable in the same space \( \Omega \).

**Definition D.1.** We say that the sequence \( \{\theta_n\} \) converges in probability to \( \theta_0 \) and write \( \theta_n \xrightarrow{P} \theta_0 \), iff for every \( \varepsilon > 0 \) we have:

\[
P(|\theta_{ni} - \theta_{0i}| > \varepsilon) \to 0, \quad \text{for all } i = 1, \ldots, d. \tag{D.1}
\]

Convergence in probability is invariant with respect to any continuous mapping:

**Proposition D.2** ([111], Proposition 6.1.4). If \( \theta_n \xrightarrow{P} \theta_0 \) and \( g : \Omega \to \mathbb{R} \) is a continuous function at \( \theta = \theta_0 \), then \( g(\theta_n) \xrightarrow{P} g(\theta_0) \).
APPENDIX D. STATISTICAL BACKGROUND

Definition D.3. We say that a sequence \{\theta_n\} converges in law (in distribution) to the random variable \(\tilde{\theta}\) and write \(\theta_n \xrightarrow{L} \tilde{\theta}\), iff the sequence of their joint CDFs, \(F_n\), point-wise converges to the joint CDF of \(\tilde{\theta}\):

\[F_n(a) = P(\theta_{1n} \leq a_1, \ldots, \theta_{nd} \leq a_d) \to F(a) = P(\tilde{\theta}_1 \leq a_1, \ldots, \tilde{\theta}_d \leq a_d) \quad \forall a \in C_F \subseteq \mathbb{R}^d,\]

where \(C_F\) is the set of continuity points of the CDF \(F\).

Equation (D.2) means that for large values of \(n\), the distribution of \(\theta_n\) can be well approximated by the distribution of \(\tilde{\theta}\). Note that throughout this paper, for simplicity, we say that a random sequence \{\theta_n\} converges to a distribution with density function \(p(\theta)\), or write \(\theta_n \xrightarrow{L} p(\theta)\), instead of fully saying that \{\theta_n\} converges in law to a random variable with that distribution.

Note that \(\theta_n \xrightarrow{P} \theta_0\) suggests that \(\theta_n - \theta_0 \xrightarrow{L} \delta(\theta)\) where \(\delta\) is the Kronecker delta function, which can be viewed as the density function of a degenerate distribution at \(\theta = 0\). This, however, does not give any information about the speed with which \(\theta_n\) converges to \(\theta_0\). In order to take the speed into account, we consider the convergent distribution of the sequence \(a_n \cdot (\theta_n - \theta_0)\), where \(a_n\) is any sequence of positive integers and \(a_n \to \infty(n \to \infty)\). In practice \(a_n\) is usually considered to have the form \(n^r\) with \(r > 0\).

Definition D.4. Assume \(\theta_n \xrightarrow{P} \theta_0\). We say that the sequence \{\theta_n\} converges to \(\theta_0\) with rate of convergence \(r > 0\), iff \(n^r(\theta_n - \theta_0)\) converges in law to a random variable with non-degenerate distribution. Furthermore, the non-degenerate distribution is the asymptotic distribution of \(\theta_n\).

Next, we discuss some of the classic results in asymptotic statistics:

Theorem D.5 (Law of Large Numbers, [111]). Let \(\theta_1, \ldots, \theta_n\) be a set of independent and identically distributed (i.i.d) samples. If \(\mathbb{E}[\theta_i] = \mu\), then

\[\bar{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} \theta_i \xrightarrow{P} \mu.\]

Theorem D.6 (Central Limit Theorem, [99]). Let \(\theta_1, \ldots, \theta_n\) be a set of i.i.d samples with mean \(\mathbb{E}[\theta_i] = \mu\) and covariance \(\text{Cov}[	heta_i] = \Sigma\) (with a symmetric and positive semi-definite matrix \(\Sigma\)), then the sequence of sample averages \{\bar{\theta}_n\} with \(\bar{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} \theta_i\) converges to the true mean with convergence rate \(1/2\). Moreover, its asymptotic distribution is a zero-mean Gaussian distribution.
APPENDIX D. STATISTICAL BACKGROUND

with covariance matrix $\Sigma$, that is:

$$\sqrt{n} \cdot (\hat{\theta}_n - \mu) \xrightarrow{L} \mathcal{N}(0, \Sigma).$$ (D.4)

The following results are very useful when deriving the asymptotic distribution of a random sequence under a continuous mapping:

**Theorem D.7. (Multivariate Delta Method, first order, [99])** Let $\{\theta_n\}$ be a sequence of random variables such that it converges to $\theta_0$ with rate of convergence $1/2$ and a normal asymptotic distribution, that is $\sqrt{n} \cdot (\theta_n - \theta_0) \xrightarrow{L} \mathcal{N}(0, \Sigma)$. If $g : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable mapping and $\nabla_\theta g(\theta_0) \neq 0$, then

$$\sqrt{n} \cdot \left[ g(\theta_n) - g(\theta_0) \right] \xrightarrow{L} \mathcal{N} \left( 0, \nabla_\theta^\top g(\theta_0) \Sigma \nabla_\theta g(\theta_0) \right).$$ (D.5)

