Proactive Learning: Towards Learning with Multiple Imperfect Predictors

Pinar Donmez

CMU-LTI-10-002

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Jaime G. Carbonell, Chair
Tom Mitchell
Jeff Schneider
Guy Lebanon, Georgia Institute of Technology

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Keywords: Active learning, proactive learning, multiple noisy predictors, unsupervised risk estimation, unsupervised supervised learning

To dad and mom.
Mom - This work would not be complete without your constant love and support.
Dad - I wish you could see this day that I will be earning my Ph.D. You'd have been proud.
Sizi çok seviyorum!

Abstract

Label scarcity is a serious problem in many machine learning applications. In many domains such as classifying texts, images, etc., unlabeled data is readily available whereas labels are fairly expensive to obtain due to labeler availability, cost, and difficulty. Active learning is a paradigm that addresses this challenge by carefully selecting instances to be labeled. The goal is to improve the generalization performance of the learner with fewer labeling requests.

Although active learning is well studied in the literature, it makes unrealistic assumptions. For instance, active learning assumes there is a unique omniscient oracle that works for free or charges uniform fee. In many real-world applications, it is quite possible there are multiple imperfect predictors with differing but unknown qualities. These qualities may vary from providing incorrect labels, failing to provide a label at all or charging non-uniform fees. The proactive learning paradigm addresses these problems to bridge the gap between active learning and more practical real-life scenarios.

In this thesis, we first propose novel active learning methods for classification and rank learning problems that are shown to be quite effective in various real-world domains. We then describe a decision-theoretic framework that addresses learning with multiple predictors having non-ideal characteristics mentioned above with no apriori information, introducing proactive learning and how it can address various scenarios. Later, we focus on more specific aspects of proactive learning, especially coping with multiple fallible (noisy) predictors. We are interested in estimating the labeling accuracy of the predictors to select the most reliable ones in the absence of ground truth. We propose two novel approaches that achieve this goal when first the labeling accuracies are stationary and second when they vary with time. Our empirical evaluation demonstrates the ability to infer the predictor accuracy without any prior information in both synthetic and real-world datasets.

Finally, we frame the problem as unsupervised risk estimation of multiple predictors as an alternative to the above approaches. The benefit is that it allows risk to be defined as a parametric function where the parameter governs the generation of the noisy predictor output. We propose maximum likelihood estimation framework as a solution. One of the most crucial consequences of this framework is to train classifiers that will minimize margin-based risk without using a single labeled example. The likelihood maximization framework yields statistically consistent estimators and hence effective estimation and training capabilities as supported by the thorough empirical evaluation.

Acknowledgements

There are many people who have influenced my life as a PhD student. First and foremost, I would like to thank my advisor Jaime Carbonell. He has been a great mentor through the entire time. He has taught me how to see the big picture, how to approach a problem and systematically tackle it and pretty much anything that I now know about doing research. He has also given me the freedom to pursue what is interesting to me and guided me when I am stuck. I am grateful for his unlimited support, encouragement and mentorship and I feel special to be one of his students.

I am also greatly thankful to my thesis committee members: Tom Mitchell, Jeff Schneider and Guy Lebanon. Tom has always asked the most insightful questions, forcing me to think about the important issues. He has provided several invaluable comments to make the thesis more complete and identified issues that I would have otherwise missed. I appreciate Jeff for his great ideas and timely support. He has always caught the main problems with my approach and suggested many useful alternatives. The research I have conducted turned out to be more concrete because of his feedback. Last but not the least, I am grateful to Guy for his detailed guidance and his tremendously novel ideas which always yielded the most exciting results. He has helped me a great deal on so many technical levels and explained his reasoning to help me understand several important issues. I consider myself extremely privileged to have collaborated with him.

In addition, special thanks to my mentors Chris Burges and Krista Svore at Microsoft Research when I was an intern in 2008. They have introduced me to the excitement and challenges of industrial research. Their advice, criticisms and constructive feedback were critical in my research career to make me a better researcher. I would also like to thank Paul Bennett for his numerous valuable suggestions and comments when I first started to work on active learning. Paul has always been very enthusiastic and open to share his ideas and given me very useful advice on crucial aspects of my research. I owe many heartful thanks to a lot of good friends at CMU Jon Elsas, Jaime Arguello, Andy Schlaikjer, Andreas Zollmann, Pradipta Ray, Selen Uguroglu, Matt Bilotti, Grace Yang, and in Pittsburgh Oznur Tastan, Tankut Dogrul, Soner Yaldiz, Mehmet Gerceker, Leman Akoglu, Hasan Akyol, Emre Karagozler, Ozge Gokbayrak, Kerem Goren, Basak Isin, Umut Arslan and many others for

their amazing friendship and support. I am also deeply indebted to my dear mother and sister for their endless love and encouragement.

Last but definitely not the least, I am thankful beyond words to my soon-to-be-husband Volkan Ediz for he has always believed in me even at the times when I did not believe in myself. Without his love, I would not have been the person I am today.

External Collaborators

In addition to the acknowledgments, I would like to share credit with my external collaborators. Paul Bennett (Microsoft Research) has contributed to formulate the switching point efficiently and effectively in the dynamic ensemble method, DUAL, in Chapter 3. He has also helped to better shape the representation of the experimental results for DUAL. Krishnakumar Balasubramanian (Georgia Institute of Technology) has contributed, in Chapter 7, to the technical formulations for estimating regression error rates and the experiments regarding analyzing the estimation error in the regression case. Furthermore, he has conducted the experiments regarding the unsupervised grid search in Chapter 8 as well as analyzing the sensitivity of the proposed solution to misspecifications in the prior $p(Y)$.

Notation and Basic Definitions

- $x \in \mathcal{X} = \mathbb{R}^d$: input data vector $d = 1, 2, \ldots, M$
- $y \in \mathcal{Y} = \{1, \ldots, l\}$: target label
- $\hat{y} \in \mathcal{Y}$: noisy output label
- $f : \mathcal{X} \to \mathcal{Y}$: predictor function
- r_i: rank position of data point x_i
- ϑ: real-valued decision threshold for classification
- $\bar{\vartheta}$: integer-valued rank threshold for ranking
- $p(x)$: data density distribution
- $p(y \mid x)$: posterior class label distribution
- $p(x \mid y)$: class label conditional density
- F: set of linear functions, i.e. $f(x) =< w, x >$
- w: weight vector
- $Lb \subseteq (\mathcal{X}, \mathcal{Y})$: set of labeled data
- $Un \subseteq \mathcal{X}$: set of unlabeled data
- I_l: set of indices for labeled data
- I_u: set of indices for unlabeled data
- $G(V, E)$: graph with V (set of nodes) and E (set of edges)
- $p \in V^l$: path of length $l = |p|$
- $P_{i,j}$: set of all paths that connect x_i and x_j

- DS: dissimilarity matrix with $DS_{ij} = d(x_i, x_j)$ where d is any well-defined distance function
- $||.||$: Euclidean distance function
- λ: regularization parameter for regularized optimization
- U: utility function
- $ET(Y \mid x, w)$: conditional entropy of the target label y given instance x and model w
- D_t: distribution over $\mathcal{X}x\mathcal{X}$ at time t (for boosting)
- $H(x)$: final scoring function for RankBoost, which is a weighted sum of weak rankings
- B: budget for label acquisition

Above is the general notation used throughout the thesis. Individual chapters or sections may have additional notation which is introduced where mentioned. Different variations of the general notation is also noted where used.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In most machine learning domains, unlabeled data is available in abundance, but obtaining class labels or ranking preferences requires extensive human effort, sometimes from experts with very limited availability. For instance, it is easy to crawl the web, but much more costly to pay human annotators to carefully examine the web documents in order to assign topics or relevance-based judgments in a document retrieval scenario. It is also simple to collect images, but much harder to obtain linguistic content labels. For tasks such as classifying galaxies in the Sloan Sky Catalog, scarce expertise is required. Thus, it is crucial to design methods that will considerably reduce the labeling effort without sacrificing a significant loss of generalization accuracy.

The active learning paradigm addresses this challenge. In active learning, a few labeled instances are typically provided together with a large set of unlabeled instances. The objective is first to select optimal instance(s) for an external oracle to label, and then re-run the learning method with the additional labels to minimize the prediction error, i.e. to improve performance. The active learning task attempts to optimize learning by selecting the most informative instances to be labeled, where informativeness is typically defined as maximal expected improvement in accuracy. Several studies including our own [Tong and Koller, 2000; Roy and McCallum, 2001; Nguyen and Smeulders, 2004; Donmez *et al.*, 2007; Donmez and Carbonell, 2008b] show that active learning greatly helps reduce the labeling effort in various domains. However, active learning relies on unrealistic assumptions, largely swept under the proverbial carpet thus far. For instance, active learning assumes there is a unique omniscient oracle. In real life, it is possible and more general to have multiple sources of information with differing reliabilities or areas of expertise. Active learning also assumes that the single oracle is perfect, always providing a correct answer when requested. In reality, though, a "predictor" (we use the term to refer to any source of information) may be incorrect (fallible) with some probability (either fixed or time-varying or dependent on the difficulty of the question). Moreover, a predictor may be reluctant - it may refuse to

answer if it is too uncertain or too busy. Finally, active learning presumes the predictor is either free or charges uniform cost in label elicitation. Such an assumption is naive since cost is likely to be regulated by difficulty (amount of work required to formulate an answer) or other factors. We propose, in this thesis, *proactive learning* as a new paradigm that relaxes these assumptions and enables active learning to reach practical applications.

Figure 1 depicts a geneology of the thesis work. First, we developed active learning algorithms for classification and ranking learning using machine learning techniques. For classification, our main focus was to combine the uncertainty principle with the underlying instance-space density to sample instances in the maximally uncertain and highly dense regions [Donmez *et al.*, 2007; Donmez and Carbonell, 2008a]. We also investigated different strategies for active rank learning. We adopted a theoretically motivated loss minimization framework leading to effective, efficient and fast converging results. We proposed sampling based on maximizing the estimated loss differential over unlabeled data in [Donmez and Carbonell, 2008b]. Similarly in [Donmez and Carbonell, 2009], we proposed a sampling strategy that selects instances minimizing the hinge rank loss in a SVM rank learning setting. Both methods lead to promising results with significant improvements against state-of-the-art baselines in the literature [Brinker, 2004; Yu, 2005; Amini *et al.*, 2006; Rajaram *et al.*, 2007].

More recently, we studied proactive learning with reluctant, fallible and variable-cost predictors by formulating active label sampling as inherently a decision-theoretic problem. We assumed that the different predictor properties can be defined as a function of the query difficulty, i.e. the level of difficulty to classify the sampled instance, and analyzed each property: reluctance, reliability, and cost variability. We proposed an effective decision-theoretic approach to sample the optimal instance to be labeled as well as the optimal predictor to label it. Our empirical evaluation on benchmark datasets show promising results.

The remainder of the thesis focuses on learning with multiple fallible predictors. Many inductive learning applications rely on accurately labeled instances since they aim at maximizing the classification accuracy based on a set of training instances. The maximum accuracy achieved depends strongly on the quality of the labeling process. However, what happens if there are multiple predictors with different but unknown labeling accuracies? With the recent advent of inexpensive and scalable online annotation tools, such as Amazon's Mechanical Turk (http://www.mturk.com), the labeling process has become more vulnerable to noise - and without prior knowledge of the accuracy of each individual predictor. For some simple tasks, these tools are shown to be reliable in producing quality labels on average [Snow *et al.*, 2008], but for other tasks quality is highly variable [Ambati *et al.*, 2010]. It is still unclear to what extent such tools can be trusted for tasks that require expertise higher than unvetted average annotators can handle. For instance, multiple experts might not agree on the medical diagnosis of a clinical case, or on the primary topic of a document, even if we hypothesize the existence of a ground truth (or consensus).

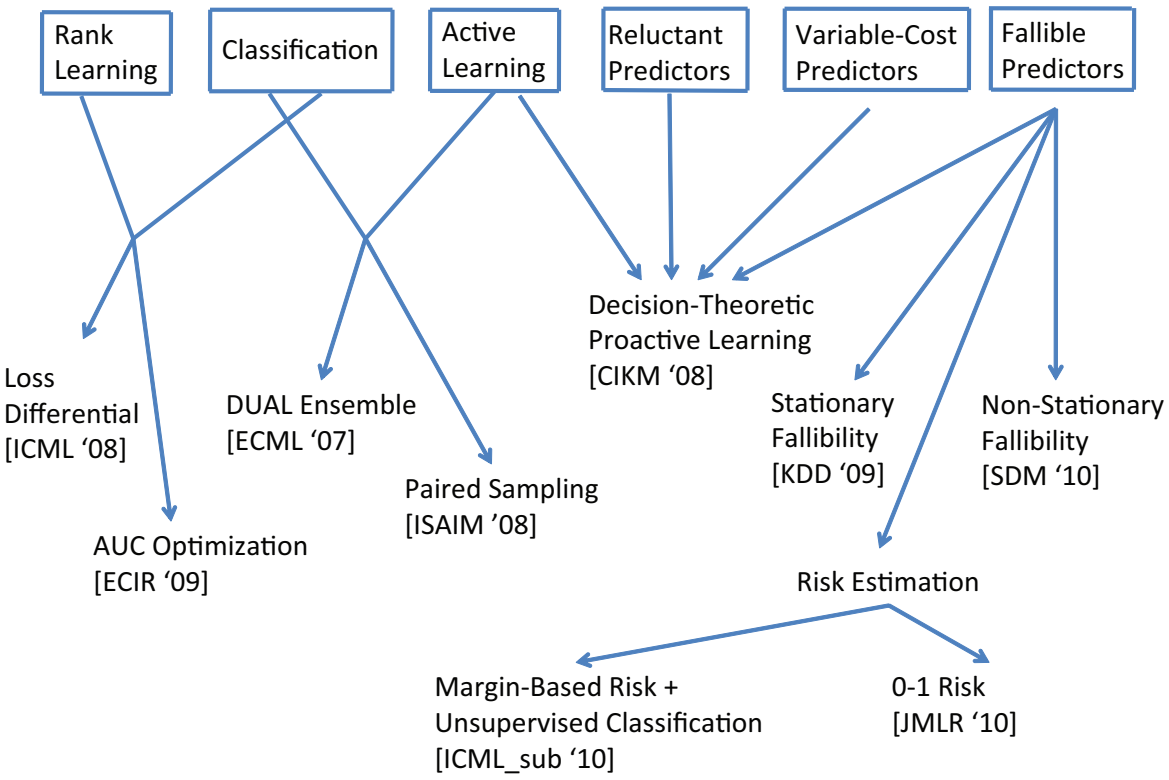


Figure 1.1: Geneology of the thesis work

In remote-sensing applications, image analysis is often a manual process with subjective labeling by multiple labelers. Furthermore, using human annotators is not the only way to acquire labels. Extensive machinery and different lab experiments are often applied in protein structure prediction to determine which fold a protein sequence assumes. In medical diagnosis, blood or urine tests, certain type of biopsy, etc. are used to diagnose a patient (i.e. to produce diagnosis labels). These procedures range from cheap and partially reliable to more costly and more reliable. Hence, it is essential to use multiple sources to increase confidence while maintaining a trade-off between quality and cost. This requires estimating the accuracy of multiple prediction sources and identifying the highly accurate ones to rely on.

We have analyzed learning with multiple noisy predictors in the absence of gold standard labels in [Donmez *et al.*, 2009; Donmez *et al.*, 2010a]. Donmez *et al.* [2009] considers the case where the labeling accuracies of the predictors are fixed over time. It proposes an Interval Estimation learning technique which naturally balances the exploration vs. exploitation trade-off. It relies on majority voting over the responses of individual predictors themselves to evaluate the performance of each. Another dimension we address is that in many application areas providers of information may exhibit varying accuracy over time. It is, therefore, useful to track such variation in order to know when and how much to rely upon their answers. For instance, scientific equipment used over time may lose calibration, and thus gradually increase measurement error, until calibrated or upgraded causing a return to earlier peak performance or better. Loan officers at financial institutions may apply decision criteria increasingly at odds with the economic environment (e.g. leading to financial crises). Text topic labelers, on the other hand, become sloppier with time due to fatigue effects, but may be refreshed in future labeling sessions. The same is true for clinicians diagnosing ailments after pro-longed on-call sessions. Lab technicians may hone their skills over time, and thus run more accurate experiments with increasingly reliable results. In [Donmez *et al.*, 2010a], we tackle the problem of learning from multiple predictors with time-varying accuracies. We propose a framework based on Sequential Bayesian Estimation to learn the expected accuracy at each time step while simultaneously deciding which predictors to query for a label in an incremental learning framework. The experimental results demonstrate the strength of the proposed method in terms of estimating the time-varying reliability of multiple predictors and producing quality labels without extensive label queries.

Finally, we have considered risk estimation using exclusively unlabeled data. Traditionally, the risk of a predictor $f : \mathcal{X} \rightarrow Y$ is estimated through empirical evaluation using the true labels. However, there are various reasons that lead us to use only unlabeled data to estimate the risks. Consider organizations using training sets with private labels to construct predictors. Then, labeled data is not available due to privacy concerns. For example, in medical diagnosis prediction, the predictors $f_1, \ldots, f_k$ may be obtained by k hospitals, each using private internal labeled data. Each hospital releases its predictor to the public which

uses a separate unlabeled set to estimate the risks $R(f_1), \ldots, R(f_k)$. Companies releasing predictors to clients as black boxes (without their training data) are other examples, such as entity labelers for text mining products. The main reason for them not to share their training data is to protect their intellectual property. This is a common practice in the consulting and analytics industry.

Domain adaptation or transfer learning in general is another example to motivate the use of unlabeled data for risk estimation. Transfer learning addresses the challenge of learning classifiers (or regressors) in one domain and applying them in a different but related domain, e.g. training classifiers in the newswire domain and testing them in the biomedical documents. This phenomenon is important in many different areas, especially in NLP since it is often the case that labeled data is available in one domain (source domain) but the aim is to make predictions in another domain (target domain) [Daumé III, 2007]. Furthermore, predictors might be trained on labeled data drawn from the past but are used at test time to predict data drawn from a different distribution associated with the present time. Since the training and test datasets are drawn from different distributions, training error will not provide an accurate estimate for the test error of the predictor. A similar situation might arise in personalization tasks such as personalized spam filtering. For instance, some messages could be considered either spam or nonspam by different people. Even optimal filters will perform poorly on such emails especially if they are trained considering only the preferences of a specific set of users. Therefore, it is crucial to estimate how well such predictors perform on new datasets for which no labels are available. We propose in [Donmez *et al.*, 2010c] a maximum likelihood estimator for 0-1 risk (for classification) and mean squared error (for regression) that is proven to be statistically consistent. One of the side benefits of the proposed framework is that it allows making predictions on the new test examples using the estimated risk and the noisy outputs of the predictor(s).

In [Donmez *et al.*, 2010b], we extend the risk estimation framework to work with more general risks such as margin-based risk functions. We focus on estimating margin-based risk functions such as log-loss, hinge-loss, exponential loss, etc. through a likelihood maximization technique over the unlabeled data. We prove that the technique is statistically consistent for high-dimensional linear classifiers, especially suitable for text and image domains. The unsupervised risk estimate framework allows building unsupervised algorithms that aims to minimize the estimated risk without using a single labeled example. This framework further leads to an unsupervised clustering with known class priors $p(y)$. Our empirical analyses demonstrate that the estimated risk minimizers yield highly accurate class partitioning of the data in different large-scale real-world benchmark datasets. The details are described in [Donmez *et al.*, 2010b] and Chapter 8.

The rest of the thesis is organized as follows. In the next chapter, we review the literature on active and proactive learning to give the reader the necessary background for the following material. In Chapter 3, we describe our contributions on active learning

for classification. We propose two novel ensemble methods. Our first method, DUAL, estimates the best operating range for density-weighted uncertainty sampling and standard uncertainty sampling. Our second contribution defines a utilify function which favors data points with close proximity to a large number of uncertain points. Furthermore, it aims to sample pairs with opposite class labels to maximize straddling the decision boundary. It relies on the cluster assumption that the decision boundary should lie in low density regions of the input space to estimate the distant pairs with opposite labels. In Chapter 4 we describe the methods we have developed for active rank learning. These methods both consider efficient ways to estimate the effect of candidate examples on the loss by their contribution to the current loss or its future estimate. We then introduce in Chapter 5 a decision-theoretic approach to proactive learning concerning different scenarios to investigate predictor properties such as fallibility, reluctance and variable-cost. Chapter 6 deals with noisy predictors and analyzes joint predictor accuracy estimation and predictor selection for both stationary and non-stationary cases of predictor accuracy. The common attribute of both approaches is their ability to balance the exploration vs. exploitation trade-off. Chapters 7 and 8 both consider unsupervised risk estimation where the former focuses on classification and regression error rates and the latter focuses on margin-based risk estimation for linear classifiers. A likelihood maximization framework is adopted in both cases, leading to statistically consistent estimators. Chapter 8 also proposes a novel unsupervised class partitioning using exclusively unlabeled data based on the margin-based risk estimation framework. Finally, we offer our conclusions and summarize the thesis contributions in Chapter 9.

Chapter 2

Literature Review

2.1 Related Work on Active Learning

Active learning has received considerable interest among researchers over the past 15 years. There have been several studies investigating the use of active learning in classification, rank learning, regression, and function optimization. In this chapter, we review the literature on active learning for classification and rank learning including our own contributions.

For classification, there are a number of active learning algorithms. The Query-by-Committee algorithm [Seung *et al.*, 1992; Freund *et al.*, 1997] is among the earliest work. The method samples a set of classifiers from a fixed prior distribution over hypotheses. Then, the example leading the highest disagreement between the committee of classifiers is chosen to be labeled. Liere and Tadepalli [2007] uses a similar notion of a commmittee to solve active learning for text categorization using Winnow classifiers. Cohn *et al.* [1995] initiated a statistically optimal method that selects the example, once labeled and added to the training data, is expected to result in the lowest error rate on future test examples. Unfortunately, this method cannot scale well to very large datasets due to its repeatedly re-training nature. Roy and McCallum [2001] proposed a practical solution to this problem by using Monte Carlo estimation of error reduction, and described an efficient incremental training procedure for Naive Bayes classifier. This technique is claimed to be applicable to any learning method where incremental training is efficient. However, it is still not practical for many large scale real-life applications, such as rank learning for document retrieval. In order to address this issue, we proposed sampling based on an estimation of loss differential without having to re-train the learning method for each candidate [Donmez and Carbonell, 2008b]. We have observed very encouraging results on benchmark test

corpora for document retrieval. Details are provided in Section 4.3.

Uncertainty Sampling originated by [Lewis and Gale, 1994] chooses the example which the current learner is most uncertain about. Tong and Koller [2000] adopted the same idea and applied it to a support vector machine (SVM) classifier. They provided a theoretical motivation based on shrinking the version space as much as possible; in other words, selecting the example that will minimize the maximum expected size of the version space. In case of linear discriminant functions, this corresponds to the selection of the unlabeled example with the smallest margin [Tong and Koller, 2000]. Despite the theoretical justification of the version-space reduction methods, they suffer from wasting time eliminating areas of the parameter space that have no direct effect on the error rate, but may have indirect effects. Hence, these methods are not immune to selecting outliers [McCallum and Nigam, 1998] since they have high uncertainty, but getting their labels do not help the learner improve the generalization performance. Nevertheless, uncertainty sampling plays a key role in many studies in the literature, including our previous work, which will be explained in detail in Chapter 3.

There are also several active learning strategies that incorporate the underlying data distribution into the selection mechanism. Most of these methods propose ways to trade-off an uncertainty measure with the density of the sample. Xu *et al.* [2003] uses the k-means algorithm to cluster the samples lying inside the margin of an SVM classifier trained on the current labeled set. The cluster centers are then selected for labeling. Shen and Zhai [2005] adopts the same idea in an information retrieval scenario, and uses the k-medoid algorithm for the top relevant examples. Similar active learning schemes are proposed by [Tang *et al.*, 2002] for natural language parsing, and by [Zhang and Chen, 2002] for content-based information retrieval. In [McCallum and Nigam, 1998], a naive Bayes classifier is trained on both labeled and unlabeled data using an EM algorithm. Under the assumption that unlabeled data dominates the labeled data, the training algorithm clusters the dataset where labeled data is used for initialization only. Clustering information, then, contributes to the selection such that an uncertainty measure is weighed with the density of the example. Another active learning approach that utilizes the uncertainty and the density criteria is our paired sampling strategy described in [Donmez and Carbonell, 2008a]. The difference between our method and the density-based methods introduced above is that our method relies on balanced sampling on both sides of a decision boundary rather than sampling disproportionately on one side. Furthermore, it exploits the natural grouping (clustering) of the data to effectively reduce distance as a function of local density, and maximizes a utility-based conditional entropy criterion for sampling. We have shown that this new density-sensitive method yields significantly superior performance over multiple datasets against other popular active learning sampling methods including representative sampling, uncertainty sampling and density-only sampling.

Nguyen and Smeulders [2004] suggested a probabilistic framework where clustering is combined with a discriminative model. The motivating idea behind their method is that

examples lying on the classification boundary, in other words most uncertain examples, are informative, but using information about the underlying data distribution helps to select better examples. Their method favors higher density examples lying close to the decision boundary. Those examples are assumed to have the largest contribution to the current error. Though their strategy works well in practice, especially by reducing the error quickly, it exhibits very slow additional learning after substantial sampling. On the other hand, standard uncertainty sampling initially has a slower learning rate, but gradually outperforms their method, as shown in our previous work [Donmez *et al.*, 2007]. The reason is that density based methods sample from maximal-density unlabeled regions, and thus help establish the initial classification boundary where it affects the most remaining unlabeled data. On the other hand, uncertainty sampling fine tunes a decision boundary by sampling the regions where the classifier is least certain, regardless of the distribution of the unlabeled data. Our algorithm, DUAL [Donmez *et al.*, 2007], tackles this issue by proposing a principled ensemble-based approach that selects which sampling method to apply based on estimated residual classification error reduction. DUAL incorporates a robust combination of density weighted and uniform uncertainty sampling. We have empirically shown that DUAL combines the best of both worlds, and leads to superior performance across various domains [Donmez *et al.*, 2007]. Details of the DUAL implementation can be found in Section 3.2.

Baram *et al.* [2003] presented an online algorithm (COMB) that selects among three alternative active learning strategies using a variant of the multi-armed bandit algorithm to decide the strategy to be used at each iteration. Albeit the similarities between DUAL and COMB, there is a major distinction. COMB aims to select which sampling method is optimal for a given dataset, whereas DUAL focuses on selecting the operating range among the sampling methods. Other ensembled active learning methods have appeared in literature. Melville and Mooney [2003] extends the query-by-committee algorithm by constructing diverse committees by employing artificial training examples. Co-testing [Muslea *et al.*, 2000] is another ensemble active learning strategy inspired by the multi-view approach called co-training [Blum and Mitchell, 1998]. It utilizes two redundant views of the training data to create an ensemble and selects the unlabeled instance where two classifiers disagree.

