ACTIVE LEARNING METHODS BASED ON STATISTICAL LEVERAGE SCORES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Cem Orhan
July 2016

ACTIVE LEARNING METHODS BASED ON STATISTICAL LEVERAGE SCORES

By Cem Orhan

July 2016

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Öznur Taştan Okan(Advisor)

Çiğdem Gündüz Demir

Tolga Can

Approved for the Graduate School of Engineering and Science:

Levent Onural
Director of the Graduate School

ABSTRACT

ACTIVE LEARNING METHODS BASED ON STATISTICAL LEVERAGE SCORES

Cem Orhan
M.S. in Computer Engineering
Advisor: Öznur Taştan Okan
July 2016

In many real-world machine learning applications, unlabeled data are abundant whereas the class labels are expensive and/or scarce. An active learner aims to obtain a model with high accuracy with as few labeled instances as possible by effectively selecting useful examples for labeling. We propose two novel active learning approaches for pool-based active learning setting: `ALEVS` for querying single example at each iteration and `DBALEVS` for querying a batch of examples. `ALEVS` and `DBALEVS` select the most influential instance(s) based on statistical leverages scores of examples. The rank-k statistical leverage score of i-th row of an $n \times n$ kernel matrix $\mathbf{K}$ is the squared norm of the i-th row of the matrix $\mathbf{U}$ whose columns are the top-k eigenvectors of $\mathbf{K}$. Statistical leverage scores are shown to be useful in matrix approximation algorithms in finding influential rows of a matrix. `ALEVS` and `DBALEVS` assess the influence of the examples by the statistical leverage scores of kernel matrix computed on the examples of the pool. Additionally, through maximizing a submodular set function at each iteration `DBALEVS` selects a diverse a set of examples that are highly influential but are dissimilar to selected labeled set. Extensive experiments on diverse datasets show that the proposed methods, `ALEVS` and `DBALEVS` offer more effective strategies in comparison to other single and batch mode active learning approaches, respectively.

Keywords: Machine Learning, Active Learning, Binary Classification, Statistical Leverage Scores, Kernel Methods.

ÖZET

İSTATİSTİKSEL KALDIRAÇ DEĞERLERİNE DAYALI ETKİN ÖĞRENME METOTLARI

Cem Orhan
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Öznur Taştan Okan
Temmuz 2016

Yapay öğrenme metodlarının gerçek hayattaki birçok uygulamasında bol miktarda etiketlenmemiş veri bulunmasına karşılık etiketlenmiş veriler pahalı ve/veya sınırlı sayıdadır. Bir etkin öğrenici, etiketleme için yararlı örnekler seçerek mümkün olduğunca az etiketli örnek kullanımı ile yüksek doğrulukta bir model elde etmeyi amaçlamaktadır. Bu tezde havuz tabanlı etkin öğrenme kurgusu için iki yeni metot önerilmektedir: Her adımda bir tane etiketlenmemiş örneği seçerek etiketini sorgulayan (tek-seçimli) `ALEVS` metodu ve her adımda bir grup etiketlenmemiş örneği seçerek etiketlerini sorgulayan (grup-seçimli) `DBALEVS` metodu. `ALEVS` ve `DBALEVS` metodları örneklerin istatistiksel kaldıraç değerlerini kullanarak en etkili örneği/örnekleri seçer. $n \times n$ boyutlu bir $\mathbf{K}$ çekirdek matrisinin i-inci satırına ait k-kerte istatistiksel kaldıraç değerleri, kolonları $\mathbf{K}$ matrisinin üst-k özdeğer vektörlerinden oluşan $\mathbf{U}$ matrisinin i-inci satır düzgesinin karesidir. İstatistiksel kaldıraç değerlerinin etkili satırları seçerek düşük-kerte matris yaklaşıklama algoritmalarında yararlı oldukları gösterilmiştir. `ALEVS` ve `DBALEVS` metodları örneklerin önemini ölçmek için havuzdaki örnekler kullanılarak hesaplanmış çekirdek matrisinin istatistiksel kaldıraç değerlerini kullanır. Bunlara ek olarak, `DBALEVS` her adımda bir altmodüler küme fonksiyonunu maksimize ederek etkili, ama etiketlenmiş örneklere ve birbirlerine benzemeyen örnekleri seçmeye çalışır. Farklı verisetleri üzerinde yapılan deneylerle, `ALEVS` ve `DBALEVS` metodlarının karşılaştırılan diğer tek-seçimli ve grup-seçimli metodlara kıyasla data etkili yöntemler olduğu gösterilmiştir.

Anahtar sözcükler: Yapay Öğrenme, Etkin Öğrenme, İkili Sınıflandırma, İstatistiksel Kaldıraç Değerleri, Çekirdek Yöntemleri.

Acknowledgement

This thesis (and probably my enthusiasm for research in machine learning) would not be existed if I did not have the opportunity (first, to be taught by, and then) to work with my supervisor, Öznur Taştan, who supported me all through the enlightening but tough way of masters study with her kind, endlessly patient and wise supervision. Thanks for all the joyful research meetings and evenings; but biggest thank is for being more than a professor to me and giving me a purpose and courage to pursue a place in scientific world. Without you, there would be no Corhan of an academic kind.

I would like to thank Çiğdem Gündüz Demir and Tolga Can for showing the kindness to be a part of my jury committee.

Friends that I met in this hopeless and boring desert of Ankara, thank you for making this city sufferable.

Friends that I met in perhaps the most miserable, desperate but cozy place on earth, Yatakhane, thank you for the best five years of my life and everlasting friendship.

Mom and dad, you always supported my educative enthusiasm and respected my decisions since childhood; thank you for everything you sacrificed for me and for turning me from a baby into an individual. Uğur, thanks for being the most charismatic, funny and warm-hearted brother in the world.

Lastly, I would like to dedicate this thesis in memory of Fırat Bayır, one of the most influential minds that I have ever met, one of the greatest teachers that I have been taught by, the person who showed me how to enjoy maths and life together, the person who taught me that alms for knowledge is given by teaching, the person who taught me the beauty of the sigmoid function for the first time and the person who always wanted me to be an outlier with having a life that would make a difference.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every passing second, vast amount of raw data in a variety of fields such as medicine, biology, social media and marketing are being produced. For example, on the microblogging site Twitter[1], around 6,000 tweets are posted per second, which corresponds to 500 million tweets per day[2]. Luckily, in the past few decades, advances in data capture, the increasing availability of computational power and storage endowed us with basic tools and hardware to handle such data. In response to the increasing need for making sense of raw data, machine learning emerged as a field offering the much needed computational tools and theoretical framework to tackle a plethora of tasks such as image and speech recognition, text classification, drug discovery etc.

Machine learning approaches can be broadly categorized into two categories as *supervised* and *unsupervised* learning. In supervised learning, the goal is to predict the value of an output variable based on a number of input variables. In unsupervised learning, input variables are still present but there are no output variable to supervise the learning process; this approach exploit the intrinsic structure of the data. In supervised learning, the machine learning algorithm is presented with training examples as pairs of input features and a set of output

[1] `http://www.twitter.com`
[2] `http://www.internetlivestats.com/twitter-statistics/`

labels [1, 2]. The task is to infer a mapping between the input features and the output labels with good generalization performance so that the induced model will accurately predict the output labels of unseen examples reliably. In classification problems, these output labels are discrete class labels representing different classes. For example, in a medical diagnosis task the features can be the results of various medical tests or basic patient information such as age or weight, and the class label can be whether the patient has a particular disease or not.

Learning a supervised model with good predictive performance requires sufficiently large number of labeled data. However, in many real-world applications, the unlabeled data is cheaply and easily acquired but obtaining labels is difficult. For a twitter sentiment analysis, annotating the 500 million tweets per day with sentiment labels requires tremendous human effort. Moreover, labeling all examples will be redundant, as some examples do not add further information. There are also applications where labeling data is expensive because they require expert knowledge, i.e speech recognition data requires linguistic experts and medical image analysis requires pathology experts. Active learning is motivated from these applications where unlabeled data is abundant but labeling is costly, time consuming or requires expertise. Active learning algorithms reduce the labeling cost through intelligently interacting with an oracle for label acquisition [3].

The active learner aims to learn a model using as few training examples as possible with greater model accuracy. There are several scenarios in which active learner may operate in. One common scenario is the pool-based active learning, where a large number of unlabeled examples together with a small number of labeled examples are initially available. The learner interacts with an oracle (i.e. human expert) that provides labels when queried (Fig. 1.1). At each step the active learner chooses unlabeled example(s) from the unlabeled pool and request labels of these queries from the oracle. Once the algorithm is provided with the new labels annotated by the oracle, the training data is augmented with the newly labeled data and the classifier is retrained. This iterative procedure is repeated until a stopping criterion (i.e. budget constraint, desired accuracy, etc.) is met.

Active learning procedure can also differ by the number of queries submitted to

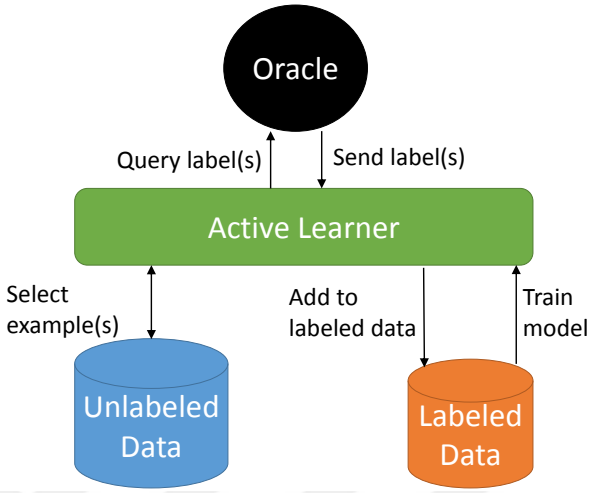

Figure 1.1: A standard pool-based active learning framework. In sequential-mode active learning, active learner queries a single example at each iteration whereas in batch-mode setting labels for a set of examples are requested from the oracle.

the oracle. At each iteration active learner may request label for a single instance, which is referred as *sequential mode*; or alternatively it may ask labels pertaining to a set of examples, *batch-mode* learning.

The critical component of an active learning algorithm is to decide which example to query. Different approaches have been suggested for the pool-based active learning problem in the last two decades [3]. Although improving upon a simple random selection strategy is not trivial [4], such approaches have demonstrated that active learning can significantly reduce the labeling cost [3, 5].

In this thesis, we focus on supervised learning and binary classification and pool-based learning scenario. We propose two novel active learning algorithms for sequential-mode and batch-mode selection. These algorithms employ statistical leverage scores for the selection of points. Statistical leverage score is not a new concept and it has found use in modern matrix approximation algorithms [6, 7]. The basic motivation behind these methods is to assign relative importance to the columns/rows of a matrix based on their statistical leverage scores. The sampling is carried out in accordance with the computed scores so that columns/rows with high statistical leverage scores are included in the final approximation. In

this study we exploit leverage scores to identify instances with important feature vectors, and thereby the examples associated with these features vectors are queried. This is the first study wherein the usefulness of statistical leverage scores is explored in the context of active learning.

This thesis documents two main and novel contributions:

- We present a sequential-mode active learning algorithm, **A**ctive Learning by Statistical **Lev**erage **S**ampling (`ALEVS`). This method uses statistical leverage scores utilizing both the true labels that are available and the predicted labels from currently trained classifier. Our extensive empirical experiments demonstrate that `ALEVS` can outperform some of the standard baselines and state-of-the art approaches on several binary classification benchmark datasets in terms of accuracy and computational runtime.
- We present a batch-mode active learning algorithm, **D**iverse **B**atch-mode **A**ctive Learning by Statistical **Lev**erage **S**ampling (`DBALEVS`), using statistical leverage scores. `DBALEVS` not only selects a batch of examples that are influential but also a set that is diverse. We achieve this by encoding these properties in a set function, and we prove that this set function is submodular and monotone; therefore, we can utilize a submodular maximization algorithm that is provably near-optimal.

We describe the notations used in this thesis in Appendix Table A.1.

The organization of the thesis is as follows:

- Chapter 2 provides background and discusses related work regarding active learning, statistical leverage scores and submodularity.
- Chapter 3 describes `ALEVS`, the proposed sequential-mode active learning method.
- Chapter 4 presents the results of empirical performance tests of `ALEVS` algorithm and the set of experiments conducted on several datasets.

- Chapter 5 describes `DBALEVS`, the proposed batch-mode active learning method.
- Chapter 6 presents the results of empirical performance tests of `DBALEVS` algorithm and the set of experiments conducted on several datasets.
- Finally, Chapter 7 presents conclusions and lists possible future directions.

Chapter 2

Background and Related Work

In this chapter, we will discuss related work and provide background information on active learning. First we will explain the active learning paradigm and different scenarios it is used. Additionally, we will give preliminary information on statistical leverage scores and submodular functions and how they are utilized in machine learning.

2.1 Active Learning

In active learning, there are three types of paradigms that are used for data sampling. The first one is referred to as "membership query synthesis", wherein the learner synthesizes new examples and queries the labels of these examples [3, 8, 9]. The active learner, by using the input space, generates the examples and these examples do not need to correspond to real-world examples. For query synthesis to be computable, one requirement is that the input space should be finite [9]. Although the generated examples could be effective in improving the classification performance if their labels were obtained, the generated images may not be labeled by an annotator [10]. For illustrating this point, an example often used in the literature [3, 10] is the following: consider an optical character recognition

task, which involves the classification of handwritten characters. Assume that the input space comprises gray scale images, and an active learner synthesizes the images for the input space. Most of these images will be too complex for a human labeler if the aim is to recognize the digits in them.

In the other two active learning scenarios, selective sampling and pool-based sampling, the examples are picked from the existing example distribution in lieu of de novo generation [3]. Selective sampling and pool-based sampling rely on different data point acquisition strategies, but both methods are commonly based on the assumption that unlabeled data comes at no cost or comes cheaper than labeled data (or labeling).

The selective sampling approach assumes that the learner receives streams of data, that is, examples arrive one by one and the active learner should decide whether to query or disregard the unlabeled example. This paradigm is more suitable whenever there is a limitation on processing capability and/or main memory [3] or the learner receives signals from the environment consistently. There are many methods that operate within the stream-based setting. Freund et al. [11], in their query-by-committee approach, train a committee of classifiers, and when a new example arrives, the method asks the label of the unlabeled data to this committee of predictors. If the committee disagrees on the label, then it queries the data to the annotator. CAL algorithm [12] defines a region of uncertainty for the current version space (set of hypotheses that are consistent with available labeled data) and it asks the label of incoming data based on this region. The uncertainty region corresponds to a specific area in feature space wherein the set of hypotheses do not agree with the labels of the data from this region. More specifically, upon receiving a new example, if the data point is found to reside in the uncertainty region defined by the current version space, then it asks its label to oracle. Intuitively, by querying points in this manner, version space is reduced, and the number of hypotheses that are consistent with the training data is decreased; hence we are closer to achieve the optimal hypothesis. Beygelzimer et al. proposed an algorithm [13] to address the selective sampling problem as well, but this time it assigns a probability for querying the label of the incoming data, which they call importance weighting. The labeling probability for a data

point is assigned proportional to the maximum difference between the loss function evaluations of the two hypotheses inside version space on this data point. Assuming the hypothesis class comprises linear separators and the loss function is convex, this problem can be solved through convex optimization.