**Theorem D.8** (Multivariate Delta Method, second order). Let $\{\theta_n\}$ be a sequence of random variables such that it converges to $\theta_0$ with rate of convergence $1/2$ and a normal asymptotic distribution, that is $\sqrt{n} \cdot (\theta_n - \theta_0) \xrightarrow{L} \mathcal{N}(0, \Sigma)$. If $g : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable mapping where $\nabla_\theta g(\theta_0) = 0$ and $\nabla^2_\theta g(\theta_0)$ is non-singular in a neighborhood of $\theta_0$, then the sequence $\{g(\theta_n) - g(\theta_0)\}$ converges in law to a mixture of random variables with first-degree Chi-square distributions, and the rate of convergence is one. More specifically,

$$n \cdot \left[ g(\theta_n) - g(\theta_0) \right] \xrightarrow{L} \sum_{i=1}^{d} \lambda_i \chi^2_1,$$ (D.6)

where $\lambda_i$’s are eigenvalues of $\Sigma^{1/2} \nabla_\theta g(\theta_0) \Sigma^{1/2}$. Moreover, variance of this asymptotic distribution can be written as

$$\frac{1}{2} \left\| \Sigma^{1/2} \nabla^2_\theta g(\theta_0) \Sigma^{1/2} \right\|_F^2,$$ (D.7)

where $\| \cdot \|_F$ is the Frobenius norm.

**Proof.** For proof see Appendix E. □

### D.2 Parameter Estimation

Now suppose that the set of independent and identically distributed (i.i.d) set of samples $x_1, \ldots, x_n$ are generated from an underlying distribution that belongs to a parametric family, for which the
APPENDIX D. STATISTICAL BACKGROUND

density function \( p(x \mid \theta) \) can be represented by a multivariate parameter vector \( \theta \). Assume the true parameter is \( \theta_0 \), that is \( \{x_i\} \sim p(x \mid \theta_0) \), \( i = 1, \ldots, n \). An estimator \( \theta_n = \theta(x_1, \ldots, x_n) \) is a function that maps the observed random variables to a point in the parameter space \( \Omega \). The subscript \( n \) in \( \theta_n \) indicates its dependence on the sample size. Since the observations are generated randomly, the estimators are also random and thus \( \{\theta_n\} \) can be viewed as a sequence of random variables. There are some reserved terms for such a sequence, which we introduce in the remaining of this section:

Definition D.9 (Consistency). We say that an estimator \( \theta_n \) is consistent iff \( \theta_n \xrightarrow{P} \theta_0 \).

Based on Theorem D.5, sample average of the observation set is a consistent estimator of the true mean of the samples. Another important characteristic of estimators is based on the following bound over their covariance matrices:

Theorem D.10 (Cramér-Rao, [99]). Let \( x_1, \ldots, x_n \sim p(x \mid \theta_0) \) and \( \theta_n = \theta(x_1, \ldots, x_n) \) be an estimator. If the first moment of \( \theta_n \) is differentiable with respect to the parameter vector and its second moment is finite, then the following inequality holds for every \( \theta \in \Omega \):

\[
\text{Cov}[\theta_n] \succeq -(\nabla_{\theta}\mathbb{E}[\theta_n])^\top I(\theta)^{-1} \nabla_{\theta}\mathbb{E}[\theta_n]. \quad (D.8)
\]

The right-hand-side of (D.8) is called the Cramer-Rao bound of the estimator, where the middle term is the inverse of the Fisher information matrix of the parametric distribution \( p(x \mid \theta) \):

\[
I(\theta) = \mathbb{E} \left[ \nabla_{\theta} \log p(x \mid \theta) \cdot \nabla_{\theta}^\top \log p(x \mid \theta) \right]
\]

Theorem D.10 suggests that for an unbiased estimator \( \theta_n \), the inequality over the covariance matrix becomes: \( \text{Cov}[\theta_n] \succeq I(\theta)^{-1}, \forall \theta \in \Omega \).

Definition D.11 (Efficiency). We say that an estimator \( \theta_n \) is efficient, iff it attains the Cramér-Rao bound, that is \( \text{Cov}[\theta_n] \) achieves the lower-bound in (D.8) for every \( n = 1, 2, \ldots \). Furthermore, we say that \( \theta_n \) is asymptotically efficient, iff the lower bound is attained asymptotically (when \( n \to \infty \)).
Appendix E

Proof of Second-order Multivariate Delta Method

In order to prove this theorem, we have to formulate the statistical Taylor expansion. This, in turn, needs a brief introduction of stochastic order notations.

E.1 Stochastic Order Notations

The stochastic order notations are denoted by $o_p$ and $O_p$, where the former is equivalent to convergence in probability (Definition D.1) and the latter implies boundedness in probability. In what follows, if otherwise stated, $\{\theta_n\}$ is a sequence of multivariate random variables lying in $\Omega \subseteq \mathbb{R}^d$ and $\{a_n\}$ is a sequence of strictly positive real numbers. The skipped proofs can be found in many textbooks on asymptotic theory, such as [111, Chapter 6].

Definition E.1. We write $\theta_n = o_p(a_n)$ iff

\[
\frac{\theta_{in}}{a_n} = o_p(1), \quad \text{for all } i = 1, \ldots, d
\]  

(E.1)

Definition E.2. We write $\theta_n = O_p(a_n)$ iff the sequence $\left\{\frac{\theta_{in}}{a_n}\right\}$ is bounded in probability for every $i = 1, \ldots, d$, that is, for every $\epsilon > 0$ there exists $\delta_\epsilon$ such that

\[
P\left(\left|\frac{\theta_{in}}{a_n}\right| > \delta_\epsilon\right) < \epsilon, \quad n = 1, 2, \ldots
\]  

(E.2)

We also need the following propositions:
APPENDIX E. PROOF OF SECOND-ORDER MULTIVARIATE DELTA METHOD

**Proposition E.3 ([111]).** Let \( \{ \theta_n \} \) and \( \{ \eta_n \} \) be two sequences of scalar random variables, and \( \{ a_n \} \) and \( \{ b_n \} \) be two sequences of positive real numbers. If \( \theta_n = O_p(a_n) \) and \( \eta_n = o_p(b_n) \), then

(i) \( \theta_n^2 = O_p(a_n^2) \)

(ii) \( \theta_n \eta_n = o_p(a_n b_n) \)

**Proposition E.4.** The followings are true\(^1\):

(i) \( \theta_n = o_p(a_n) \) \( \iff \) \( \| \theta_n \| = o_p(a_n) \).