Recently, there have been attempts to address the challenges in active sampling for rank learning. Brinker [2004] uses a notion of the margin as an approximation to reducing the volume of the version space. The margin in the ranking scenario is defined as the minimum difference of scores between two instances assuming the ranking solution is a real-valued scoring function. Yu [2005] adopted the same notion of margin for SVM rank learning and proposed a batch mode selection that minimizes the sum of the rank score differences of all data pairs within the batch of samples. Yu proposed an efficient implementation which considers only the rank-adjacent pairs and showed that this strategy is optimal in terms of selecting the most ambiguous set of samples with respect to the ranking function. The major drawback of this margin-based sampling method of [Brinker, 2004;

Yu, 2005] is that a scoring function for ranking may assign very similar scores to two relevant or two non-relevant instances. Such instances do not carry any additional information for the rank learner to distinguish between the relevant and the non-relevant data; hence, cannot help improve the current ranking function. A similar observation is made by [Amini *et al.*, 2006], which proposed a divergence-based active sampling method for rank learning. The proposed method selects the samples at which two different ranking functions maximally disagree. One of the two functions is the current ranking function trained on the labeled data, and the other is a randomized function obtained by cross-validation [Amini *et al.*, 2006]. The divergence-based strategy is effective only when a sufficiently large initial labeled data is available, which is impractical for many real-life ranking applications, such as document retrieval.

We have proposed two different sampling methods for rank learning in [Donmez and Carbonell, 2009] and [Donmez and Carbonell, 2008b]. Both methods take into account the loss minimization factor due to its direct effect on the performance of the ranker. To the best of our knowledge, none of the sampling methods for ranking have directly addressed loss minimization before. Hence, our work is the fist such attempt that relies on loss minimization for active learning in ranking. Our method in [Donmez and Carbonell, 2009] relies on the relationship between the area under the ROC curve (AUC) and loss minimization. Steck [2007] has shown that minimizing hinge loss is an accurate approximation for maximizing AUC. In retrospect, SVMs should be good rankers since they optimize a ranking quality measure, namely the AUC. We use this relationship to sample instances with the largest expected hinge rank loss (a ranking variant of hinge loss defined in [Steck, 2007]. Our empirical results demonstrate that maximizing the AUC is well correlated with maximizing other rank evaluation metrics, such as MAP and NDCG. Our sampling method also significantly outperforms the margin-based heuristic and the divergence-based sampling method of [Amini *et al.*, 2006].

The method we propose in [Donmez and Carbonell, 2008b] targets the expected loss of a ranking function, and aims to reduce it by sampling the instances that will have a positive effect on the performance. Specifically, the instances with the highest potential to update the current ranking function in a favorable way are sampled via maximizing the expected loss differential. This differential is defined as a function of the estimated error of a ranker introduced by the addition of the candidate instance. This principle is applied using RankSVM and RankBoost algorithms as the baselines. Our empirical evaluations on two TREC benchmark corpora show a significant advantage favoring our method against the margin-based heuristic of [Brinker, 2004; Yu, 2005] and random sampling.

2.2 Related Work on Proactive Learning

Proactive learning is a virgin area that has not yet been fully explored. Proactive learning addresses a major disability in active learning to be practical in real-world situations. Thus far, active learning methods assume there is a perfect predictor which always provides an answer at no cost and is never wrong. Such assumptions are not realistic in real-life. First of all, there is almost always a cost for labeling; i.e. immense lab effort is needed to derive the topological structure of a large protein. Second, the predictor may not be reliable; i.e. it may give incorrect answers or no answer at all depending on the query difficulty. Third, not all queries cost the same; some queries might be genuily harder to find an answer, hence costs more. Last but not the least, there is no reason to believe there should exist one and only one predictor. In fact, there might be multiple predictors which charge different fees, have different expertise and different reliability. Therefore, proactive learning requires joint optimization of the expected value of information of learning and its cost depending on the query difficulty, predictor reliability, and so on. This suggests an interestingly complex decision-theoretic problem where the most cost-effective (highest utility) instance should be sampled instead of the one with the highest value of information.

We addressed fallible predictors together with reluctant and variable-cost ones in [Donmez and Carbonell, 2008c]. That work assumes two experts with differing costs: e.g. one is the perfectly reliable expert whereas the other is a noisy expert whose reliability is conditioned on the instances. We further assume that the fallible expert provides a confidence score together with the label. The confidence score is used to assess the quality of the expert. The instance-conditional reliability of the fallible expert is estimated via an exploration phase where the most representative instances are queried and the confidence is propagated through the neighbors. The paper provides a decision-theoretic framework to make the optimal instance-expert selection.

Melville *et al.* [2005] also addresses the cost-sensitivity in active learning in the context of feature-value acquisition. In some machine learning tasks, the training data has missing feature values which are often quite expensive to obtain. The goal of active feature-value acquisition is to incrementally select feature values that are most cost-effective for improving the performance. Melville *et al.* propose a selection approach based on the expected utility of acquiring the value of a feature. The utility of an acquisition is defined in terms of the improvement in model accuracy per unit cost [Melville *et al.*, 2005]. Since the true values for the model accuracy (accuracy on the unseen test data) is unknown, it is estimated by the training set accuracy. If the feature costs are assumed to be equal, this strategy is similar to the loss reduction principles presented earlier in several research studies [Nguyen and Smeulders, 2004; Donmez and Carbonell, 2008b; Roy and McCallum, 2001].

When there is no available gold standard labels, collecting multiple annotations is

becoming a more common practice in the literature. Repeated labeling on the same data point has been considered by [Sheng *et al.*, 2008; Smyth *et al.*, 1994; Smyth *et al.*, 1995] because the labels may not be reliable. The focus of [Smyth *et al.*, 1994; Smyth *et al.*, 1995] is to learn from probabilistic labels in the absence of ground truth in an image processing application. In their task, the domain experts examine an image and provide subjective class labels. They provide a probabilistic framework to model the subjective labeling process and use EM to estimate the model parameters as maximizers of a likelihood function [Smyth *et al.*, 1995]. Sheng *et al.* [2008] and Snow *et al.* [2008] show that it can be effective to use multiple, potentially noisy labels in the absence of gold standard. Sheng *et al.* relies on an active learning framework that uses repeated labeling and provides conditions where repeated labeling can be effective for improving data quality. Their results point out that repeated labeling can give additional benefit especially when the labeling quality is low. Snow *et al.*, on the other hand, collected labeled data through Amazon Mechanical Turk (www.mturk.com) for simple natural language understanding tasks. They empirically show high agreement between the existing gold-standard labels and non-expert annotations. Their analysis is carried out on fairly straightforward tasks and collecting a reasonably large number of non-experts labels. However, it is unclear if their conclusions would generalize to other tasks that require better-than-average expertise and under high label acquisition cost which restricts the total number of labelings one can afford.

Raykar *et al.* [2009] propose an EM-based algorithm to estimate the error rate of multiple annotators assuming conditional independence of the annotator judgments given the true label. Their method iteratively estimates the gold standard, and measures the performance of multiple annotators and update the gold standard based on the performance measures. Dekel and Shamir [2009] offer a solution to identify low-quality or malicious annotators. But their framework is rather limited in the sense that it is based only on Support Vector Machines (SVMs). More importantly, they assume that each annotator is either good or bad, not in a continous distribution and not time-varying. Good annotators assign labels based on the marginal distribution of the true label conditioned on the instance whereas bad annotators provide malicious answers.

We directly address learning with multiple noisy predictors, each of which has unknown labeling accuracy in [Donmez *et al.*, 2009]. The goal of our work is to estimate each labeler's accuracy and use these estimates to select the highest quality labeler(s) for additional label acquisition. Hence, it balances an exploration vs. exploitation tradeoff to acquire information about predictors and to select the best quality ones. It adopts Interval Estimation Learning and proposes a thresholding mechanism to narrow down the set of potentially good predictors and improves the estimation accuracy with many fewer exploratory trials.

In [Donmez *et al.*, 2010a] we offer a new framework which differs from the previous body of work in a variety of ways. The previous works all assume the accuracy (or the labeling quality) of the annotators are fixed. Our framework explicitly deals with non-

stationary labeling accuracy. We model the accuracy of each annotator as a time-varying unobserved state sequence without directional bias. For instance, an annotator might learn the task over time and get better in labeling. Also, she might occasionally get tired and her performance drops but it increases again after enough rest and so on. Our framework provides the necessary tools to track this change with the assumption that the maximal degree of the change is known. Another example is the case where the annotators are trained classifiers. As one can imagine, the performance of a classifier improves with more training data but it may decrease due to noise in the labels or over-fitting. Hence, such classifiers are good examples for annotators with time-varying accuracies.

Donmez *et al.* [2010c] and Donmez *et al.* [2010b] formulate the problem as an unsupervised risk estimation task. Donmez *et al.* [2010c] focuses on estimating 0-1 risk (for classification) and mean squared error (for regression) of multiple predictors in the absence of true labels. The paper proposes a maximum likelihood estimator which is proven to be statistically consistent and shown to be very effective on synthetic and real-world data. Donmez *et al.* [2010b], on the other hand, extends this work by estimating a more general risk, namely the risk concerned with continuous loss functions such as log loss, hinge loss, etc. The most profound consequence of this risk estimation framework is that it leads to an unsupervised partition of the data into class labels via estimating margin-based risk functions without a single labeled example. The theoretical and empirical evaluation demonstrates the effectiveness of unsupervised risk estimation and class partition without any labeled data whatsoever. Xu and Schuurmans [2005] and Bie and Cristianini [2003] have proposed two-class clustering principles for SVMs. The idea is to find a labeling such that the obtained margin over subsequent runs of SVM would be maximal over all possible labelings. A reformulation of this problem allows semidefinite techniques to be applied to find a solution [Xu and Schuurmans, 2005; Bie and Cristianini, 2003]. Although their method achieves performance exceeding or competitive with spectral clustering in most cases, it is limited to SVMs. Our framework, on the other hand, is general to work with any margin-based risk functions; hence, provides substantial modeling flexibility with desirable performance.

Chapter 3

Active Learning for Classification

3.1 Introduction

As mentioned before, obtaining class labels to train supervised machine learning techniques is costly and time-consuming. On the other hand, unlabeled data is available in abundance in many domains. For instance, it is relatively simple to collect images, but much harder to obtain semantically sound content labels. It is also easier to obtain geological data pertaining to regions that may contain oil, but much more costly to drill multiple deep test holes to classify the ones that really contain oil. Active learning consists of optimizing sampling strategies over unlabeled data with respect to an optimization criterion while minimizing the number of samples required for definitive categorization for training. Typically, the learner starts with a very small number of labeled examples, trains a classifier and selects new sample(s) to be labeled, re-trains the classifier and iterates.

In this chapter, we introduce two novel active sampling methods for classification problems. The first section that follows describes the first approach which is an ensemble-based strategy that utilizes the best strategy at each time during the evolving training process, depending on the sample size. More specifically, our strategy is a context-sensitive sampling method whose primary focus is to improve active learning for the later portion of the process, rather than traditional methods that concentrate primarily on the initial dataset labeling. Our experimental evaluation shows that 1) the proposed strategy is reliably better than the best of the single strategies of the ensemble, and 2) it is better across various domains and for both minimal and copious labeled data volumes.

The last section describes also an ensemble sampling method with the following specific goals: 1)maximizing the likelihood of straddling the decision boundary with paired samples, 2) a transformed distance function to effectively reduce distance as a function of local density, and 3) rely on a utilitybased conditional-entropy maximization criterion to combine factors

in making the sampling decision. Our experimental evaluation shows the effectiveness of this sampling scheme against popular active sampling methods.

3.2 The Dual Strategy Active Learning

3.2.1 Motivation for DUAL Ensemble Approach

Nguyen and Smeulders [2004] suggest a probabilistic framework where clustering information is incorporated into the active sampling scheme. They argue that data points lying on the classification boundary are informative, but using information about the underlying data distribution helps to select better examples.

They assume higher density samples lying close to the decision boundary are more informative. We call their method density weighted uncertainty sampling, or DWUS for short. DWUS uses the following active selection criterion:

$$s \quad = \quad \arg\max_{i \in I_u} E[(\hat{y}_i - y_i)^2 \mid x_i] p(x_i) \tag{3.1}$$

where $E[(\hat{y}_i - y_i)^2 \mid x_i]$ and $p(x_i)$ are the expected error and density of a given data point x_i, respectively. I_u is the index for the unlabeled data. (3.1) selects for labeling the unlabeled data point with the maximum density-weighted uncertainty. Hence, it favors points with the largest contribution to the current classification error. In contrast, one can use an uncertainty-based criterion within the same probabilistic framework as illustrated by the following formula:

$$s \quad = \quad \arg\max_{i \in I_u} E[(\hat{y}_i - y_i)^2 \mid x_i] \tag{3.2}$$

We refer to (3.2) as Uncertainty Sampling for the rest of Section 3.2. Consider Fig. 3.1, which displays the performance of DWUS and Uncertainty Sampling on two of the datasets that we explore in more detail later. Combining uncertainty with the density of the underlying data is a good strategy to reduce the error quickly. However, we have empirically observed that, after rapid initial gains, DWUS exhibits very slow additional learning while uncertainty sampling continues to exhibit more rapid improvement.[1] A theoretical insight into why this might be the case remains an open question. A similar behavior is also evident in [Xu *et al.*, 2003] where their representative sampling method increases accuracy in the initial phase while uncertainty sampling has a slower learning rate, but gradually outperforms their method.

[1]Although a quick drop in classification error for DWUS is also observed in [Nguyen and Smeulders, 2004], they did not compare with uncertainty sampling.

We investigated the Spearman's ranking correlation over candidates to be labeled by density and uncertainty in our scenario, and found that they seldom reinforce each other, but instead they tend to disagree on sample point selection. At early iterations, many points are highly uncertain. Thus, DWUS can pick high density points which are lower down in the uncertainty *ranking* but have a high absolute uncertainty score. Later, points with high absolute uncertainty are no longer in dense regions. DWUS picks points that have moderate density but low uncertainty because such points are scored highly according to (3.1). Hence, it wastes effort picking instances with no large effect on error rate reduction.

Fortunately, we can do better across the full spectrum of labeled instances by our algorithm DUAL which adopts a dynamically reweighed mixture of density and uncertainty components and achieves performance superior to its competitors over a variety of datasets. In the following section, we review essential parts of DWUS and then describe DUAL.

3.2.2 Density Weighted Uncertainty Sampling (DWUS)

Nguyen and Smeulders [2004] assume a clustering structure of the underlying data. $x \in \mathcal{R}^d$ is the data and $y \in \{+1, 0\}$ is the class label. The cluster label $k \in \{1, 2, .., K\}$ indicates the hidden cluster information for every single data point where K is the number of total clusters. In order to calculate the posterior $P(y \mid x)$, they use the following marginalization:

$$P(y \mid x) = \sum_{k=1}^{K} P(y, k \mid x) = \sum_{k=1}^{K} P(y \mid k, x) P(k \mid x) \tag{3.3}$$

where $P(y \mid k, x)$ is the probability of the class label y given the cluster k and the data point x, and $P(k \mid x)$ is the probability of the cluster given the data point. But once k is known, y and x are independent since points in one cluster are assumed to share the same label as the cluster; hence knowing the cluster label k is enough to model the class label y. Thus:

$$P(y \mid x) = \sum_{k=1}^{K} P(y, k \mid x) = \sum_{k=1}^{K} P(y \mid k) P(k \mid x) \tag{3.4}$$

$P(k \mid x)$ is calculated only once unless the data is re-clustered, whereas $P(y \mid k)$ is updated each time a new data point is added to the training set. Before explaining how to estimate these two distributions, we illustrate below how the algorithm works:

1. Cluster the data.
2. Estimate $P(y \mid k)$.
3. Calculate $P(y \mid x)$ (Equation 3.4).

4. Choose an unlabeled sample based on (Equation 3.1) and label.
5. Re-cluster if necessary.
6. Repeat steps 2-5 until stop.

We first explain how to induce $P(k \mid x)$ according to [Nguyen and Smeulders, 2004]. A Gaussian mixture model is used to estimate the data density using the clustering structure such that $p(x)$ is a mixture of K Gaussians with weights $P(k)$. Hence, $p(x) = \sum_{k=1}^{K} p(x \mid k)P(k)$. where $p(x \mid k)$ is a multivariate Gaussian sharing the same variance σ^2 for all clusters k:

$$p(x \mid k) = (2\pi)^{-d/2}\sigma^{-d}\exp\{\frac{-\left|\left|x - c_k\right|\right|^2}{2\sigma^2}\} \tag{3.5}$$

where c_k is the centroid of the k-th cluster which is determined via the K-medoid algorithm [Struyf *et al.*, 1997]. It is similar to the K-means algorithm since they both try to minimize the squared error between the points assigned to a cluster and the cluster centroid. In K-means, the centroid is the average of all points in the cluster, whereas in K-medoid the most centrally located point in the cluster is the centroid. Moreover, K-medoid is more robust to noise or outliers.

Once the cluster representatives are identified, an EM procedure is applied to estimate the cluster prior $P(k)$ using the following two steps:

E-step:

$$P(k \mid x_i) = \frac{P(k)\exp\{\frac{-\left|\left|x_i - c_k\right|\right|^2}{2\sigma^2}\}}{\sum_{\acute{k}=1}^{K} P(\acute{k})\exp\{\frac{-\left|\left|x_i - c_{\acute{k}}\right|\right|^2}{2\sigma^2}\}}$$

M-step:

$$P(k) = \frac{1}{n}\sum_{i=1}^{n} P(k \mid x_i) \tag{3.6}$$

The cluster label distribution $P(y \mid k)$ is calculated using the following logistic regression model: $P(y \mid k) = \frac{1}{1+\exp(-y(c_k.a+b))}$, $a \in \mathcal{R}^d$ and $b \in \mathcal{R}$ are logistic regression parameters[1]. c_k is the k-th cluster centroid, so $P(y \mid k)$ models the class distribution for a representative subset of the entire dataset. Points are assigned to a cluster with the probability $P(k \mid x)$ so that their labels will be affected by their cluster membership probabilities (See Equation 3.4). Hence, a distribution is learned at each cluster and no cluster purity requirement is forced.

The parameters of the logistic regression model are estimated via the following likelihood maximization:

$$l(a,b) = \sum_{i \in I_l \cup I_u} \ln p(x_i; c_1, ..., c_K, P(1), ..., P(K)) + \sum_{i \in I_l} \ln P(y_i \mid x_i; a, b) \tag{3.7}$$

where I_l and I_u are the indices for labeled and unlabeled data, respectively. The parameters of the first summand have already been determined by the K-medoid algorithm and the EM

[1]While we use logistic regression, any probabilistic classifier can be adapted.

routine in Equation 3.6. The second summand is used to estimate the parameters a and b via Equation 3.4, as follows:

$$l(a,b) = \frac{\lambda}{2}||a||^2 - \sum_{i \in I_l} \ln \left\{ \sum_{k=1}^{K} P(k \mid x_i) P(y_i \mid k; a, b) \right\} \tag{3.8}$$

The regularization parameter λ is given initially independently of the data. Since the problem is convex, it has a unique solution which can be solved via Newton's algorithm. Then we can calculate the probability $P(y_i \mid k; \hat{a}, \hat{b})$ using the logistic regression model and obtain the class posterior probability $P(y_i \mid x_i; \hat{a}, \hat{b})$ using Equation 3.4. The label $\hat{y}_i$ is predicted for each unlabeled point x_i according to Bayes rule. Finally, active point selection is done by Equation 3.1. The error expectation for a given unlabeled point $E[(\hat{y}_i - y_i)^2 \mid x_i]$ in that equation is:

$$E[(\hat{y}_i - y_i)^2 \mid x_i] = (\hat{y}_i - 1)^2 P(y_i = 1 \mid x_i) + (\hat{y}_i)^2 P(y_i = 0 \mid x_i) \tag{3.9}$$

Since the probability $P(y_i \mid x_i)$ is unknown, its current approximation $P(y_i \mid x_i; \hat{a}, \hat{b})$ is used instead. Additionally, data points are re-clustered into smaller clusters as the expected error reduces. The reason is that it is important to make significant changes in the decision boundary during the early iterations of active sampling. Later the classification boundary becomes more stable and thus needs to be finely tuned. Additional details can be found in [Nguyen and Smeulders, 2004].

3.2.3 Description of the DUAL Algorithm

DUAL works as follows: It starts executing DWUS up until it estimates a cross-over point with uncertainty sampling by predicting a low derivative of the expected error, e.g. $\frac{\partial \epsilon(DWUS)}{\partial x_t} \leq \delta$. The derivative estimation need not be exact, requiring only the detection of diminishing returns which we explain soon. Then, it switches to execute a combined strategy of density-based and uncertainty-based sampling. In practice, we do not know the future classification error of DWUS, but we can approximate it by calculating the average expected error of DWUS on the unlabeled data. It will not give us the exact cross-over point, but it will provide a rough estimate of when we should consider switching between methods. The expected error of DWUS on the unlabeled data can be evaluated as follows:

$$\hat{\epsilon}_t(DWUS) = \frac{1}{n_t} \sum_{i \in I_u} E[(\hat{y}_i - y_i)^2 \mid x_i] \tag{3.10}$$

where $E[(\hat{y}_i - y_i)^2 \mid x_i]$ is calculated as in Equation 3.9. Moreover, it is re-calculated at each iteration of active sampling. t is the iteration number, and n_t is the number of unlabeled instances at the t-th iteration and I_u is the set of indices of the unlabeled points at time

t. By monitoring the average expected error at every single iteration, we can estimate when DWUS' performance starts to saturate, i.e., $\frac{\partial \hat{\epsilon}(DWUS)}{\partial x_t} \leq \delta$. δ is assigned a fixed small value in our evaluations [See Section 4.2 for how it was estimated]. When it is near zero, this is equivalent to detecting when a method is stuck in local minima/plateau in gradient descent methods. In fact, this principle is flexible enough to work with any two active learning methods where one is superior for labeling the initial data and the other is favorable later in the process. It generalizes to N sampling methods by introducing additional estimated switchover points based on estimated derivative of expected error for each additional sampling strategy.

We know that the strength of DWUS comes from the fact that it incorporates the density information into the selection mechanism. However, as the number of iterations increases uncertainty sampling outperforms DWUS and DWUS exhibits diminishing returns. We propose to use a mixture model for active sampling after we estimate the cross-over:

$$x_s^* = \arg\max_{i \in I_u} \pi_1 * E[(\hat{y}_i - y_i)^2 \mid x_i] + (1 - \pi_1) * p(x_i) \tag{3.11}$$

It is desirable for the above model to minimize the expected future error. If we were to select based on only the uncertainty, then the chosen point would be $x_{US}^* = \arg\max_{i \in I_u} E[(\hat{y}_i - y_i)^2 \mid x_i]$. After labeling x_{US}^*, the expected loss is:

$$f_{US} = \frac{1}{n} \sum_j E_{Lb+\{x_{US}^*, y\}}[(\hat{y}_j - y_j)^2 \mid x_j] \tag{3.12}$$

The subscript $Lb + \{x_{US}^*, y\}$ indicates that the expectation is calculated from the model trained on the data $Lb + \{x_{US}^*, y\}$. Assume f_{US}=0, then we can achieve the minimum expected loss by forcing $\pi_1 = 1$; hence $x_s^* = x_{US}^*$. The appropriate weight in this scenario is inversely related with the expected error of uncertainty sampling. Thus, we can replace the weights by $\pi_1 = 1 - f_{US}$, and $1 - \pi_1 = f_{US}$, and obtain the following model:

$$x_s^* = \arg\max_{i \in I_u} (1 - f_{US}) * E[(\hat{y}_i - y_i)^2 \mid x_i] + f_{US} * p(x_i) \tag{3.13}$$

Achieving the minimum expected loss is guaranteed only for the extreme case where the expected error, f_{US}, of uncertainty sampling is equal to 0. However, correlating the weight of uncertainty sampling with its generalization performance increases the odds of selecting a better candidate after the cross-over.
In the real world, we do not know the true value of f_{US}. So we need to approximate it. After estimating the cross-over, we are interested in giving higher priority to uncertainty, reflecting how well uncertainty sampling would perform on the unlabeled set. Therefore, we approximate f_{US} as $\hat{\epsilon}(US)$, the average expected error of uncertainty sampling on the unlabeled portion of the data. This leads us to the following selection criterion for DUAL:

$$x_s^* = \arg\max_{i \in I_u} (1 - \hat{\epsilon}(US)) * E[(\hat{y}_i - y_i)^2 \mid x_i] + \hat{\epsilon}(US) * p(x_i) \tag{3.14}$$

$\hat{\epsilon}(US)$ is updated at every iteration t after the cross-over. Its calculation is exactly the same as in Equation 3.10. However, the data to sample from is restricted to the already labeled examples by active selection. We construct a set with the actively sampled examples by DWUS until the cross-over, and call it set A. Uncertainty sampling is allowed to choose the most uncertain data point from only among elements in set A by estimating the posterior $P(y_i \mid x_i; \hat{a}, \hat{b})$ over the initially labeled data. The chosen point is added to to the initial labeled set for uncertainty sampling and removed from set A. The average expected error of uncertainty sampling is calculated on the remaining unlabeled data. Then, DUAL selects the next data point to label via the criterion in Equation 3.14. This labeled point is also added to set A. Hence, set A is dynamically updated at each iteration with the actively sampled points. Consequently, in order to calculate the expected error of uncertainty sampling the algorithm never requests the label of a point that has not already been sampled during the active learning process. Such a restriction will prevent an exact estimate of the expected error. But, it is a reasonable alternative, and introduces no additional cost of labeling. The pseudo-code for the DUAL algorithm is given in Algorithm 1.