Pool-based active learning as the third approach is appropriate when large collections of unlabeled data are available. Recall the Twitter example introduced in Chapter 1, where a large amount of uncategorized tweets are available but sentiment analysis labels are scarce due to high annotation cost. There are various settings where a large pool of unlabelled data can be cheaply and easily gathered. In the pool-based scenario the active learner is provided with this unlabeled pool and at each active selection step it chooses one or a batch of instances from this pool and retrain the model [3]. The overall strategy of pool-based active learner is described in Algorithm 1 [5]. Since our proposed methods are for pool-based active learning, we will discuss the related work regarding pool-based active learning in Section 2.1.1 and Section 2.1.2 for sequential-mode and batch-mode cases, respectively.

Algorithm 1 `ActiveLearning`

Input: $\mathbf{U}$: pool of unlabeled data; $\mathbf{L}$: available labeled data; $\mathcal{O}$: labeling oracle; c: stopping criterion.
Output: h^*: final classifier.
if $\mathbf{L} = \emptyset$ **then**
  Randomly pick examples(s) $\mathbf{x} \in U$
  Query $\mathbf{x}$ to $\mathcal{O}$ for label(s)
  $\mathbf{U} = \mathbf{U} \setminus \mathbf{x}$
  $\mathbf{L} = \mathbf{L} \cup \mathbf{x}$
end if
repeat
  Train a classifier h using $\mathbf{L}$
  Query example(s) $\mathbf{x} \in \mathbf{U}$ that satisfies some predefined query criteria to $\mathcal{O}$
  $\mathbf{U} = \mathbf{U} \setminus \mathbf{x}$
  $\mathbf{L} = \mathbf{L} \cup \mathbf{x}$
until c is satisfied
$h^* \leftarrow h$

Active learning is also used in problems other than classification, such as regression [14, 15, 16], rank learning [17, 18, 19], clustering [20, 21]. In these domains

too, active learning techniques lead to solutions with reduced labelling cost. In this thesis, we propose two pool-based active learning algorithm for classification. Therefore, we will review these methods. In one of the methods, the active learner requests one example at a time, and we will refer to this scenario as *sequential-mode*. In the second setting, a set of examples will be queried, which we will refer to as *batch-mode*. Note that in the existing literature selective sampling is sometimes termed as sequential sampling as well.

2.1.1 Sequential-mode active learning

In sequential-mode, one example is queried at each active learning iteration (Algorithm 1).

One of the oldest and widely used methods in sequential-mode active learning is uncertainty sampling [22]. In uncertainty sampling, the classifier samples examples from the regions where the classifier is least certain about, without paying attention to the density of the unlabeled data. By not querying the examples the classifier is confident about, uncertainty sampling focuses on regions where the decision boundary needs to be clarified. Uncertainty can be quantified in various ways. One way is to measure the distance to the decision boundary, which is applicable only for classifiers that provides such a distance measure. A probabilistic version is to query the instance whose predicted output is the least confident by calculating the posterior probability of the predicted class label and looking at the difference to the certainty [3, 22]. Another version is calculating the margin between the most likely predicted classes posterior probabilities. Another commonly used method is calculating Shannon's entropy [23] of predicted class labels. Information theoretic works for active data sampling are discussed in [24, 25]. Variants of this idea are margin based sampling for linear classifiers [26, 27], expected gradient length and Fisher information [28]. As empirical analyses of uncertainty sampling, Fisher information and expected gradient change algorithms (along with a method that incorporates both information and similarity, which will be discussed later) are presented in [28]. One important drawback

for these algorithms, particularly at early iterations, is that the classifier will be uncertain about many points and the decision boundary formed with the classifier will not be reliable as only few training examples are available. As pointed out in the literature [29], uncertainty sampling can also be fooled by outliers easily.

Dasgupta et al. [5], noted that uncertainty sampling introduces a sampling bias; the selected examples might not be representative at all. To address this problem, methods that select representative instances are proposed. [30] tries to find the clusters in data using hierarchical clustering algorithm, and queries unlabeled points accordingly. In [31], the active learner clusters the data and assigns probability for each instance to use them later for selecting queries.

Combining informativeness and representativeness is also studied in literature. [28] uses a weighting strategy that incorporates similarity to other points with informativeness of a point. A similar method, using density and entropy [32] is applied to a text classification problem. Donmez et al. [33] proposed a hybrid active learner that switches between informativeness and representativeness. For informativeness measure, they employed uncertainty sampling and for representativeness they employed density based sampling as proposed in [31]. Another algorithm, by Huang et al. [34] optimizes an objective function where both informativeness and representativeness are taken into account simultaneously. It is empirically shown in aforementioned studies, selecting instances that are both informative and representative is really effective to obtain a high accuracy classifier with minimal number of examples.

2.1.2 Batch-mode active learning

In batch-mode setting the active learner selects a set of examples and query this batch at once in each step of active selection. Methods proposed for this problem either adapt the sequential-mode setting to batch-mode setting by simply selecting top-b examples (where b is number of elements in batch) that maximizes the sampling metric (e.g. uncertainty and density), or try to directly optimize an objective function that represents a good quality batch.

The method introduced in [32] is applicable to both sequential and batch-mode active learning problems, where the method selects top b examples that satisfies an objective function that combines density and entropy. [35] poses the problem of batch selection as optimization, where the objective is to select a batch of examples so that the best discriminative classification performance is achieved. [36] uses Fisher information matrix to select examples. More specifically, its objective is to minimize the Fisher information between the selected set of unlabeled examples and the whole set of unlabeled examples. [37] aims to select instances so that the mutual information between labeled and unlabeled examples is maximized. This method uses Gaussian processes to reduce the problem to matrix partitioning. Posing the problem as minimizing the difference between labeled and unlabeled data distributions (using maximum mean discrepancy), [38] results a good empirical generalization performance. [39] introduces a method for selection of a batch of data points to minimize the error bound of the resulting classifier.

Diverse selection of unlabeled examples is also considered important and studied in the literature. [40] addresses the problem of selecting a batch for labeling as a submodular function maximization (which will be discussed deeper in Section 2.3 and Chapter 5) which results in a set that has a high value of information and it is diverse at the same time. A method that incorporates uncertainty and the divergence among selected set is introduced in [41], where they proposed an NP-hard optimization problem with bounded and guaranteed convex relaxations, which results in a diverse and informative set of instances. Another work that utilizes uncertainty and diversity is [42], where they represent the data points as a graph, and use a random walk based definition of information and incorporate the ranking of points in terms of similarity into equation for diversity.

2.2 Statistical Leverage Scores

The leverage scores are first introduced in the context of regression diagnostics. Statistical leverage score is defined as the diagonal entries of the hat matrix and

are used for detecting outliers [43, 44]:

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$
$$\ell = diag(\mathbf{X})$$

In more recent studies, statistical leverage scores are used and shown to be effective in low-rank matrix approximation and data analysis algorithms. Here, the statistical leverage score is defined as the squared Euclidean norm of the rows of the matrix that consists of top left singular vectors of arbitrary nxd matrices [7]. For a symmetric positive semi-definite (SPSD) the statistical leverage scores relative to the best rank-k approximation to input matrix is defined in [45] as follows:

Definition 1 (Leverage scores for an SPSD matrix)**.** Let $\mathbf{K}$, an arbitrary $m \times m$ SPSD matrix with the eigenvalue decomposition $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$. $\mathbf{U}$ can be partitioned as follows:

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{pmatrix}$$

where $\mathbf{U}_1$ comprises k orthonormal columns spanning the top k-dimensional eigenspace of $\mathbf{K}$. Let $\lambda_1(\mathbf{K}) \geq \lambda_2(\mathbf{K}) \geq \cdots \geq \lambda_m(\mathbf{K})$ be the eigenvalues of $\mathbf{K}$ ranked in descending order. Given $\mathbf{K}$ and a rank parameter k, the statistical leverage scores of $\mathbf{K}$ relative to the best rank-k approximation to $\mathbf{K}$ is equal to the squared Euclidean norms of the rows of the $m \times k$ matrix $\mathbf{U}_1$:

$$\ell_i := \|(\mathbf{U}_1)_{(i)}\|_2^2 \tag{2.1}$$

for $i \in \{1, \ldots, m\}$, where $\ell_i \in [0, 1]$, and $\sum_{i=1}^{m} \ell_i = k$.

Mahoney and others showed that in low-rank matrix approximation task, the column subset selection is improved if the columns of the matrices are selected based on a probability distribution weighted by the leverage scores of the columns [46, 47]. Along with these randomized algorithms, Papailiopoulos et al. [48] demonstrated that deterministically selecting a subset of the matrix columns with the largest leverage scores results in a good low-rank matrix approximation.

In a different study, Mahoney and Drineas showed that CUR decomposition is improved with statistical leverage scores [6]. In CUR decomposition [49], a matrix $\mathbf{A}$ is approximated with a product $\mathbf{A} = \mathbf{CUR}$, where $\mathbf{C}$ and $\mathbf{R}$ represent small subsets of the columns and rows of the parent matrix $\mathbf{A}$. To construct $\mathbf{C}$ (or $\mathbf{R}$), they compute statistical leverage scores for each column (row) of the input matrix, and randomly sample a small number of columns (rows) from the input matrix with a probability proportional to their leverage scores. In this line of work the leverage scores are used as importance scores for the columns (or rows) in a matrix. Similarly, Nyström extensions are sampling based randomized low-rank approximations to (SPSD) matrices. Gittens et al. analyzed different Nyström sampling strategies for SPSD matrices (with radial basis function, laplacian and linear kernels) and showed that sampling based on leverage scores are quite effective [45]. [50] used leverage score to sample matrix columns with missing data. [51] empirically analyzed different methods including sampling with probability proportional to their leverage scores for matrix column subset selection problem.

One application in supervised classification domain that uses statistical leverage scores is [52]. In their work, leverage scores are used as sampling distribution for column subset selection in random forest classifier. In another work, Bi and Wok [53] used leverage score based subspace sampling method in a multi-label classification problem. In multi-label classification, each sample can be associated with multiple labels. Some of these labels can be redundant, this work selects a small subset of class labels that can approximately span the original label space using statistical leverage scores. To our knowledge these are the only two methods that uses statistical leverage scores in supervised learning and to-date there exists no active learning method that employ this concepts.

Statistical leverage scores reflect the influence of the rows or the columns in the matrix by capturing the dominant part of the matrix. Finding the columns (or rows) with highest leverage scores reveals the core information lurking in the parent matrix. Motivated from the recent work in the statistical leverage score sampling, we set out to find influential data points in a data distribution based on leverage scores, both in sequential-mode and batch-mode.

2.3 Submodularity in Machine Learning

2.3.1 Submodular set functions

In our batch-mode active learning method, we use submodular function maximization. Therefore, we shall concisely provide background information on submodularity.

Submodular set functions are defined as follows [54, 55, 56]:

Definition 2 (Submodularity)**.** Let $A \subseteq B \subseteq V$, where V denotes the ground set and let $x \in V \setminus B$ be an element. A set function $f : 2^V \to \mathbb{R}$ is called *submodular* if the following holds:

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) \tag{2.2}$$

Informally, if the marginal gain of adding a new item to a smaller set is larger than adding the same element to a bigger set, then the function is called submodular. Intuitively, this type of functions has diminishing returns property.

2.3.2 Applications of submodularity

Submodular property of functions has variety of applications in computer science and machine learning. Depending on the problem, both maximization and minimization of submodular functions are used to optimize an objective function. In tasks where the aim is to select a diverse set, maximization of submodular functions is used. Examples for these type of applications are maximizing the spread of influence in social networks [57], document summarization [58], feature selection [59], dictionary selection for sparse coding [60], information gathering [61], sensor placement [62], etc.

In literature, for capturing properties like coherence and regularization, submodular minimization is used. Example applications are image segmentation

[63], clustering [64], etc.

For active learning setting, there are applications of submodular functions for selecting points to query their labels. One example is [40], where they benefit from a notion called "Adaptive Submodularity" [55]. In adaptive submodularity, rather then an element has diminishing returns property in terms of marginal gain when added to a larger set (function itself being submodular), its conditional expected marginal benefit has diminishing returns property. Using this concept, submodularity has been used in adaptive planning, policy learning and stochastic optimization [55, 65, 66, 67].

Chapter 3

ALEVS: Sequential-mode Querying with Statistical Leverage Scores

In this chapter, we present an algorithm for sequential-mode pool-based active learning, called Active Learning by Statistical Leverage Scores (`ALEVS`)

3.1 Problem Set up

In binary classification the objective is to learn an accurate classifier, $h : \mathcal{X} \to \mathcal{Y}$, from a finite set of labeled training samples. Here, $\mathcal{X}$ denotes the instance space and $\mathcal{Y}$ is the set of class labels. The training data is represented as $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ wherein each example $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ is the class label of $\mathbf{x}_i$. We focus on the pool-based active learning setting, where few labeled examples are provided along with a large pool of unlabeled examples. We iteratively select one example, $\mathbf{x}_\mathrm{q}$ from the unlabeled pool and query its label. The labeling oracle $\mathcal{O}$ is assumed to be perfect and upon receiving the labeling request for $\mathbf{x}_\mathrm{q}$ the oracle responds with the true label y_q. We assume all examples have the same

cost of labeling. We denote the labeled set of training examples at iteration t with $\mathcal{D}_{\text{l}}^t$ and the set of unlabeled examples with $\mathcal{D}_{\text{u}}^t$. $\mathcal{D}_{\text{l}}^t$ comprises labeled $(\mathbf{x}_i, y_i)$ pairs; whereas $\mathcal{D}_{\text{u}}^t$ include only $\mathbf{x}_i$. The objective is to attain a good accuracy classifier h^* by minimizing the number of examples queried to the oracle thus reducing the labeling cost, with the constraint that only one example can be selected for querying in each iteration.

3.2 Proposed Methodology

Active Learning by Statistical Leverage Scores (`ALEVS`) is based on finding influential data examples in the pool based on statistical leverage scores. Below we describe `ALEVS` in steps:

Divide the pool based on class membership: At the iteration t, the classifier, h^t is exclusively trained with the labeled training examples $\mathcal{D}_{\text{l}}^t$ with a supervised method, the class membership of the unlabeled examples are predicted with h^t. At this step, the training examples are divided into two subsets based on class memberships and two separate feature matrices are formed accordingly. $\mathbf{X}_+^t$ is an $m \times d$ feature matrix, where the rows are the feature vectors of examples with positive class membership at iteration t. These examples are those whose true labels are known along with the examples for which the true labels are not known but are predicted to be in the positive class based on the prediction of h^t. $\mathbf{X}_-^t$ is similarly constructed from negatively predicted and labeled examples.

Compute kernel matrices of the training data: After the prediction of the labels of unlabeled data, `ALEVS` computes a kernel matrix over $\mathbf{X}_+^t$ and $\mathbf{X}_-^t$ separately. A kernel function, $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ gives the dot product of the input vectors in a typically higher dimensional transformed feature space, $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ [68]. Let $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$. For a given m number of examples, the kernel matrix is defined as $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{m \times m}$. `ALEVS` computes one kernel matrix on positive class examples, $\mathbf{X}_+^t$, which we will denote with $\mathbf{K}_+^t$. Similarly, for the negatively labeled feature matrix $\mathbf{X}_-^t$ a kernel matrix $\mathbf{K}_-^t$ is defined. These two

matrices encode the similarity of examples to other examples that are in the same class.