(ii) \( \theta_n = O_p(a_n) \) \( \iff \) \( \| \theta_n \| = O_p(a_n) \).

**Proof.** The proof of part (i) can be found in [111, Proposition 6.1.2]. Here, we only prove part (ii):

\( ii, \Rightarrow : \) Since \( \theta_n = O_p(a_n) \), for every \( \varepsilon > 0 \) and for every \( i = 1, \ldots, d \), there exists a coefficient \( \delta_i > 0 \) such that

\[
P(\| \theta_{ni} \| > a_n \cdot \delta_i) < \frac{\varepsilon}{d}, \quad n = 1, 2, \ldots. \tag{E.3}
\]

Define \( \delta_{\text{max}} = \max\{ \delta_1, \ldots, \delta_d \} \) and note that we can write

\[
\left\{ \theta_n : \sum_{i=1}^{d} |\theta_{ni}|^2 > (d \cdot a_n \cdot \delta_{\text{max}})^2 \right\} \subseteq \left[ \bigcap_{i=1}^{d} \{ \theta_n : |\theta_{ni}| \leq a_n \cdot \delta_{\text{max}} \} \right]^c = \bigcup_{i=1}^{d} \{ \theta_n : |\theta_{ni}| > a_n \cdot \delta_{\text{max}} \} \tag{E.4}
\]

implying that

\[
P\left( \| \theta_n \|^2 > (d \cdot a_n \cdot \delta_{\text{max}})^2 \right) \leq P\left( \bigcup_{i=1}^{d} \{ \theta_n : |\theta_{ni}| > a_n \cdot \delta_{\text{max}} \} \right) \leq \sum_{i=1}^{d} P(|\theta_{ni}| > a_n \cdot \delta_{\text{max}}) \tag{E.5}
\]

Furthermore, for every \( i = 1, \ldots, d \) we have \( \delta_{\text{max}} \geq \delta_i \), consequently the interval \( (a_n \delta_{\text{max}}, \infty) \) is a subset of \( (a_n \delta_i, \infty) \) and \( P(|\theta_{ni}| > a_n \delta_{\text{max}}) \leq P(|\theta_{ni}| > a_n \delta_i) \). This implies that

\[
P\left( \| \theta_n \|^2 > (d \cdot a_n \cdot \delta_{\text{max}})^2 \right) \leq \sum_{i=1}^{d} P(|\theta_{ni}| > a_n \cdot \delta_i) < \varepsilon. \tag{E.6}
\]

\(^1\)Unless subscripted, \( \| \cdot \| \) denotes the \( L_2 \) norm in all the equations.
Therefore, for every \( \varepsilon > 0 \), we can choose \( \delta_\varepsilon = d \cdot \delta_{\text{max}} \) such that \( P\left( \frac{\| \theta_n \|}{a_n} > \delta_\varepsilon \right) < \varepsilon \) for every \( n = 1, 2, \ldots \), that is \( \| \theta_n \| = O_p(a_n) \).

\((ii, \Leftarrow)\): Suppose \( \| \theta_n \| = O_p(a_n) \), that is for every \( \varepsilon > 0 \) we can find \( \delta_\varepsilon > 0 \) such that

\[
P\left( \frac{\| \theta_n \|}{a_n} > a_n \cdot \delta_\varepsilon \right) < \varepsilon , \quad n = 1, 2, \ldots \tag{E.7}
\]

It is clear that for any given \( i \in \{1, \ldots, d\} \) we have

\[
\{ \theta_n : |\theta_{ni}| > a_n \cdot \delta_\varepsilon \} \subseteq \{ \theta_n : \| \theta_n \| > a_n \cdot \delta_\varepsilon \} \tag{E.8}
\]

hence

\[
P(|\theta_{ni}| > a_n \cdot \delta_\varepsilon) \leq P\left( \frac{\| \theta_n \|}{a_n} > a_n \cdot \delta_\varepsilon \right) < \varepsilon , \quad n = 1, 2, \ldots \tag{E.9}
\]

meaning that \( \theta_{ni} = O_p(a_n), i = 1, \ldots, d \) or equivalently \( \theta_n = O_p(a_n) \).

**Proposition E.5.** If \( \theta_n = O_p(a_n) \) and \( a_n \to 0(n \to \infty) \), then \( \theta_n = o_p(1) \).

**Proof.** The goal is to show \( \theta_n = o_p(1) \) or equivalently \( \| \theta_n \| = o_p(1) \) by proving that \( P\left( \| \theta_n \| > \varepsilon \right) \to 0(n \to \infty) \) for every \( \varepsilon > 0 \). Fix \( \varepsilon \) to a positive real number. In order to have the sequence of probability numbers \( \{ P\left( \| \theta_n \| > \varepsilon \right) \} \) converging to zero, for every \( \varepsilon_0 > 0 \) there should exist a positive integer \( N > 0 \) such that

\[
P\left( \| \theta_n \| > \varepsilon \right) < \varepsilon_0 \quad \forall n > N. \tag{E.10}
\]