Algorithm 1 The DUAL Algorithm

Input: Labeled data Lb, Unlabeled data Un, max number of iterations T, and δ.
Output: A set S of actively sampled data points.
Program
Initialize: $t = 0$ and $S = \{\}$.
while while(not switching point) **do**
  Run DWUS algorithm and compute $\frac{\partial \hat{\epsilon}(DWUS)}{\partial x_t}$.
  if $\frac{\partial \hat{\epsilon}(DWUS)}{\partial x_t} > \delta$ **then**
    $x_s^* = \arg\max_{i \in I_u} E[(\hat{y}_i - y_i)^2 \mid x_i] p(x_i)$
    Add the chosen point to set S: $S = S \cup x_s^*$
    t=t+1 (Increment counter t)
  else
    Hit the switching point.
  end if
end while
while while($t < T$) **do**
  Compute $E[(\hat{y} - y)^2 | x]$, $p(x)$ via DWUS, and $\hat{\epsilon}_t(US)$ via uncertainty sampling.
  $x_s^* = \arg\max_{i \in I_u} (1 - \hat{\epsilon}_t(US)) * E[(\hat{y}_i - y_i)^2 \mid x_i] + \hat{\epsilon}_t(US) * p(x_i)$
  Add the chosen point to set S: $S = S \cup x_s^*$
  t=t+1
end while

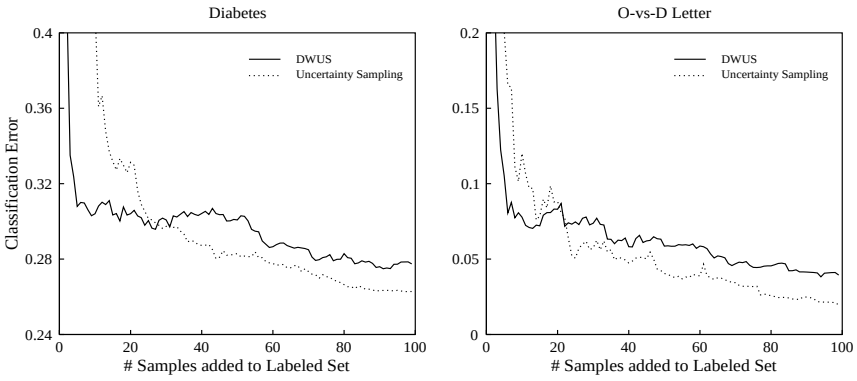


Figure 3.1: Comparison of Density Weighted versus (standard) uniformly weighted Uncertainty Sampling on two UCI benchmark datasets

3.2.4 Experimental Evaluation

To evaluate the performance of DUAL, we ran experiments on UCI benchmarks: diabetes, splice, image segment, and letter recognition [Newman *et al.*, 1998]. Some of these problems are not binary tasks so we used the random partitioning into two classes as described by [Rätsch *et al.*, 2001]. For the letter recognition problem, we picked three pairs of letters (M-vs-N, O-vs-D, V-vs-Y) that are most likely to be confused with each other. Thus, we examine six binary discrimination tasks. For each dataset, the initial labeled set is 0.4% of the entire data and contains an equal number of positive and negative instances. For clustering, we followed the same procedure used by [Nguyen and Smeulders, 2004] where the initial number of clusters is 20 and clusters are split until they reach a desired volume. The values of the parameters are given in Table 6.1 along with the basic characteristics of the datasets. These parameters and the δ parameter used for switching criteria were estimated on other data sets and held constant throughout our experiments, in order to avoid over-tuning. We compared the performance of DUAL with the following baselines:

1. DWUS
2. Uncertainty sampling: It selects the most uncertain instance by the classifier to be labeled; i.e., $x_s^* = \arg\max_{i \in I_u} E[(\hat{y}_i - y_i)^2 \mid x_i]$
3. Density-based sampling: It adopts the same probabilistic framework as DWUS but uses only the density information for active data selection; i.e., $x_s^* = \arg\max_{i \in I_u} p(x_i)$
4. Representative sampling [Xu *et al.*, 2003]: The unlabeled points that fall inside the margin are clustered using k-means in a linear SVM framework. The centroid of the largest cluster among $k = 10$ clusters is chosen to be labeled.

Table 3.1: Characteristics of the Datasets, Values of the Parameters and p-value for significance tests after 40 iterations

DATASET	TOTAL SIZE	+/- RATIO	DIMS(D)	SIGMA(σ)	LAMBDA(λ)	DUAL>DWUS
DIABETES	768	0.536	8	0.5	0.1	$p < 0.0001$
SPLICE	3175	0.926	60	3	5	$p < 0.0001$
IMAGE	2310	1.33	18	0.5	0.1	$p < 0.0001$
M-VS-N	1575	1.011	16	0.1	0.1	$p < 0.0001$
O-VS-D	1558	0.935	16	0.1	0.1	$p < 0.0001$
V-VS-Y	1550	0.972	16	0.1	0.1	$p < 0.0001$

5. COMB method of [Baram *et al.*, 2003]: uses an ensemble of uncertainty sampling, sampling method of [Roy and McCallum, 2001], and a distance-based strategy choosing the unlabeled instance that is farthest from the current labeled set. COMB uses SVM with Gaussian kernel for all three strategies.

The performance of each algorithm was averaged over 4 runs. At each run, a different initial training set was chosen randomly. At each iteration of each algorithm, the active learner selected a sample from the unlabeled pool to be labeled. After it has been added to the training set, the classifier is re-trained and tested on the remaining unlabeled data and the classification error is reported. We also conducted significance tests between DUAL and DWUS to report whether they perform significantly different. In order to determine whether two active learning systems differ statistically significantly, it is common to compare the difference in their errors averaged over a range of iterations [Melville and Mooney, 2004; Guo and Greiner, 2007]. Comparing performance over all 100 iterations would suppress detection of statistical differences since DUAL executes DWUS until cross-over. We conducted the comparison when they start to differ, which is on average after 40 iterations; we compute the two-sided paired t-tests by averaging from the 40th to 100th iteration. Table 6.1 shows that DUAL statistically outperforms DWUS in that range. For the remaining comparisons, we compute 2-sided paired t-tests over the full operating range since we want to know if DUAL is superior to the other methods more generally and DUAL does not execute these other methods at any iteration. Figure 3.2 presents the improvement in error reduction using DUAL over the other methods. We only display representative results on 4 datasets for consistency. For the results on all datasets see `www.cs.cmu.edu/~pinard/DualResults`. DUAL outperforms DWUS and representative sampling both with $p < 0.0001$ significance. DUAL outperforms COMB with $p < 0.0001$ significance on 4 out of 6 datasets, and with $p < 0.05$ on Image and M-vs-N data sets. We also calculate the error reduction of DUAL compared to the strong baseline DWUS. In each graph after 3/4 of the sampling iterations after cross-over occurs, we observe 40% relative error reduction on O-vs-D data, 30% on Image, 50% on M-vs-N, 27% on V-vs-Y, 10% on Splice, and 6% on Diabetes dataset. These results are significant both statistically and also with respect to the magnitude reduction in relative residual error. DUAL is superior to Uncertainty sampling ($p < 0.001$) on 5 out of

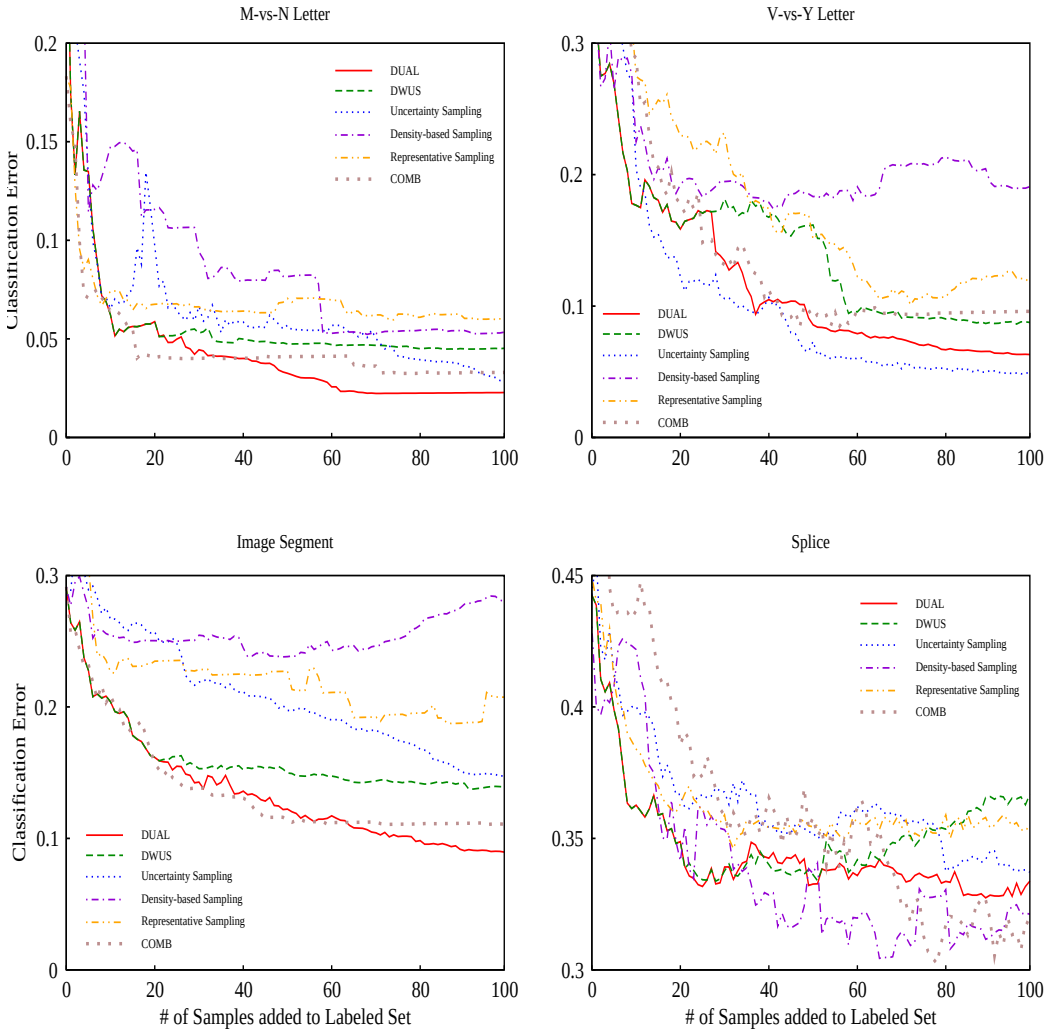


Figure 3.2: Results on 4 different UCI benchmark datasets

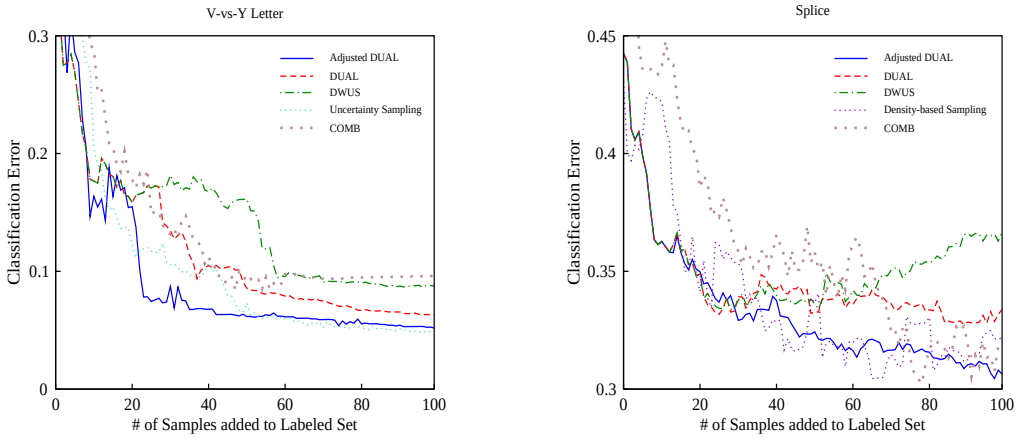


Figure 3.3: Left: Results after adjusting the switching point for DUAL on the V-vs-Y Letter data. Right: Results when DUAL is adjusted using Equation 3.15 on the splice data.

6 datasets. We see on the V-vs-Y data that the cross-over between DWUS and uncertainty sampling occurs at a very early stage, but the current estimate of the expected error of DWUS to switch selection criteria is not accurate at the very early points in that dataset. Clearly, DUAL might have benefited from changing its selection criterion at an earlier iteration.

As part of a failure analysis and in order to test this hypothesis, we conducted another set of experiments where we simulated a better relative error estimator for strategy switching. Fig. 3.3 demonstrates that DUAL outperforms all other methods when the true cross-over point is identified, indicating that better error estimation is a profitable area of research. In fact, one hypothesized solution is to switch when $P(error(M_2) \mid X) < P(error(M_1) \mid X) + \epsilon$, which considers the probability that over future selected instances method 2, M_2, will have less error than method 1, M_1. Studying more robust switching criteria is out of the scope of this thesis and left as future work.

DUAL outperforms Density-based sampling ($p < 0.0001$) on all but splice data. Density-based sampling performs worst for almost 40 iterations but then beats all of the others thereafter, totally breaking the pattern observed in the other datasets. Currently, DUAL only estimates how likely the uncertainty score is to lead to improvement, but the density-based method may also be likely to improve. One strategy is to calculate the expected error $\hat{\epsilon}(DS)$ of density-based sampling and modify Equation 3.14 to obtain the following:

$$x_s^* = \arg\max_{i \in I_u} \{\hat{\epsilon}(DS) * E[(\hat{y}_i - y_i)^2 \mid x_i] + (1 - \hat{\epsilon}(DS)) * p(x_i)\} \tag{3.15}$$

Fig. 4 presents the result after the modification in Equation 3.15. The adjustment helps DUAL make a significant improvement on the error reduction. Moreover, it consistently decreases the error as more data is labeled, hence its error reduction curve is smooth as

opposed to the higher variance of density-based sampling. This suggests that pure density-based sampling is inconsistent in reducing error since it only considers the underlying data distribution regardless of the current model. Thus, we argue that DUAL may be more reliable than individual scoring based on density due to its combination formula that adaptively establishes balance between two selection criteria. Even though a strategy such as uncertainty or density based sampling performs well individually, Figures 3.2 and 3.3 illustrate that it is more advantageous to use their combination.

3.3 The Density-Sensitive Paired Sampling

This section describes the density-sensitive paired sampling technique developed based on two key principles: a) Balanced sampling on both sides of the decision boundary is more effective than sampling one side disproportionately, and b) exploiting the natural grouping (clustering) of unlabeled data establishes a more meaningful non-Euclidean distance function with respect to estimated category membership. In the sections that follow, we first outline a transformation of the data exploiting the cluster hypothesis, which states that the decision boundary should lie in low density regions (i.e. inter-cluster, vs intra-cluster). In section 3.3.2, we derive a sampling criterion that favors pairs of points straddling the decision boundary with maximum utility. We present experimental results in section 3.3.3 that demonstrate the superiority of the proposed method.

3.3.1 Density-Sensitive Distance Estimation

In order to sample points that are likely to be maximally informative to an active learner, we first seek to maximize the chance that we will sample on both sides of a decision boundary – sampling disproportionately on either side will not optimize boundary placement in the learning process. Maximizing the distance between two points is a step in the right direction, but Euclidean distance may not be the optimal measure; instead we investigate density-sensitive distance functions.

According to the cluster hypothesis, the decision boundary should lie in low density regions, and hence should not cut clusters [Chapelle, 2005]. Our goal is to represent the data in such a way that points in separate clusters are assigned high-distances (equivalent to low similarities). In order to enforce this criterion, we chose to derive pairwise similarities/dissimilarities in a fully-connected graph-based representation of the data. Let $G = (V, E)$ be a graph where V is the set of nodes each of which denotes a data point and E denotes the edges between nodes. Edge weights are Euclidean distances, i.e. $\|x-y\|$. $p \in V^l$ is defined as a path of length $l = |p|$ that connects the nodes x_i and x_j if $(p_k, p_{k+1}) \in E$ for $1 \leq k < l$, and $p_1 = x_i$ and $p_l = x_j$. Points in the same cluster can be connected via a path

traveling in that cluster, thereby a high density region. Conversely, any path connecting points in different clusters has to travel along a low density region. The density-sensitive distance between any two points can be approximated by first selecting the longest distance edge along each path, i.e. the weakest link, then repeating this process for every path that connects these two points, and finally finding the minimum among the longest distance edges. This approach was first proposed by [Fischer *et al.*, 2004] and used for clustering:

$$d(x_i, x_j) = \min_{p \in P_{i,j}} \max_{1 \leq k < |p|} \|p_k - p_{k+1}\| \tag{3.16}$$

where $P_{i,j}$ is the set of all paths that connects x_i and x_j. The above formulation does not take into account the length of the paths. A long path connecting two points in different clusters might have a very short edge; hence that single outlier would dramatically disrupt the distance approximation. In order to avoid this problem, we incorporate the path length into the above equation by taking the sum over the edge distances instead of the maximum:

$$d(x_i, x_j) = \frac{1}{\rho} \left\{ \ln(1 + \min_{p \in P_{i,j}} \sum_{k=1}^{|p|-1} (e^{\rho \|p_k - p_{k+1}\|} - 1)) \right\} \tag{3.17}$$

Equation 3.17 is proposed by Chapelle [2005]. Equation 3.16 and 3.17 are equivalent when $\rho \to \infty$. For large values of ρ, the distances between points in the same cluster are decreased whereas the distances between points in different clusters are still dominated by the gaps between clusters. For small values of ρ, every edge contributes to the distance calculation. We follow their approach by applying Multidimensional Scaling (MDS) [Cox and Cox, 1994] to the dissimilarity matrix DS, where $DS_{ij} = d(x_i, x_j)$ in Equation 3.17 to obtain a Euclidean representation of a set of objects while preserving their distance relationships. MDS first transforms the distance matrix DS into a new matrix A by defininining $a_{ij} = -\frac{1}{2} DS_{ij}^2$. Matrix A is used to derive matrix $\Delta = [\delta_{ij}]$ such that $\delta_{ij} = a_{ij} - \bar{a}_i - \bar{a}_j + \bar{a}$, where $\bar{a}_i$ and $\bar{a}_j$ are row and column means of A, respectively; and $\bar{a}$ is the mean of all elements in A. The eigenvalues $(\nu_1, \nu_2, ..., \nu_k)$ and eigenvectors $(u_1, u_2, ..., u_k)$ of Δ are computed, and the latter is scaled so that $\sqrt{u_k' u_k} = \sqrt{\nu_k}$. Chapelle [2005] showed that it is safe to discard the eigenvectors with small eigenvalues; hence we followed their formulization by taking only the first p eigenvectors that satisfy the following inequality:

$$\sum_{i=1}^{p} \nu_i \geq (1 - \delta) \sum max(0, \nu_i) \;\text{ where }\; \nu_p \leq \delta \nu_1 \;\text{ and }\; \nu_1 \geq ... \geq \nu_n \geq 0 \tag{3.18}$$

The δ parameter is fixed at 0.1 as specified in [Chapelle, 2005], though it could potentially be optimized. Let Λ be an n x p matrix whose columns are the scaled eigenvectors, then the rows of Λ are the coordinates of the objects in MDS space, i.e. $\widetilde{x}_{i,\cdot} = \Lambda_{i,\cdot}$. The time complexity to compute the distance matrix DS is $O(n^2(n + logn))$ when Dijkstra's shortest

path length algorithm is adopted to implement the search for the next closest unexplored node in the graph using a binary heap [Chapelle, 2005]. This is the implementation we used in the paper. The MDS transformation takes $O(n^3)$ time since it computes the eigenvectors of an n x n matrix. However, if a k nearest neighbor graph is used instead of a fully-connected graph, and if only the first p eigenvectors are considered, the time complexity for both steps can be reduced.

3.3.2 Density-Sensitive Paired Sampling

Given a set of training data points in MDS space $(X, y) = \{(x_1, y_1), ..., (x_m, y_m)\}$, we use logistic regression to obtain the posterior class distribution. But our approach is designed to be used with any probabilistic classifier including Gaussian processes or Bayesian optimal classifiers. We focus on binary problems in our evaluations, though our method can be easily adapted to multi-class cases. We provide information on handling multi-class problems as appropriate throughout the section. The logistic regression model is

$$P(y \mid x, w) = \sigma(yw^T x) = \frac{1}{1 + exp(-yw^T x)} \tag{3.19}$$

where $y \in \{-1, +1\}$. We use the regularized version to find the parameter vector w which minimizes the negative log-likelihood:

$$l(w) = \sum_{i=1}^{m} \log(1 + exp(-y_i w^T x_i)) + \frac{\nu}{2} w^T w \tag{3.20}$$

The minimization problem is convex so it can be solved by a number of iterative algorithms. We use iteratively reweighted least squares method: $w_{new} = w_{old} - \boldsymbol{H}^{-1}\boldsymbol{g}$, where $\boldsymbol{g}$ and $\boldsymbol{H}$ are the gradient and Hessian of $l(w)$, respectively:

$$\begin{aligned} \frac{\partial l(w)}{\partial w} &= \lambda w + \sum_{i=1}^{m} -y_i x_i (1 - p(y_i \mid x_i, w)) \\ \frac{\partial^2 l(w)}{\partial^2 w} &= \lambda + \sum_{i=1}^{m} x_i x_i^T p(y_i \mid x_i, w)(1 - p(y_i \mid x_i, w)) \end{aligned} \tag{3.21}$$

If there are m instances of d dimensions, it takes $O(md^2)$ time per iteration.

In order to maximize the likelihood of straddling the decision boundary, and to halve the computational time, we sample a pair of points to label at a time, in contrast to the traditional active learning methods that select one point at each iteration. Figure 3.4 illustrates the motivation for paired sampling in active learning. Here we assume for simplicity the data is

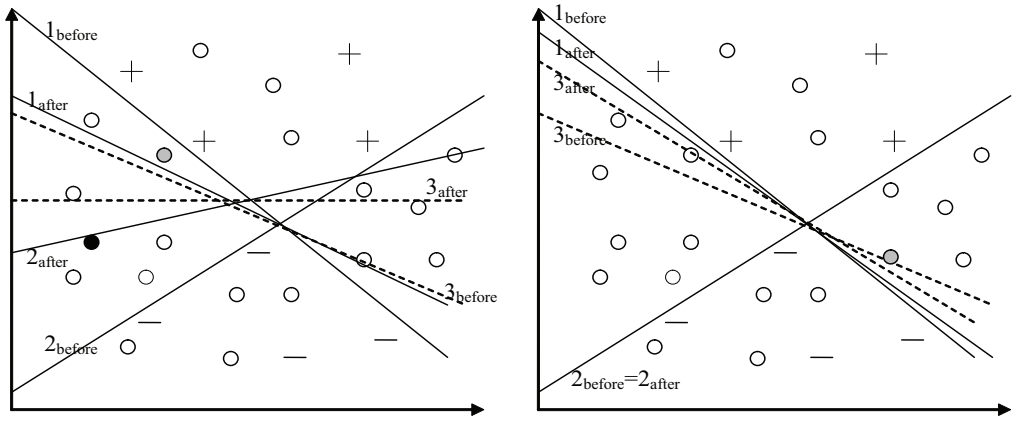


Figure 3.4: Illustrative Example: The plus (minus) sign and circles indicate the positively (negatively) labeled points and unlabeled data, respectively. x_{after} and x_{before} indicate the line before and after data is sampled for labeling. The selected points are labeled either positive (shown in grey) or negative (shown in black). This example illustrates our motivation to sample two points with opposite labels at a time instead of a single point.

linearly separable. The dashed line shows the current decision boundary while the two solid lines define the region where the true boundary is expected to lie; namely the version space. The left figure in Figure 3.4 is an example of sampling a pair for labeling from opposite sides of the current boundary. It greatly reduces the version space since both points affect how the version space will be bounded. The current boundary also shifts significantly. On the other hand, the figure on the right shows that only a single point is sampled for labeling. The amount of shift in the current hypothesis is relatively small. The version space is not reduced as significantly as in the previous scenario since only one point contributes to the reduction. These two scenarios illustrate why it is more advantageous to straddle the decision boundary in order to reduce the set of candidate hypotheses rapidly. With this goal in mind, we strive to sample two points with opposite class labels. In multi-class scenarios, this is equivalent to sampling as many points as the number of classes at each iteration of active learning, seeking to maximize the chance of sampling each class once per round. Since the labels of the unlabeled data are unknown, we need to approximate the likelihood that any two points have opposite class labels, $P(y_i \neq y_j \mid x_i, x_j)$, for all $i, j \in I_u$ where I_u is the set of indices of the unlabeled points in the data. By our cluster assumption, points in different clusters are likely to have different labels. In the new representation of the data, points in different clusters are assigned low similarity. It is then reasonable to define $P(y_i \neq y_j \mid x_i, x_j)$ as proportional to the distance between x_i and x_j, i.e. $P(y_i \neq y_j \mid x_i, x_j) \propto \|x_i - x_j\|^2$. We

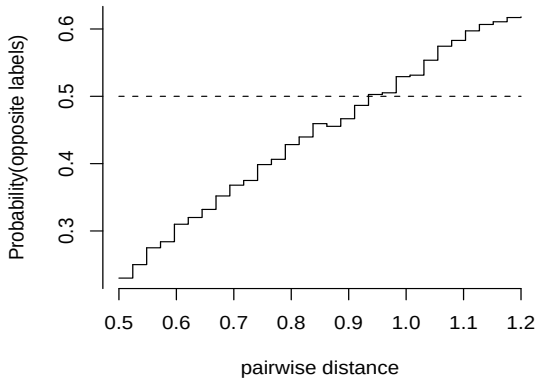


Figure 3.5: Graph of $\hat{P}(y_i \neq y_j \mid x_i, x_j)$ versus $\|x_i - x_j\|$ on g50c dataset

provide here an empirical analysis justifying this claim.

We estimated the probability $P(y_i \neq y_j \mid x_i, x_j)$ as a function of the pairwise distance $\|x_i - x_j\|$. Figure 3.5 is generated on g50c dataset. We sorted the pairwise distances in increasing order and divided them into 30 equal intervals. For each interval, all pairs (x_i, x_j) with distance $\|x_i - x_j\|$ falling within that interval were examined. $P(y_i \neq y_j \mid x_i, x_j)$ was estimated as the relative frequency of pairs in that interval with opposite class labels. As shown in Figure 3.5, $P(y_i \neq y_j \mid x_i, x_j)$ monotonically increases with the pairwise distance. This analysis empirically shows that $\hat{P}(y_i \neq y_j \mid \|x_i - x_j\|) \geq \hat{P}(y_i \neq y_k \mid \|x_i - x_k\|) \Leftrightarrow \|x_i - x_j\| \geq \|x_i - x_k\|$. The curve may differ for other datasets, but if the class membership is a well-defined (e.g. smooth) function, the same principle applies. The dotted line is the probability $P(y_i \neq y_j) = P(y_i = 1, y_j = -1) + P(y_i = -1, y_j = 1)$, independent of any knowledge regarding the data distribution. It is equal to 0.5 in this dataset. The absolute difference between the two curves at any point indicates the loss $|P(y_i \neq y_j) - \hat{P}(y_i \neq y_j \mid \|x_i - x_j\|)|$ introduced by relying on $\|x_i - x_j\|$. Hence, sampling distant pairs increases the likelihood that they have opposite class labels without sacrificing a large penalty. This procedure is conducted only to support our claim, i.e. $P(y_i \neq y_j \mid x_i, x_j) \propto \|x_i - x_j\|^2$; the proposed active sampling strategy is carried on an unsupervised manner.

As the goal of active learning is to learn the model parameters accurately with the least number of labeled examples, the selected instances need to be informative, e.g. the points whose labels we are most uncertain about. Uncertainty-based active learning strategies have been proposed by a number of researchers [Lewis and Gale, 1994; Tong and Koller, 2000; Campbell *et al.*, 2000; Schohn and Cohn, 2000]. Such strategies work fairly well in practice, and have nice theoretical properties related to VC dimension reduction [Tong and Koller, 2000]. Thus, in order to obtain a faster learning rate we need to select two points that are likely to have opposite labels *and* high uncertainty. We first define a scoring function for

each pair of unlabeled points as follows:

$$\begin{aligned} S(i,j) &= P(y_i \neq y_j \mid x_i, x_j) * U(i,j) \\ &= c\|x_i - x_j\|^2 * U(i,j) \end{aligned} \tag{3.22}$$

where c is a normalization constant for $P(y_i \neq y_j \mid x_i, x_j)$, and $U(i,j)$ is a complex utility score which will be explained soon. Before doing so, let us give an outline of how our method works:

1. Compute the distance matrix DS using Equation 3.17 and transform the entire data into the MDS space
2. Compute the pairwise Euclidean distances, $\|x_i - x_j\|$, of the transformed data
3. Train the logistic regression classifier using the current training set in its transformed form and estimate the posterior class probabilities $P(y \mid x, \hat{w})$
4. For all $i \neq j \in I_u$
   (a) Compute the score $S(i, j)$ using Equation 3.22
5. Choose for labeling the points x_{i^*}, x_{j^*} which have the highest score $S(i, j)$, add them to the training set and remove i^*, j^* from I_u.
6. Repeat 3-5 until a desired amount is sampled

Another important factor for active sampling is to select points from high density regions. It is shown to boost the performance in various studies [Cohn *et al.*, 1995; Zhang and Chen, 2002; Xu *et al.*, 2003; Nguyen and Smeulders, 2004; Donmez *et al.*, 2007]. Obtaining the label of an instance with high density has the advantage that it will significantly increase our confidence in the labels of the neighbors. One drawback with this approach is that it does not take into account the current learner's predictions. High density points may already be correctly labeled by the current learner with high confidence. In this case, there is no much benefit in querying points with dense neighborhoods because it will not provide much information about the labels of the remaining unlabeled instances.