Query based on statistical leverage scores: `ALEVS` finds the example that imparts the strongest influence on the kernel matrices. To assess the importance of an example, we use statistical leverage scores of the kernel matrices $\mathbf{K}_+^t$ and $\mathbf{K}_-^t$. Leverage scores of an SPSD matrix are defined in Definition 1. If the leverage scores are computed on a low-rank, then their summation is equal to the low-rank parameter k, if computed on full rank, then the sum of the leverage scores is equal to m. To be able to compare leverage scores of examples computed on matrices with different m and k values, we use the scaled leverage scores:

$$\ell_i = \frac{m}{k} \|(\mathbf{U}_1)_{(i)}\|_2^2 \ . \tag{3.1}$$

which makes the average leverage score equals to 1 by scaling the summation to m. At iteration t `ALEVS` computes leverage scores for $\mathbf{K}_+^t$ and $\mathbf{K}_-^t$, and the unlabeled example that corresponds to the highest leverage score row in these matrices is selected for query:

$$\mathbf{x}_{\mathrm{q}} = \arg\max_{\mathbf{x}_i \in \mathcal{D}_{\mathrm{u}}^t} \ell_i \ . \tag{3.2}$$

Steps of the algorithm is illustrated in Fig. 3.1.

3.3 Selection of Target Rank

One important parameter is the target rank parameter k. The parameter is selected by setting a threshold on the sum of the top eigenvalues at each iteration. Given a threshold parameter τ that determines the proportion of variance explained by the top first k eigenvalues, the low-rank parameter k is selected as follows:

$$k^* = \arg\min_{k \in \{1,\ldots,m\}} \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq \tau \ . \tag{3.3}$$

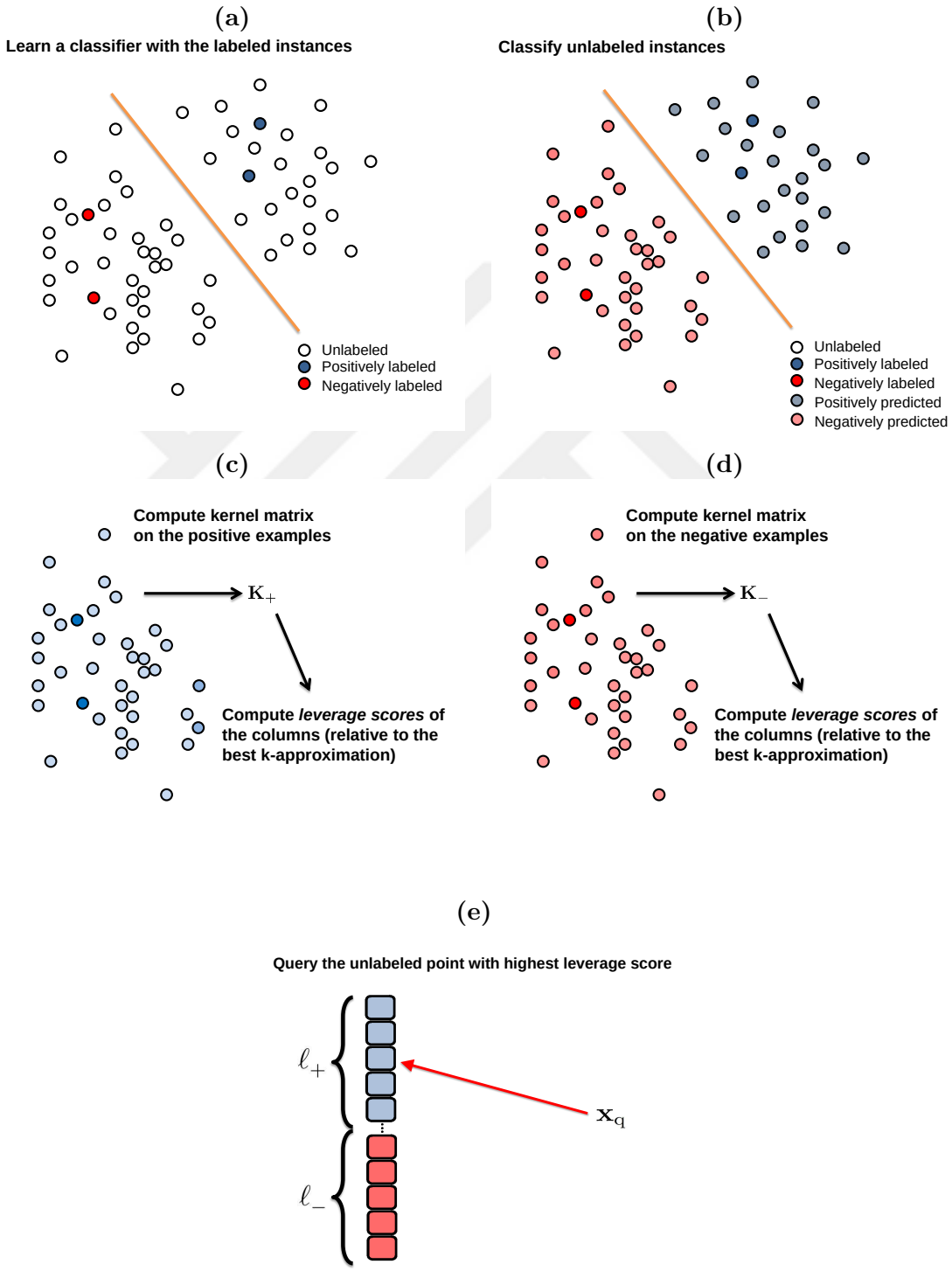

Figure 3.1: Illustration of the steps of `ALEVS` algorithm.

Leverage computation and target rank selection is summarized in `ComputeLeverage` (Algorithm 3) and `RankSelector` (Algorithm 2) algorithms respectively. The overall procedure of `ALEVS` is summarized in `ALEVS` (Algorithm 4) algorithm.

Algorithm 2 `RankSelector`

Input: λ: $m \times 1$ vector containing eigenvalues; τ: eigenvalue threshold.
Output: k: target rank.
$\lambda \leftarrow \text{sort}(\lambda,\text{'descend'})$
$k \leftarrow 1$
while $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} < \tau$ **do**
  $k \leftarrow k + 1$
end while

Algorithm 3 `ComputeLeverage`

Input: $\mathbf{K}$: $m \times m$ kernel matrix; τ: eigenvalue threshold.
Output: ℓ: leverage scores.
$\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$
$\lambda \leftarrow diag(\Sigma)$
$k \leftarrow \texttt{RankSelector}(\lambda,\tau)$
$\mathbf{U} = \begin{pmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{pmatrix}$, where $\mathbf{U}_1$ spans top k-eigenspace of $\mathbf{K}$ and is $m \times k$
for $i = 1$ **to** m **do**
  $\ell_i = \frac{m}{k} \|(\mathbf{U}_1)_{(i)}\|_2^2$
end for

Algorithm 4 `ALEVS`: Active Learning with Leverage Score Sampling

Input: $\mathcal{D}$: a training dataset of m instances; $\mathcal{O}$: labeling oracle; τ: eigenvalue threshold; p: kernel parameters.
Output: h^*: final classifier.
Initialize:
$\mathcal{D}_{\text{l}}^0$ // initial set of labeled instances
$\mathcal{D}_{\text{u}}^0 \leftarrow \mathcal{D} \setminus \mathcal{D}_{\text{l}}^0$ // the pool of unlabeled instances
$t \leftarrow 0$
repeat
——— **Classification** ———
$h^t \leftarrow \text{train}(D_{\text{l}}^t)$
$\hat{\mathbf{y}}_{\text{u}}^t \leftarrow \text{predict}(h^t, \mathcal{D}_{\text{u}}^t)$
——— **Sampling** ———
Based on $\hat{\mathbf{y}}_{\text{u}}^t$ and $\mathbf{y}_{\text{l}}^t$, construct $\mathbf{X}_+^t$ and $\mathbf{X}_-^t$
$\mathbf{K}_+^t \leftarrow \texttt{ComputeKernel}(\mathbf{X}_+^t, p)$
$\mathbf{K}_-^t \leftarrow \texttt{ComputeKernel}(\mathbf{X}_-^t, p)$
$\ell_+^t \leftarrow \texttt{ComputeLeverage}(\mathbf{K}_+^t, \tau)$
$\ell_-^t \leftarrow \texttt{ComputeLeverage}(\mathbf{K}_-^t, \tau)$
$\ell^t \leftarrow \ell_+^t \cup \ell_-^t$
$\mathbf{x}_{\text{q}}^t = \arg\max_{\mathbf{x}_j \in \mathcal{D}_{\text{u}}^t} \ell_j^t$
$y_{\text{q}}^t \leftarrow \text{query}(\mathcal{O}, \mathbf{x}_{\text{q}}^t)$
——— **Update** ———
$\mathcal{D}_{\text{l}}^{t+1} \leftarrow \mathcal{D}_{\text{l}}^t \cup (\mathbf{x}_{\text{q}}^t, y_{\text{q}}^t)$
$\mathcal{D}_{\text{u}}^{t+1} \leftarrow \mathcal{D}_{\text{u}}^t \setminus \mathbf{x}_{\text{q}}^t$
$t \leftarrow t + 1$
until stopping criterion
$h^* \leftarrow h^t$

Chapter 4

ALEVS Experimental Results

In this chapter we evaluate performance of ALEVS introduced in Chapter 3. We will first describe the competing methods that we compare ALEVS against, next describe the experimental set up, and finally present empirical results both in terms of accuracy and runtime.

4.1 Baselines and Compared Methods

We compare ALEVS with the following approaches: (1) Random Sampling: randomly selects query instances, (2) Uncertainty Sampling: selects the instance with maximal uncertainty [22], (3) Leverage sampling on all data (LevOnAll): computes the leverage score on the whole pool at the beginning of the iteration without paying attention to class membership and selects unlabeled queries in the order of their leverage scores, and (4) QUIRE: Active Learning by Querying Informative and Representative Examples [34]. In Uncertainty Sampling, to find the most uncertain unlabeled data point based on the SVM output, we estimate the posterior probabilities of each unlabeled instance with Platt's algorithm [69]. The most uncertain point is the one with maximal $(1 - p(y^* \mid \mathbf{x}))$; here y^* is the predicted class label for that point. LevOnAll baseline is designed to check

whether separating the examples based on their class membership has any value or not. The last method to compare, `QUIRE`, is chosen because it stands out among the state-of-the-art algorithms [34] and because an implementation is provided. `QUIRE` finds an instance that is both informative and representative by optimizing an objective function in a min-max view. `QUIRE` implementation is obtained from[1].

In our experiments we employ different kernel functions: linear, polynomial and Radial Basis Function (RBF) kernels. Over a set of data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{\mathrm{d}}$, the linear kernel matrix $\mathbf{K}$ is computed as follows:

$$\mathbf{K}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \ . \tag{4.1}$$

For polynomial kernel:

$$\mathbf{K}_{ij} = (\mathbf{x}_i \mathbf{x}_j^{\mathrm{T}} + c)^d \ . \tag{4.2}$$

In the above equation, c and d corresponds to the kernel parameters; the constant and the degree of the polynomial respectively.

RBF kernel matrix is defined by:

$$\mathbf{K}_{ij} = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2}{2\sigma^2}\right) \ , \tag{4.3}$$

where σ is a nonnegative real number that determines the scale of the kernel.

4.2 Datasets

To evaluate performance of `ALEVS` we run experiments on eight different datasets. The *digit1*, *g241c*, *USPS* datasets are from[2] [70]. The *spambase* and *letter* datasets are from [71]. The *letter* dataset is a multi-class dataset, we select a letter pair that are difficult to distinguish: *UvsV*. Similarly, we sample 3 and 5

[1] `http://lamda.nju.edu.cn/code_QUIRE.ashx`
[2] `http://olivier.chapelle.cc/ssl-book/benchmarks.html`

Table 4.1: Datasets, their dimensions and used parameters (Sequential-mode).

Dataset	Size	Dim.	+/-	Kernel	τ
digit1	1500	241	1.00	RBF, $\sigma = 2$	0.50
g241c	1500	241	1.00	Linear	0.50
UvsV	1577	16	1.10	RBF, $\sigma = 1$	0.75
USPS	1500	241	0.25	Polynomial $c = 1, d = 4$	0.50
twonorm	2000	20	1.00	RBF, $\sigma = 1$	0.50
ringnorm	2000	20	1.00	RBF, $\sigma = 1$	0.50
spambase	2000	57	0.66	RBF, $\sigma = 1$	0.75
3vs5	2000	784	1.20	RBF, $\sigma = 2$	0.75

digits from the *MNIST* dataset as *3vs5*, since they are one of the most confused pairs in the *MNIST* dataset [72]. Finally, *twonorm* and *ringnorm* are culled from[3] and[4] which are implementations of [73]. We use a random subsample of 2000 examples for *ringnorm*, *twonorm*, *spambase*, and *3vs5* because the running time for `QUIRE` was prohibitively long. The description of these datasets and the parameters used by the algorithm for each of the dataset are given in Table 4.1.

4.3 Experimental Setup

Each dataset is divided into training and held out test sets. We start with 4 randomly selected labeled examples, 2 from each class. At each iteration, the classifier is updated for all the methods with the training data, and the accuracies are calculated on the same held-out test data. In all experiments, an SVM classifier with RBF kernel is used with scale parameter set to 1. For each dataset the experiment is repeated 50 times with random splitting of the training and the test data and random initial selection of labeled examples. The accuracies reported in Fig. 4.1 and Fig. 4.2 are the average accuracies computed for these random trials. In calculating leverage scores we experiment with linear, polynomial and RBF kernels. We also experiment with different eigenvalue thresholds and kernel parameters. Here best performing cases are provided.

[3] `http://www.cs.toronto.edu/~delve/data/twonorm/desc.html`

[4] `http://www.cs.toronto.edu/~delve/data/ringnorm/desc.html`

4.4 Results and Discussion

4.4.1 Performance

Fig. 4.1 and Fig. 4.2 show the average classification accuracies of `ALEVS` and other approaches at each iteration of active sampling. Table 4.2 and Table 4.3 summarize the win, tie and lost counts of `ALEVS` versus each of the competing methods on the 1-sided paired sample t-test at the significance level of 0.05.

Table 4.2: Win/Tie/Loss counts for the first 50 iterations (Sequential-mode).

Dataset	vs.QUIRE	vs.LevOnAll	vs.Random	vs.Uncertainty
digit1	18/25/7	42/8/0	45/5/0	46/4/0
g241c	0/28/22	25/25/0	30/20/0	34/16/0
USPS	10/31/9	35/14/1	35/12/3	14/36/0
ringnorm	45/5/0	48/2/0	48/2/0	48/2/0
spambase	19/10/21	21/24/5	21/20/9	2/46/2
3vs5	1/41/8	39/11/0	42/8/0	46/4/0
UvsV	0/3/47	47/3/0	30/20/0	9/8/33
twonorm	49/1/0	50/0/0	50/0/0	50/0/0

Table 4.3: Win/Tie/Loss counts for iterations between 50 and 100 (Sequential-mode).