Because of the assumption of being bounded by \( a_n \), that is \( \theta_n = O_p(a_n) \) or equivalently \( \| \theta_n \| = O_p(a_n) \), we can choose a real number \( \delta_0 > 0 \) such that

\[
P\left( \| \theta_n \| > a_n \delta_0 \right) < \varepsilon_0 \quad n = 1, 2, \ldots \tag{E.11}
\]

On the other hand, since \( a_n \to 0(n \to \infty) \), there exists a large enough number \( N_0 > 0 \) such that

\[
0 < a_n < \frac{\varepsilon}{\delta_0} \text{ for all } n > N_0.
\]

Therefore we get:

\[
[0, a_n \delta_0] \subseteq [0, \varepsilon] \quad \forall n > N_0 \tag{E.12}
\]

implying that

\[
P\left( \| \theta_n \| \leq a_n \delta_0 \right) \leq P\left( \| \theta_n \| \leq \varepsilon \right) \quad \forall n > N_0 \tag{E.13}
\]
APPENDIX E. PROOF OF SECOND-ORDER MULTIVARIATE DELTA METHOD

From inequalities (E.11) and (E.13), and noticing that the latter holds for all $n$ whereas the former is satisfied when $n > N_0$, one can write:

$$P(\|\mathbf{\theta}_n\| > \varepsilon) \leq P(\|\mathbf{\theta}_n\| > a_n\delta_0) < \varepsilon_0 \quad \forall n > N_0$$  \hspace{1cm} (E.14)

Therefore, for every $\varepsilon_0 > 0$, equation (E.10) is guaranteed if $N$ is chosen to be equal to $N_0$ so that inequality (E.13) is satisfied. Similarly, this can be written for every $\varepsilon > 0$, thus the proof is complete. \hfill $\Box$

**Proposition E.6** ([112], Chapter 1). Let $\{\mathbf{\theta}_n\}$ be a sequence of random variables. If there exists a random variable $\mathbf{\theta}_0$ such that $\mathbf{\theta}_n \xrightarrow{L} \mathbf{\theta}_0$, then $\mathbf{\theta}_n = O_p(1)$.

### E.2 Second-order Statistical Taylor Expansion

Now we are ready to establish the second-order statistical Taylor expansion.

**Theorem E.7.** Let $\{\mathbf{\theta}_n\}$ be a sequence of random vectors in a convex and compact set $\Omega \subseteq \mathbb{R}^d$ and $\mathbf{\theta}_0 \in \Omega$ be a constant vector such that $\mathbf{\theta}_n - \mathbf{\theta}_0 = O_p(a_n)$ where $a_n \to 0(n \to \infty)$. If $g : \Omega \to \mathbb{R}$ is a $C^3$ function, then

$$g(\mathbf{\theta}_n) = g(\mathbf{\theta}_0) + \nabla_{\mathbf{\theta}} g(\mathbf{\theta}_0)(\mathbf{\theta}_n - \mathbf{\theta}_0) + \frac{1}{2}(\mathbf{\theta}_n - \mathbf{\theta}_0)^T \nabla_{\mathbf{\theta}^2} g(\mathbf{\theta}_0)(\mathbf{\theta}_n - \mathbf{\theta}_0) + o_p(a_n^2).$$  \hspace{1cm} (E.15)

**Proof.** Since $g$ is twice continuously differentiable in a neighborhood of $\mathbf{\theta}_0$, it can be written in terms of the Taylor expansion as

$$g(\mathbf{\theta}) = g(\mathbf{\theta}_0) + (\mathbf{\theta} - \mathbf{\theta}_0)^T \nabla_{\mathbf{\theta}} g(\mathbf{\theta}_0) + \frac{1}{2}(\mathbf{\theta} - \mathbf{\theta}_0)^T \nabla_{\mathbf{\theta}^2} g(\mathbf{\theta}_0)(\mathbf{\theta} - \mathbf{\theta}_0) + r_2(\mathbf{\theta}, \mathbf{\theta}_0)$$  \hspace{1cm} (E.16)

where $r_2(\mathbf{\theta}, \mathbf{\theta}_0)$ is the Lagrange remainder of second order. Based on Taylor’s polynomial theorem for multivariate functions, there exists a number $t \in [0, 1]$ such that $\mathbf{\theta}^* = t \mathbf{\theta} + (1 - t) \mathbf{\theta}_0 \in \Omega$ (due to convexity of $\Omega$) and

$$r_2(\mathbf{\theta}, \mathbf{\theta}_0) = \frac{1}{6} \sum_{1 \leq i,j,k \leq d} \frac{\partial^3 g(\mathbf{\theta}^*)}{\partial \mathbf{\theta}_i \partial \mathbf{\theta}_j \partial \mathbf{\theta}_k} (\mathbf{\theta}_i - \mathbf{\theta}_0i)(\mathbf{\theta}_j - \mathbf{\theta}_0j)(\mathbf{\theta}_k - \mathbf{\theta}_0k).$$  \hspace{1cm} (E.17)
APPENDIX E. PROOF OF SECOND-ORDER MULTIVARIATE DELTA METHOD

But since $\Omega$ is compact and $g \in C^3$, the third derivative of $g$ is bounded\(^2\) and therefore there exists $M > 0$ such that

$$\left| \frac{\partial^3 g(\theta)}{\partial \theta_i \partial \theta_j \partial \theta_k} \right| \leq M, \forall \theta \in \Omega, \forall i, j, k \in \{1, \ldots, d\}$$  \hspace{1cm} (E.18)