For a given point x, $p(x)$ can be estimated as the average similarity to the remaining points, $\frac{\frac{\sum_{i=1}^{n} exp(-\|x-x_i\|^2)}{n-1}}{Z_n}$, where n is the total number of points, and Z_n is the normalization constant. From an active learning point of view, however, we are more interested in the close neighborhood of a point since it will directly be affected by the labeling of that point. Thus, we constrain the density estimation to the points in a local neighborhood. That is, the density estimate for a given point will depend only on those unlabeled neighbors whose distance to the point is smaller than a pre-defined threshold:

$$\hat{p}(x) = \frac{\sum_{k \in N_x} exp(-\|x - x_k\|^2)}{Z'_n} \tag{3.23}$$

where $N_x = \{r \in I_u | \ \|x - x_r\| < t\}$ is the set of indices of the unlabeled points whose distance to x is smaller than the threshold t. Z'_n is again the normalization constant. Note that Equation 3.23 is not an average; it does not divide by the size of the neighborhood; $|N_x|$. By enforcing the estimate in Equation 3.23, we guarantee that it depends on *the number of neighbors* as well as *their proximity*. As we discussed earlier, a density measure

itself cannot fully capture the information content of a point in terms of the amount of surprise we would get if we knew the true label. The conditional entropy of the unknown label y given the instance x and the model w is:

$$ET(Y \mid x, w) = -\sum_{y} P(y \mid x, w) \log P(y \mid x, w) \tag{3.24}$$

It measures the amount of information (uncertainty) of the discrete random variable Y, and is maximum when $P(y \mid x, w) = \frac{1}{|Y|}$, where $|Y|$ is the number of values that the class variable Y can get. For binary problems, i.e., $y \in \{-1, +1\}$, we have the following equality:

$$\arg\max_{i \in I_u} ET(Y_i \mid x_i, w) = \arg\max_{i \in I_u} \{ \min_{y_i \in \{\pm 1\}} \{P(y_i \mid x_i, w)\}\} \tag{3.25}$$

We adopted this equation for the experiments reported in this section. For multi-class problems, the conditional entropy can be equivalently used. Since we do not know the true model w, we used its approximation $\hat{w}$ from the logistic regression classifier trained with the data seen up to the present point. Finally, we propose using an uncertainty weighted density measure:

$$\hat{p}(x) = \sum_{k \in N_x} \exp(-\|x - x_k\|^2) * \min_{y_k \in \{-1, +1\}} \{P(y_k \mid x_k, \hat{w})\} \tag{3.26}$$

For simplicity, we leave out the normalization constant since we are interested in the relative density rather than the absolute density. Equation 3.26 captures both the density of a given point and also the information content of its neighbors. Furthermore, each neighbor's contribution to the density score is weighed by its uncertainty; hence it reduces the effect of the neighbors at which the current learner has high confidence. Formally, we define the utility $U(i, j)$ of a pair of points as the sum of the density estimate for each point. By the definition of N_x, it includes the point x in consideration. Hence, Equation 3.26 includes the uncertainty of the point itself, $\min_{y \in \{-1, +1\}} \{P(y \mid x, w)\}$, as a summand with weight equals to $\exp(-\|x - x\|^2) = 1$. We propose to give more flexibility to that uncertainty term by introducing a regularization coefficient. It quantifies a trade-off of the information content of an instance with the proximity weighted information content of its neighbors. This allows us to define the utility function as follows:

$$\begin{aligned} U(i, j) = \log\{\hat{p}(x_i) + \hat{p}(x_j)\} = \\ \log\Bigg\{ & \sum_{k \neq i \in N_{x_i}} \exp(-\|x_i - x_k\|^2) * \min_{y_k \in \{\pm 1\}} \{P(y_k \mid x_k, \hat{w})\} \\ & + \sum_{r \neq j \in N_{x_j}} \exp(-\|x_j - x_r\|^2) * \min_{y_r \in \{\pm 1\}} \{P(y_r \mid x_r, \hat{w})\} \\ & + s * (\min_{y_i \in \{\pm 1\}} \{P(y_i \mid x_i, \hat{w})\} + \min_{y_j \in \{\pm 1\}} \{P(y_j \mid x_j, \hat{w})\}) \Bigg\} \end{aligned} \tag{3.27}$$

Note x_i and x_j are treated separately in the last summand where s is the regularization constant. We tried a range of values from 1 to 3 for s on another dataset that is not reported in this paper. Different values did not effect the results in any significant way; hence we picked $s = 2$ which is reasonable given the restriction on the size of the neighborhood. Equation 3.27 is substituted into Equation 3.22 to get the final score $S(i, j)$. Thus, our strategy is to select instances for labeling that have the largest score:

$$\{i^*, j^*\} = \arg\max_{i \neq j \in I_u} S(i, j) = \arg\max_{i \neq j \in I_u} \|x_i - x_j\|^2 * U(i, j) \tag{3.28}$$

The pseucode of the algorithm is given as Algorithm 2.

Algorithm 2 Paired Sampling

Input: Data $(X, y) = \{(x_1, y_1), ..., (x_m, y_m)\}$
Output: Logistic Regression Classifier
Program
Compute the distance matrix D
for all $(x_i, x_j) \in X$ **do**
  $D_{ij} = \frac{1}{\rho}\left\{\ln(1 + \min_{p \in P_{i,j}} \sum_{k=1}^{|p|-1}(e^{\rho\|p_k - p_{k+1}\|} - 1))\right\}$
end for
Apply MDS to D to obtain the data in MDS space $(\widetilde{X}, y) = \{(\widetilde{x}_1, y_1), ..., (\widetilde{x}_m, y_m)\}$
Divide the data into training set T and unlabeled set U s.t. $(\widetilde{X}, y) = T \cup U$
repeat
  Train logistic regression on T to get $P(y \mid \widetilde{x}, \hat{w})$
  for all $i \neq j \in I_u$ **do**
    Compute $S(i, j) = \|\widetilde{x}_i - \widetilde{x}_j\|^2 * U(i, j)$ using Equation 3.27
  end for
  Pick $\{i^*, j^*\} = \arg\max_{i \neq j \in I_u} S(i, j)$
  Update $T = T \cup \{(\widetilde{x}_{i^*}, y_{i^*}), (\widetilde{x}_{j^*}, y_{j^*})\}$ and
  $I_u = I_u - \{i^*, j^*\}$
until stopping criterion

3.3.3 Experimental Evaluation

We conducted a set of experiments in order to evaluate our method on six benchmarks, details of which can be found in Table 3.3.3. We compared our proposed method with four other strategies:

1. Most Uncertain: We rank the unlabeled points according to their uncertainty, i.e., $\min_y\{P(y \mid x, \hat{w})\}$ (via Equation 3.19), in descending order. Then, we select the top two points with the most uncertainty.

Data	Breast	Heart	Flare	Face	Glass2	g50c
Size	277	270	1066	2500	163	550
+/-	0.413	0.800	1.234	1	1.144	1
Dim	9	13	13	400	9	50

Table 3.2: Properties of the datasets used in Paired Sampling

2. Density Only: It differs from the proposed method by considering only the proximity of the neighbors for computing the density.

3. Representative Sampling [Xu *et al.*, 2003]: We used a similar implementation explained in the previous section except that the centroids of the two largest clusters are chosen to be labeled. Also, penalty factor C in SVM, and k in clustering are optimized minimizing the test error to obtain the best possible performance[2].

4. Random Sampling

For each dataset, we conducted 10 runs. For each run, we randomly picked just 2 instances, one from each class, to form the initial training set. This number is usually larger for many active learning studies including [Nguyen and Smeulders, 2004; Schein and Ungar, 2005]. We left the remaining data as the unlabeled pool. We ran each active learning method for 20 iterations and at each iteration we selected 2 instances to label. Hence, we actively sampled 40 instances in total. We note that the scoring function $S(i, j)$ needs to be computed for each pair of unlabeled points, which takes $O(|I_u|^2)$ time per iteration. In order to reduce the computational cost, we rank the unlabeled points from most to least uncertain.The top $p\%$ is selected and pairwise scores are computed for this subset. The algorithm then picks instances to label from this representative subset of unlabeled data. This is only enforced on the Flare and Face datasets by setting $p\%$ to 30% and 20%, respectively. Every time a new pair of samples is added to the training set, the classifier is re-trained and evaluated on the remaining unlabeled portion of the data. At each iteration, we reported the error of the active sampling method. We averaged those results over 10 runs for comparison.

Figure 3.6 shows the results on the UCI Breast data comparing five methods. Our method has the steepest decrease as well as the lowest final error rate which is very close to the optimal achieved using the entire training data. Figure 3.7 shows the results on four of the remaining datasets. We only show three methods in each graph to ease visual legibility. The top two graphs in Figure 3.7 compare our method against uncertainty sampling and representative sampling, whereas the bottom two graphs compare it against the density only version and representative sampling. Our method outperforms the others on each data. The density only version performs slightly better than our method for the initial iterations on

[2]Parameter tuning minimizing the test error has only been used for representative sampling. Parameters in other methods are tuned as explained.

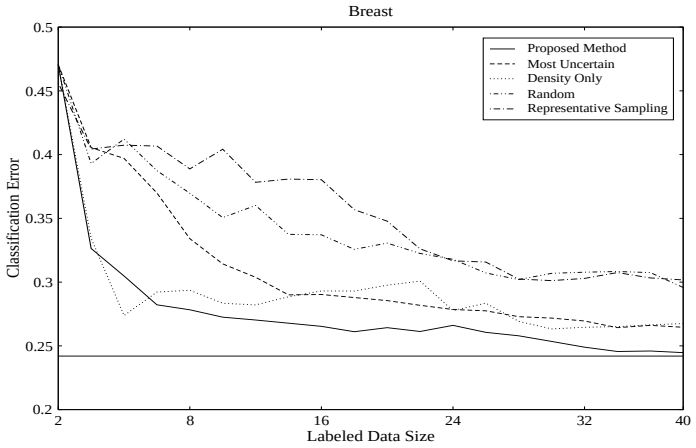


Figure 3.6: Results on UCI Breast data. The solid horizontal line indicates the 10-fold cross-validation error using the entire data as the training data.

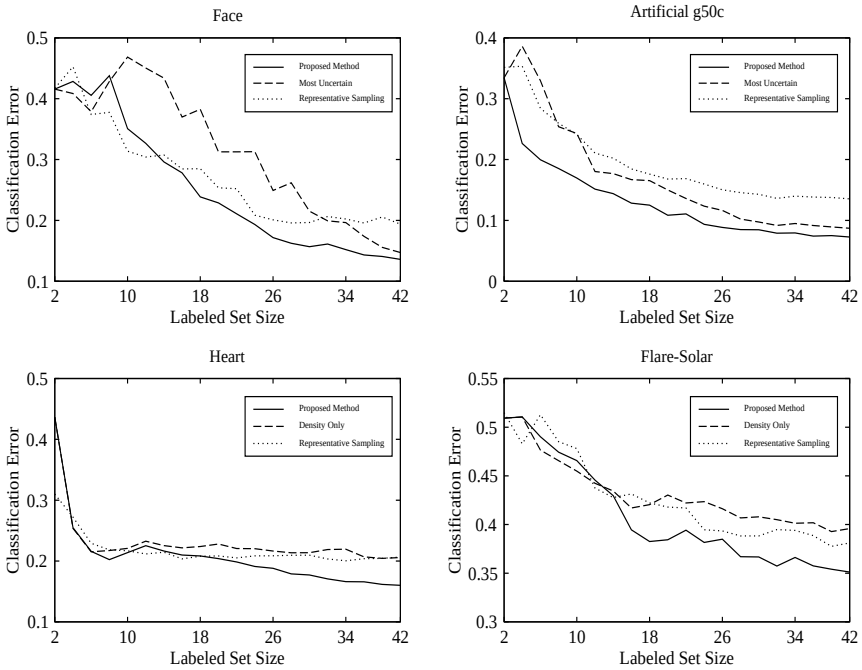


Figure 3.7: Results on four different datasets

Table 3.3: Comparison of five different active learners on all datasets

Data	Proposed Method	Most Uncertain	Density Only	Representative	Random
Breast 5	**0.278 (-24.6%)**	0.334 (-9.04%)	0.293 (-20.5%)	0.380 (+2.9%)	0.369
Breast 11	**0.264 (-20%)**	0.285 (-13.6%)	0.297 (-10%)	0.347 (+5.1%)	0.330
Breast 17	**0.249 (-18.8%)**	0.269 (-12.3%)	0.264 (-14%)	0.302 (-1.6%)	0.307
Heart 5	**0.213 (-18.3%)**	0.245 (-6.1%)	0.220 (-15.7%)	0.216 (-17.2%)	0.261
Heart 11	**0.198 (-4.3%)**	0.208 (+0.4%)	0.220 (+6.2%)	0.205 (-0.9%)	0.207
Heart 17	0.166 (-13.5%)	**0.164 (-14.5%)**	0.219 (+14%)	0.20 (+4.1%)	0.192
Flare 5	0.465 (+5.2%)	0.454 (+2.7%)	0.454 (+2.7%)	0.478 (+8.1%)	**0.442**
Flare 11	**0.394 (-1.6%)**	0.451 (+10%)	0.422 (+2.9%)	0.417 (+1.7%)	0.410
Flare 17	**0.366 (-8.7%)**	0.449 (+11.9%)	0.401 (0%)	0.393 (-1.9%)	0.401
Face 5	0.350 (-1.9%)	0.468 (+31%)	0.420 (+17.6%)	**0.313 (-12.3%)**	0.357
Face 11	**0.210 (-23.3%)**	0.312 (+13.8%)	0.287 (+4.7%)	0.252 (-8%)	0.274
Face 17	**0.151 (-32.5%)**	0.196 (-12.5%)	0.189 (-15.6%)	0.202 (-9.8%)	0.224
Glass2 5	0.339 (-11%)	0.442 (+16%)	0.392 (+2.8%)	**0.326 (-14.4%)**	0.381
Glass2 11	0.317 (-7%)	0.341 (0%)	0.324 (-4.9%)	**0.31 (-9%)**	0.341
Glass2 17	**0.266 (-8.9%)**	0.292 (0%)	0.275 (-5.8%)	0.30 (+2.7%)	0.292
g50c 5	**0.169 (-46.3%)**	0.242 (-23.1%)	0.187 (-40.6%)	0.241 (-23.4%)	0.315
g50c 11	**0.110 (-37.8%)**	0.136 (-23.1%)	0.128 (-27.6%)	0.168 (-5%)	0.177
g50c 17	**0.079 (-34.1%)**	0.094 (-21.6%)	0.102 (-15%)	0.139 (+15.8%)	0.120

Flare-Solar, and similarly representative sampling performs slightly better on early iterations on Face detection. But our method readily achieves significantly better performance on both cases as more data is sampled. A more thorough comparison of all methods on six datasets is given in Table 3.3.3. In Table 3.3.3, we show the error rates for each method at three different points in iteration: 5th, 11th and 17th iterations. The first column in Table 3.3.3 shows the dataset and the corresponding iteration at which the error rates are compared. The percentage error reduction against the random sampling baseline is given in parenthesis. Lowest error rates are given in bold. Our method wins on the majority of the cases. Whenever it loses, there is only a slight difference between our method and the winner so our method is still comparable on cases where it is not the best.

We see that our method is the best on all except few cases. To quantify this, we did a 2-sided paired t-test at the 95% confidence level on the entire reported operating range to test the hypothesis that our method has significantly lower error than each of its competitors. Thus, it was tested against each method separately and the corresponding p-values were recorded. Our method always performed significantly better ($p < 0.001$) than the density only version on all datasets. It also outperformed most uncertain with $p < 0.001$ on all except the Heart data where $p < 0.05$. It outperformed random sampling on Flare with

$p < 0.05$, on Face with $p < 0.01$ and with $p < 0.001$ on the rest. Moreover, it outperformed representative sampling with $p < 0.001$ on Breast, Flare, g50c, and with $p < 0.05$ on Face whereas both are comparable on Glass2 and Heart datasets. However, Table 2 shows that our method improves more steeply and wins in the later iterations on these two datasets. When we only compared the errors for the last 10 iterations on Glass2 and Heart, then our method wins with $p < 0.05$ and $p < 0.001$, respectively.

We also conducted another set of experiments to evaluate the cluster assumption. We reran our method without transforming the data. In other words, we computed the Euclidean pairwise distances in the original input space, and selected the instances to label according to Equation 3.28. It performed worse than or comparable with our original method. On Heart and g50c they both did equally well. In fact, the average absolute difference between the errors of the two methods on Heart data is 0.016 ± 0.009, and 0.01 ± 0.005 on g50c data. On Glass2, Flare, Face and Breast datasets the untransformed version is outperformed by our method with $p < 0.001$ significance.

3.4 Chapter Conclusions

Ensemble techniques are effective in supervised learning tasks in machine learning literature. We have demonstrated in this chapter that it is also very effective in active learning tasks especially if the right techniques are merged at the right granularity. The first ensemble technique, DUAL, investigated robust combinations of uncertainty and density information. Empirical evaluation demonstrates that, in general, this approach leads to more effective sampling than strong state-of-the-art baselines. Xu *et al.* also propose a hybrid approach to combine representative sampling and uncertainty sampling. Their method, however, only applies to SVMs and only tracks the better performing strategy rather than outperforming both individual strategies. Baram *et al.* also reports comparable performance for COMB to the best individual sampling strategy, but it is sometimes marginally better, and sometimes marginally worse and hence is not consistently the best performer. Our performance, on the contrary, exceeds that of the individually best sampling strategy in most cases by statistically significant margins. Hence, DUAL clearly goes beyond COMB in terms of lower classification error and faster convergence. Furthermore, our framework is general enough to fuse active learning methods that exhibit differentiable performance on the whole operating range. It can also be easily generalized to multi-class problems: one can estimate the error reduction globally or per-class using class-weighted or instance-weighted average, and then use the same cross-over criterion.

The main contributions of DUAL are in estimating the error of one method using the labeled data selected by another, and robustly integrating their outputs when one method is dominant (Equation 3.14 vs. Equation 3.15). Directions for future work might include generalization of DUAL using a relative success weight across multiple (more than two)

strategies, maximizing ensemble diversity [Melville and Mooney, 2003; Melville and Mooney, 2004; Baram *et al.*, 2003]. Moreover, investigation of better methods for estimating the cross-over, such as estimating a smoothed version of $\frac{\partial \hat{\epsilon}}{\partial x_t}$ rather than a local-only version might be necessary to build more robust systems.

The second half of this chapter deals with exploring a proximity-weighted conditional-entropy-based criterion for active learning. Our contributions are two-fold: First, our technique combines the density, uncertainty and dissimilarity-across-classification-boundary strategies into a unified framework. Second, it uses a density-sensitive distance metric to measure the dissimilarity between pairwise instances, maximizing the likelihood of sampling both sides of a decision boundary in a totally unsupervised process. Distances of points within the same cluster are reduced while those from different clusters are dominated by the inter-cluster distances. Our empirical results in various domains demonstrate that our method outperforms others in terms of both error reduction and fewer number of labeling queries required to obtain a certain level of accuracy. One drawback of the proposed method is the time complexity of the data transforming process that it prohibits the application to very large datasets. Future work could address efficiency improvements, for instance by extending kd-trees and by computing a k-nearest-neighbor fanout graph, vs the full graph. Moreover, extensions to other probabilistic classifiers, such as Gaussian Processes and to different kernels might lead to valuable further developments.

Chapter 4

Active Learning for Rank Learning

4.1 Introduction

Classification problems are not the only examples where active sampling can help greatly reduce the labeling effort. Dynamic ranking-based problems also require lots of training examples together with their corresponding permutations. For instance, search engines must rank results for each query; review and recommendation sites rank competing products; Netflix ranks movie preferences based on prior user selections and feedback; Amazon ranks books based on collaborative filtering; service recommendation sites rank providers based on match to user requests, price, quality or reliability as judged by others, and geographical distance to the user. Obtaining the ordered preferences of the customers or the relevance judgments of web documents given queries is an extremely time-consuming and expensive task. While we have discussed in the previous chapter that this constitutes a serious problem for classification, it is even a bigger issue in the ranking domain. In ranking, the target domain is the set of permutations and hence it is more costly to acquire a complete preference order rather than absolute class labels.

Learning to rank has recently drawn broad attention among machine learning researchers [Joachims, 2002; Freund *et al.*, 2003; Cao *et al.*, 2006]. The ranking task is to induce a mapping (ranking function) from a predefined set of instances to a set of partial (or total) orders. Rank learning thus far has mostly been applied to improving document retrieval, where a global ordering of documents is constructed based on the relevance scores of each document to each given query. Like many other supervised learning tasks, this requires a human expert to carefully examine the documents in order to assign relevance-based permutations. Considering the size and time complexity of the rank labeling process, it becomes crucial to design methods that will considerably reduce the labeling effort without significantly sacrificing ranking accuracy. As previous chapter discusses, active

learning paradigm helps reduce the labeling effort, sometimes by orders of magnitude, via incrementally sampling from an unlabeled pool of instances and requesting the labels (or rank decisions) of the selected ones with the goal of maximizing the information value to the learning function. Active learning approaches studied in the context of classification generally aim to minimize the error rate; hence do not take into account the rank oder which is essential in ranking tasks (e.g. an error at the top of the rank order is more consequential than one further down). Moreover, ranking problems, especially in document retrieval, often deal with very skewed data distributions with relevant data being a small minority of the total data. In this chapter, we try to address these issues by proposing two active sampling methods for rank learning in the context of document retrieval.

First section describes an active sampling method for SVM rank learning [Joachims, 2002] (RankSVM in short). SVMs are regarded as good rankers, which is a claim theoretically justified by [Steck, 2007], since they implicitly optimize a ranking quality measure, namely the area under the ROC curve (AUC). More specifically, Steck shows that minimizing hinge loss is an accurate approximation for maximizing AUC. Our method relies on this relationship but goes well beyond it by presenting a robust loss estimation that is crucial for typically highly skewed ranking datasets.

The second section proposes a sampling framework for RankSVM [Joachims, 2002] and RankBoost [Freund *et al.*, 2003] based on maximizing the estimated loss differential over unlabeled data. The proposed framework considers the capacity of an unlabeled example to update the current model if rank-labeled and added to the training set. We define this capacity as a function that estimates the error of a ranker introduced by the addition of a new example. The capacity function takes different forms in RankSVM and RankBoost due to different formulations of the corresponding ranking function. However, in both cases, the goal is to select samples which are estimated to produce a faster convergence to the true ranking. Experimental results indicate a significant advantage favoring both sampling strategies over strong baselines on real-life corpora.

4.2 Active Learning via Optimizing AUC

4.2.1 Motivation

Before we start explaining the details of our algorithm, we recapitulate the relationship between the AUC and the hinge rank loss as proposed by [Steck, 2007]. Understanding this relationship is essential in building our RankSVM active sampling method since we rely on it as a theoretical justification for our loss minimization framework. The hinge loss of a real-valued classifier is defined as $L^H = \sum_{i=1}^{N}[1 - y_i(c_i - \vartheta)]_+$. $c_i \in \mathbb{R}$ is the classifier output, $y_i \in \{-1, +1\}$ are the binary class labels, ϑ is the real-valued decision threshold, and N is the total number of training instances. $[.]_+$ denotes the positive part, i.e. $[a]_+ = a$

if $a > 0$, and 0 otherwise. Let the classifier outputs c_i be sorted in ascending order, i.e. the smallest output value is assigned the lowest rank. Then, the rank version of the standard hinge loss proposed by [Steck, 2007] becomes:

$$L^{HR} = \sum_{i=1}^{N} [\frac{1}{2} - y_i(r_i - \bar{\vartheta})]_+ \tag{4.1}$$

r_i is the rank of the data point x_i, $\bar{\vartheta}$ is the rank threshold defined as $\bar{\vartheta} = max\{r_i : c_i \leq \vartheta\} + \frac{1}{2}$ which is half way between two neighboring rank positions where one belongs to the positive(negative) class, and the other belongs to the other class. Note that L^{HR} increases linearly in r_i tracking the standard hinge loss in c_i.

The AUC measure is equivalent to the probability that a randomly chosen member of class $+1$ will have a smaller estimated probability of belonging to class -1 than a randomly chosen member of class -1 [Hand and Till, 2001]. Moreover, AUC is equivalent to the Wilcoxon-Mann-Whitney test statistic [Mann and Whitney, 1947; Wilcoxon, 1945]; thus it can be written in terms of pairwise comparison of ranks:

$$A = \frac{1}{n^+ n^-} \sum_{j=1}^{n^+} \sum_{i=1}^{n^-} I(r_j^+ > r_i^-) \tag{4.2}$$

where I is the indicator function where $I(a) = 1$ if a is true, and 0 otherwise. n^+ and n^- denote the number of positive(relevant) and negative(nonrelevant) examples, respectively. Steck shows that AUC can be written in terms of the hinge rank loss defined in Equation 4.1 as follows [Steck, 2007]:

$$A \geq 1 - \frac{L^{HR} - C}{n^+ n^-} \tag{4.3}$$

where C is a constant, independent of the rank order (see [Steck, 2007] for further details). The hinge rank loss is the decisive term in the lower bound on the AUC. Hence, minimizing the hinge rank loss guarantees maximizing the AUC. Similarly, the bipartite ranking error R adopted by [Rajaram *et al.*, 2007] is directly coupled with the AUC; i.e. $R = 1 - AUC$. Hence, effectively reducing the bipartite loss guarantees an increase in the AUC. This is supported empirically in Section 4.2.3 where both our method and the bipartite ranking loss based method of [Rajaram *et al.*, 2007] improve the AUC.

4.2.2 SVM Active Learning for Ranking

Relying on the relationship between the hinge rank loss and the AUC, we propose selecting examples that will minimize the expected hinge rank loss in order to maximize rank-learning as measured by the AUC. Expected loss minimization has been studied before for

active learning, but in classification [Donmez *et al.*, 2007; McCallum and Nigam, 1998; Nguyen and Smeulders, 2004], rather than in ranking. Unfortunately, active sampling designed for classification error cannot directly apply to the ranking scenario. Ranking loss is based on the relative position of the entities instead of the absolute class label. The rank position of an error matters significantly since the top of the ordered list is more important than the bottom. Moreover, it is crucial to take into account the data skew typical in ranking datasets when designing sampling algorithms for ranking. In this section, we describe a loss minimization algorithm for active learning in ranking to address these issues.