Dataset	vs.QUIRE	vs.LevOnAll	vs.Random	vs.Uncertainty
digit1	0/0/50	0/38/12	0/38/12	0/22/28
g241c	31/17/2	50/0/0	50/0/0	50/0/0
USPS	0/5/45	50/0/0	50/0/0	0/32/18
ringnorm	30/20/0	43/7/0	42/8/0	35/15/0
spambase	0/9/41	3/47/0	5/45/0	0/5/45
3vs5	3/15/32	44/6/0	31/19/0	2/19/29
UvsV	0/0/50	0/50/0	0/50/0	0/0/50
twonorm	50/0/0	50/0/0	50/0/0	50/0/0

The results indicate that `ALEVS` outperforms random sampling and uncertainty sampling approaches in most of the datasets. Exceptions to these are the *UvsV* dataset, for which `ALEVS` performs worse than uncertainty sampling and *spambase* dataset, where `ALEVS` performs as good as the uncertainty sampling but not better. `ALEVS`' performance is consistently better than random sampling in the

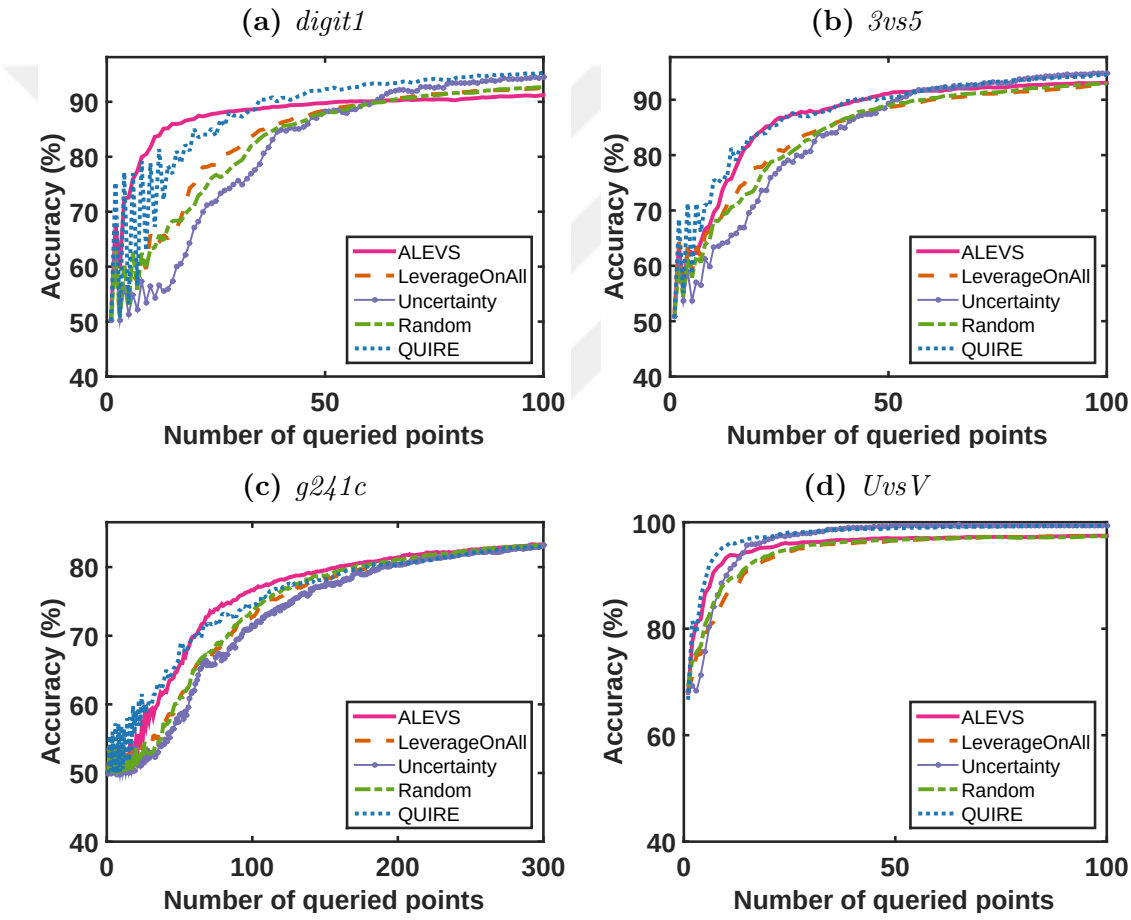

Figure 4.1: Comparison of `ALEVS` with other methods on classification accuracy - 1 (Sequential-mode).

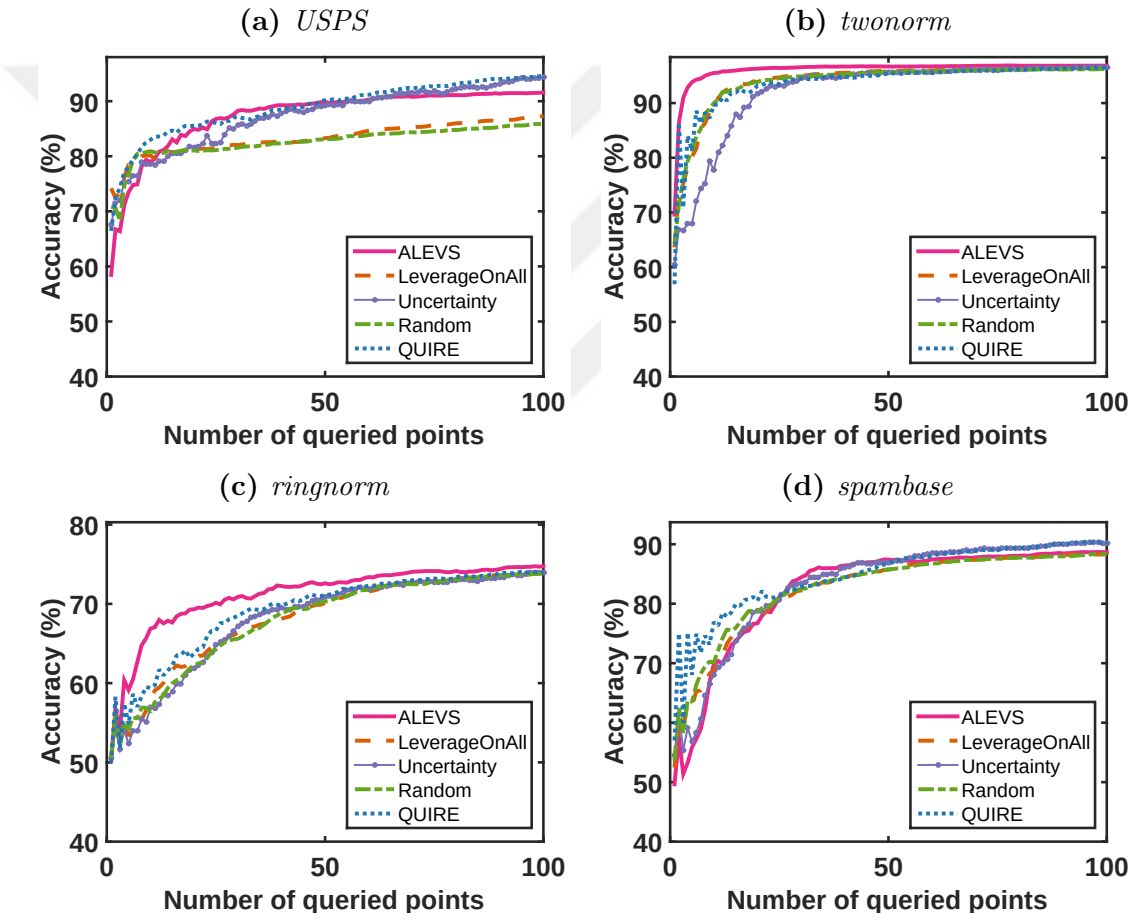

Figure 4.2: Comparison of `ALEVS` with other methods on classification accuracy - 2 (Sequential-mode).

first 50 iterations of active sampling (Table 4.2). For iterations between $50-100$, the two methods tie in *digit1*, *spambase* and *UvsV* datasets (Table 4.3).

Secondly, when comparing the performance of `ALEVS` against `QUIRE`, there are three different groups of datasets for which `ALEVS` performance varies. First group of datasets comprise *ringnorm* and *twonorm*; for which `ALEVS` decisively outperforms `QUIRE`. When tested in the second group of datasets, `ALEVS` either outperforms `QUIRE` or ties with it at a subset of the iterations. In *digit1*, `ALEVS` outperforms in the first 50 iterations. For the *g241c* dataset the reverse holds; `ALEVS` do worse than `QUIRE` in early iterations, then holds up in later stages. In the *3vs5* dataset `QUIRE` and `ALEVS` tie in most of the 50 iterations. There are also datasets, where ALEVS lags behind `QUIRE`. These include *UvsV*, *USPS* and *spambase*. For the *spambase* dataset, `ALEVS` shows promising performance around iterations 30 and 40 but it is not sustained in the later steps.

When compared against LevOnAll baseline, we observe that `ALEVS` consistently prevails. This shows that calculating the leverage scores within each class is better at finding influential data points than singling them out from the whole pool ignoring the class membership. Naturally, the underlying class structure has an effect on the leverage scores.

One observes that generally `ALEVS` manages to find effective examples for querying in the early iterations. Therefore, a strategy that combines `ALEVS` with a method that performs poorly in early iterations but do better in later iterations – such as uncertainty sampling – could lead to a strong active learner. This will be explored in the future work.

Lastly, to understand whether increase in the leverage scores correlates with increase in accuracy, we plot the scaled leverage scores pertaining to the queried examples across the active sampling iterations (Fig. 4.3). In this graph, a value of 1 corresponds to the case where the leverage scores of the kernel matrix are all uniform. When Fig. 4.3 and Fig. 4.4 are compared with Fig. 4.1 and Fig. 4.2, it is evident that the regions where there is a notable increase in accuracy overlaps with regions wherein the leverage scores ramp up. When the queried leverage

scores stabilize, the accuracy also stabilizes.

4.4.2 Effect of target rank

One parameter that has a large impact on the performance of `ALEVS` is the target low-rank parameter k. In this work, we adaptively select the value of k for negative and positive kernel matrices at each iteration by setting a threshold on the variance for top-k dimensional eigenspace as described in `RankSelector` algorithm. We further analyze the effect of τ by varying these thresholds; experimented on four datasets with three different τ values. Accuracies shown in Fig. 4.7 are averages computed over 10 random experiments. Selecting the full rank option for computing leverage scores does not necessarily provide the best performance. The low-rank representation ignores the noise and focus on the core dimensions that matters in the datasets. Different τ values results are the best choice for different datasets. For the datasets *digit1*, *twonorm*, *ringnorm*, $\tau = 0.5$ works best, whereas for *3vs5* threshold value 0.75 is a better choice and $\tau = 0.5$ is the worst choice. This difference is expected, as the eigenvalue spectrums of the matrices are different. Here, we selected τ by experimenting different choices of the thresholds. Other strategies will be explored in future work.

4.4.3 Runtime comparisons

We performed the experiments in Matlab on a computer with 2.6 GHz CPU (24-core) and 64 GB of memory running Ubuntu 14.04 LTS operating system.

The querying step of `ALEVS` involves the calculation of eigenvalue decomposition of the kernel matrices; however, in practice this does not cause a computational bottleneck. We summarize the average CPU times for selecting one example in a single iteration from the unlabeled data pool in Fig. 4.8 and Fig. 4.9. As the boxplots illustrate, we can see `ALEVS` is as fast as uncertainty sampling.

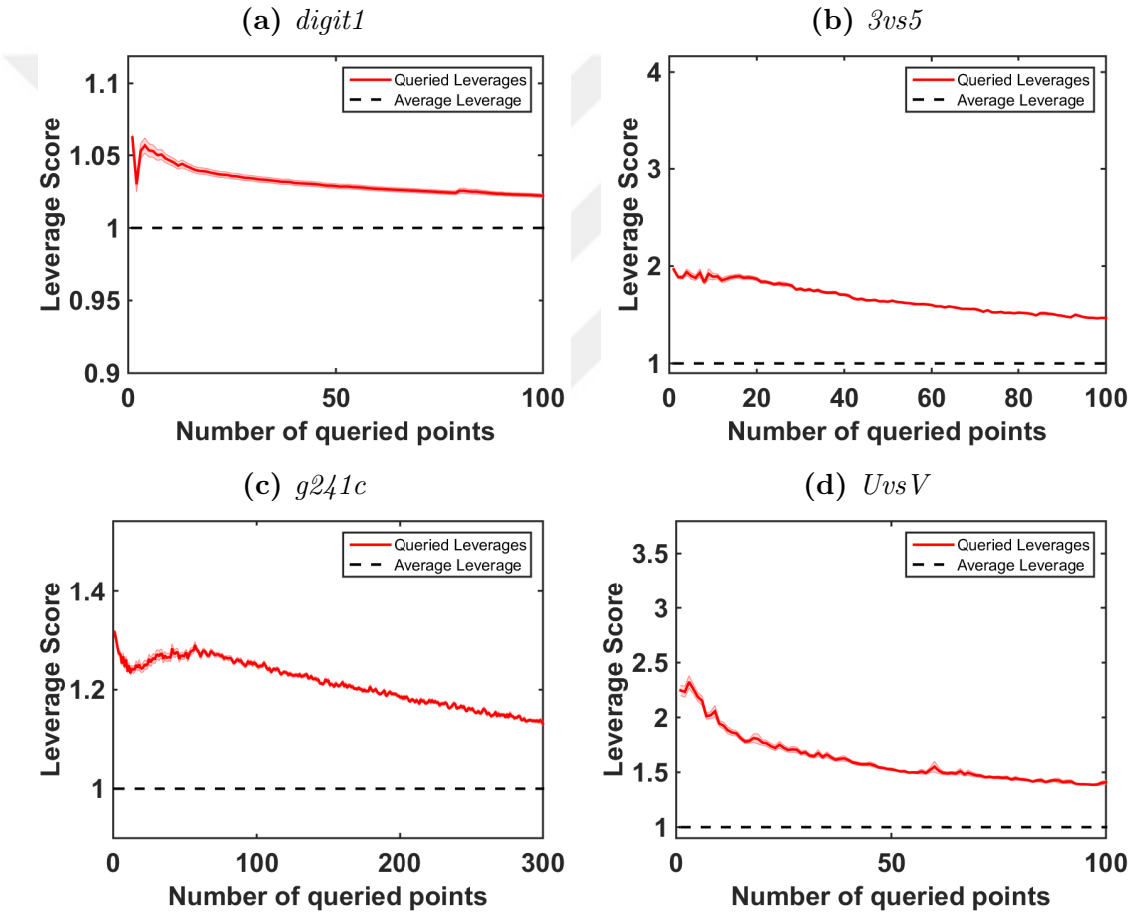

Figure 4.3: Comparison of queried leverage scores in each iteration with average leverage scores - 1 (Sequential-mode).