Hence the Lagrange remainder can be bounded by

$$|r_2(\theta, \theta_0)| \leq \frac{M}{6} \sum_{1 \leq i, j, k \leq 3} |\theta_i - \theta_{0i}| \cdot |\theta_j - \theta_{0j}| \cdot |\theta_k - \theta_{0k}|$$

$$= \frac{M}{6} \left\| \theta - \theta_0 \right\|_1^3$$

$$\leq \frac{M'}{6} \left\| \theta - \theta_0 \right\|_3^3$$ \hspace{1cm} (E.19)

where $M' = c_u M$ and $c_u$ is obtained from the equivalence of norms in $\mathbb{R}^d$. Now define the function $h : \Omega \to \mathbb{R}$ as below

$$h(\theta) := \begin{cases} \frac{r_2(\theta, \theta_0)}{\left\| \theta - \theta_0 \right\|^2/2}, & \theta \neq \theta_0 \\ 0, & \theta = \theta_0 \end{cases}$$  \hspace{1cm} (E.21)

Note that $h(\theta)$ is continuous at $\theta = \theta_0$: due to boundedness of $r_2(\theta, \theta_0)$, $h(\theta)$ is also bounded by

$$|h(\theta)| \leq \frac{M'}{3} \left\| \theta - \theta_0 \right\|.$$  \hspace{1cm} (E.22)

Hence, for every $\varepsilon > 0$, we can select $\delta_\varepsilon = \frac{3\varepsilon}{M'}$ such that the following continuity condition holds

$$\left\| \theta - \theta_0 \right\| < \delta_\varepsilon \Rightarrow |h(\theta)| \leq \varepsilon.$$  \hspace{1cm} (E.23)

Continuity of $h(\theta)$ at $\theta = \theta_0$ implies $\lim_{\theta \to \theta_0} h(\theta) = h(\theta_0) = 0$. Furthermore, since $\theta_n - \theta_0 = O_p(a_n)$ and $a_n \to 0 (n \to \infty)$, Proposition E.5 suggests that $\theta_n - \theta_0 = o_p(1)$. These two enable us

\(^2\)This is because of the following Theorem in real analysis:

**Theorem E.8.** Let $X$ and $Y$ be two vector spaces. If $g : X \to Y$ is continuous and $X$ is compact, then $f(X)$ is compact in $Y$.

In special case of this theorem, when $Y = \mathbb{R}$, compactness of $f(X)$ is equivalent to boundedness and closedness.

\(^3\)Two norm functions $\| \cdot \|_{(1)}$ and $\| \cdot \|_{(2)}$, in a vector space $\Omega$, are called equivalent if there exist constants $c_u \geq c_d > 0$ such that

$$c_d \| \theta \|_{(2)} \leq \| \theta \|_{(1)} \leq c_u \| \theta \|_{(2)}, \forall \theta \in \Omega.$$  \hspace{1cm} (E.20)
to use Proposition D.2 and write
\[
h(\theta_n) - h(\theta_0) = h(\theta_n) = o_p(1).
\] (E.24)

Finally, from equation (E.21) and Propositions E.3, E.4 and E.5, we can write that
\[
r_2(\theta_n, \theta_0) = h(\theta_n) \cdot \frac{\|\theta_n - \theta_0\|^2}{2} = o_p(1) \cdot O_p(a_n^2) = o_p(a_n^2)
\] (E.25)

\section*{E.3 Second-order Multivariate Delta Method}

Finally, here is the proof of second-order multivariate Delta method (Theorem D.8):

\textbf{Proof.} From assumption of the Theorem, \(\sqrt{n}(\theta_n - \theta_0) \xrightarrow{L} \mathcal{N}(0, \Sigma)\), and Proposition E.6, one conclude that \(\sqrt{n}(\theta_n - \theta_0) = O_p(1)\) and therefore \(\theta_n - \theta_0 = O_p\left(\frac{1}{\sqrt{n}}\right)\). Thus we can use Theorem E.7 with \(a_n = \frac{1}{\sqrt{n}}\) to write:
\[
g(\theta) = g(\theta_0) + (\theta - \theta_0)\cdot \nabla_{\theta} g(\theta_0) + \frac{1}{2}(\theta - \theta_0)^\top \nabla^{2}_{\theta} g(\theta_0)(\theta - \theta_0) + o_p\left(\frac{1}{n}\right),
\] (E.26)

hence
\[
n\left[ g(\theta) - g(\theta_0) \right] = \frac{1}{2} \left[ \sqrt{n} \cdot (\theta - \theta_0) \right]^\top \nabla^{2}_{\theta} g(\theta_0) \left[ \sqrt{n} \cdot (\theta - \theta_0) \right] + o_p(1)
\]
\[
\xrightarrow{L} \frac{1}{2} \mathcal{N}(0, \Sigma)^\top \nabla^{2}_{\theta} g(\theta_0) \mathcal{N}(0, \Sigma)
\]
\[
= \frac{1}{2} \mathcal{N}(0, I_d)^\top \left[ \Sigma^{1/2} \nabla^{2}_{\theta} g(\theta_0) \Sigma^{1/2} \right] \mathcal{N}(0, I_d)
\] (E.27)