The expected loss minimization criterion requires each unlabeled example to be tested separately in order to calculate the expected future error if it were chosen for a rank-label. Clearly, this is not efficient for large datasets. Nguyen and Smeulders [2004] proposed selecting the examples that have the largest contribution to the current estimated error instead of choosing the sample that produces the smallest future error; $s = \arg\max_{i \in I_U} E_{y|x}[(y_i - \hat{y}_i)^2 \mid x_i]$ where I_U is the set of indices of the unlabeled data. We adopt a similar approach but our selection criterion is based on the hinge rank loss rather than the typical loss functions used for classification such as the squared loss. The optimization problem for SVM Rank Learning [Cao *et al.*, 2006; Joachims, 2005] can be written as a loss minimization problem as follows:

$$L^H = \sum_{i=1}^{l} [1 - z_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle]_+ \tag{4.4}$$

plus a complexity penalty [1]. $x_i^{(1)}$ and $x_i^{(2)}$ correspond to one relevant and one nonrelevant example, respectively, for a given query (we omit the query subindex for notational simplicity). $z_i = +1$ if $x_i^{(1)} \succ x_i^{(2)}$, and $z_i = -1$ otherwise. By algebraic reformulations:

$$L^H = \sum_{i=1}^{l} [1 - z_i \langle w, x_i^{(1)} \rangle + z_i \langle w, x_i^{(2)} \rangle]_+$$

$$L^H = \sum_{i=1}^{l} [(1 - y_i^{(1)} \langle w, x_i^{(1)} \rangle) + (1 - y_i^{(2)} \langle w, x_i^{(2)} \rangle) - 1]_+$$

[1] The decision threshold ϑ is typically chosen as 0 without loss of generality.

where $y_i^1 = z_i$ and $y_i^2 = -z_i$. The rank version of the above loss function then becomes[2]:

$$L^{HR} = \sum_{i=1}^{l} [(\frac{1}{2} - y_i^{(1)}(r_i^{(1)} - \bar{\vartheta})) + (\frac{1}{2} - y_i^{(2)}(r_i^{(2)} - \bar{\vartheta})) - 1]_+$$

$$L^{HR} \leq \sum_{i=1}^{l} [\frac{1}{2} - y_i^{(1)}(r_i^{(1)} - \bar{\vartheta})]_+ + [\frac{1}{2} - y_i^{(2)}(r_i^{(2)} - \bar{\vartheta})]_+ \quad (4.5)$$

where the rank threshold $\bar{\vartheta}$ is specific to a given query q. Since the RankSVM implementation takes as input vectors corresponding to individual data points, we use, for convenience, the right hand side of the above inequality as the loss function instead of Equation 4.4 that uses pairwise difference vectors. This corresponds to selecting the example pair that has the largest expected hinge rank loss $E_{y|x}[[\frac{1}{2} - y_i^{(1)}(r_i^{(1)} - \bar{\vartheta})]_+ + [\frac{1}{2} - y_i^{(2)}(r_i^{(2)} - \bar{\vartheta})]_+ \mid (x_i^{(1)}, x_i^{(2)})]$, where the expectation is taken over the posterior distribution of y given x. However, picking an optimal pair requires $O(n^2)$ comparisons in a set of size n; hence it is impractical for large-scale ranking applications. Therefore, we proceed with selecting individual example(s) per query with the largest expected loss. A selected example may not be optimal compared to the pair selected according to (4.5); however, it is a reasonable choice for performance-time tradeoff. In fact, our empirical results show that this strategy is quite effective for learning a good ranker with few labeled instances.

$$E\big[[\frac{1}{2} - y_k(r_k - \bar{\vartheta})]_+ \mid x_k\big] =$$

$$\hat{P}(y_k = 1 \mid x_k)[\frac{1}{2} - (r_k - \bar{\vartheta})]_+ + \hat{P}(y_k = -1 \mid x_k)[\frac{1}{2} + (r_k - \bar{\vartheta})]_+ \quad (4.6)$$

Equation 4.6 favors points with the highest uncertainty. RankSVM optimizes pairwise preferences, and it may not learn a reasonable decision threshold. Thus, the estimated decision boundary may not be in correspondence with the true rank threshold. This bias may not affect the ranking performance as long as the correct order is obtained. However, it presents a larger problem in the active-learning-to-rank context. The most uncertain points in ranking problems can be considered as the points whose rankings are closest to the rank threshold. This corresponds to multiple thresholds in a multi-level rating scenario with uncertain points being specific to each threshold. Therefore, the rank threshold should define the decision boundary. In order to simulate this effect, we propose a normalized rank

[2]The transformation from Equation 4.4 to 4.5 is possible when the data has binary relevance judgments, which is the case for the majority of the benchmark test collections including ones used in this work.

distance measure and incorporate it into Equation 4.6 to obtain the following:

$$E\big[[\frac{1}{2} - y_k(r_k - \bar{\vartheta})]_+ \mid x_k\big] = \Big\{ \hat{P}(y_k = 1 \mid x_k) \frac{[\frac{1}{2} - (r_k - \bar{\vartheta})]_+}{|r_{min} - \bar{\vartheta}|}(1-\eta) + \\ \hat{P}(y_k = -1 \mid x_k) \frac{[\frac{1}{2} + (r_k - \bar{\vartheta})]_+}{|r_{max} - \bar{\vartheta}|}\eta \Big\} \quad (4.7)$$

where $r_{min} = 1$ and $r_{max} = |I_{U_q}|$ since the most relevant examples have the highest rank and vice versa. $|I_{U_q}|$ denotes the size of the unlabeled set for the query q. The normalization in Equation 4.7 regularizes the effect of the points that are ranked further below in the rank order, and those ranked at the top. Generally, the number of points that are ranked above the threshold would be small since there are only a handful of positive(relevant) examples compared to the large amount of negative(non-relevant) examples in tasks such as document retrieval. Without normalization, the points with rank $r_k > \bar{\vartheta}$ have little chance of being selected since the rank distance $r_k - \bar{\vartheta}$ is small. Dividing both distances by their maximum renormalizes them into the same scale, favoring a more balanced estimation. $0 < \eta < 1$ is a trade-off parameter that controls the weight of the examples on either side of the rank threshold. Setting $\eta > 0.5$ gives more weight to the examples that are mistakenly ranked above the threshold but are in fact negative(nonrelevant). We tuned the η parameter on a small dataset not reported in this paper and that resulted in fixing η at 0.6. Better tuning on a validation set could further improve our results. The outline of our selection algorithm is given in Algorithm 3.

Algorithm 3 Active Sampling using Hinge Rank Loss

Input: Labeled data Lb, Unlabeled data Un, # rounds T
Output: A ranking function $f(x) = \langle w, x \rangle$
Program
for all $t = 1 : T$ **do**
Learn a ranking function f on Lb
Rank the examples $x_k \in Un$ in ascending order acc. to $f(x_k)$
Estimate their posterior, i.e. $\hat{P}(y_k \mid x_k)$
Select the top l examples, $Un^{(l)}$, when sorted in descending order w.r.t.:

$x^* = \arg\max_x \hat{P}(y_k = 1 \mid x_k) \frac{[\frac{1}{2} - (r_k - \bar{\vartheta})]_+}{|r_{min} - \bar{\vartheta}|}(1-\eta) + \hat{P}(y_k = -1 \mid x_k) \frac{[\frac{1}{2} + (r_k - \bar{\vartheta})]_+}{|r_{max} - \bar{\vartheta}|}\eta$
Remove x^* from Un and update $Lb = Lb + \{x^*, y^*\}$
end for

The class probability $\hat{P}(y_k \mid x_k)$ in step 3 of Algorithm 3 can be estimated by fitting a sigmoid to the ranking function output:

$$\hat{P}(y_k \mid x_k) = \frac{1}{1 + exp(-y_k * f(x_k))} \quad (4.8)$$

In this paper, we propose a simple method to construct a calibrated estimate for the posterior class distribution. First, we propose a way to estimate the rank threshold $\bar{\vartheta}$ and then we use it to calibrate the posterior. We assume that the true ranking function maximizes the score difference between the lowest ranked relevant and the highest ranked non-relevant examples. We sort the data in ascending order of rank scores and compute the absolute difference of the scores of two neighboring examples. The threshold is then chosen as summarized in Algorithm 4.

Algorithm 4 Posterior Calibration

Input: a ranking function f, unlabeled data Un
Output: the posterior class distribution $\hat{P}(y \mid x)$
1. Sort the examples $x \in Un$ acc. to $f(x)$ to obtain a rank order, i.e. $x_1 \prec x_2 \prec ... \prec x_{r_{max}}$
2. Compute $|f(x_i) - f(x_{i+1})| \; \forall i = 1, 2, ..., r_{max} - 1$
3. The threshold then becomes: $\hat{\bar{\vartheta}} = \arg\max_{i=1,...,r_{max}-1} |f(x_i) - f(x_{i+1})|$
4. $\hat{P}(y \mid x) = \frac{1}{1+exp(-y*f(x)+f(x_{\hat{\bar{\vartheta}}}))}$

Now we can calibrate the estimate in Equation 4.8 by adding the output score of the instance whose rank is equal to the estimated threshold, i.e.

$$\hat{P}(y_k \mid x_k) = \frac{1}{1 + exp(-y_k * f(x_k) + f(x_{\hat{\bar{\vartheta}}}))} \tag{4.9}$$

We substitute the above estimate into Equation 4.7 for active instance selection. Now, it should be clear that Equation 4.7 favors points with the highest uncertainty with respect to the current ranker. This is consistent with many other active sampling methods proposed for classification in which uncertainty-based selection criterion plays an effective role [Donmez *et al.*, 2007; Tong and Koller, 2000; Xu *et al.*, 2003], although none of them has previously adopted a normalized uncertainty-based criterion for rank-learning.

4.2.3 Experimental Evaluation

In order to assess the effectiveness of our active-sampling method, we used the Learning to Rank (LETOR) Benchmark dataset [Liu *et al.*, 2007]. We report results of our studies on the TREC 2003 and TREC 2004 topic distillation tasks [Craswell *et al.*, 2003; Craswell and Hawking, 2004] in LETOR, namely TD2003 and TD2004. The relevance assessments are binary and created by human judges. There are 44 features for each document-query pair. In our evaluation, we used query-based normalization into the $[0, 1]$ interval for the features, as suggested by the producers of the LETOR [Liu *et al.*, 2007] package. There are 50 and 75 queries, each with ~ 1000 documents, in TD2003 and TD2004, respectively. The percentage of relevant documents is 1% in TD2003 and 0.6% in TD2004. The TD2003 and

TD2004 datasets come with standard train and test splits divided into 5 folds. In each fold, we randomly picked 11 documents (one relevant and 10 non-relevant) for each query from the given training data to construct the initial labeled set. The remaining training data is used as the unlabeled set. Each sampling method selects $l = 5$ unlabeled instances per query at each round. Then, the selected instances are labeled and added to the current training set. The performance of the ranker is re-evaluated on the testing data. This procedure is repeated for 20 iterations on every fold, and the averaged results are reported.

We tested the performance of our method (denoted by *LossMin*) against four baselines: the entropy-based sampling method of [Rajaram *et al.*, 2007] (denoted by *Entropy*), the uncertainty sampling heuristic of [Yu, 2005] (denoted by *Uncertain*), the divergence-based sampling strategy of [Amini *et al.*, 2006] (denoted by *Diverse*), and random sampling (denoted by *Random*). *Entropy* method [Rajaram *et al.*, 2007] samples the most confusing instances for the current ranker which are identified via estimating the bipartite ranking error [Freund *et al.*, 2003] that counts an error each time a relevant instance is ranked lower than an irrelevant one. The selection mechanism of [Yu, 2005] favors the most ambiguous set of samples (data pairs that are closest in the rank scores and thus most ambiguous) with respect to the current ranker. *Diverse* method selects samples exhibiting maximal divergence (disagreement) between the current hypothesis and a randomized one [Amini *et al.*, 2006]. We report AUC, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) as the evaluation measures. The NDCG measure was evaluated at the 10^{th} rank cut-off. The performance at the beginning is the same for all methods since they start with the same initial random samples. The ranking implementation in SVMLight [Joachims, 1999] was used with a linear kernel and default parameter settings. Figure 4.1 shows the performance comparison on the TD2004 and TD2003 datasets. Our method outperforms the others on both datasets. In fact, these results are significant ($p < 0.0001$ on TD2003 and $p < 0.001$ on TD2004 w.r.t. MAP and NDCG10) according to a two-sided paired t-test at the 0.95 confidence conducted over the entire operating range. Furthermore, we can order the methods according to the significance of the results with respect to three evaluation metrics. We denote $p < 0.01$ significance level by $\gg$, $p < 0.05$ significance by $>$, and indifference by $\approx$. For MAP score, LossMin $\gg$ Entropy $\gg$ Diverse $\gg$ Uncertain on TD2004, and LossMin $\gg$ Entropy $\gg$ Uncertain $\gg$ Diverse on TD2003 dataset. For NDCG10, LossMin $\gg$ Entropy $\gg$ Diverse $\approx$ Uncertain on TD2004, and LossMin $\gg$ Entropy $\gg$ Uncertain $>$ Diverse on TD2003. Finally for AUC, LossMin $>$ Entropy $>$ Diverse $\gg$ Uncertain, and LossMin $\approx$ Entropy $\gg$ Uncertain $>$ Diverse on TD2004 and TD2003 datasets, respectively. Unfortunately, the uncertainty sampling and the divergence-based sampling have low performance. Uncertainty sampling selects instances with the most similar scores, but ignores the fact that examples with the same rank label are likely be assigned similar scores. However, such examples do not provide any additional information to the rank learner, leading to a poor performance. A similar behavior is also observed by [Amini *et al.*, 2006]. On the other hand, the low performance of the divergence-based sampling is

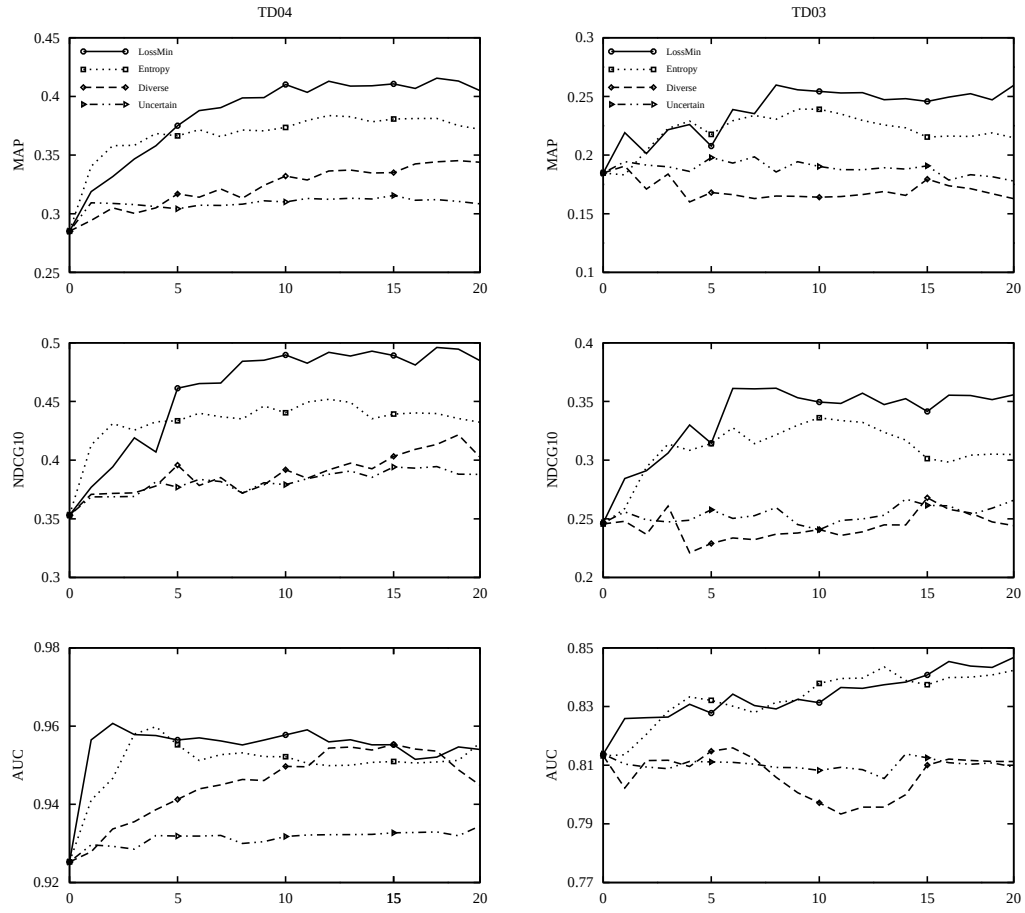


Figure 4.1: Average results on TD2004 (left figure) and TD2003 (right figure). X-axis shows the # of iterations. 5 instances per query are selected per round.

perhaps due to the heavy dependence of this algorithm to a sufficiently large initial training set. Divergence-based sampling divides the initial training set into folds and a ranking function is trained on each fold. These functions are not reliable when the training set size is small, which is the case in our empirical setting. Our method, on the other hand, effectively increases the performance even starting with minimal labeled data, which makes it quite useful for many ranking applications that otherwise would require extensive labeling effort, such as document retrieval. We also conducted a final comparison which demonstrates the superiority of our method against random sampling. Our method also gradually increases the AUC on both datasets. This supports the theoretical claim that our method optimizes the AUC metric by sampling the instances that have the largest effect on the expected hinge rank

Table 4.1: Performance and Selection Time Comparison. Iter: the # of iterations. LossMin: the proposed method, Ent: entropy-based method, Diverse: Divergence-based sampling, Un: maximum-uncertainty sampling. Time: training time + ranking time + instance selection time.

Time (cpu-sec)				MAP				Iter
LossMin	Ent	Diverse	Un	LossMin	Ent	Diverse	Un	
1.347	1.347	1.347	1.347	0.184	0.184	0.184	0.184	0
1.977	45.234	36.072	1.947	0.219	0.183	0.190	0.194	1
2.003	48.576	39.127	1.953	0.201	0.204	0.171	0.191	2
2.038	52.865	41.973	1.968	0.221	0.222	0.183	0.189	3
2.110	55.038	44.665	2.001	0.226	0.228	0.160	0.186	4

loss, and thereby on maximizing the expected AUC score. A similar monotonic improvement is also apparent for the MAP and NDCG@10 metrics, indicating that they might be well correlated with the AUC. On the other hand, *Entropy* method achieves comparable AUC score to ours even though our method has significantly better MAP and NDCG scores. This is not a very surprising result since the *Entropy* method is suited for the bipartite ranking loss R, which is inversely correlated with the AUC, i.e. $R = 1 - AUC$. But, the main advantage of our method is to use the normalized rank distance resulting in a more balanced selection for the highly skewed datasets. This sampling favors the mistakenly ranked instances at the top of the ordered list, hence boosts the metrics sensitive to the high ranks, such as MAP and NDCG.

Table 4.1 summarizes the average results on TD2003 dataset. It shows the results for the first 5 iterations and evaluate the MAP scores and the average selection time for each method. The selection time is calculated as the time period each algorithm spends to train the rank learner on the current labeled set, and then to assign scores to the unlabeled examples and finally to select new instances to be labeled. Our method achieves greater learning efficiency with modest computation time in comparison with the other baselines. Particularly, our method has very similar performance to that of the entropy-based method for the early iterations despite the greater complexity of the latter. Nevertheless, our method reaches a significantly better final performance.

4.3 Optimizing Estimated Loss Reduction for Active Sampling

4.3.1 Motivation

This section introduces our motivation for focusing on maximizing the estimated loss differential which forms the basis of the underlying sampling framework. Roy and McCallum

2001 argue that an optimal active learner is the one that asks for the labels of the examples that, once incorporated into training, would result in the lowest expected error on the test set. The expected error on the test set can be estimated using the posterior distribution $\hat{P}_{Lb}(y \mid x)$ of class labels estimated from the training set using some loss function L

$$E_{\hat{P}_{Lb}} = \int_x L(P(y \mid x), \hat{P}_{Lb}(y \mid x))P(x) \tag{4.10}$$

Their aim is then to select the point x^* such that when added to the training set with a chosen label y^*, the classifier trained on the new set $\{Lb + (x^*, y^*)\}$ would have less error than any other candidate x.

$$\forall(x, y)E_{\hat{P}_{Lb+(x^*,y^*)}} \leq E_{\hat{P}_{Lb+(x,y)}} \tag{4.11}$$

Since the true label y^* is unknown, the expectation calculation is carried out by calculating the estimated error for each possible label $y \in Y$, and then taking the average weighted by the current learner's posterior $\hat{P}_{Lb}(y \mid x)$. The naive implementation of this method would be quite inefficient and almost intractable on large datasets. Roy and McCallum 2001 address this problem using fast updates for a Naive Bayes classifier. Although efficient re-training procedures are available for some other learners such as SVMs [Cauwenberghs and Poggio, 2000], it would still be infeasible for ranking tasks, especially considering the interactive nature of ranking systems. In this paper, we propose a method to estimate how likely the addition of a new example will result in the lowest expected error on the test set *without any re-training on the enlarged training set*. Our method is based on the likelihood of an example to change the current hypothesis significantly. There are a number of reasons why we believe this is a reasonable indicator for estimating that error:

- Adding a new data point to the labeled set can only change the error on the test set if it changes the current learner.
- The more significant that change, the greater chance to learn the true hypothesis faster.
- We note that a big change in the current hypothesis might not always lead to better generalization. However, as more data is sampled and the hypothesis gets closer to the truth, it is less likely that a single outlier could hurt the performance noticeably.

In the following sections, we briefly review the RankSVM and the RankBoost algorithms and propose a novel active learning method for each.

4.3.2 SVM Rank Learning

Assume $f \in F$ is a linear function, i.e. $f(x) = \langle \vec{w}, x \rangle$, that satisfies

$$x_i \succ x_j \Leftrightarrow \langle \vec{w}, x_i \rangle > \langle \vec{w}, x_j \rangle$$

The SVM model targeting this problem can be formulated as a Quadratic Optimization problem:

$$\min_{\vec{w}} \frac{1}{2}\|\vec{w}\|^2 + B\sum \xi_{ij} \text{ subject to } \langle \vec{w}, x_i \rangle \geq \langle \vec{w}, x_j \rangle + 1 - \xi_{ij} \text{ , } \xi_{ij} \geq 0 \; \forall i, j$$

The above optimization can be equivalently written by re-arranging the constraints and substituting the trade-off parameter B for $\lambda = \frac{1}{2B}$ as follows:

$$\min_{\vec{w}} \sum_{k=1}^{K} \left[1 - z_k \left\langle \vec{w}, x_k^1 - x_k^2 \right\rangle\right]_+ + \lambda \|\vec{w}\|^2 \tag{4.12}$$

where $[.]_+$ indicates the standard hinge loss. $x^1 - x^2$ is a pairwise difference vector whose label z is positive, i.e., $z = +1$ if $x^1 \succ x^2$ and $z = -1$ otherwise. K is the total number of such pairs in the training set. Finally, a ranked list is obtained by sorting the instances according to the output of the ranking function in descending order.

4.3.3 Active Sampling for RankSVM

Let us consider a candidate example $x \in Un$, where Un is the set of unlabeled examples. Assume x is incorporated into the labeled set with a rank label $y \in Y$. We denote the total loss on the instance pairs that include x by a function of x and $\vec{w}$, i.e. $NL(x, \vec{w}) = \sum_{j=1}^{J_y} \left[1 - z_j \left\langle \vec{w}, x_j - x \right\rangle\right]_+$ where J_y is the number of examples in the training set with a different label than the label y of x. For instance, J_y is the number of negative(non-relevant) examples in the training set if y is assumed to be positive(relevant), and vice versa. The objective function to be minimized by RankSVM then becomes:

$$\min_{\vec{w}} \left\{ \lambda \|\vec{w}\|^2 + \sum_{k=1}^{K} \left[1 - z_k \left\langle \vec{w}, x_k^1 - x_k^2 \right\rangle\right]_+ + NL(x, \vec{w}) \right\} \tag{4.13}$$

Assume $\vec{w}^*$ is the solution to the optimization in Equation 4.12, and it is unique. Burges and Crisp 2000 show the necessary and sufficient conditions for the uniqueness of the SVM solution. There are only rare cases where uniqueness does not hold, thus it is a rather safe assumption to make. Since we do not actually re-run the optimization problem on the enlarged data, we restrict ourselves to the current solution (hypothesis) $\vec{w}^*$. Instead of re-optimizing, we estimate the effect of adding each candidate instance on the training loss using the current solution to tell how much incorporating x into the labeled set is likely to change the current hypothesis. First, let us consider two cases.

1. Assume $\vec{w}^* = \arg\min_{\vec{w}} NL(\vec{w}, x)$
   Then, $\vec{w}^*$ is also the solution to the optimization problem in Equation 4.13, combining the assumption with $\vec{w}^*$ being the solution to Equation 4.12. That means, adding x to the training set would not change the current hypothesis. From an active learning point of view, this example is useless since the learning algorithm is indifferent to its inclusion.

2. Assume $\vec{w}^* \neq \arg\min_{\vec{w}} NL(\vec{w}, x)$
   This is the situation where the current solution could be different if that example x were incorporated into training. The magnitude of the difference depends on the magnitude of the deviation of $NL(\vec{w}^*, x)$ from its optimal value, $min_{\vec{w}} NL(\vec{w}, x)$.

We now study the second case in more detail. Let $\hat{\vec{w}}$ be the weight vector that minimizes $NL(\vec{w}, x)$, i.e. $\hat{\vec{w}} = \arg\min_{\vec{w}} NL(\vec{w}, x)$. Then, as the difference $\|\vec{w}^* - \hat{\vec{w}}\|$ increases it becomes less likely that $\vec{w}^*$ is optimal for Equation 4.13. In other words, the current solution $\vec{w}^*$ is in most need of updating in order to compensate for the loss on the new pairs. Let us write $\hat{\vec{w}}$ in terms of $\vec{w}^*$ as follows:

$$\hat{\vec{w}} = \vec{w}^* - \Delta w$$

Minimizing $NL(\vec{w}, x)$ requires working with the hinge loss, the direct optimization of which is difficult due to the discontinuity of the derivative. However, it can still be solved using a gradient-descent-type algorithm[3].

Recall the objective function to be minimized:

$$\min_{\vec{w}} NL(\vec{w}, x) = \min_{\vec{w}} \sum_{j=1}^{J_y} \left[1 - z_j \left\langle \vec{w}, x_j - x \right\rangle\right]_+ \tag{4.14}$$

The derivative of the above equation with respect to $\vec{w}$ at a single point x_j, $\Delta\vec{w}_j$, is:

$$\Delta\vec{w}_j = \begin{cases} 0 & \text{if } z_j \left\langle \vec{w}, x_j - x \right\rangle \geq 1 \\ -z_j(x_j - x) & \text{if } z_j \left\langle \vec{w}, x_j - x \right\rangle < 1 \end{cases} \tag{4.15}$$

We substitute $\vec{w}$ in Equation 4.15 for the current weight vector $\vec{w}^*$ to estimate how the solution of Equation 4.14 deviates from it, i.e. $\|\vec{w}^* - \hat{\vec{w}}\| = \|\Delta\vec{w}\|$. We can now write the magnitude of the total derivative as a function of x and the rank label y as follows:

$$\begin{aligned} g(x, y) &= \|\Delta\vec{w}\| = \sum_j \|\Delta\vec{w}_j\| \qquad (4.16) \\ &= \sum_{j=1}^{J_y} \begin{cases} 0 & \text{if } z_j \left\langle \vec{w}^*, x_j - x \right\rangle \geq 1 \\ \| - z_j(x_j - x)\| & \text{if } z_j \left\langle \vec{w}^*, x_j - x \right\rangle < 1 \end{cases} \end{aligned}$$

[3]For a detailed discussion on solving SVM rank learning using gradient descent, see [Cao *et al.*, 2006].