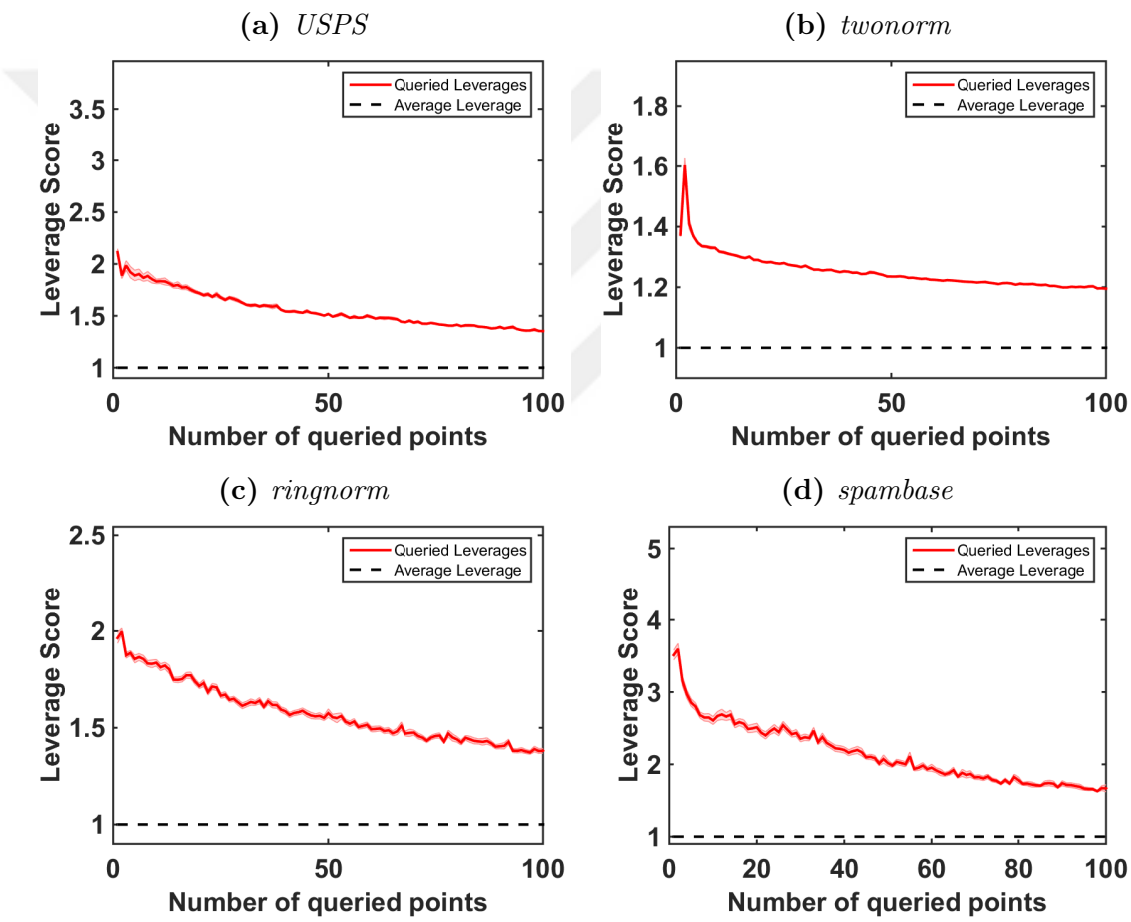

Figure 4.4: Comparison of queried leverage scores in each iteration with average leverage scores - 2 (Sequential-mode).

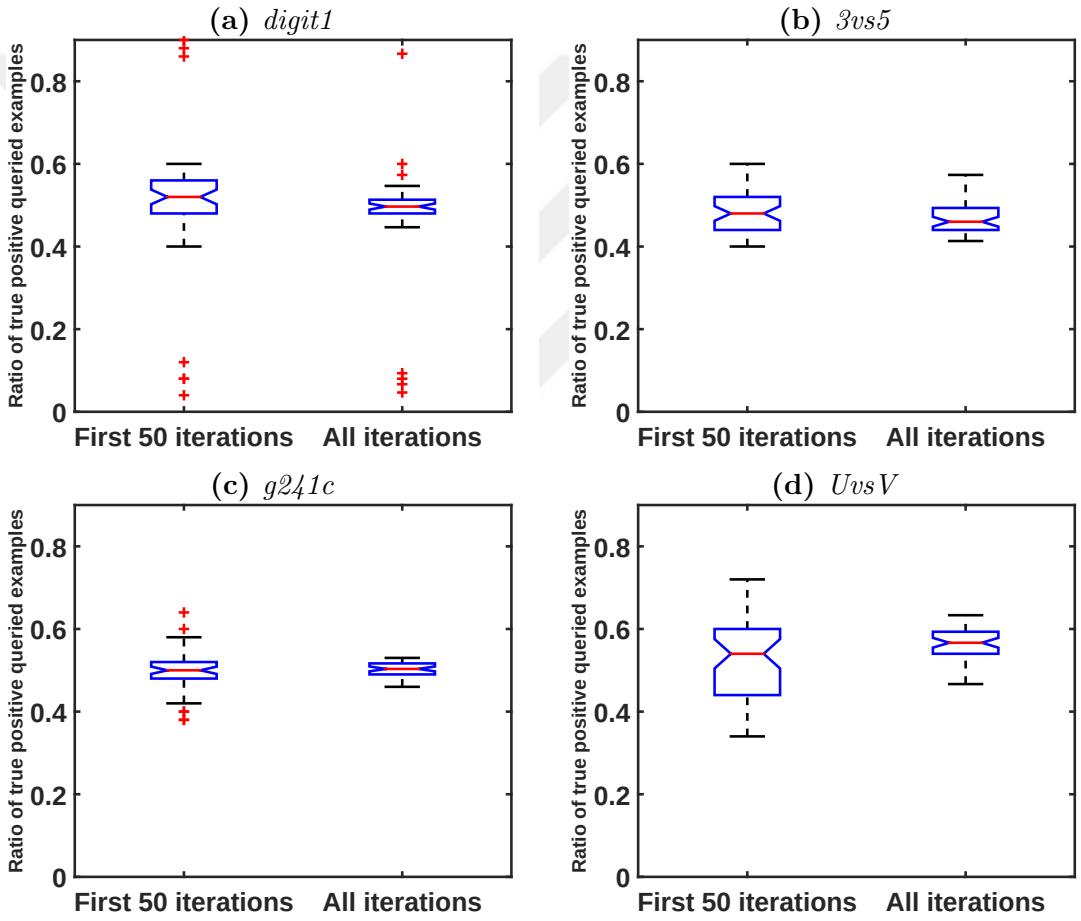

Figure 4.5: Ratio of true positive labels of queried examples for `ALEVS` - 1 (Sequential-mode).

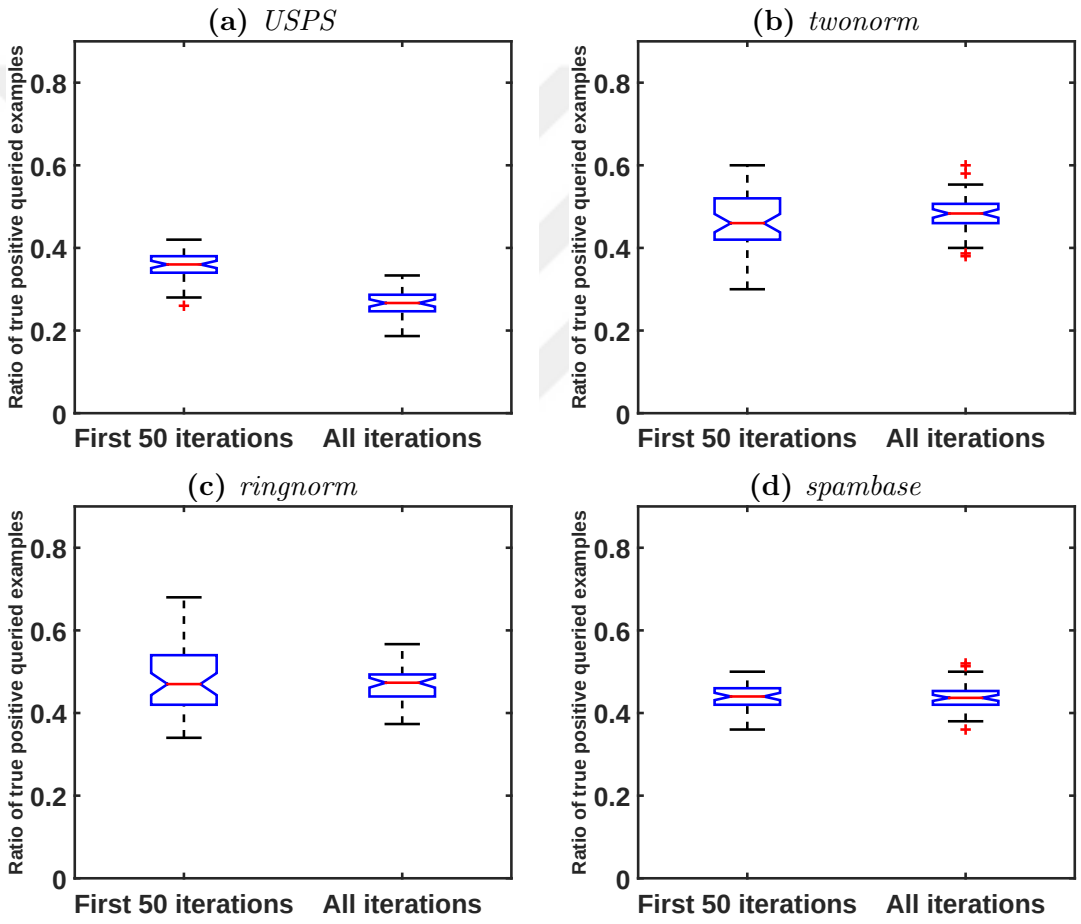

Figure 4.6: Ratio of true positive labels of queried examples for `ALEVS` - 2 (Sequential-mode).

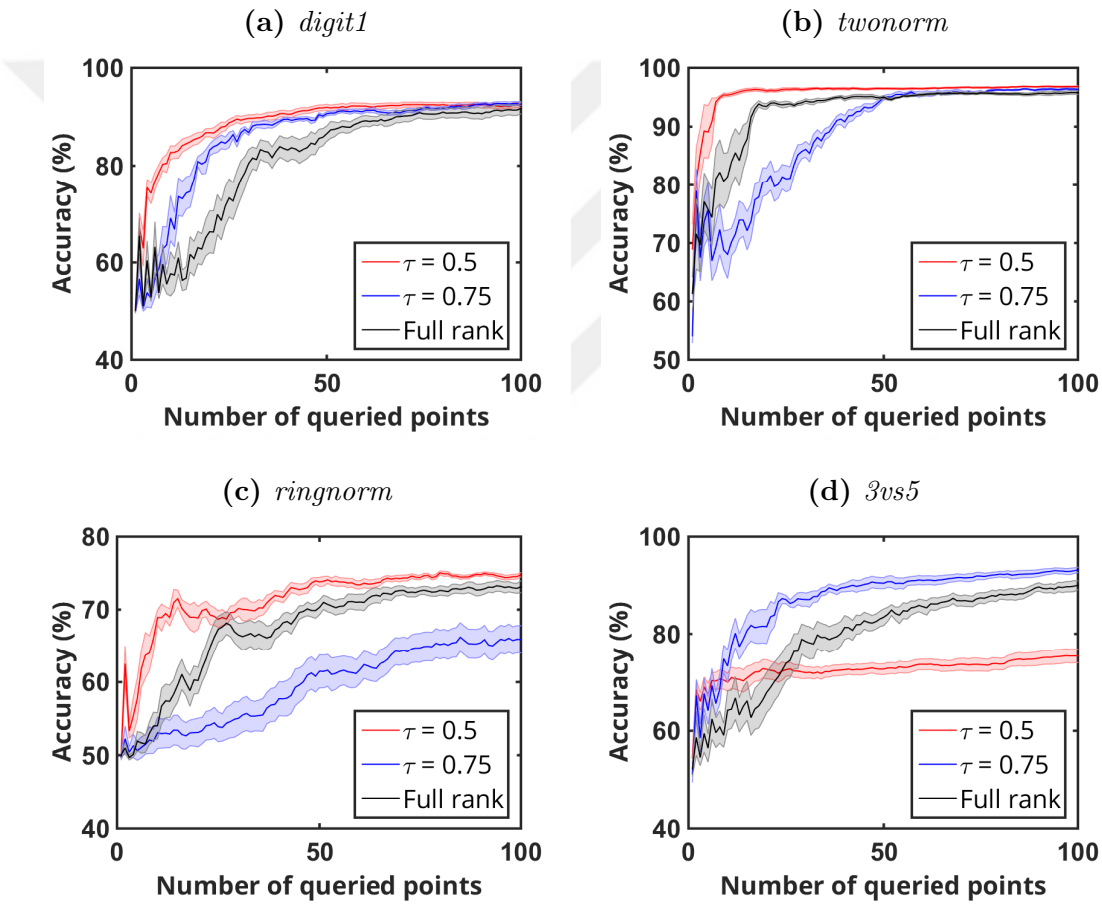

Figure 4.7: Effect of target rank k selected by threshold τ on test set accuracy for `ALEVS` (Sequential-mode).

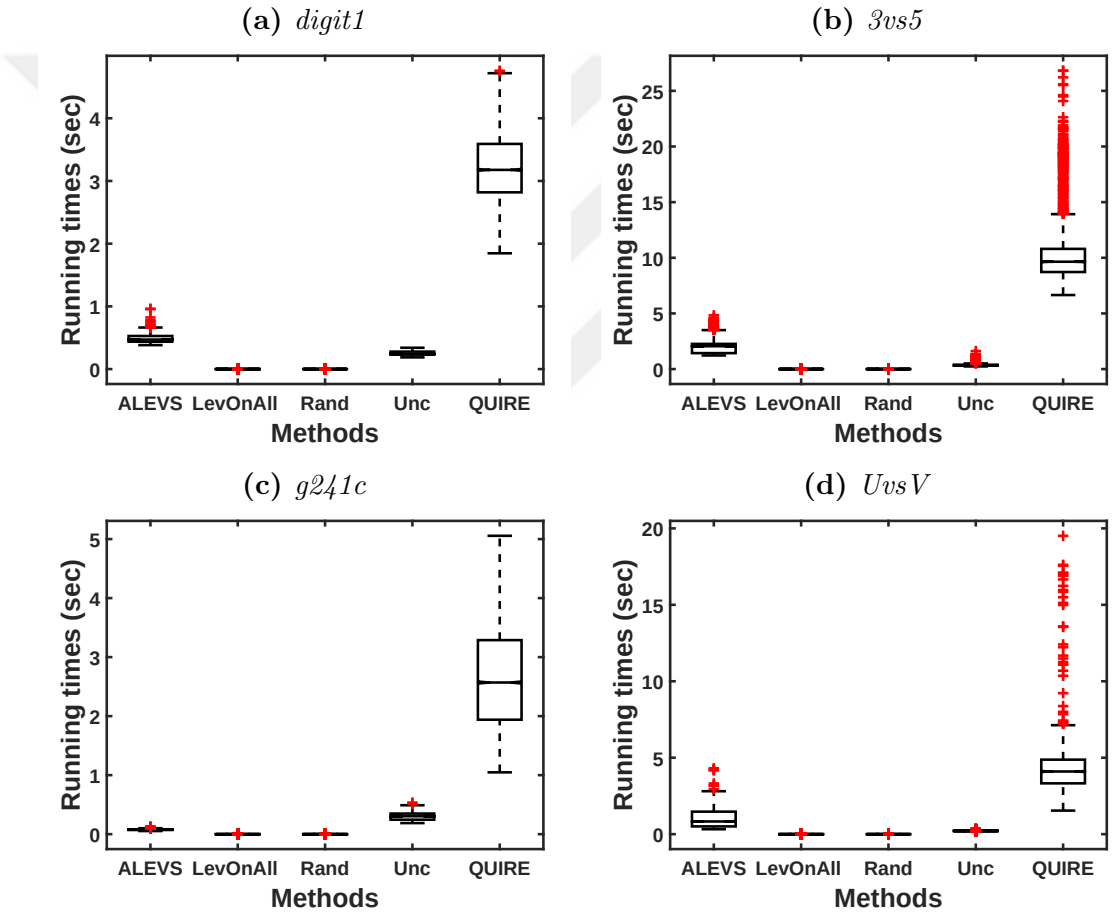

Figure 4.8: Comparison of `ALEVS` with other methods on running times - 1 (Sequential-mode).