Define \(\Gamma := \Sigma^{1/2} \nabla^{2}_{\theta} g(\theta_0) \Sigma^{1/2}\) and rewrite the right-hand-side element-wise as
\[
\frac{1}{2} \mathcal{N}(0, I_d)^\top \Gamma \mathcal{N}(0, I_d) = \frac{1}{2} \sum_{i=1}^{d} \lambda_i \mathcal{N}(0, 1)^2 = \frac{1}{2} \sum_{i=1}^{d} \lambda_i \chi_i^2,
\] (E.28)

where \(\lambda_i\)’s are eigenvalues of \(\Gamma\). Finally, noting that the terms in the Chi-square mixture are
independent, variance of the convergent random variable can be easily computed as

\[
\text{Var} \left[ \frac{1}{2} \sum_{i=1}^{d} \lambda_i \chi_i^2 \right] = \frac{1}{4} \sum_{i=1}^{d} \lambda_i^2 \cdot \text{Var} [\chi_i^2] \\
= \frac{1}{2} \sum_{i=1}^{d} \lambda_i^2 \\
= \frac{1}{2} \left\| \Sigma^{1/2} \nabla_{x}^2 g(x_0) \Sigma^{1/2} \right\|_F^2, \tag{E.29}
\]

\[\square\]
Appendix F

Proof of Lemma 4.7

We first substitute the score function of the classifier

\[ \nabla_{\theta} \log P(y|x, \theta) = \frac{\nabla_{\theta} P(y|x, \theta)}{P(y|x, \theta)} \]

into formulation Monte-Carlo approximation of \( I_q \) to get:

\[
\hat{I}(\theta; Q) = \frac{1}{|Q|} \sum_{x \in Q} \sum_{y=1}^{c} P(y|x, \theta) \cdot \frac{\nabla_{\theta} P(y|x, \theta) \nabla_{\theta}^T P(y|x, \theta)}{P(y|x, \theta)^2} + \delta \mathbb{I}_d \quad (F.1)
\]

\[
\hat{I}(\theta; Q) = \frac{1}{|Q|} \sum_{x \in Q} \sum_{y=1}^{c} \frac{\nabla_{\theta} P(y|x, \theta) \cdot \nabla_{\theta}^T P(y|x, \theta)}{P(y|x, \theta)} + \delta \mathbb{I}_d \quad (F.2)
\]

Define the vector \( v_{\theta}(x, y) := \frac{\nabla_{\theta} P(y|x, \theta)}{\sqrt{P(y|x, \theta)}} \) and rewrite \( \hat{I}(\theta; Q) \) as:

\[
\hat{I}(\theta; Q) = \frac{1}{|Q|} \sum_{x \in Q} \sum_{y=1}^{c} v_{\theta}(x, y) \cdot v_{\theta}(x, y)^T + \delta \mathbb{I}_d. \quad (F.3)
\]
On the other hand, since \( Q \subset X \) we can write \( \hat{\mathbf{I}}(\theta; X) \) in terms of \( \hat{\mathbf{I}}(\theta; Q) \) by breaking the summation over \( X \) into summations over \( Q \) and \( X - Q \) as follows:

\[
\hat{\mathbf{I}}(\theta; X) = \frac{|Q|}{|X|} \left[ \frac{1}{|Q|} \sum_{x \in Q} \sum_{y=1}^{c} \mathbf{v}_\theta(x, y) \cdot \mathbf{v}_\theta(x, y)^\top + \delta \cdot \mathbb{I}_d \right] \\
+ \frac{1}{|X|} \sum_{x \in X - Q} \sum_{y=1}^{c} \mathbf{v}_\theta(x, y) \cdot \mathbf{v}_\theta(x, y)^\top + \delta \left( \frac{|X| - |Q|}{|X|} \right) \cdot \mathbb{I}_d \\
= \left( \frac{|Q|}{|X|} \right) \cdot \hat{\mathbf{I}}(\theta; Q) + \frac{1}{|X|} \sum_{x \in X - Q} \sum_{y=1}^{c} \mathbf{v}_\theta(x, y) \cdot \mathbf{v}_\theta(x, y)^\top \\
+ \delta \left( \frac{|X| - |Q|}{|X|} \right) \cdot \hat{\mathbf{I}}(\theta; Q)^{-1} 
\]  

(F.4)

Now that we related the Fisher information matrices to each other, we can compute the product of \( \hat{\mathbf{I}}(\theta; X) \) and \( \hat{\mathbf{I}}(\theta; Q)^{-1} \):

\[
\hat{\mathbf{I}}(\theta; Q)^{-1} \hat{\mathbf{I}}(\theta; X) = \left( \frac{|Q|}{|X|} \right) \cdot \mathbb{I}_d + \frac{1}{|X|} \sum_{x \in X - Q} \sum_{y=1}^{c} \mathbf{v}_\theta(x, y) \cdot \mathbf{v}_\theta(x, y)^\top \\
+ \delta \left( \frac{|X| - |Q|}{|X|} \right) \cdot \hat{\mathbf{I}}(\theta; Q)^{-1} 
\]  

(F.5)

Applying the trace function to both sides of the equation will result:

\[
\text{tr} \left[ \hat{\mathbf{I}}(\theta; Q)^{-1} \hat{\mathbf{I}}(\theta; X) \right] = \frac{|Q| \cdot d}{|X|} + \frac{1}{|X|} \sum_{x \in X - Q} \sum_{y=1}^{c} \text{tr} \left[ \hat{\mathbf{I}}(\theta; Q)^{-1} \mathbf{v}_\theta(x, y) \cdot \mathbf{v}_\theta(x, y)^\top \right] \\
+ \delta \left( \frac{|X| - |Q|}{|X|} \right) \cdot \text{tr} \left[ \hat{\mathbf{I}}(\theta; Q)^{-1} \right] \\
\approx \frac{|Q| \cdot d}{|X|} + \frac{1}{|X|} \sum_{x \in X - Q} \sum_{y=1}^{c} \mathbf{v}_\theta(x, y)^\top \hat{\mathbf{I}}(\theta; Q)^{-1} \mathbf{v}_\theta(x, y), 
\]  