$g(x, y)$ estimates how likely the current hypothesis is to be updated to minimize the loss introduced as a result of the addition of the example x with the rank label y. Thus, we use this function to estimate the ability of each unlabeled candidate example to change the current learner if incorporated into training. Since the true labels of the candidate examples are unknown, we use the current learner to estimate the true label probabilities. Then, we can take the expectation of $g(x, y)$ by taking the weighted sum over the current posterior $\hat{P}(y \mid x)$ for all $y \in Y$. Among all the unlabeled examples, we choose the one with the highest value for that expectation:

$$x^* = \arg\max_{x \in U} \sum_{y \in Y} \hat{P}(y \mid x) g(x, y) \tag{4.17}$$

$$= \arg\max_{x \in U} \left\{ \hat{P}(y = 1 \mid x) g(x, y = 1) + \hat{P}(y = -1 \mid x) g(x, y = -1) \right\} \tag{4.18}$$

4.3.4 RankBoost Learning

RankBoost is a boosting algorithm designed for ranking problems. Like all algorithms in boosting family, RankBoost learns a weak learner on each round, and maintains a distribution D_t over the ranked pairs, $X \times X$, to emphasize the pairs whose relative order is the hardest to learn. An outline of the algorithm is given in Algorithm 5. Z_t is a normalization constant, and the final ranking is a weighted sum of the weak rankings $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. For more details and theoretical discussion see [Freund *et al.*, 2003].

Algorithm 5 RankBoost

Input: initial data distribution D_1 over $\mathcal{X} \times \mathcal{X}$
for $t = 1$ **to** T **do**
  Train a weak learner on D_t
  Obtain the weak ranking $h_t : \mathcal{X} \mapsto \mathbb{R}$
  Choose a weight $\alpha_t \in \mathbb{R}$ for h_t
  $D_{t+1}(x^1, x^2) = \frac{D_t(x^1, x^2) exp(-\alpha_t (h_t(x^1) - h_t(x^2)))}{Z_t}$
end for

4.3.5 Active Sampling for RankBoost

This section introduces a similar method for active sampling for the RankBoost algorithm [Freund *et al.*, 2003]. Consider a candidate point $x \in Un$ and assume it is merged into the training set with rank label $y \in \mathcal{Y}$. Unlike RankSVM, RankBoost algorithm does not directly

operate with an optimization function. But the ranking loss with respect to the distribution at time t can be written as:

$$\sum_{x^1,x^2} D_t(x^1,x^2)I(H(x^2) \geq H(x^1)) \tag{4.19}$$

where I is defined to be 1 if the predicate holds and 0 otherwise. Hence, this is a sum over misranked pairs, assuming $x^1 \succ x^2$. The distribution at time $T+1$ can be written as:

$$D_{T+1}(x^1,x^2) = D_1(x^1,x^2)\frac{exp(H(x^2)-H(x^1))}{\prod_t Z_t} \tag{4.20}$$

The initial distribution term D_1 can be dropped without loss of generality, assuming it is uniform (which is reasonable given the fact that we do not have prior information about the data). Similarly to RankSVM, we would like to estimate how much the current ranking function would change if the point x were in the training set. We estimate this deviation by the difference in the ranking loss after enlarging the current labeled set with each example $x \in Un$. The ranking loss on the enlarged set with respect to the distribution D_{T+1} is:

$$\sum_{x^1,x^2} \frac{exp(H(x^2)-H(x^1))}{\prod_t Z_t} I(H(x^2) \geq H(x^1)) + \sum_{x^j,x} \frac{exp(H(x^j)-H(x))}{\prod_t Z_t} I(H(x^j) \geq H(x)) \tag{4.21}$$

Note that the rank label y of x is assumed to be positive (relevant) with $x \succ x_j$ in this case. We have a similar calculation for the case where y is assumed to be negative (non-relevant). We adopt the distribution D_{T+1} because 1) it can easily be written in terms of the final ranking function, 2) it contains information about which pairs remain the hardest to determine after the iterative weight updates. Then, the difference in the ranking loss between the current and the augmented set simply becomes:

$$\Delta L(x, y=1) = \sum_{x^j,x} \frac{exp(H(x^j)-H(x))}{\prod_t Z_t} I(H(x^j) \geq H(x)) \tag{4.22}$$

This difference indicates how much the current ranking function needs to be modified to compensate for the loss incurred by including this example. Note that $I(x \geq 0) \leq e^x$ for $\forall x \in \mathbb{R}$ [Freund *et al.*, 2003]. Therefore, the upper bound on ΔL can be written as:

$$\Delta L(x, y=1) \leq \sum_{x^j,x} \frac{exp(2(H(x^j)-H(x)))}{\prod_t Z_t} \tag{4.23}$$

$\Delta L(x, y = -1)$ can be similarly bounded, e.g. $\Delta L(x, y = -1) \leq \sum_{x,x^m} \frac{exp(2(H(x)-H(x^m)))}{\prod_t Z_t}$. Now, the loss difference can be estimated by taking the expectation over the possible rank labels of x with respect to the current ranker's posterior, $\hat{P}(y \mid x)$:

$$E_{\hat{P}}(\Delta L(x)) = \hat{P}(y = 1 \mid x)\Delta L(x, y = 1) + \hat{P}(y = -1 \mid x)\Delta L(x, y = -1) \tag{4.24}$$

Note the similarity with Equation 4.17 in the SVM case. Finally, we select the instance x that has the highest expected loss differential, e.g. $x^* = \arg\max_x E_{\hat{P}}(\Delta L(x))$. For notational clarity, we take the maximum over the upper bound in Equation 4.23 as follows:

$$x^* = \arg\max_{x \in U} \Big\{ \hat{P}(y = 1 \mid x)(\sum_{x^j, x} exp(2(H(x^j) - H(x)))) + \hat{P}(y = -1 \mid x)(\sum_{x, x^m} exp(2(H(x) - H(x^m)))) \Big\} \tag{4.25}$$

For simplicity, we leave out the normalization constant $\prod_t Z_t$ since we are interested in the relative expectation rather than the absolute expectation.

4.3.6 Final Selection

The sample selection in both RankSVM and RankBoost requires estimating a posterior label distribution. We adopt a sigmoid function to estimate that posterior in the SVM case, as suggested by [Platt, 1999]:

$$\hat{P}(y \mid x) = \frac{1}{1 + exp(-y * f(x) + K)}$$

where $f(x)$ is the real-valued score of the ranking algorithm, and K is a constant for calibrating the estimate. K is tuned on a separate corpus not used for evaluation in this paper. The final ranking in RankBoost is a sum of weak learners with the corresponding weights. When the weights are too small (or too large), the posterior gets close to the extreme (either 0 or 1) regardless of the example. Hence, we normalize the RankBoost output dividing by the maximum possible rank score without changing the rank order:

$$\hat{P}(y \mid x) = \frac{1}{1 + exp(-y * \frac{H(x)}{\sum_{t=1}^{T} \alpha_t} + K)}$$

Note $max_x H(x) = max_x \sum_{t=1}^{T} \alpha_t h_t(x) = \sum_{t=1}^{T} \alpha_t$ since the weak learner $h_t(x)$ in RankBoost is a {0,1}-valued function defined on the ordering information provided by the corresponding feature [Freund *et al.*, 2003].

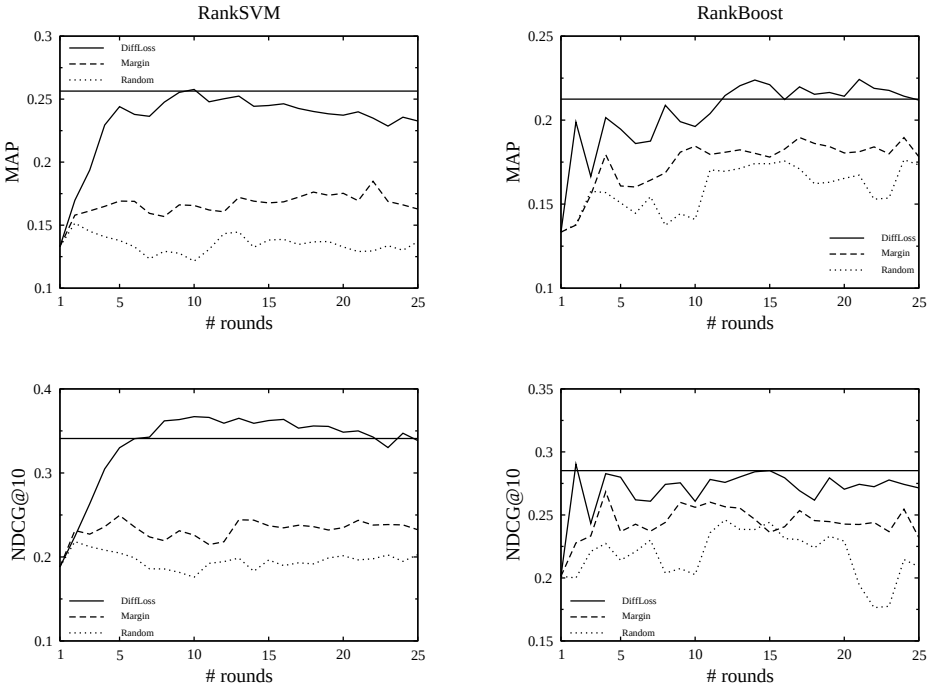


Figure 4.2: Comparison of different active learners on TD2003. The horizontal line indicates the performance when the entire training data is used. Only $\sim 15\%$ of the training data is actively labeled in total by each method.

4.3.7 Experimental Evaluation

We used the same datasets, namely TD2003 and TD2004, as we used for the experiments in Section 4.2. Figures 4.2 and 4.3 plot the performance of the proposed method (denoted by *DiffLoss*), and as comparative baselines, the margin-based sampling and random sampling strategies on TD2003 and TD2004 datasets. DiffLoss has a clear advantage over margin-based and random sampling in all cases with respect to different evaluation metrics. The differences over the entire operating range are also statistically significant ($p < 0.0001$) according to a two-sided paired t-test at 95% confidence level. DiffLoss especially achieves 30% relative improvement over the margin-based sampling for RankSVM on TD2003 dataset.

The horizontal line in each figure indicates the performance if all the training data was used, which we call the "optimal" performance. The performance of DiffLoss for RankBoost is comparable to the "optimal" on TD2003 and TD2004 datasets. In case of RankSVM, DiffLoss is close to the "optimal" on TREC 2003, and outperforms it on TREC 2004 dataset.

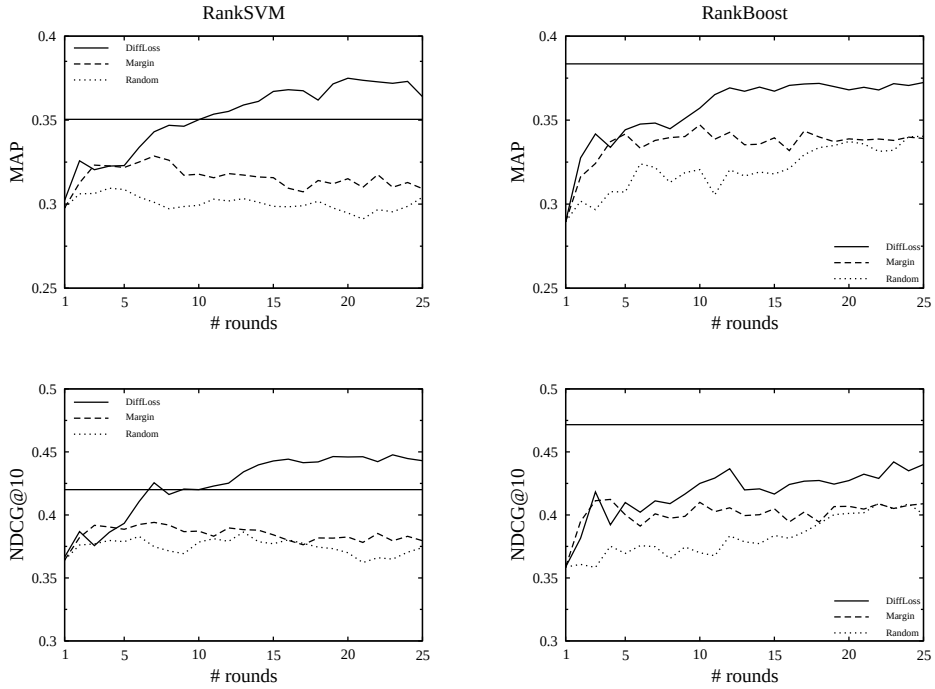


Figure 4.3: Comparison of different active learners on TD2004. The horizontal line indicates the performance when the entire training data is used. Only $\sim 15\%$ of the training data is actively labeled in total by each method.

More precisely, DiffLoss using RankSVM reaches the optimal performance (even surpassing it on TD2004) after 10 rounds of labeling on average (labeling 5 documents per query at each round). DiffLoss using RankBoost, on the other hand, reaches 95% and 90% of the optimal performance on MAP and NDCG@10, respectively on TD2004 dataset after 10 rounds. This suggests that carefully chosen samples might lead to a higher level of accuracy than blindly using large amounts of training data. This is an important development over traditional supervised rank learning since it not only reduces the expensive labeling effort, but also may lead to greater generalization power.

4.4 Chapter Conclusions

Rank learning is a more complicated task than classification since it requires learning a complete (or partial) ordering of data rather than finding absolute class assignments. The

target domain of a set of permutations is more complex than that of absolute classes. Hence, it is even more crucial to select the most informative instances to be labeled (or ordered) to learning a ranking model with fewer training examples. This chapter is devoted to two novel active sampling frameworks for rank learning. The main contribution of the first method, LossMin, lies in estimating the expected normalized loss minimization. The normalization yields a more balanced estimation, which is essential in highly skewed datasets, as typical in document retrieval. Experimental results offer that LossMin works well in practice, successfully learning a ranking function with many fewer labeling requests than the competitive baselines. There are open questions regarding potential future directions. While LossMin optimizes for AUC, it significantly improves two other popular IR evaluation metrics, namely MAP and NDCG@10. Studying the relationship between AUC and these measures might be useful for developing sampling strategies that directly optimize these measures.

Our second contribution considers the magnitude of the change in a ranking function resulting from an addition of a new labeled instance. Our framework efficiently estimates the risk of the ranking function after adding a new instance with all possible labels. The samples with the largest expected loss differential are selected to maximize the degree of fast learning. Empirical analysis shows that the proposed framework significantly reduces the required number of labeled examples to learn an accurate ranking function.

For both strategies, ranking problems with a complete order pose further challenges. Eliciting human judgments concerning complete orders is even more time-consuming and costly. Even when only a partial (relative) order of a set of points is acquired, deciding the set of points to be judged remains an important open research question.

Chapter 5

From Active to Proactive Learning

5.1 Introduction

We cannot emphasize enough how acquiring class labels or ranking preferences requires extensive effort while unlabeled data is often available in abundance. Thus far, we have discussed the active learning paradigm which attempts to optimize performance by selecting the most informative instances to label. Informativeness is defined in various ways (e.g. maximal expected improvement in prediction). We have described in detail novel machine learning approaches for active sampling in classification and rank learning problems.

However, traditional active learning relies on unrealistic assumptions, including our previous work. These assumptions have largely been ignored in the previous literature. Chapter 1 discusses these assumptions and explains the consequences at great length. We will not rehearse them here; instead, we will focus on the new machine learning paradigm, *proactive learning*. As discussed, proactive learning bridges the gap between traditional active learning and many practical problems. The main purpose of proactive learning is to reach out the appropriate predictor(s) with the appropriate query at the appropriate cost. This chapter takes a first step towards dealing with predictors having various characteristics including fallibility, reluctance and cost-variability. In the following chapters, we purely concentrate on fallible predictors and learning in the absence of ground truth labels.

In this chapter, we formulate the problem as inherently a decision-theoretic problem, and focus on three scenarios. These scenarios are designed to explore different predictor types in a multi-predictor setting, i.e. predictors reluctant to give answers, predictors that charge non-uniform cost, and fallible predictors that might provide wrong answers. We assume that each of these properties can be defined as a function of the query difficulty, i.e. the level of difficulty to classify the sampled instance. Each scenario analyzes a single property; i.e. reluctance, non-uniform cost and fallibility. In multi-predictor proactive

sampling, it is crucial to select the optimal data instance(s) to be queried as well as the optimal predictor. We achieved promising results on benchmark classification datasets by transforming the problem into expected utility maximization. We further assume a pre-defined and fixed budget; hence, the task becomes a constraint optimization problem. The results demonstrate the effectiveness of joint sampling of the optimal predictor-example pair as compared to sampling with respect to a single predictor.

5.2 Predictor and Instance Selection

In this section, we present a proactive learning method to select the optimal predictor instance pair for classification problems. Hence, the objective is to find the most informative instance to be labeled and the most appropriate and cost-effective predictor to elicit the label. We focus on three scenarios embodying the notion of multiple predictors with differing properties and costs. Let us begin by explaining "Scenario 1".

5.2.1 Scenario 1: Reluctance

In this scenario, we assume there exists one reliable predictor and one reluctant predictor. The reliable predictor gives an answer every time it is invoked with a query, and the answer is always correct. The reluctant predictor, on the other hand, does not always provide an answer, but when it answers it does so correctly. The probability of getting an answer from the reluctant predictor depends on the difficulty of the classification task. Not surprisingly, they charge different fees: the reliable predictor is more expensive than the reluctant one. We experimented with various cost combinations to simulate different real-world situations, with results in Section 5.3.

Rather than fixing the number of instances to sample, as in standard active learning, proactive learning fixes a maximum budget envelope since instances and predictors may have variable costs. Now, let us formulize the problem step by step as a joint optimization of which instance(s) to sample and which predictor to use to purchase their labels. The objective is to maximize the information gain under a pre-defined budget:

$$\text{maximize } E[V(S)] \text{ subject to } B$$

where B is the budget, S is the set of instances to be sampled, and $E[V(S)]$ is the expected value of information of the sampled data to the learning algorithm. $V(S)$ is a value function that can be replaced with any active selection criterion. For instance, it could be the estimated uncertainty of the current learning function at S, or a density weighted uncertainty score, or the estimated error on the unlabeled data if S is labeled and added to the training set.

The above equation can be rewritten by incorporating the budget constraint into the objective function:

$$\max_{S \subseteq Un} E[V(S)] - \zeta(\sum_k t_k * C_k) \quad \text{s.t. } \sum_k t_k * C_k \leq B \ , \ \sum_k t_k = |S|$$

where the subscript $k \in K$ denotes the chosen predictor from the set of predictors, K, and ζ is the parameter controlling the relative importance of maximizing the information and minimizing the cost. For simplicity, we assumed $\zeta = 1$ in this paper. C_k and t_k indicate the cost of the chosen predictor and the number of times it is invoked, respectively. Un is the set of unlabeled examples, $|S|$ is the total size of the sampled set[1]. Although this formulation is appealing, there is a major drawback. It is at best difficult to optimize directly due to the fact that the maximization is over the entire set of potential sampling sequences, an exponentially large number. However, the learning function is updated with each additional example, which affects which examples will be sampled in the future, though we can only calculate this effect after we know which examples are chosen and labeled. Thus, we cannot decide all the points to be sampled at once. A tractable alternative is a greedy approximation that will perform the optimal strategy at each round where only a single example or a small batch of examples is sampled. Now, let us see below how the greedy approach works:

$$(x^*, k^*) = \underset{x \in Un, k \in K}{\arg\max} \, (E_k[V(x)] - C_k) \tag{5.1}$$

$E_k[V(x)]$ is the expected value of information of the example x with respect to corresponding predictor k. For the remainder of this chapter, we adopted the density-sensitive sampling method described in Section 3.3. The only difference is that we adopt Euclidean distance instead of a density-sensitive distance metric and sample a single instance at a time instead of sampling in pairs.

$$U(x_i) = \log\Bigg\{ \min_{y_i \in \{\pm 1\}} \{P(y_i \mid \boldsymbol{x_i}, \hat{\boldsymbol{w}})\}\} + \sum_{k \neq i \in N_{x_i}} \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2^2) * \min_{y_k \in \{\pm 1\}} \{P(y_k \mid \boldsymbol{x_k}, \hat{\boldsymbol{w}})\} \Bigg\} \tag{5.2}$$

This results in faster computation while giving comparable performance; hence, we can focus more on the predictor selection.

We extend (5.1) by incorporating the probability of receiving an answer and obtain the following[2]:

$$(x^*, k^*) = \underset{x \in Un, k \in K}{\arg\max} \, (P(ans \mid x, k) * V(x) - C_k) \tag{5.3}$$

[1]The extension of this formulation to more than two predictors is straightforward.

[2]The expectation is equal to the actual value of information for the reliable predictor since $P(ans \mid x, reliable) = 1$ for all x.

Our goal in this scenario is to attain the maximum gain under the budget constraint. If both predictors were reliable, then the most cost-effective solution would be to use the cheapest predictor for every query. However, the cheapest predictor may not respond to every request, especially when the query is difficult. We define a utility score, $U(x, k)$, which is a function of the predictor k and the data point x:

$$U(x, k) = P(ans \mid x, k) * V(x) - C_k \tag{5.4}$$

When the utility is defined as above, it is often necessary to normalize the scores and the costs into the same range. In order to avoid the normalization, we re-define the utility of an example given the predictor as the information value of that example at unit cost:

$$U(x, k) = \frac{P(ans \mid x, k) * V(x)}{C_k} \text{ where } k \in K \tag{5.5}$$

The difference between (5.4) and (5.5) is that the latter always yields non-negative utilities whereas the former allows negative values. A similar utility function for label acquisition at unit cost is also adopted by [Melville *et al.*, 2005].

Unfortunately, there do not exist real-world datasets that have ground truth information on the reliability (in this case, $P(ans \mid x, k)$) of the labeling source (e.g. predictor, annotator). Therefore, we simulate the reliability as follows. We assume the amount of labeled training data available to a predictor determines its knowledge (expertise). For instance, the reliable (perfect) predictor resembles a system that has been trained on the entire dataset so it has perfect knowledge on each and every data point. Unlike the perfect predictor, a reluctant predictor has access only to a small portion of the data; therefore, it is not knowledgeable for every point. Whenever it encounters an ambiguous data point to classify, it becomes reluctant to provide an answer. We train a classifier on a small random subset of the entire data to obtain a posterior class distribution $P(y \mid x)$. For its simplicity and probabilistic nature, we adopted logistic regression in our experiments to calculate the class posterior. The class posterior is then used for measuring uncertainty, $\min_{y \in \mathcal{Y}} P(y \mid x)$, where $\mathcal{Y}$ is the set of target labels. We assume that the chance of obtaining an answer from the reluctant predictor is low when the uncertainty is high and vice versa. We explain how we design the reluctance in Section 5.3.1 in more detail.

In order to calculate the utility as shown in Equation 5.5, we need to know the answer probability of the reluctant predictor. However, it is unrealistic to be given each predictor's knowledge level and response characteristics apriori, so we estimate these properties in a discovery phase. First, we cluster the unlabeled data using kmeans clustering [Hartigan and Wong, 1979]. The number of clusters depends on the pre-defined budget available for this phase and the cost of the reluctant predictor. Second, for each cluster, we inquire the label of the data point closest to the centroid. The number of successful inquiries (i.e. the number of data points that we obtain the labels of) varies depending on the reluctance

of the predictor [3]. We hypothesize that if the predictor does not provide the label of a data point then it is unlikely to provide the labels for the nearby points since we assume that similar points share similar posterior class probabilities. Therefore, it is reasonable to estimate the answer probability of the reluctant predictor by inquiring the labels of the cluster centroids.

For each cluster, if we obtain the label of the centroid, then we increase the answer probability of the points in this cluster. Similarly, we decrease the answer probability of the points in the clusters whose centroids we did not obtain the labels of. This step can be regarded as a belief propagation step. If we receive the label of a centroid, then we propagate our belief in receiving a label to similar points and vice versa. Initially, we assume the answer probability for each unlabeled point is 0.5, which indicates a random guess. Then, we adopt the following update to estimate the answer probability of each point so that it changes as a function of the proximity of the point to the cluster centroid and predictor responsiveness:

$$\hat{P}(ans \mid x, reluctant) = \frac{0.5}{Z} * exp\left(\frac{h(x_{c_t}, y_{c_t})}{2} \ln \frac{max_d - \|x_{c_t} - x\|}{\|x_{c_t} - x\|}\right) \forall x \in Cl_t \qquad (5.6)$$

where Z is a normalization constant. x_{c_t} is the centroid of the cluster Cl_t that includes x. $h(x_c, y_c) \in \{1, -1\}$ is an indicator function which is equal to 1 when we receive the label y_c for the centroid x_c, and -1 otherwise. $\|x_c - x\|$ is the Euclidean distance between the cluster centroid x_c and the point x, and $max_d := \max_{x_{\acute{c}}, x} \|x_{\acute{c}} - x\|$ is the maximum distance between any cluster centroid and data point.

We substitute the estimated answer probability into the utility function, i.e. $\hat{U}(x, k) = \frac{\hat{P}(ans|x,k) * V(x)}{C_k}$. The joint sampling of the predictor-example pair can now be performed as shown in Algorithm 6.

The algorithm works in rounds till the budget is exhausted. Each round corresponds to a single label acquisition attempt where sampling persists until obtaining a label. One important point to note here is that we need to restrain from spending too much on a single attempt by adaptively penalizing the reluctant predictor every time it refuses to answer. At any given round, if the algorithm chooses the reluctant predictor and does not receive an answer, the utility of remaining examples with respect to this predictor decreases by the amount spent thus far at this round:

$$\hat{U}(x, reluctant) = \frac{\hat{P}(ans \mid x, reluctant) * V(x)}{C_{round}}$$

where C_{round} is the amount spent thus far in the given round. This penalization only applies to the reluctant predictor since the reliable predictor always provides the label.

[3] We experimented with varying reluctance levels for a thorough investigation.

Algorithm 6 Scenario 1: Reluctance

Input: a classifier f, labeled data Lb, unlabeled data Un, entire budget B, clustering budget $B_C < B$, two predictors, each with a cost C_k, $k \in K = \{reliable, reluctant\}$
Output: f
- Cluster Un into $p = B_C/C_{reluctant}$ clusters
- Let x_{c_t} be the data point closest to its cluster centroid, $\forall t = 1, ..., p$
- Query the label y_{c_t} for each cluster centroid x_{c_t}
- Identify $\{x_{c_1}, ..., x_{c_g}\}$ for which we obtain the labels
- Estimate $\hat{P}(ans \mid x, reluctant)$ via Equation 5.6
- Update $Lb = Lb \cup \{x_{c_t}, y_{c_t}\}_{t=1}^{g}$, $Un = Un \setminus \{x_{c_t}, y_{c_t}\}_{t=1}^{g}$
- cost spent so far $C_T = B_C$
while $C_T < B$ **do**
  - Train f on Lb
  - Initialize the cost of this round $C_{round} = 0$ and the set of queried examples $Q = \{\}$
  - $\forall k \in K, x \in Un$ estimate utility $\hat{U}(x, k)$
  repeat
    1. Choose $k^* = \arg\max_{k \in K} \max_{x \in Un \setminus Q} \{\hat{U}(x, k)\}$
    2. Choose $x^* = \arg\max_{x \in Un \setminus Q} \{\hat{U}(x, k^*)\}$
    3. Update $C_{round} = C_{round} + C_{k^*}$
    4. $Q = Q \cup \{x^*\}$
    5. Query the label y^* with probability $P(ans \mid x^*, k^*)$
  until label y^* is obtained
  - Update $C_T = C_T + C_{round}$
  - Update $Lb = Lb \cup (x^*, y^*)$ and $Un = Un \setminus (x^*, y^*)$
end while

Algorithm 6 selects the maximum utility examples. This framework leads to an incrementally optimal solution in the sense that the most useful data is sampled at the minimum cost.