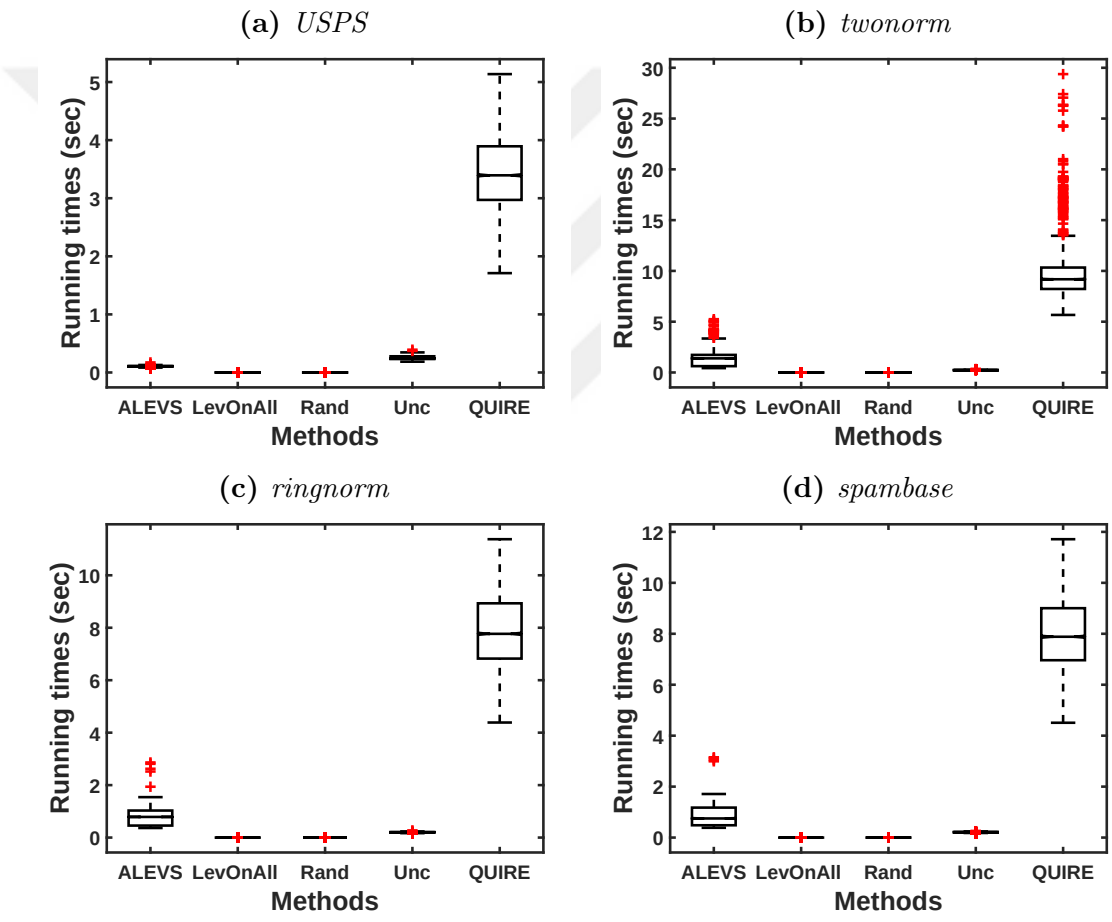

Figure 4.9: Comparison of `ALEVS` with other methods on running times - 2 (Sequential-mode).

Chapter 5

DBALEVS: Batch-mode Querying with Statistical Leverage Scores

In this chapter, we present an algorithm for selecting a batch of unlabeled examples, called Diverse Batch-mode Active Learning by Statistical Leverage Scores (DBALEVS).

5.1 Problem Set up

The problem set up is the same as ALEVS: we deal with a binary supervised classification setting in the pool-based active learning with a perfect oracle. Here, instead of selecting a single example at each iteration, we consider the case where the active learner is allowed to select b unlabeled examples at each iteration and request labels of these examples at once. Let there be m examples in the unlabeled pool; batches of size $b < m$ are sequentially picked where b is specified a priori by the user. We will refer the set of examples queried at iteration t as S_q^t. Similar to the sequential-mode, in the batch mode setting our aim is to attain a

good accuracy classifier h^* by minimizing the total number of examples queried thereby reducing the labeling cost.

5.2 Proposed Methodology

For selecting b examples at each iteration, we need to evaluate the quality of possible batches. A high quality batch should contain highly influential examples so that the data distribution is spanned accurately. Even though some of the examples can be highly influential on an individual basis, they might contain redundant information. Therefore, the batch should also a diverse set of examples. We encode these properties in a set scoring function and use it to select batches at each iteration.

5.2.1 Set scoring function for batch selection

Similar to `ALEVS`, the influence of each example is assessed by the statistical leverage scores. The sum of leverage scores assess the total usefulness of a set of examples. To select a diverse set, we incorporate a term that penalizes the selection of examples that are similar to each other. For evaluating the similarity of examples, we use the kernel function. Let $k(i, j)$ denote the kernel function evaluated for the i and j example pair. If the i and j are dissimilar to each other, the kernel value will be small. If the two examples are strongly similar the kernel value will be large.

We define set scoring function, $F : 2^V \rightarrow \mathbb{R}$, as follows:

Definition 3 (Set scoring function)**.** Given a set S that is a subset of ground set V, $S \subseteq V$ and ℓ_i denoting the leverage score of point i with low-rank parameter k, and $k(i, j)$ denoting the kernel function evaluation of points i and j with $1 \geq k(i, j) \geq 0$, diversity trade-off parameter $\alpha \in [0, 1]$, and maximum selectable set size as $M \geq |S|$, the set function for computing the score of a set is defined

as follows:

$$F(S) = \sum_{i \in S} (\ell_i + 1) - \frac{\alpha}{M} \sum_{\substack{i,j \in S \\ i \neq j}} k(i,j) \tag{5.1}$$

The first part of this function evaluates the influence of examples. The second part of the function penalizes the selection of highly similar instances. Here S is a set, ℓ_i denotes the leverage score of example i. The influence of the diversity term can be adjusted by the trade-off parameter α. Setting $\alpha = 0$ translates into selecting examples with the highest leverage scores. M is the size of the maximum selectable set, and is therefore a cardinality constraint. The first term in the equation and M is critical in solving this problem in an efficient way.

5.2.2 Selection strategy

At each iteration we will select b examples from the unlabeled data pool with the best set function value. In optimal batch-mode selection, we would like to select a batch that maximizes the F:

$$\begin{aligned} S^* &= \arg\max F(S) \\ s.t. \quad & |S| = b \end{aligned} \tag{5.2}$$

where b denotes the size of the batch and S^* is the optimal solution. This is a subset selection problem and except for small numbers of n and b the exhaustive search for the optimal batch will be intractable. To tackle this computational challenge, we first prove that our proposed set scoring function is submodular. Although submodular maximization is also NP-hard in general [74] we will resort to a greedy algorithm that returns a solution guaranteed to be close to the optimal solution. Nemhauser et al. [75] show that for a submodular non-decreasing set function satisfying $F(\emptyset) = 0$, a greedy algorithm finds a solution close to the optimal value within a bound (Theorem 1).

Theorem 1 (Nemhauser et al., 1978 [56, 75])**.** *For a monotone, non-negative, submodular function $f : 2^V \rightarrow \mathbb{R}$, and a cardinality constraint b, the greedy*

approximation yields to:

$$f(S_b) \geq (1 - \frac{1}{e}) \max_{|S| \leq b} f(S) \tag{5.3}$$

where S_b denotes the greedily selected set with cardinality b, $|S_b| = b$, and e is the base of natural logarithm.

The greedy algorithm in Algorithm 5, adds elements to the solution that gives the maximum increase at each step.

Algorithm 5 `GreedyAlgorithm`

Input: f: a set function; b: cardinality constraint; V ground set.
Output: S: selected set with $|S| = b$.
$S_0 = \emptyset$
$i \leftarrow 1$
while $i \leq b$ **do**
  $S_i \leftarrow S_{i-1} \cup \arg\max_{x \in V \setminus S_{i-1}} f$
  $i \leftarrow i + 1$
end while
$S \leftarrow S_i$

To be able to use this greedy algorithm with aforementioned approximation bound, we need to show that F is a submodular, monotonically non-decreasing and non-negative function.

Proposition 1 (Submodularity)**.** *F is submodular.*

Proof. Consider any two set functions A and B such that $A \subseteq B \subseteq V$ and an example $x \in V \setminus B$. For F to be submodular as defined in Definition 2, the following should hold:

$$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \ . \tag{5.4}$$

Using Definition 3 for F, consider

$$\begin{aligned} &F(A \cup \{x\}) - F(A) \\ &= \Big(\sum_{i \in A \cup \{x\}} (\ell_i + 1) - \frac{\alpha}{M} \sum_{\substack{i,j \in A \cup \{x\} \\ i \neq j}} k(i,j) \Big) - \Big(\sum_{i \in A} (\ell_i + 1) - \frac{\alpha}{M} \sum_{\substack{i,j \in A \\ i \neq j}} k(i,j) \Big) \end{aligned}$$

$$= \Big(\sum_{i \in A} \ell_i + \ell_x + |A| + 1 - \frac{\alpha}{M} \Big(\sum_{\substack{i,j \in A \\ i \neq j}} k(i,j) + \sum_{i \in A} k(x,i) \Big) \Big) - \Big(\sum_{i \in A} \ell_i + |A| - \frac{\alpha}{M} \sum_{\substack{i,j \in A \\ i \neq j}} k(i,j) \Big)$$

Rearranging the terms we end up with the following expression:

$$F(A \cup \{x\}) - F(A) = \Big(\ell_x + 1 - \frac{\alpha}{M} \sum_{i \in A} k(x,i) \Big)$$

If we do the same simplification for the right hand side, we arrive to a similar expression for set B. Therefore,

$$\begin{aligned} & F(A \cup \{x\}) - F(A) - (F(B \cup \{x\}) - F(B)) \\ &= \Big(\ell_x + 1 - \frac{\alpha}{M} \sum_{i \in A} k(x,i) \Big) - \Big(\ell_x + 1 - \frac{\alpha}{M} \sum_{i \in B} k(x,i) \Big) \\ &= \frac{\alpha}{M} \sum_{i \in B} k(x,i) - \frac{\alpha}{M} \sum_{i \in A} k(x,i) \\ &= \frac{\alpha}{M} \sum_{i \in B \setminus A} k(x,i) \end{aligned}$$

Since $k(i,j) \geq 0$ and $\alpha \in [0,1]$ and $M > 0$, $\frac{\alpha}{M} \sum_{i \in B \setminus A} k(x,i) \geq 0$. Therefore, $F(A \cup \{x\}) - F(A) \geq (F(B \cup \{x\}) - F(B))$. Hence, F is submodular. □

To apply the greedy algorithm with an approximation guarantee, we need to also show that F is a monotonically non-decreasing and non-negative function under reasonable conditions.

Definition 4 (Monotonicity)**.** $f : 2^V \to \mathbb{R}$ is *monotonically non-decreasing* set function if $f(A) \leq f(B)$ for every $A \subseteq B \subseteq V$.

Proposition 2 (Monotonicity)**.** *F is a monotonically non-decreasing set function when input sets can be at most size M.*

Proof. Consider two arbitrary sets $A \subseteq B \subseteq V$ and let $t = |B| - |A|$, and $|A| \leq M$ and $|B| \leq M$. We need to show that the following holds:

$$F(A) \leq F(B) \tag{5.5}$$

Using Definition 3 for F:

$$\begin{aligned}
F(A) - F(B) &= \Big(\sum_{i\in A}(\ell_i + 1) - \frac{\alpha}{M}\sum_{\substack{i,j\in A\\ i\neq j}} k(i,j)\Big) - \Big(\sum_{i\in B}(\ell_i + 1) - \frac{\alpha}{M}\sum_{\substack{i,j\in B\\ i\neq j}} k(i,j)\Big) \\
&= \Big(\sum_{i\in A}\ell_i + |A| - \frac{\alpha}{M}\sum_{\substack{i,j\in A\\ i\neq j}} k(i,j)\Big) - \Big(\sum_{i\in B}\ell_i + |B| - \frac{\alpha}{M}\sum_{\substack{i,j\in B\\ i\neq j}} k(i,j)\Big) \\
&= \frac{\alpha}{M}\Big(\sum_{\substack{i,j\in B\\ i\neq j}} k(i,j) - \sum_{\substack{i,j\in A\\ i\neq j}} k(i,j)\Big) - \sum_{i\in B}\ell_i + \sum_{i\in A}\ell_i - |B| + |A| \\
&= \frac{\alpha}{M}\Big(\sum_{\substack{i\in B\setminus A\\ j\in A}} k(i,j) + \sum_{\substack{i,j\in B\setminus A\\ i\neq j}} k(i,j)\Big) - \sum_{i\in B\setminus A}\ell_i - t
\end{aligned}$$

The leftmost summation calculated over $t\,|A|$ terms and the second one sums over t^2 terms. Using the fact $k(i,j) \leq 1$, these terms can be at most $t\,|A|$ and t^2, respectively. Additionally, since $0 \leq \ell_i \leq 1$ the minimum value that $\sum_{i\in B\setminus A}\ell_i$ can take is 0. Then the following holds

$$\begin{aligned}
F(A) - F(B) &\leq \frac{\alpha}{M}\Big(t\,|A| + t^2\Big) - t \\
&\leq t\Big(\frac{\alpha}{M}(|A| + t) - 1\Big) \\
&\leq t(\alpha - 1) \\
&\leq 0
\end{aligned}$$

From second line to third line we used the fact that, $|B| \leq M$. $F(A) - F(B) \leq 0$; therefore, F is monotonically decreasing for sets with sizes smaller than M. □

Proposition 3 (Non-negativity)**.** *F is a non-negative set function.*

Proof. For F to be non-negative, the following statement should hold for sets smaller than M:

$$\forall S \subseteq V, F(S) \geq 0$$

$F(S)$ is defined as follows:

$$
\begin{aligned}
F(S) &= \sum_{i \in S} (\ell_i + 1) - \frac{\alpha}{M} \sum_{\substack{i,j \in S \\ i \neq j}} k(i,j) \\
&= \sum_{i \in S} \ell_i + |S| - \frac{\alpha}{M} \sum_{\substack{i,j \in S \\ i \neq j}} k(i,j) \\
&\geq 0 + |S| - \frac{\alpha}{M} |S| \\
&\geq |S| \left(1 - \frac{\alpha}{M}\right)
\end{aligned}
$$

Since $\alpha \in [0,1]$,$(1 - \frac{\alpha}{M}) \geq 0$; thereby, $F(S) \geq 0$. This concludes the proof that $F(S)$ is non-negative. □

5.2.3 Querying strategy

The overall procedure for querying a batch is similar to `ALEVS`. First, the labeled and unlabeled pool division is made based on class labels. For labeled examples the true labels are used, whereas for unlabeled examples the predicted class labels are used. At the iteration t, the classifier, h^t is exclusively trained with the labeled training examples $\mathcal{D}_l^t$ with a supervised method, and the class membership of the unlabeled examples are predicted with h^t. The examples whose true labels are known along with the instances for which the true labels are not known but are predicted to be in the positive class based on the prediction of h^t form a positive class group, $\mathbf{X}_+^t$. $\mathbf{X}_-^t$ is similarly constructed from negatively predicted and labeled examples.