where the last term is dropped since the overloading constant, \( \delta \), is assumed to be small. Furthermore, the term including \( \hat{\mathbf{I}}(\theta; Q)^{-1} \) can be approximated by replacing the weighted harmonic mean of the eigenvalues of \( \hat{\mathbf{I}}(\theta; Q) \) by their weighted arithmetic mean [4]:

\[
\mathbf{v}_\theta(x, y)^\top \hat{\mathbf{I}}(\theta; Q)^{-1} \mathbf{v}_\theta(x, y) \approx \frac{\| \mathbf{v}_\theta(x, y) \|^4}{\mathbf{v}_\theta(x, y)^\top \hat{\mathbf{I}}(\theta; Q) \mathbf{v}_\theta(x, y)}. 
\]  

(F.7)
APPENDIX F. PROOF OF LEMMA 4.7

Note that this approximation becomes exact when the condition number of \( \hat{I}(\theta; Q) \) is one. Substituting \( \hat{I}(\theta; Q) \) from equation (F.3) into the denominator of the approximation above yields:

\[
v_\theta(x, y) \hat{I}(\theta; Q) v_\theta(x, y) = \frac{1}{|Q|} \sum_{x' \in Q} \sum_{y' = 1}^c \left[ v_\theta(x, y) v_\theta(x', y') \right]^2 + \delta \| v_\theta(x, y) \|^2 \tag{F.8}
\]

Integrating this approximation with equation (F.6), and assuming that the value of \( \theta \) is not located at the stationary point of the conditional density \( P(y | x, \theta) \) (hence \( v_\theta(x, y) \) is not the zero vector), results:

\[
\text{tr} \left[ \hat{I}(\theta; Q)^{-1} \hat{I}(\theta; X) \right] \approx \frac{|Q| \cdot d}{|X|} + \frac{1}{|X|} \sum_{x \in X - Q} \sum_{y = 1}^c \delta \cdot \| v_\theta(x, y) \|^{-2} + \frac{1}{\sum_{x' \in Q} g_\theta(x, y, x')}
\]

where

\[
g_\theta(x, y, x') := \frac{1}{|Q|} \sum_{y' = 1}^c \left[ \frac{v_\theta(x, y) v_\theta(x', y')}{\| v_\theta(x, y) \|^2} \right]^2 \tag{F.10}
\]

Finally (F.9), implies that constrained optimization of both sides are equivalent. Note that with the constraint \( |Q| = k \), we treat \( |Q| \) as a constant for a given \( k \). In the following, we discard the constants from the right-hand-side.

\[
\arg\min_{Q \subseteq X, |Q| = k} \text{tr} \left[ \hat{I}(\theta; X_q)^{-1} \hat{I}(\theta; X_p) \right] = \arg\max_{Q \subseteq X, |Q| = k} \sum_{x \in X - Q} \sum_{y = 1}^c \delta \cdot \| v_\theta(x, y) \|^{-2} + \frac{1}{\sum_{x' \in Q} g_\theta(x, y, x')}
\]

(F.11)
Appendix G

Proof of Theorem 4.8

Proof of this Theorem is a generalization of the discussion by [4], with clarification of all the assumptions and approximations made.

First, note that the function $f$ can be broken into simpler terms $f(Q) = \sum_{y=1}^{c} f(Q; y)$, where

$$f(Q; y) = \sum_{x \in X - Q} \frac{-1}{\delta \cdot \|v_{\theta}(x, y)\|^2 + \sum_{x' \in Q} g_{\theta}(x, y, x')}, \quad \forall Q \subseteq X. \quad (G.1)$$

Therefore, in order to prove submodularity and monotonicity of $f_{\theta}$, it suffices to prove these properties for $f_{\theta}(\cdot; y)$ for all $y \in \{1, \ldots, c\}$. Fix $y$ and take any subset $Q \subseteq X$ and $\xi \in X - Q$. Then, we can write:

$$f_{\theta}(Q \cup \{\xi\}; y) = \sum_{x \in X - (Q \cup \{\xi\})} \frac{-1}{\delta \cdot \|v_{\theta}(x, y)\|^2 + \sum_{x' \in Q \cup \{\xi\}} g_{\theta}(x, y, x')} \quad (G.2)$$

$$= \sum_{x \in X - Q} \frac{-1}{\delta \cdot \|v_{\theta}(x, y)\|^2 + \sum_{x' \in Q \cup \{\xi\}} g_{\theta}(x, y, x')} + \frac{1}{\delta \cdot \|v_{\theta}(\xi, y)\|^2 + \sum_{x' \in Q \cup \{\xi\}} g_{\theta}(\xi, y, x')}.$$
We then form the discrete derivative of \( f_\theta(\cdot; y) \) at \( Q \) to get:

\[
\rho_{f_\theta(\cdot; y)}(Q; \xi) = f_\theta(Q \cup \{\xi\}; y) - f_\theta(Q; y)
= \sum_{x \in X - Q} \left[ \frac{-1}{\|v_\theta(x, y)\|^2} + \sum_{x' \in \mathcal{Q} \cup \{\xi\}} \frac{\delta_{v_\theta(x, y)} \|v_\theta(x, y')\|^2 + 1}{\sum_{x' \in \mathcal{Q}} g_\theta(x, y, x')} \right]
+ \frac{1}{\|v_\theta(x, y)\|^2} \sum_{x' \in \mathcal{Q} \cup \{\xi\}} g_\theta(x, y, x').
\]