5.2.2 Scenario 2: Fallibility

In real-world, there might also be fallible predictors which answer each query, but the credibility of the answer is questionable. We simulate this setting by two predictors; one reliable and one unreliable predictor. The reliable predictor is the perfect predictor that always provides the correct answer to any query. The unreliable predictor in this scenario is fallible that it may provide the wrong label for a given example, depending on the difficulty. Specifically, if an example approaches the decision boundary, the probability of correct classification approaches 0.5 (random guess). The probability of acquiring a correct label,

Algorithm 7 Scenario 2: Fallibility

Input: a classifier f, labeled data Lb, unlabeled data Un, entire budget B, clustering budget $B_C < B$, two predictors, each with a cost C_k, $k \in K = \{reliable, fallible\}$
Output: f
- Cluster Un into $p = B_C/C_{fallible}$ clusters
- Let x_{c_t} be the data point closest to its cluster centroid, $\forall t = 1, ..., p$
- Query the label y_{c_t} for each cluster centroid x_{c_t}
- Identify $\{x_{c_1}, ..., x_{c_h}\}$ for which the fallible predictor has high confidence
- Estimate $\hat{P}(correct \mid x, fallible)$
- Update $Lb = Lb \cup \{x_{c_t}, y_{c_t}\}_{t=1}^{h}$, $Un = Un \setminus \{x_{c_t}, y_{c_t}\}_{t=1}^{h}$
- cost spent so far $C_T = B_C$

while $C_T < B$ **do**
1. Train f on Lb
2. $\forall k \in K, x \in Un \quad \hat{U}(x,k) = \frac{\hat{P}(correct|x,k)*V(x)}{C_k}$
3. Choose $k^* = \arg\max_{k \in K} \max_{x \in Un} \{\hat{U}(x,k)\}$
4. Choose $x^* = \arg\max_{x \in Un} \{\hat{U}(x,k^*)\}$
5. Update $C_T = C_T + C_{k^*}$
6. Update $Lb = Lb \cup (x^*, y^*)$ and $Un = Un \setminus (x^*, y^*)$ where y^* is the correct label with probability $P(correct \mid x^*, k^*)$

end while

$P(correct \mid x, fallible)$ is modeled the same way as in "Scenario 1". The solution we propose is similar to the method introduced for "Scenario 1", with slight variations. For instance, the learning method receives a random label for the queried example x with probability $1 - P(correct \mid x, fallible)$. Moreover, we use the clustering step exploiting the fallible predictor to estimate the correctness probability $P(correct \mid x, fallible)$. Similar to the previous scenario, we inquire the labels of the cluster centroids. Unlike the reluctant predictor, the fallible predictor provides the label together with its confidence. The confidence is its posterior class probability for the provided label, $P(y \mid x)$. If the class posterior is within an uncertainty range, then we decide not to use the provided label since it is likely to be noisy (See Section 5.3.1 for details). We decrease the correctness probability for the points in the cluster whose centroid has a class posterior in the uncertainty range. We increase the correctness probability for the points in the clusters with highly confident centroids; i.e. $\hat{P}(correct \mid x, fallible) = \frac{0.5}{Z} * exp\left(\frac{\tilde{h}(x_{c_t}, y_{c_t})}{2} \ln \frac{max_d - \|x_{c_t} - x\|}{\|x_{c_t} - x\|}\right) \forall x \in Cl_t$ where $\tilde{h}(x_{c_t}, y_{c_t}) = -1$ if $P(y \mid x_{c_t})$ is in the uncertainty range, and 1 otherwise. The pseudocode of the algorithm is given in Algorithm 7.

5.2.3 Scenario 3: Non-uniform Cost

Thus far, we have only considered the settings where a uniform fee is charged for every query by a predictor, although each predictor may charge differently. Fraud detection in banking transactions is a good example for this setting. The customer records are saved in the bank database so it takes the same amount of time and effort, hence the same cost, to look up any entry in the database. On the contrary, it is possible that the costs are distributed non-uniformly over the set of instances. For instance in text categorization, it might be relatively easy for an annotator to categorize a web page; hence the cost is modest. On the other hand, assigning a book into a category incurs a considerable reading time and therefore cost. Another example of a non-uniform cost scenario is medical diagnosis. Some diseases such as herpes are easy to diagnose. Such diagnoses are not costly since there is usually a major definitive symptom, i.e. outbreak of blisters on the skin. On the other hand, diagnosing hepatitis can be very costly since it may require blood and urine tests, CT scans, or even a liver biopsy. In "Scenario 3", we explore the problem of deciding which instances to query for the labels when label acquisition cost varies with the instance. We assume two predictors one of which has a uniform and fixed cost for each query whereas the other charges according to the task difficulty. We further assume that these predictors always provide an answer and both are perfectly reliable in their answers.

In order to simulate the variable-cost (non-uniform) predictor, we model the cost of each example x as a function of the posterior class distribution $P(y \mid x)$. We use the class posterior calculated similarly in the previous scenarios. The non-uniform cost $C_{non-unif}(x)$ per instance is then defined as follows:

$$C_{non-unif}(x) = 1 - \frac{\max_{y \in \mathcal{Y}} P(y \mid x) - 1/|\mathcal{Y}|}{1 - 1/|\mathcal{Y}|}$$

The cost increases as the instance approaches the decision boundary and vice versa. In other words, the predictor charges based on how valuable the instance is to the learner. This may not be the case in the real world, but this sets up a more challenging decision in terms of the utility-cost trade-off. The utility score in this scenario is calculated as the difference between the information value and the cost instead of the information value per unit cost[4]. This is to avoid infinitely large utility scores as a result of the division by small ϵ-cost. Thus, the revised utility score per predictor is given as follows:

$$\begin{aligned} U(x, unif) &= V(x) - C_{unif} \\ U(x, non-unif) &= V(x) - C_{non-unif}(x) \end{aligned} \tag{5.7}$$

where C_{unif} is the fixed cost of the uniform-cost predictor. The pseudocode of the algorithm is given in Algorithm 8. Note that there is no clustering phase in Algorithm 8 since

[4]In general, if the cost and information value are not assessed in the same units, then they can be normalized into the same range.

we assume we know the cost of every instance, which is realistic for many real-world applications.

Algorithm 8 Scenario 3: Non-uniform Cost

Input: a classifier f, labeled data Lb, unlabeled data Un, entire budget B, two predictors, each with a cost C_k, $k \in K = \{unif, non-unif\}$
Output: f
cost spent so far $C_T = 0$
while $C_T < B$ **do**
1. Train f on Lb
2. $\forall k \in K, x \in Un$ calculate $U(x,k)$ via Equation 5.7.
3. Choose $k^* = \arg\max_{k \in K} \max_{x \in Un}\{U(x,k)\}$
4. Choose $x^* = \arg\max_{x \in Un}\{U(x,k^*)\}$
5. Update $C_T = C_T + C_{k^*}$
6. Update $Lb = Lb \cup (x^*, y^*)$, $Un = Un \setminus (x^*, y^*)$

end while

5.3 Experimental Evaluation

5.3.1 Setup for All Three Scenarios

In order to simulate the reliability of the labeling source (predictor), we assume that a perfectly reliable predictor resembles by a classifier trained on the entire data. An unreliable predictor, then, resembles a classifier trained on only a small subset of the entire data. Hence, we randomly sampled a small subset from each dataset and trained a logistic regression classifier on this sample to output a posterior class distribution that represents an unreliable predictor. Then, we identified the instances whose class posterior falls into the uncertainty range, i.e. $\min_y P(y \mid x) \in [0.45, 0.5]$, assuming $y \in \{0, 1\}$. This range is used to filter the instances that the reluctant predictor does not answer or the fallible predictor outputs a random label. One can argue that the same effect can be achieved by randomly picking such instances. However, our simulation forces a trade-off between the reliability and the information value of an instance since uncertain instances are generally informative for active learners. In order to cover a wider spectrum, we varied the percentage of instances that fall into the uncertainty range $[45\%, 5\%]$. The second column in Table 5.1 shows the different percentages used in our experiments. The cost of the unreliable predictor is inversely proportional to its reliability. We choose higher cost ratios for the fallibility scenario since receiving a noisy label should be penalized more than receiving no label at all. The tradeoff between cost and unreliability is crucial to have an incentive to choose between predictors rather than exploiting a single one. See Table 5.1 for details.

Table 5.1: Predictor properties and costs. B_C is the clustering budget, B is the entire budget. Uncertain % is the percentage of the uncertain data points. Cost Ratio is the ratio of the cost of the unreliable predictor to the cost of the reliable one.

Scenario	Uncertain %	Cost Ratio	B_C	B
Scenario 1	45-55%	1:3	20	300
	55-60%	1:4	30	
	65-70%	1:5	50	
Scenario 2	45-55%	1:5	20	300
	55-60%	1:6	30	
	65-70%	1:7	50	

The other case we need to simulate is the uniform and non-uniform cost predictors. The cost of each instance for the variable-cost predictor is defined as a function of the class posterior, i.e. $C_{non-unif}(x) = 1 - \frac{\max_{y \in \mathcal{Y}} P(y|x) - 1/|\mathcal{Y}|}{1 - 1/|\mathcal{Y}|}$. This indicates a positive relationship between the difficulty of classifying an instance with its cost, which is realistic for many real-world situations. The cost of labeling each instance is known to the learning algorithm. Thus, we do not need any clustering phase in Scenario 3. We choose the cost of the uniform-cost predictor within the range of instance costs for the variable-cost predictor. Hence, the costs will be comparable in the same range. We varied the fixed cost such that there is always an incentive to choose between predictors instead of fully exploiting a single one.

We compared our method against sampling with randomly chosen predictors and sampling with a single predictor. Each baseline samples the cluster centroids initially for a fair comparative analysis. However, only our method estimates the predictor unreliability to help sampling the optimal predictor-example pair.

All the results reported are averaged over 10 runs. At each run, we start with one randomly chosen labeled example from each class. The rest of the data is considered unlabeled. The learner selects one example at each iteration to be labeled, and the learning function is tested on the remaining unlabeled set once the label is obtained. The learner pays the cost of each queried example regardless of whether a label is obtained. To show the effectiveness of each method, the learning curves display the classification error versus the data elicitation cost. The budget is fixed at 300 in Scenario 1 and 2, and at 20 in Scenario 3. A small budget is enough for the latter since the cost of individual instances can be very small depending on the posterior probability. We have observed that 20 is more than enough to reach a desirable accuracy in this scenario. The clustering budget, on the other hand, varies according to the unreliability, but is the same for each baseline under the same scenario (See Table 5.1). The number of clusters, though, is determined by dividing the clustering budget by the cost of the predictor used during this phase. The unreliable predictor is used in our method and the unreliable-predictor baselines for the

Table 5.2: Overview of Datasets. +/- is the positive/negative ratio. Dim is the dimensionality.

Data	Face	Spambase	Adult	VY-letter
Size	2500	4601	4147	1550
+/-	1	0.65	0.33	0.97
Dim	400	57	48	16

initial clustering phase. Thus, they obtain the same labeled data during this step, which results in the same error rate. The random predictor baseline uses a fixed number of clusters, but for each cluster centroid it randomly chooses the predictor to invoke and continues until the clustering sub-budget exhausts.

5.3.2 Datasets

We study the performance of the proposed methods on various real-world benchmark datasets. The face detection dataset [Pham *et al.*, 2002] has a total number of 393360 images, which we used a random subsample of size 2500 as in Section 3.3. UCI-Letter is another image dataset for recognizing English capital letters where we labeled the letter V as the positive class and the letter Y as the negative class. This is one of the most ambiguous pairs in the data. The Spambase and the Adult datasets are also popular datasets available from the UCI Machine Learning Repository [Newman *et al.*, 1998]. The Spambase data contains 4601 instances and 57 condition attributes. It is used to classify emails as spam and non-spam. Most of the attributes indicate whether a certain word or character appears frequently in emails. For the Adult dataset, we adopted the smaller version constructed for the IJCNN 2007 Workshop on Agnostic Learning [agn, 2007]. This version has 48 features and 4147 instances in total. The task of Adult data is to discover high revenue people from the census bureau. A summary of datasets is provided in Table 5.2.

5.3.3 Results

We conducted a thorough analysis to examine the performance of our method under various conditions. Due to the lack of existing work that is directly comparable, we compared our method against active sampling with randomly chosen predictors and active sampling with a single predictor. We denote our method of jointly optimizing predictor and instance selection *Joint*, the random sampling of predictors *Random*. *Reliable*, *Reluctant*, and *Fallible* refer to the corresponding single predictor baseline.

Before discussing the results, we first clarify why the maximum cost of data elicitation, shown in Figures 5.1-5.6, differ in various tasks and scenarios. The results are averaged

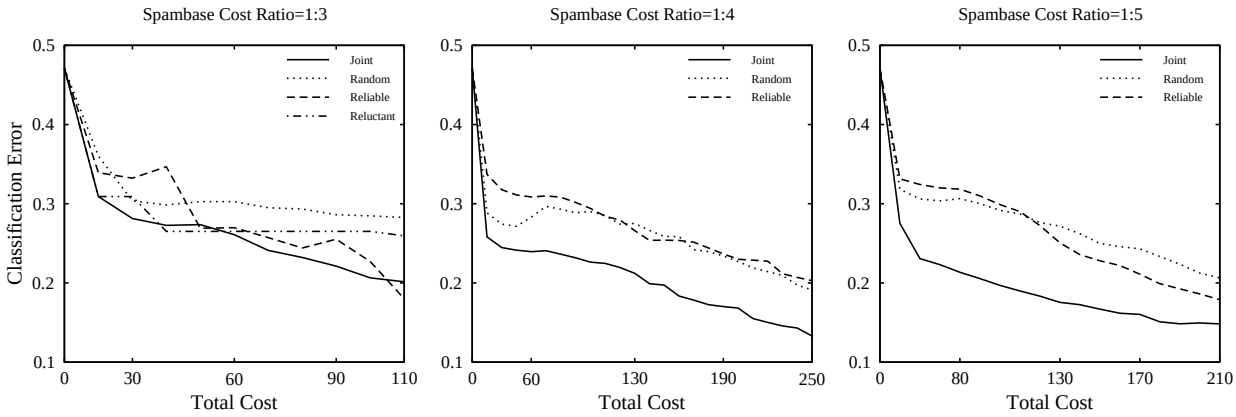


Figure 5.1: Performance Comparison for Scenario 1 (Reluctance) on the Spambase dataset. The cost ratio is indicated above each plot.

over 10 runs for each experiment. At each run, the total number of iterations to spend the entire budget may differ depending on how the budget is allocated between predictors. In order to take the average of the results, we rely on the minimum number of iterations attained over 10 runs for each experiment. This ensures that all runs equally contribute to the average. This also results in different maximum elicitation costs smaller than the budget for different experiments. Nevertheless, the *Joint* strategy outperforms the others even after spending only a small amount in most cases.

Figure 5.1 shows the results for the reluctance scenario on the Spambase dataset. Each plot indicates a different cost ratio. Our method outperforms the others on every case while the performance gap increases with the cost ratio. The cost ratio denotes the relative price of each predictor against the other, i.e. the ratio of the cost of the unreliable predictor to the cost of the reliable one. This is largely because the predictor differences leave more room for improvement via predictor selection in the latter case. When the unreliability gets higher, the reluctant predictor tends to spend almost the entire budget on a single label acquisition attempt. This leads to acquiring only a small amount of labeled data; hence, its poor performance. As a result, we do not report the reluctant predictor baseline except in its best case, the $1:3$ cost ratio.

Figures 5.2 and 5.3 show the comparison between *Joint* and the other baselines for Scenario 1 on the Adult and VY-Letter datasets, respectively. For the Adult dataset, *Joint* outperforms the others when the cost ratio is $1:3$ while it tracks the best performer for the other cost ratios. Generally, *Joint* tracks the best performer when the best performer is a clear winner for the entire operating range. This pattern is also evident in Figure 5.3 for the cost ratio $1:3$. For the other cost ratios, *Joint* significantly outperforms the other baselines

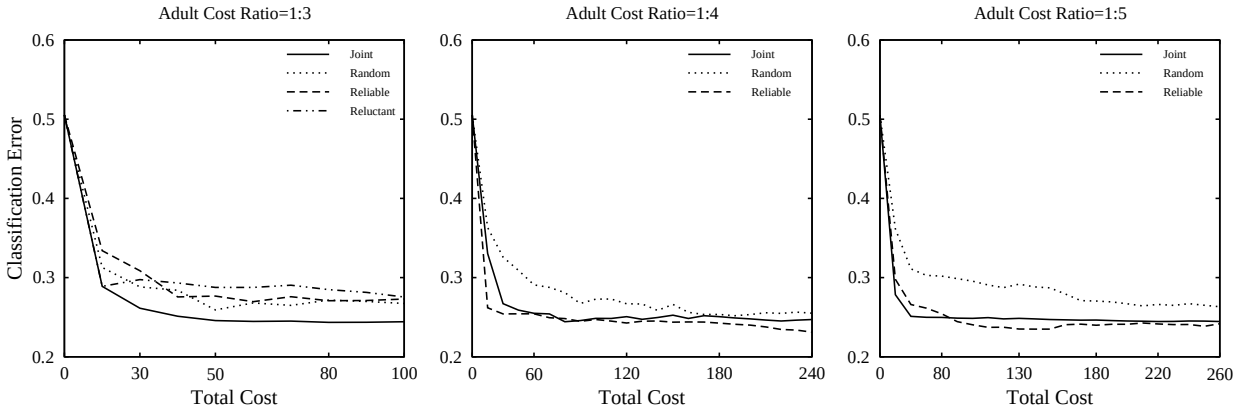


Figure 5.2: Performance Comparison for Scenario 1 (Reluctance) on the Adult dataset. The cost ratio is indicated above each plot.

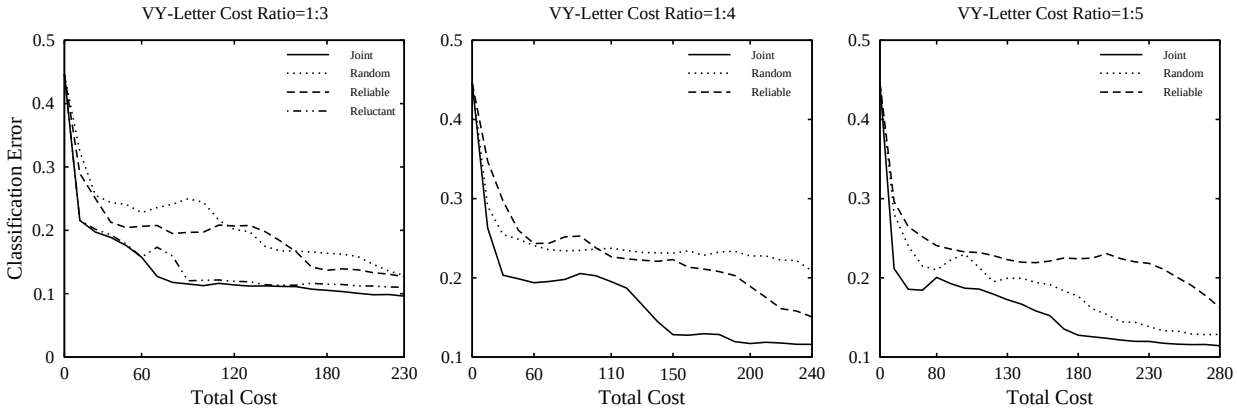


Figure 5.3: Performance Comparison for Scenario 1 (Reluctance) on the VY-Letter dataset. The cost ratio is indicated above each plot.

on the VY-Letter dataset.

Figure 5.4 compares the performances for Scenario 2 on the VY-Letter dataset. The Fallible predictor in this scenario performs poorly when the relative cost ratio is high. As shown in Table 5.1, the cost ratio increases with the number of unreliable instances. In other words, a higher cost ratio indicates a more unreliable predictor. Thus, the Fallible predictor may increase the classification error with more labeled data since the labels are increasingly likely to be noisy. This pattern is especially evident in Figure 5.4 for the cost

Table 5.3: Results on different datasets for two scenarios. Cost column shows the total cost spent to reach the corresponding error rate. The best result on each row is given in bold.

			Error Rate			
Scenario	Dataset & Cost Ratio	Cost	Joint	Random	Reliable	Unreliable
Scenario 1	Face & 1:4	60	0.195	0.294	0.347	**0.188**
		120	**0.179**	0.275	0.261	0.192
		180	**0.144**	0.201	0.163	0.178
		240	0.119	0.137	**0.118**	0.168
Scenario 2	Face & 1:5	70	**0.250**	0.294	0.468	0.343
		130	**0.233**	0.298	0.298	0.271
		190	**0.165**	0.330	0.193	0.250
		250	**0.152**	0.215	0.153	0.233
	Spambase & 1:7	70	0.285	0.335	**0.264**	0.369
		120	**0.243**	0.328	0.289	0.373
		170	**0.185**	0.311	0.279	0.357
		220	**0.151**	0.281	0.262	0.337
	Adult & 1:6	70	0.334	0.386	**0.302**	0.363
		130	0.309	0.358	**0.295**	0.362
		190	0.288	0.300	**0.284**	0.350
		250	**0.269**	0.278	0.281	0.342

ratio $1:7$. On the other hand, *Joint* strategy is quite effective for reducing the error in this scenario, indicating that it is capable of reducing the risk of introducing noisy data through strategic selection between predictors.

We present the rest of the results in Table 5.3. We selected a representative cost ratio for each dataset. The values in bold correspond to the winning methods. *Joint* wins frequently (i.e. 10 out of 16) and is a close runner-up for the cases where it does not achieve the best result.

Figure 5.6 presents the evaluation results when the cost varies non-uniformly across the set of instances. We experimented with different assignments of the fixed cost, each of which is a function of the average instance cost, denoted avg, for the non-uniform cost predictor. We present two representative assignments for each dataset: Cost1:$= avg/1.5$ and Cost2:$= avg/2$. The remaining cost values are not included since they are similar to those reported here. On the Face and the Spambase datasets, *Joint* is the best performer throughout the full operating range. Moreover, *Joint* predominantly outperforms the others on the VY-letter dataset. The performance difference between *Joint* and each baseline is also statistically significant based on a paired two-sided t-test ($p < 0.01$). For the Adult dataset,

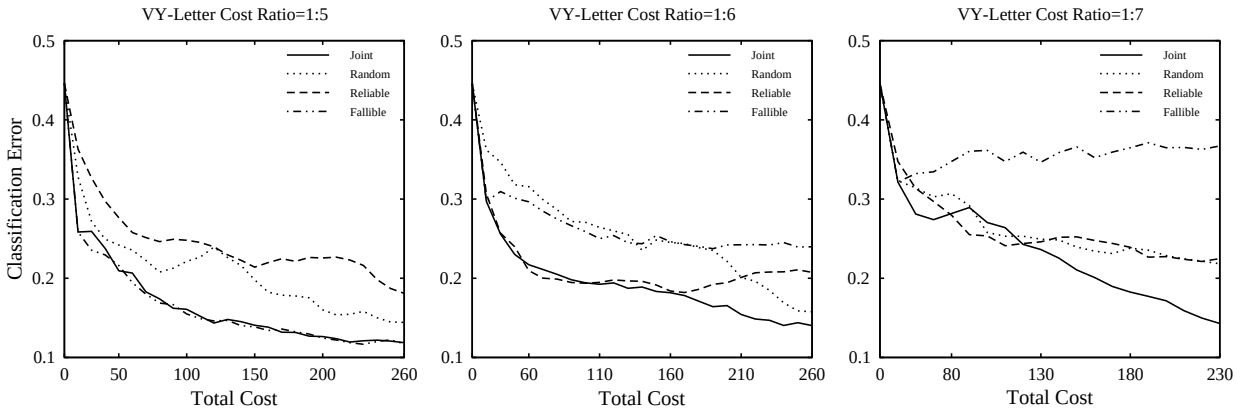


Figure 5.4: Performance Comparison for Scenario 2 (Fallibility) on the VY-Letter dataset. The cost ratio is indicated above each plot.

both cost cases performed equivalently; there was no opportunity for "Joint" to optimize further and thus was not reported.

In order to investigate if the initial clustering phase helps all the baselines, we re-ran each baseline excluding the clustering step. In this case, there is no separate clustering budget; hence, the entire budget is spent in rounds for data elicitation. Figure 5.5 compares each baseline with the clustering restriction on the Spambase dataset for Scenario 1. Every baseline significantly benefits from clustering, with the biggest boost in improvement occurring for the Reluctant predictor. Hence, both the baselines and the "Joint" strategy benefit from the diversity-based sampling via clustering in their initial steps. Without pre-clustering, the Reluctant predictor is prone to spend too much on a single elicitation attempt due to unsuccessful labeling requests. It can, however, maximize the chance of receiving a label through diversity sampling during the clustering step instead of getting stuck in one round for a single label.

5.4 Chapter Conclusions

This chapter focused on proactive learning to overcome the unrealistic assumptions of active learning. We introduced three scenarios that analyze the effect of multiple imperfect predictors with differing properties and costs on selective sampling. The proposed methods formulated in a decision-theoretic framework rely on expected utility maximization across predictor-instance pairs. The empirical results demonstrate the effectiveness of this approach against random predictor selection and exploitation of a single predictor, even the best one.

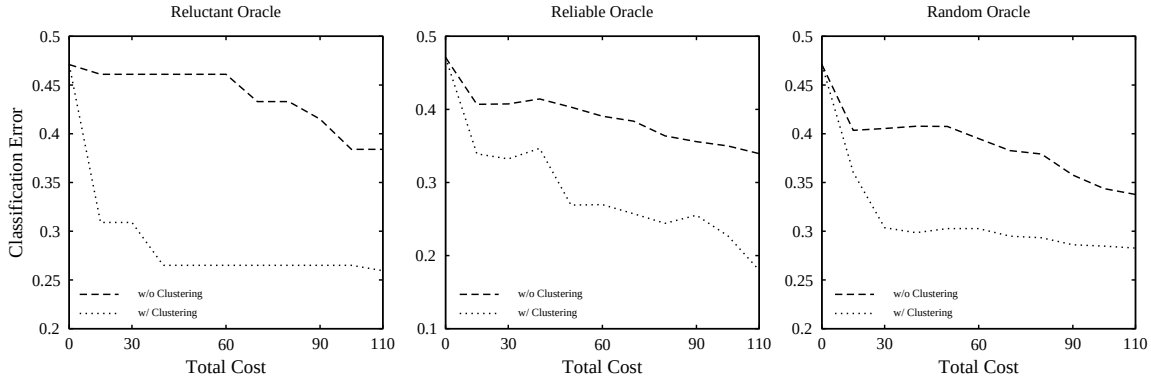


Figure 5.5: Change in performance of each baseline with and without clustering on Spambase. The type of baseline is given in the title. The cost ratio is 1:3.