Having divided the pool based on class memberships, the kernel matrices for each class are computed. The kernel matrix is used both for computing the leverage scores and for evaluating the set scoring function, in other words $\mathbf{K}_+^t$ is formed using $\mathbf{X}_+^t$, and $\mathbf{K}_-^t$ is formed using $\mathbf{X}_-^t$. Then leverage scores are computed using Definition 1 for each class, without scaling the leverage scores with $\frac{m}{k}$.

`DBALEVS` selects half of the batch from the positive side, and half of the points from the negative side. For this selection, the method uses the set scoring function

(Definition 3) and greedy maximization (Algorithm 5) on each class. For greedy maximization, the method uses the available labeled data for positive ($\mathbf{L}_{+}^{t}$) and negative ($\mathbf{L}_{+}^{t}$) class as the initial set. The intuition behind keeping track of past data is that, the method will be forced to select a set which is also diverse with respect to the already labeled examples.

The modified greedy maximization is summarized in Algorithm 6 and the procedure of batch-mode selection is summarized in Algorithm 7.

Algorithm 6 `B-GreedyAlgorithm`

Input: F: set scoring function in Definition 3; b: batch size; A: initial set; V: ground set; ℓ: leverage scores; $\mathbf{K}$: kernel matrix; α: diversity parameter for F.
Output: S: selected set.
$S_0 \leftarrow A$
$M \leftarrow |A| + b$
$i \leftarrow 1$
while $i \leq b$ **do**
  $S_i \leftarrow S_{i-1} \cup \arg\max_{x \in V \setminus S_{i-1}} F(\ell, \mathbf{K}, \alpha, M)$
  $i \leftarrow i + 1$
end while
$S \leftarrow S_i \setminus A$

Algorithm 7 `DBALEVS`: Diverse Batch-mode Active Learning with Leverage Score Sampling

Input: $\mathcal{D}$: a training dataset of m instances; $\mathcal{O}$: labeling oracle; τ: eigenvalue threshold; p: kernel parameters; F: set scoring function in Definition 3; b: batch size; α: diversity trade-off parameter for F.
Output: h^*: final classifier.
Initialize:
$\mathcal{D}_{\mathrm{l}}^0$ // initial set of labeled instances
$\mathcal{D}_{\mathrm{u}}^0 \leftarrow \mathcal{D} \setminus \mathcal{D}_{\mathrm{l}}^0$ // the pool of unlabeled instances
$t \leftarrow 0$
repeat
——— **Classification** ———
$h^t \leftarrow \mathrm{train}(D_{\mathrm{l}}^t)$
$\hat{\mathbf{y}}_{\mathrm{u}}^t \leftarrow \mathrm{predict}(h^t, \mathcal{D}_{\mathrm{u}}^t)$
——— **Sampling** ———
Based on $\hat{\mathbf{y}}_{\mathrm{u}}^t$ and $\mathbf{y}_{\mathrm{l}}^t$, construct $\mathbf{X}_+^t$ and $\mathbf{X}_-^t$
Based on $\mathbf{y}_{\mathrm{l}}^t$, construct $\mathbf{L}_+^t$ and $\mathbf{L}_-^t$ //labeled class matrices
$\mathbf{K}_+^t \leftarrow \texttt{ComputeKernel}(\mathbf{X}_+^t, p)$
$\mathbf{K}_-^t \leftarrow \texttt{ComputeKernel}(\mathbf{X}_-^t, p)$
$\ell_+^t \leftarrow \texttt{ComputeLeverage}(\mathbf{K}_+^t, \tau)$
$\ell_-^t \leftarrow \texttt{ComputeLeverage}(\mathbf{K}_-^t, \tau)$
$S_{\mathrm{q}}^+ \leftarrow \texttt{B-GreedyAlgorithm}(F, b/2, \mathbf{L}_+^t, \ell_+^t, \mathbf{K}_+^t, \alpha)$
$S_{\mathrm{q}}^- \leftarrow \texttt{B-GreedyAlgorithm}(F, b/2, \mathbf{L}_-^t, \ell_-^t, \mathbf{K}_-^t, \alpha)$
$S_{\mathrm{q}}^t \leftarrow S_{\mathrm{q}}^+ \cup S_{\mathrm{q}}^-$
$\mathbf{y}_{\mathrm{q}}^t \leftarrow \mathrm{query}(\mathcal{O}, S_{\mathrm{q}}^t)$
——— **Update** ———
$\mathcal{D}_{\mathrm{l}}^{t+1} \leftarrow \mathcal{D}_{\mathrm{l}}^t \cup (S_{\mathrm{q}}^t, \mathbf{y}_{\mathrm{q}}^t)$
$\mathcal{D}_{\mathrm{u}}^{t+1} \leftarrow \mathcal{D}_{\mathrm{u}}^t \setminus S_{\mathrm{q}}^t$
$t \leftarrow t + 1$
until stopping criterion
$h^* \leftarrow h^t$

Chapter 6

DBALEVS Experimental Results

In this chapter we evaluate performance of DBALEVS introduced in Chapter 5. We will first describe the competing methods that we compare DBALEVS against, next describe the experimental set up, and finally present empirical results both in terms of accuracy and runtime.

6.1 Baselines and Compared Methods

We compare DBALEVS with the following approaches: (1) Random Sampling: randomly selects query instances, (2) Uncertainty Sampling: selects a batch of instances with maximal uncertainty [22], (3) Top-Lev: computes the leverage score on the whole pool at the beginning of the iteration without paying attention to class membership and selects a batch of unlabeled examples for querying in the order of their leverage scores, and (4) NearOpt: Near-Optimal Batch Mode Active Learning and Adaptive Submodular Optimization [40]. In Uncertainty Sampling, to find the most uncertain unlabeled data point based on the SVM output, we estimate the posterior probabilities of each unlabeled instance with Platt's algorithm [69]. The most uncertain point is the one with maximal $(1 - p(y^* \mid \mathbf{x}))$; here y^* is the predicted class label for that point. Top-Lev baseline is designed

to check whether separating the examples based on their class membership has any value or not and also to see the effect of diversification. The last method to compare, `NearOpt`, is chosen because it stands out among the state-of-the-art algorithms [40] and because an implementation is provided. `NearOpt` selects a batch of instances using adaptive submodular optimization. `NearOpt` implementation is obtained from[1].

6.2 Kernel Functions

Since the set scoring function defined in Definition 3 requires $0 \leq k(i,j) \leq 1$, only kernel function that is used is RBF kernel. RBF kernel function is defined by Equation 4.3 in sequential-mode experiments.

6.3 Datasets

To evaluate performance of `DBALEVS` we run experiments on six different datasets. *autos*, *hardware* and *sport* are sampled from 20-newsgroups dataset[2], we used a bag-of-words representation for features. These subtopics are picked so that the classes are relevant, hence differentiation between these classes is harder. *autos* dataset have *rec.autos* and *rec.motorcycles* topics, *hardware* dataset have *comp.sys.ibm.pc.hardware* and *comp.sys.mac.hardware topics* and lastly *sport* dataset have *rec.sport.baseball* and *rec.sport.hockey* topics. Similarly, we sample 3-5 and 4-9 digit pairs from the *MNIST* dataset as *3vs5* and *4vs9*, since they are two of the most confused pairs in the mentioned dataset [72]. Finally, *ringnorm* is culled from[3] which is an implementation of [73]. The description of these datasets and the parameters used by the algorithm for each of the dataset are given in Table 6.1.

[1] `https://www.inf.ethz.ch/~chenyux/icml13/bmal-src.zip`
[2] `http://qwone.com/~jason/20Newsgroups/`
[3] `http://www.cs.toronto.edu/~delve/data/ringnorm/desc.html`

Table 6.1: Datasets, their dimensions and used parameters (Batch-mode).

Dataset	Size	+/-	Kernel	τ	α
Autos	1986 x 11009	1.00	RBF, $\sigma = 1$	0.50	1
Hardware	1945 x 9877	1.00	RBF, $\sigma = 1$	0.50	1
Sport	1993 x 11148	1.00	RBF, $\sigma = 1$	0.50	1
Ringnorm	7400 x 20	1.00	RBF, $\sigma = 1$	0.50	0.1
3vs5	13454 x 784	1.20	RBF, $\sigma = 2$	0.75	1
4vs9	13782 x 784	1.20	RBF, $\sigma = 2$	0.75	1

6.4 Experimental Setup

Each dataset is divided into training and held out test sets. We start with 4 randomly selected labeled examples, 2 from each class. At each iteration, the classifier is updated for all the methods with the training data, and the accuracies are calculated on the same held-out test data. In all experiments, an SVM classifier with RBF kernel is used with scale parameter set to 1. For each dataset the experiment is repeated 10 times with random splitting of the training and the test data and random initial selection of labeled examples. The accuracies reported in Fig. 6.1 are the average accuracies computed for these random trials. In calculating leverage scores we experiment only with RBF kernel. We also experiment with different eigenvalue thresholds and kernel parameters. Here best performing cases are provided. For optimizing the set scoring function in Definition 3, we used submodular function optimization toolbox [76] downloaded from[4]. We performed the experiments in Matlab on a computer with 2.6 GHz CPU (24-core) and 64 GB of memory running Ubuntu 14.04 LTS operating system.

6.5 Results and Discussion

Fig. 6.1 shows the average classification accuracies of `DBALEVS` and other approaches at each iteration of active sampling. Table 6.2 summarizes the win, tie

[4] `https://las.inf.ethz.ch/sfo/`

and lost counts of DBALEVS versus each of the competing methods on the 1-sided paired sample t-test at the significance level of 0.05.

Table 6.2: Win/Tie/Loss counts (Batch-mode).

DATASET	vs.NEAROPT	vs.TOP-LEV	vs.RANDOM	vs.UNCERTAINTY
ringnorm	15/5/0	0/2/18	16/4/0	20/0/0
autos	15/15/0	22/8/0	10/20/0	19/11/0
hardware	21/9/0	22/8/0	20/10/0	23/7/0
sport	16/14/0	26/4/0	19/11/0	23/7/0
4vs9	18/2/0	19/1/0	17/3/0	18/2/0
3vs5	16/4/0	15/5/0	16/3/1	17/3/0

The results indicate that DBALEVS outperforms random sampling and uncertainty sampling approaches in all of the datasets. One interesting observation is, uncertainty sampling performs poorly in the batch-mode setting. Main cause for this performance is that, uncertainty sampling is highly depending on initial hypothesis. If initial hypothesis formed from initially labeled examples is not good, then the query selection is affected from it by choosing non-informative instances in general. Furthermore, random sampling is performing not bad and outperforming uncertainty sampling in all of the datasets. Random sampling baseline performing in this manner is also noted by [39], which addresses random sampling strategy as "unbiased label selection procedure". Random sampling performance for batch sizes larger than 5 is not noted in aforementioned research. By enlarging the size of the batch, random sampling can select points around the dataset since there is no constraint on the query region.

In addition to previous argument, when comparing the performance of DBALEVS against NearOpt, it can be observed that our proposed method is performing better than the compared method. For all of the datasets, DBALEVS wins over NearOpt except for some ties between them and we do not observe a win for NearOpt. Hence our proposed method is proven to be superior to the compared method on these datasets. It is also important to note that the performances of NearOpt and random sampling are similar to each other.

When compared against Top-Lev baseline, we observe that DBALEVS consistently prevails in general. One exception for this is *ringnorm* dataset, which

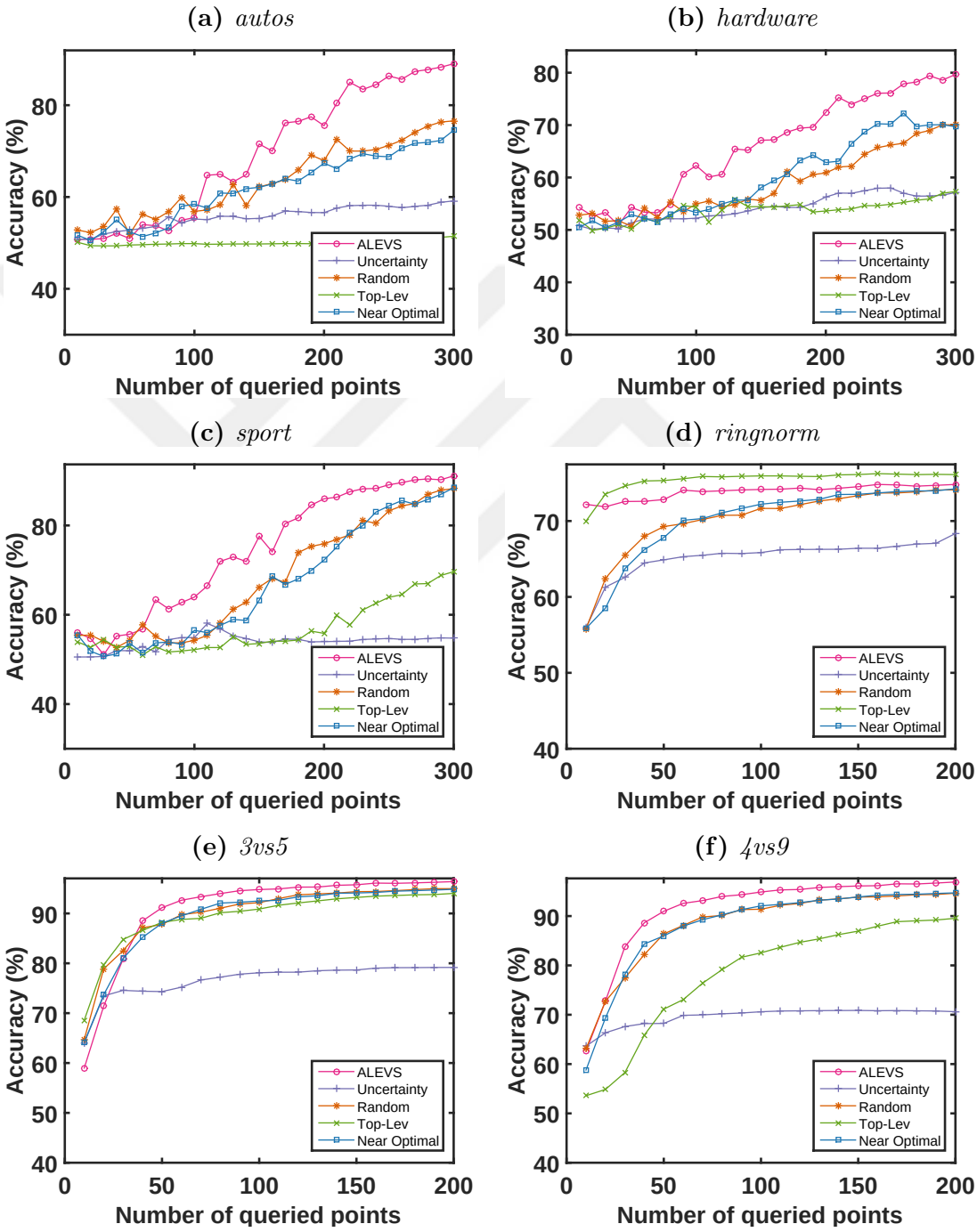

Figure 6.1: Comparison of `DBALEVS` with other methods on classification accuracy.