The right-hand-side can be rewritten as

\[
\sum_{x \in X - Q} \left[ \frac{g_\theta(x, y, \xi)}{\|v_\theta(x, y)\|^2} + \sum_{x' \in \mathcal{Q} \cup \{\xi\}} \frac{\delta_{v_\theta(x, y)} \|v_\theta(x, y')\|^2 + 1}{\sum_{x' \in \mathcal{Q}} g_\theta(x, y, x')} \right]
+ \frac{1}{\|v_\theta(x, y)\|^2} \sum_{x' \in \mathcal{Q} \cup \{\xi\}} g_\theta(x, y, x').
\]

Since by definition \( g_\theta(x, y, x') \geq 0, \forall x, y, x' \), all of the terms in (G.4) are non-negative and therefore \( \rho_{f_\theta(\cdot; y)}(Q; \xi) \geq 0 \). This is true for any \( Q \subseteq X \) hence monotonicity of \( f_\theta(\cdot; y) \) is obtained. Now let us take any superset \( Q' \) such that \( Q \subseteq Q' \subseteq X \) and \( \xi \in X - Q' \), and form the difference between their corresponding discrete derivatives. From (G.4) we will have:

\[
\rho_{f_\theta(\cdot; y)}(Q; \xi) - \rho_{f_\theta(\cdot; y)}(Q'; \xi)
= \sum_{x \in X - Q} \left[ \frac{g_\theta(x, y, \xi)}{\|v_\theta(x, y)\|^2} + \sum_{x' \in \mathcal{Q} \cup \{\xi\}} \frac{\delta_{v_\theta(x, y)} \|v_\theta(x, y')\|^2 + 1}{\sum_{x' \in \mathcal{Q}} g_\theta(x, y, x')} \right]
+ \frac{\delta_{v_\theta(x, y)} \|v_\theta(x, y)\|^2 + 1}{\|v_\theta(x, y)\|^2} \sum_{x' \in \mathcal{Q} \cup \{\xi\}} g_\theta(x, y, x')
- \sum_{x \in X - Q'} \left[ \frac{g_\theta(x, y, \xi)}{\|v_\theta(x, y)\|^2} + \sum_{x' \in \mathcal{Q}' \cup \{\xi\}} \frac{\delta_{v_\theta(x, y)} \|v_\theta(x, y')\|^2 + 1}{\sum_{x' \in \mathcal{Q}'} g_\theta(x, y, x')} \right]
- \frac{\delta_{v_\theta(x, y)} \|v_\theta(x, y)\|^2 + 1}{\|v_\theta(x, y)\|^2} \sum_{x' \in \mathcal{Q}' \cup \{\xi\}} g_\theta(x, y, x').
\]
APPENDIX G. PROOF OF THEOREM 4.8

From non-negativity of $g_{\theta}$ and that $Q \subseteq Q'$, we can conclude that for any $x \in X$ and $y \in \{1, \ldots, c\}$:

$$\sum_{x' \in Q'} g_{\theta}(x, y, x') \geq \sum_{x' \in Q} g_{\theta}(x, y, x')$$

\[
\Leftrightarrow \left[ \sum_{x' \in Q'} g_{\theta}(x, y, x') + \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} \right]^{-1} \leq \left[ \sum_{x' \in Q} g_{\theta}(x, y, x') + \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} \right]^{-1}
\]

\[
\Leftrightarrow -\left[ \sum_{x' \in Q'} g_{\theta}(x, y, x') + \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} \right]^{-1} \geq -\left[ \sum_{x' \in Q} g_{\theta}(x, y, x') + \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} \right]^{-1}
\]

(G.6)

Similarly, since $Q \cup \{\xi\} \subseteq Q' \cup \{\xi\}$ we will get:

Applying the inequalities (G.6) and (G.7) into equation (G.5) results:

$$\rho_{f_{\theta}(:,y)}(Q; \xi) - \rho_{f_{\theta}(:,y)}(X_{q'}; \xi) \geq \sum_{x \in X - Q} \left[ \frac{g_{\theta}(x, y, \xi)}{\left( \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} + \sum_{x' \in Q \cup \{\xi\}} g_{\theta}(x, y, x') \right) \left( \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} + \sum_{x' \in Q} g_{\theta}(x, y, x') \right)} + \frac{1}{\|\mathbf{v}_\theta(\xi, y)\|^2} + \sum_{x' \in Q \cup \{\xi\}} g_{\theta}(\xi, y, x') \right]$$

(G.8)

which yields\(^1\)

\[
\sum_{x \in Q' - Q} \left( \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} + \sum_{x' \in Q \cup \{\xi\}} g_{\theta}(x, y, x') \right) \left( \frac{\delta}{\|\mathbf{v}_\theta(x, y)\|^2} + \sum_{x' \in Q} g_{\theta}(x, y, x') \right) \geq 0.
\]

(G.9)

\(^1\)The inequality in (G.8) is obtained by the fact that, for every four positive real numbers $a, a_0, b$ and $b_0$, if we have $-a \geq -a_0$ and $-b \geq -b_0$ (similar to (G.6) and (G.7)), then

$$-a \cdot b = (-a) \cdot b \geq (-a_0) \cdot b = a_0 \cdot (-b) \geq a_0 \cdot (-b_0) = -a_0 \cdot b_0.$$
APPENDIX G. PROOF OF THEOREM 4.8

Inequality (G.9) holds for any $Q \subseteq X$; hence submodularity of $f_\theta(z; y)$ stands for all $y \in \{1, \ldots, c\}$ and $\theta \in \Theta$. 