DBALEVS is outperformed by Top-Lev baseline. One possible reason for this issue is the structure of the dataset, since *ringnorm* dataset is artificially created from multivariate gaussians, by querying points with maximal leverage scores the learner gets labels from dense regions in different clusters without diversification. On the other hand, since this gaussian assumption is a strong assumption, Top-Lev baseline is observed to be performing similar to uncertainty sampling in general.

Chapter 7

Conclusion and Future Work

In this study we aim to improve pool-based active learning for binary classification by introducing two novel algorithms, a sequential-mode algorithm `ALEVS` and a batch-mode algorithm `DBALEVS`. These methods rely on statistical leverage scores to find the most influential data points. Original motivation of this idea is grounded in low-rank matrix approximation algorithms. In these algorithms statistical leverage scores are used to identify influential columns for constructing a lower rank approximation to the input matrix. Novel contribution of this thesis is to use statistical leverage scores, for the first time, for effectively querying examples in active learning.

The first algorithm proposed is `ALEVS` which employs statistical leverage scores of a kernel matrix constructed by evaluating the kernel function on pairwise instances of the pool. At each iteration of the active selection, `ALEVS` trains a supervised classifier and divides the unlabeled data pool into two partitions based on the predicted class labels. Next, class specific kernel matrices are constructed by evaluating the kernel function on labeled and the unlabeled data points. The key step is the deployment of statistical leverage scores to asses the most important rows (or columns) of the kernel matrices. The corresponding examples for these rows (or columns) are the data points that are most influential in its class. Our extensive experimentation on 8 different datasets shows that `ALEVS`

achieves superior performance compared to baselines, uncertainty sampling and random selection. Moreover, `ALEVS` performs better or equally well when compared against a state-of-the art approach, `QUIRE` [34], which is documented to outperform other methods.

Our second proposed algorithm, `DBALEVS` is designed to address the batch-mode active learning setting. In this setting, the active learner selects a batch of examples at once and request the labels of this. We formulate a set scoring function in such a way that while maximizing the overall influence of the set, the similarities between the selected elements are minimized. The set soring function rewards examples with high-leverage scores and includes a term that penalizes the inclusion of similar examples into the set. Solving this problem optimally is NP-hard. We prove that this set scoring function is a submodular, monotone and non-negative, which enables us to use a greedy algorithm for solving the submodular maximization problem. The greedy algorithm produces a solution that is close to the optimal solution up to a bound. Our experiments on 6 different datasets for batch-mode active learning problem shows that the idea of incorporating leverage scores and kernel matrix entries to find an influential and diverse batch of points is a very effective idea. `DBALEVS` perform well against common baselines random sampling and uncertainty sampling; and also against a state of the art method, `NearOpt` [40], which employs a newly introduced framework called adaptive submodularity, and has a solid theoretical foundation.

We conclude that both for sequential and batch-mode settings incorporating statistical leverage scores leads to better performance by reducing the label cost and offers an efficient strategy in terms of runtime. We observe `ALEVS` and `DBALEVS` are especially effective in early iterations, where many of the algorithms fail. Therefore, a future direction can be to develop a hybrid strategy where the proposed method works in cooperation with other methods. For example the framework proposed in this study do not incorporate any knowledge about the uncertainty of the class labels. In a future study we will explore sampling examples with high leverage scores and that are also uncertain.

Secondly, one wants to know and analyze comprehensively the scenarios for

which leverage score sampling would fail. There are already two particular datasets for which `ALEVS` poorly performs. One issue could be the class imbalance associated with these datasets. The target rank parameter k and the choice of kernel have an impact on the leverage scores attained. Limitations of the proposed methods should be investigated towards this directions.

Both proposed methods are introduced for binary classification problems, yet it will be easy to extend both `ALEVS` and `DBALEVS` to the multi-class classification case. One possible future direction would be to study the adaptability of these methods to stream-based selective sampling active learning approaches.

Finally, in order to understand the effectiveness of the methods, we did not consider the more complex active learning scenarios such as non-uniform cost of labels and noisy, reluctant experts. The general framework presented here shall be further investigated to incorporate these alternative settings.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[3] B. Settles, "Active learning literature survey," Technical Report 1648, University of Wisconsin–Madison, 2009.

[4] G. C. Cawley, "Baseline methods for active learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics Workshop on Active Learning and Experimental Design*, pp. 47–57, 2011.

[5] S. Dasgupta, "Two faces of active learning," *Theoretical Computer Science*, vol. 412, no. 19, pp. 1767–1781, 2011.

[6] M. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proceedings of the National Academy of Sciences*, vol. 106, no. 3, pp. 697–702, 2009.

[7] P. Drineas, M. Magdon-Ismail, M. Mahoney, and D. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3475–3506, 2012.

[8] D. Angluin, "Queries and concept learning," *Machine Learning*, vol. 2, no. 4, pp. 319–342, 1988.

[9] D. Angluin, "Queries revisited," *Theoretical Computer Science*, vol. 313, no. 2, pp. 175–194, 2004.

[10] E. B. Baum and K. Lang, "Query learning can work poorly when a human oracle is used," in *International Joint Conference on Neural Networks*, vol. 8, p. 8, 1992.

[11] Y. Freund, H. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine Learning*, vol. 28, pp. 133–168, Sept. 1997.

[12] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, pp. 201–221, May 1994.

[13] A. Beygelzimer, S. Dasgupta, and J. Langford, "Importance weighted active learning," in *Proceedings of the 26th International Conference on Machine Learning*, pp. 49–56, 2009.

[14] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, 1996.

[15] M. Sugiyama, "Active learning in approximately linear regression based on conditional expectation of generalization error," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 141–166, 2006.

[16] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," in *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 209–218, Springer, 2007.

[17] W. Chu and Z. Ghahramani, "Extensions of gaussian processes for ranking: semisupervised and active learning," in *Proceedings of the Neural Information Processing Systems Workshop on Learning to Rank*, pp. 29–34, MIT, 2005.

[18] B. Long, O. Chapelle, Y. Zhang, Y. Chang, Z. Zheng, and B. Tseng, "Active learning for ranking through expected loss optimization," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267–274, ACM, 2010.

[19] N. Ailon, "An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 137–164, 2012.

[20] B. Eriksson, G. Dasarathy, A. Singh, and R. D. Nowak, "Active clustering: robust and efficient hierarchical clustering using adaptively selected similarities," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 8, pp. 43–45, 2011.

[21] K. Voevodski, M.-F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia, "Active clustering of biological sequences," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 203–225, 2012.

[22] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12, Springer-Verlag, 1994.

[23] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, 1948.

[24] D. J. MacKay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[25] Y. Chen, S. H. Hassani, A. Karbasi, and A. Krause, "Sequential information maximization: When is greedy near-optimal?," in *Proceedings of the International Conference on Learning Theory*, July 2015.

[26] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, pp. 45–66, November 2001.

[27] M. F. Balcan, A. Broder, and T. Zhang, "Margin based active learning," in *Proceedings of the International Conference on Learning Theory*, Springer, June 2007.

[28] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1070–1079, Association for Computational Linguistics, 2008.

[29] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proceedings of the 18th International Conference on Machine Learning*, pp. 441–448, 2001.

[30] S. Dasgupta and D. Hsu, "Hierarchical sampling for active learning," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 208–215, 2008.

[31] T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proceedings of the 21st International Conference on Machine Learning*, p. 79, 2004.

[32] J. Zhu, H. Wang, T. Yao, and B. K. Tsou, "Active learning with sampling by uncertainty and density for word sense disambiguation and text classification," in *Proceedings of the 22nd International Conference on Computational Linguistics*, pp. 1137–1144, Association for Computational Linguistics, 2008.

[33] P. Donmez, J. G. Carbonell, and P. N. Bennett, "Dual strategy active learning," in *Machine Learning: Proceedings of the 18th European Conference on Machine Learning*, pp. 116–127, Springer, 2007.

[34] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Advances in Neural Information Processing Systems*, pp. 892–900, 2010.

[35] Y. Guo and D. Schuurmans, "Discriminative batch mode active learning," in *Advances in Neural Information Processing Systems 20*, pp. 593–600, Curran Associates Inc., 2008.

[36] S. C. Hoi, R. Jin, and M. R. Lyu, "Batch mode active learning with applications to text categorization and image retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1233–1248, 2009.

[37] Y. Guo, "Active instance sampling via matrix partition," in *Advances in Neural Information Processing Systems 23*, pp. 802–810, Curran Associates Inc., 2010.

[38] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Batch mode active sampling based on marginal probability distribution matching," *ACM Transactions on Knowledge Discovery from Data*, vol. 7, no. 3, p. 13, 2013.

[39] Q. Gu, T. Zhang, and J. Han, "Batch-mode active learning via error bound minimization," in *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pp. 300–309, 2014.

[40] Y. Chen and A. Krause, "Near-optimal batch mode active learning and adaptive submodular optimization.," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 160–168, 2013.

[41] S. Chakraborty, V. Balasubramanian, Q. Sun, S. Panchanathan, and J. Ye, "Active batch selection via convex relaxations with guaranteed solution bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 1945–1958, 2015.

[42] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, "Multi-class active learning by uncertainty sampling with diversity maximization," *International Journal of Computer Vision*, vol. 113, no. 2, pp. 113–127, 2015.

[43] S. Chatterjee and A. Hadi, "Influential observations, high leverage points, and outliers in linear regression," *Statistical Science*, vol. 1, no. 3, pp. 379–393, 1986.

[44] D. Hoaglin and R. Welsch, "The hat matrix in regression and ANOVA," *American Statistician*, vol. 32, pp. 17–22, 1978.

[45] A. Gittens and M. Mahoney, "Revisiting the Nyström method for improved large-scale machine learning," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[46] P. Drineas, R. Kannan, Michael, and W. Mahoney, "Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition," *SIAM Journal on Computing*, vol. 36, p. 2006, 2004.

[47] T. Yang, L. Zhang, R. Jin, and S. Zhu, "An explicit sampling dependent spectral error bound for column subset selection," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 135–143, 2015.

[48] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis, "Provable deterministic leverage score sampling," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 997–1006, ACM, 2014.

[49] P. Drineas, M. Mahoney, and S. Muthukrishnan, "Relative-error CUR matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, pp. 844–881, 2008.

[50] Y. Wang and A. Singh, "Column subset selection with missing data via active sampling," in *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pp. 1033–1041, 2015.

[51] Y. Wang and A. Singh, "An empirical comparison of sampling techniques for matrix column subset selection," in *Proceedings of 53rd Annual Allerton Conference on Communication, Control, and Computing*, pp. 1069–1074, IEEE, 2015.

[52] A. Kyrillidis and A. Zouzias, "Non-uniform feature sampling for decision tree ensembles," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4548–4552, May 2014.

[53] W. Bi and J. T.-Y. Kwok, "Efficient multi-label classification with many labels," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 405–413, 2013.

[54] S. Fujishige, *Submodular functions and optimization*, vol. 58. Elsevier, 2005.

[55] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, pp. 427–486, 2011.

[56] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, no. 19, p. 8, 2012.

[57] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146, ACM, 2003.

[58] H. Lin and J. Bilmes, "A class of submodular functions for document summarization," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 510–520, Association for Computational Linguistics, 2011.

[59] Y. Liu, K. Wei, K. Kirchhoff, Y. Song, and J. Bilmes, "Submodular feature selection for high-dimensional acoustic score spaces," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7184–7188, IEEE, 2013.

[60] A. Krause and V. Cevher, "Submodular dictionary selection for sparse representation," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 567–574, 2010.

[61] A. Krause and C. Guestrin, "Near-optimal observation selection using submodular functions," in *Conference on Artificial Intelligence, Nectar track*, July 2007.

[62] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: Maximizing information while minimizing communication cost," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pp. 2–10, ACM, 2006.

[63] S. Jegelka and J. Bilmes, "Submodularity beyond submodular energies: coupling edges in graph cuts," in *Conference on Computer Vision and Pattern Recognition*, pp. 1897–1904, IEEE, 2011.

[64] M. Narasimhan, N. Jojic, and J. A. Bilmes, "Q-clustering," in *Advances in Neural Information Processing Systems*, pp. 979–986, MIT Press, 2006.

[65] A. Singh, A. Krause, and W. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *Proceedings of the International Joint Conference on Artificial Intelligenc*, 2009.

[66] S. Javdani, Y. Chen, A. Karbasi, A. Krause, J. A. Bagnell, and S. Srinivasa, "Near-optimal bayesian active learning for decision making," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, April 2014.

[67] Y. Chen, S. Javdani, A. Karbasi, J. A. Bagnell, S. Srinivasa, and A. Krause, "Submodular surrogates for value of information," in *Proceedings of the Conference on Artificial Intelligence*, January 2015.

[68] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT press, 2001.

[69] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, pp. 61–74, MIT Press, 1999.

[70] O. Chapelle, B. Schölkopf, and A. Zien, eds., *Semi-Supervised Learning*. MIT Press, 2006.

[71] M. Lichman, "UCI machine learning repository," 2013.

[72] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits,"

[73] L. Breiman, "Bias, variance, and arcing classifiers," technical report, 1996.

[74] A. Krause and C. Guestrin, "Near-optimal nonmyopic value of information in graphical models," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, p. 5, 2005.

[75] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[76] A. Krause, "Sfo: A toolbox for submodular function optimization," *Journal of Machine Learning Research*, vol. 11, no. Mar, pp. 1141–1144, 2010.

Appendix A

Notations

Table of notations is given in Table A.1.

Table A.1: Notation used throughout the thesis.

Symbol	Explanation
$\mathcal{X}$	input space
$\mathcal{Y}$	output space
$\mathcal{D}$	dataset
$\mathbf{x}$	feature vector of a data point
$\mathbf{x}_{\mathrm{q}}$	queried data point
y	class label
$\mathcal{O}$	labeling oracle
h	classifier
$\mathbf{X}$	input matrix
$k(i,j)$	kernel function evaluation between points i and j
$\mathbf{K}$	kernel matrix
p	kernel parameter configuration
σ	scale parameter of RBF kernel
d	degree of polynomial kernel
c	coefficient of polynomial kernel
Φ	feature mapping
$\mathbf{K}_{(i)}$	$i-th$ row of $\mathbf{K}$
$\mathbf{K}_{ij}$	element in $i-th$ row and $j-th$ column of matrix $\mathbf{K}$
λ	eigenvalue
k	rank parameter for calculating truncated leverage scores
$\mathbf{U}_1$	matrix whose columns are eigenvectors corresponding to top-k eigenvalues
ℓ	statistical leverage score
τ	threshold parameter for rank parameter selection
A, B, S, V	sets
b	batch size
S_{q}	queried batch
$\lvert A\rvert$	cardinality of set A
f, F	set functions
α	diversity trade-off parameter