In the chapters that follow, we focus our attention on learning with multiple imperfect predictors which make labeling mistakes. The challenge is that we assume the absence of both gold standard labels and a perfectly reliable predictor against which to compare the outputs of the noisy ones. In particular, we consider predictor accuracy estimation and predictor selection with no *apriori* information. We address these problems in two cases one of which deals with stationary predictor accuracies whereas the other explicitly models the time-varying accuracies. Finally, the last chapter formulates the predictor risk estimation as an optimization problem with theoretical guarantees and propose a very effective unsupervised way to train classifiers without any labeled data whatsoever.

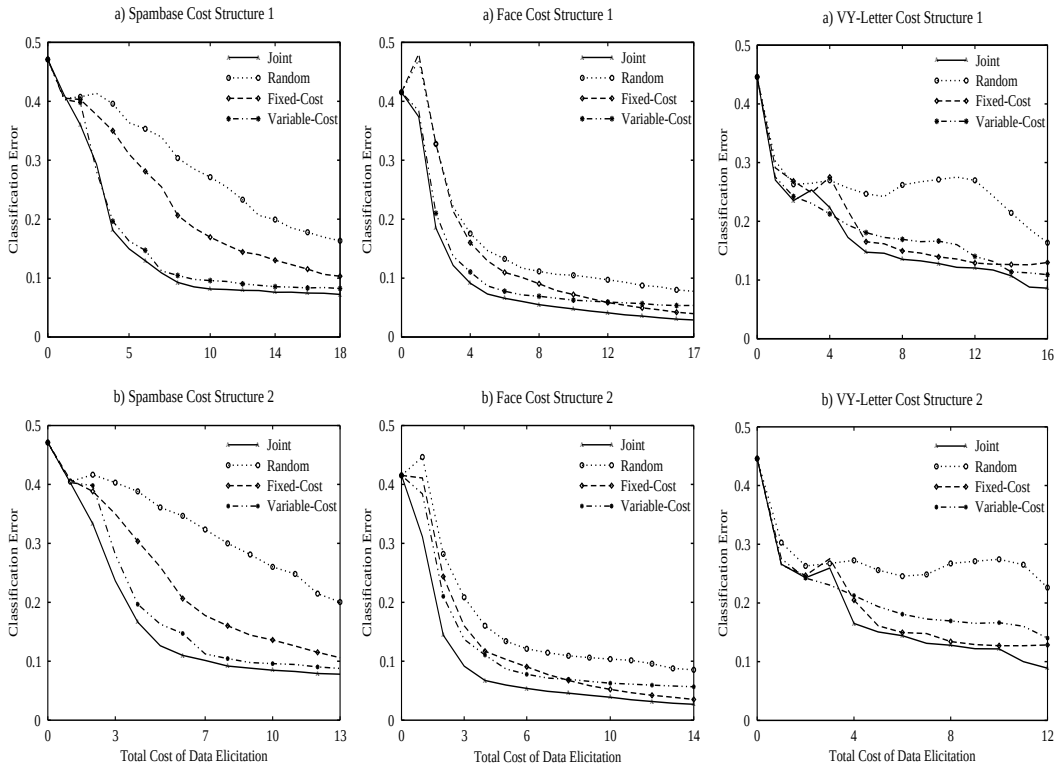


Figure 5.6: Comparison of different algorithms under non-uniform cost structures (Scenario 3) on Spambase, Face and V-Y Letter datasets, respectively. a) (Top panel) Fixed-Cost predictor has Cost1 b) (Bottom Panel) Fixed-Cost predictor has Cost2.

Chapter 6

Joint Predictor Accuracy Estimation and Predictor Selection

6.1 Introduction

Chapter 5 takes a first step introducing predictors with differing properties. It analyzes reluctance, fallibity and variable cost in predictors via simple scenarios including one perfectly reliable and one unreliable predictor. However, these scenarios are somewhat restrictive since it may not always be possible to have access to a perfect predictor. Instead, we may be given multiple ($k > 2$) predictors without any apriori information about their accuracies or the ground truth labels. Then, it becomes essential to make inference about the predictors when there is no ground truth to compare their outputs. From this chapter on, we focus on a more specific challenge which deals with learning with multiple noisy (fallible) predictors in the absence of gold standard labels. In many machine learning applications, obtaining labels for the training data might introduce noise. Supervised learning algorithms aim at maximizing the classification accuracy based on a set of training instances. The maximum accuracy achieved depends strongly on the quality of the labels. Working with multiple predictors (labelers), the quality of annotations is questionable. The question is, then, whether we can estimate the labeling accuracy of each predictor explicitly with unlabeled data and use these estimates to select the predictor(s) with the highest accuracy. This chapter addresses exactly such a challenge which we study in two different conditions. The first deals with the predictors whose accuracies are stationary with respect to time. The second case considers the situation when the predictor accuracies vary with time. For instance, a doctor may not be cognizant of a new disease and thus mislabel, or a scientist can learn from lab experience and thus improve. Fatigue and/or practice effects can also cause time variance in predictor's labeling accuracy. If the predictors are computer systems continuously re-trained with a constant stream of data, then one might expect changes in

the predictors' accuracy with new data. When the predictors are evaluated on the test data drawn from the same distribution as the training data, predictors' generalization power is most likely to improve over time. However, when the training and test data are drawn from different distributions the predictors' test accuracy might fluctuate.

In both stationary and non-stationary cases, predictor accuracy estimation requires a degree of exploration as well as exploitation. In the former case, the goal is to acquire predictor and label knowledge through repeated trials, balancing the exploration vs. exploitation tradeoff, by first favoring the former and moving gradually to increasing exploitation. In the first section, we report the first work with such properties. We adopt the Interval Estimation (IE) learning [Kaelbling, 1990; Moore and Schneider, 1995] as a building block for our framework. IE attempts to estimate the confidence interval on the expected response of an action and then selects the action with the highest upper confidence interval. In our problem, taking an action corresponds to selecting a predictor to query for labeling. We use the responses of the predictors to evaluate the performance of each predictor. Therefore, inclusion of inferior predictors can dramatically slow convergence. To overcome this issue, we propose a thesholding mechanism (IEThresh) - to filter out inferior predictors early in the process. This helps to narrow down the set of potentially good predictors and improves the estimation accuracy with many fewer exploratory trials.

To solve the problem in the non-stationary case, we present a novel algorithm based on sequential Bayesian estimation in Section 6.3. The sequential Bayesian estimation framework continuously and selectively tracks the accuracy of each predictor and selects the top quality one(s) at each time step. We assume each time step corresponds to a single example to be labeled. This framework also allows aggregating the observed labels to predict the true label for the given example. The experimental analysis shows the effectiveness of our approach for 1) improving the quality (reducing noise) of the labeled data, 2) tracking the accuracy drift of multiple predictors in the absence of ground truth, and 3) consistently identifying the low-quality predictors and reducing their effect in learning. We assume the predictor accuracy changes gradually over time without abrupt large shifts. However, we analyze the increased rate of change and report its effect on the performance. We conclude with high confidence that our method is relatively robust to higher variations, as supported by the empirical evaluation.

6.2 A Multi-armed Bandit Approach in Stationary Conditions

6.2.1 Motivation

Interval Estimation learning, like many other multi-armed bandit problems, requires an appropriate reward function. The reward of each predictor is directly related to whether the predictor makes a labeling mistake or not. Unfortunately, an exact calculation of the

reward function is impossible since the true label is unknown. A natural way to estimate the true label is to take the majority vote among the predicted labels from multiple predictors. Throughout the chapter, we assume an individual predictor accuracy is better than random guess, i.e. > 0.5 in the binary case, and the predictors' outputs are conditionally independent given the true label. Under this assumption, it is unlikely that all predictors make a labeling mistake at the same time; hence, the majority label is a close approximation to the true label. The method is robust to occasional errors by the majority vote method as demonstrated on several benchmark datasets. We add random noise to real-life datasets to simulate a set of predictors in addition to two datasets annotated with real labelers [Snow *et al.*, 2008]. For the simulated-error cases, we varied the predictor accuracies to show that our method IEThresh can detect the most accurate ones even among a uniform or skewed mix of good and bad predictors.

6.2.2 Interval Estimation Learning

Our multi-predictor active sampling method, IEThresh, builds upon Interval Estimation (IE) learning [Kaelbling, 1990; Moore and Schneider, 1995] which is useful for addressing the exploration vs. exploitation tradeoff. IE has been used extensively in reinforcement learning for action selection and in stochastic optimization problems. We first explain IE and then discuss how we extend it to learn the best predictor(s) to query, favoring exploration in the early phases and exploitation (least error-prone predictor selection) with increasing frequency. The goal of IE is to find the action a^* yielding the highest expected reward $r(a)$ with as few samples as possible; i.e. $a^* = \arg\max_a E[r(a) \mid a]$. The true expected reward is unknown and must be estimated from observed samples. Before each selection, IE estimates a standard upper confidence interval for the mean reward of each action using the sample mean and standard deviation of rewards received so far using that action:

$$UI(a) = m(a) + t_{\frac{\alpha}{2}}^{(n-1)} \frac{s(a)}{\sqrt{n}} \tag{6.1}$$

where $m(a)$ is the sample mean for a, $s(a)$ is the sample standard deviation for a, n is the number of samples observed from a, and $t_{\frac{\alpha}{2}}^{(n-1)}$ is the critical value for the Student's t-distribution with $n-1$ degrees of freedom at the $\alpha/2$ confidence level.

IE then selects the action with the highest upper confidence interval. The reason is that such an action has a high expected reward and/or a large amount of uncertainty in the reward. If an action has large uncertainty, it indicates that the action has not been taken with sufficient frequency to yield reliable estimates. Selecting this action performs exploration which will increase IE's confidence in its estimate and has the potential of identifying a high reward action. Selecting an action with a high expected reward performs exploitation. Initially, the intervals are large due to the uncertainty of the reward estimates

and action choices tend to be explorative. Over time, the intervals shrink and the choices become more exploitative. IE automatically trades off these two. α is a parameter that weights exploration more strongly when it is small and exploitation more strongly when it is large. $\alpha = 0.05$ is a common reasonable choice.

6.2.3 Interval Estimate Threshold (IEThresh)

The IE algorithm described above can be adapted to work with multiple noisy predictors. Taking an action corresponds to selecting a predictor to ask for a label in our active learning framework, assuming we have already selected an instance to label. We select the instance to label via uncertainty sampling [Lewis and Gale, 1994]. We adopt a logistic regression classifier to obtain posterior class probabilities $P(y \mid x)$. The most uncertain instance according to the posterior class distribution is selected for labeling:

$$x^* = \arg\max_x (1 - \max_{y \in \{1,0\}} P(y \mid x)) \tag{6.2}$$

One also needs to estimate a reward function for each predictor based on the labels received. The reward of each predictor should be related to the true label for the queried instance, which is not known. Hence, we need a mechanism to estimate the true label. We use a majority vote among multiple, possibly noisy predictors to infer the true label – which will be correct often, but not always. We propose the following reward function $\hat{r} : K \rightarrow \{0, 1\}$ as a mapping from the set of predictors K to a binary value. It is 1 if the predictor agrees with the majority label $\bar{y}$, and 0 otherwise.

$$\hat{r}(j) = \begin{cases} 1 & \text{if } y_j = \bar{y} \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

This reward estimate requires sampling some or all predictors to take the majority vote. Its accuracy depends on how well the majority vote represents the true label. When the individual predictor quality is high, the majority vote is a close estimate of the true label since it is unlikely that a majority of the predictors make a mistake on the same instance due to the better-than-random assumption. We propose to adopt a threshold on the upper interval to 1) filter out the less reliable predictors from the majority voting, 2) reduce the labeling cost and 3) compute the reliability estimates more efficiently. Given k predictors, we select each predictor a that has an upper bound $UI(a)$ (Equation 6.1) larger than some fraction of the maximum bound at time t:

$$S_t = \{a | UI(a) >= \epsilon * \max_a UI(a)\} \tag{6.4}$$

where S_t is the set of selected predictors to be queried for labeling. $0 < \epsilon < 1$ is a parameter tuned on a separate dataset that is not used in the experiments.[1] We smooth the confidence

[1]We note that a more sophisticated tuning could further improve the results, but our experimental results indicate that a reasonable threshold works quite effectively.

Table 6.1: Properties of six datasets used in the experiments. All are binary classification tasks with varying sizes.

Dataset	Size	+/- Ratio	Dimensions
image	2310	1.33	18
mushroom	8124	1.07	22
spambase	4601	0.65	57
phoneme	5404	0.41	5
ringnorm	7400	0.98	20
svmguide	3089	1.83	4

interval estimates by initially giving each predictor a reward of 1 and 0. At the first iteration, they have the same upper bound and all predictors are selected. As the bounds tighten, underperforming predictors are filtered out and the reliable ones are selected for labeling. The upper bound can be high because there is either little information about the predictor (high variance) or the entire interval is high and the predictor is good (high mean). It is possible that a previously filtered-out predictor will be selected again if the upper bounds of the remaining predictors lower sufficiently. We give below an outline of how IEThresh works:

1. Initialize samples for each predictor with rewards 1 and 0
2. Fit a logistic regression classifier to labeled data Lb
3. Pick the most uncertain unlabeled instance x^* for labeling (Eqn. 6.2)
4. Compute the upper confidence interval for each predictor (Eqn. 6.1)
5. Choose all predictors S_t within ϵ of the maximum upper confidence interval (Eqn. 6.4)
6. Compute the majority vote $\bar{y}$ of the selected predictors S_t
7. Update labeled data $Lb = Lb \cup \{x^*, \bar{y}\}$
8. Add calculated rewards (Eqn. 6.3) to the samples for S_t
9. Repeat 2-8

Our empirical evaluation indicates that IEThresh is very effective in filtering out the less reliable predictors early in the process and continues to sample the more reliable ones. Next, we describe our experimental results in detail.

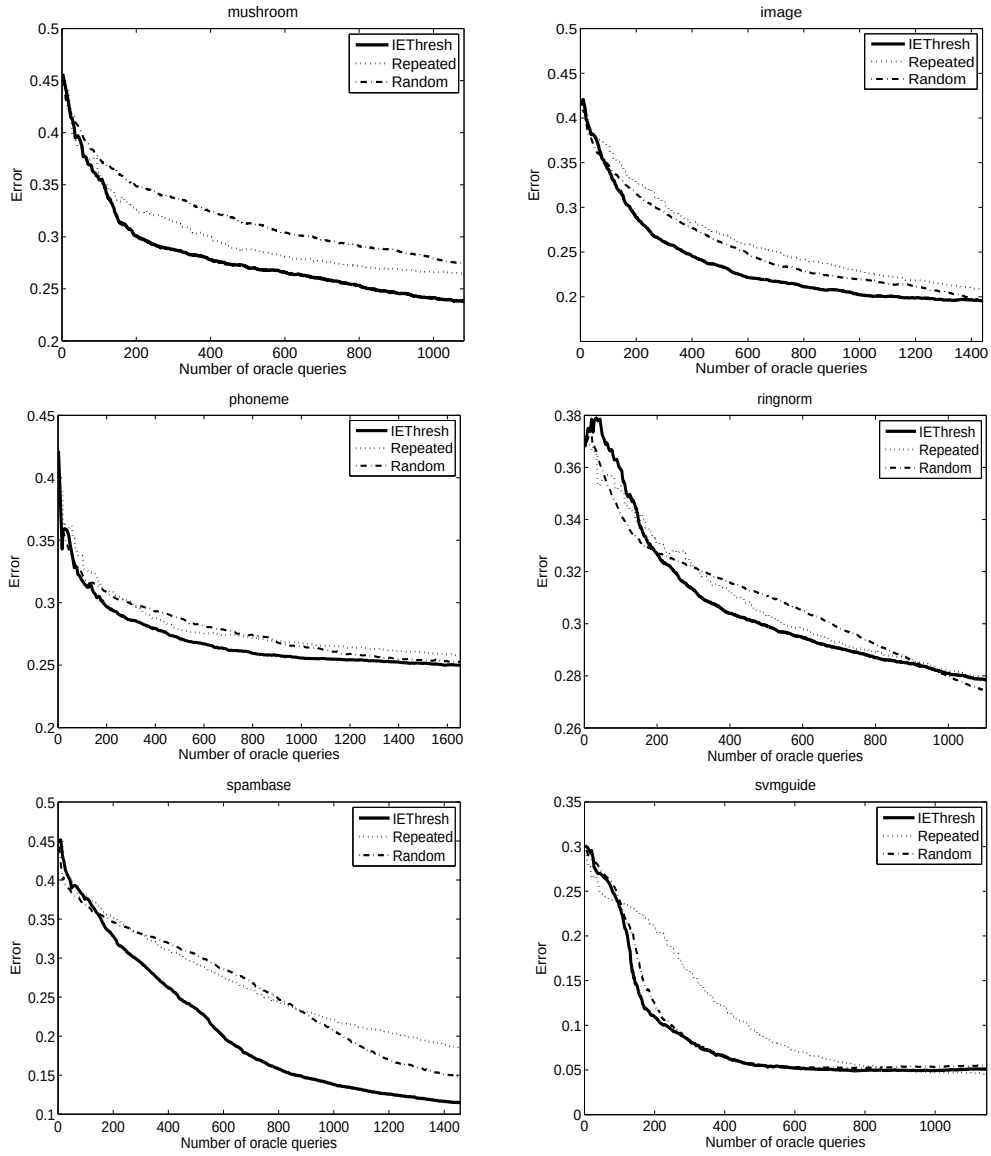


Figure 6.1: Average classification error vs. total number of predictor queries on six benchmark datasets. Number of predictors is $k = 10$ and the predictor accuracies are selected uniformly at random within the range $[.5, 1]$. The solid curve indicates IEThresh in all graphs. The differences are statistically significant based on a two-sided paired t-test at 95% confidence level.

Table 6.2: The size and the annotator accuracies for each AMT dataset.

Data	Size	Annotator Accuracies
TEMP	190	0.44, 0.44, 0.54, 0.92, 0.92, 0.93
RTE	100	0.51, 0.51, 0.58, 0.85, 0.92

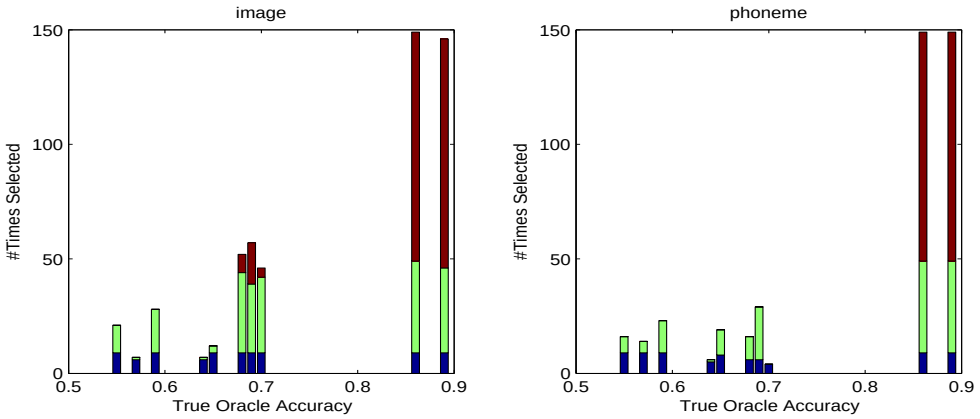


Figure 6.2: Number of times each predictor is queried vs. the true predictor accuracy. Each predictor corresponds to a single bar. Each bar is multicolored where each color shows the relative contribution. Blue corresponds to the first 10 iterations, green corresponds to an additional 40 iterations and red corresponds to another additional 100 iterations. The bar height shows the total number of times an predictor is queried for labeling by IEThresh during first 150 iterations.

6.2.4 Experimental Evaluation

We conducted a thorough analysis on eight benchmark datasets from [Newman *et al.*, 1998; Rätsch *et al.*, 2001; Snow *et al.*, 2008]. Six of these datasets are classification problems with characteristics given in Table 6.1. If the dataset was not originally binary, we converted it using random partitioning into two classes as described in [Rätsch *et al.*, 2001]. We partition each of these datasets into $70\%/30\%$ train/test splits. For each dataset, the initial labeled set includes one true positive and one true negative instance so that each method has the same initial performance before active learning. The rest of the training set is used as the unlabeled pool. We compared *IEThresh* with two baselines: asking all the predictors as introduced in [Sheng *et al.*, 2008] (we refer it as *Repeated*), and asking a randomly chosen predictor (which is referred as *Random*). Each time an unlabeled instance is selected by the active learner, a label is generated according to the true accuracy q of the selected predictors(s), i.e. the true label $y \in \{1, 0\}$ is assigned with probability q and $1-y$ is assigned with probability $1-q$. If more than one predictor is chosen, then the majority vote is

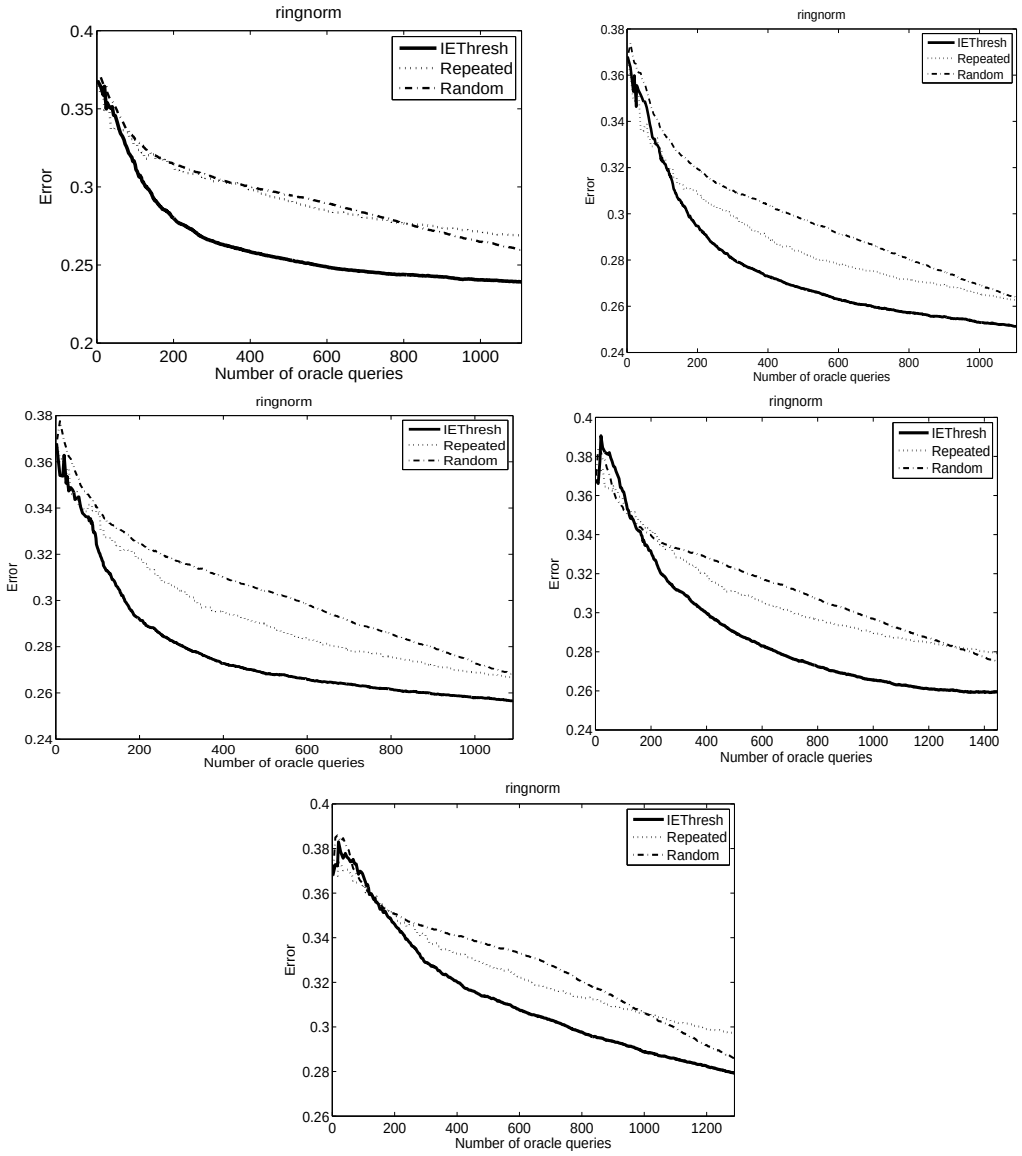


Figure 6.3: Average classification error vs. total number of predictor queries on *ringnorm* dataset. For the top left figure, accuracy $\in [.8, 1]$ for $k_{good} = 5$ predictors and accuracy $\in [.5, .7]$ for the remaining $k_{bad} = 5$ predictors. k_{good} decreases down to 1 and k_{bad} increases up to 9 from left to right, top to bottom.

assigned as the label for that instance (ties are broken randomly). We set the total number of predictors to $k = 10$. After labeling, the instance is added to the training set and the classifier is re-trained on the enlarged set. The classifier is tested on the separate test set every time a new instance is added, and the classification error is reported. The results are averaged over 100 runs.

The remaining two datasets in our experiments are from the natural language understanding tasks introduced in [Snow *et al.*, 2008]. This collection was created using Amazon's Mechanical Turk (AMT) for data annotation. AMT is an online tool where remote workers are paid to complete small labeling and annotation tasks. We selected two binary tasks from this collection: the textual entailment recognition (RTE) and temporal event recognition (TEMP) tasks. In the former task, the annotator is presented with two sentences for each question. He needs to decide whether the second sentence can be inferred from the first. The original dataset contains 800 sentence pairs with a total of 165 annotators who contributed to the labeling effort. The latter task involves recognizing the temporal relation in verb-event pairs. The annotator decides whether the event described by the first verb occurs before or after the second. The original dataset contains 462 pairs with a total of 76 annotators. For both datasets, the quality (accuracy) of annotators are measured by comparing their annotations with the gold standard labels. Unfortunately, most of the annotators completed only a handful of tasks. Therefore, we selected a subset of these annotators for each dataset such that each annotator has completed at least 100 tasks. They have differing accuracies ranging from as low as 0.44 to over 0.9. Due to the lack of a large amount of data, we selected only the instances for which all annotators provided an answer, to enable our method to select one, several or all the annotators, and to have consistent baselines. The annotator accuracies and the size on each dataset is reported in Table 6.2.

We again compared our method *IEThresh* against *Repeated* and *Random* baselines on these two datasets. We randomly selected 50 instances from each dataset to be used by IEThresh as training data to infer estimates for the annotator accuracies. The remaining instances are held out as the test set. At the end of the training, the annotator with the best estimated accuracy is chosen to be employed on the test set. The total number of queries are then calculated as a sum of the number of queries issued during training and the number of queries issued to the chosen annotator during testing. Repeated and Random baselines do not need a training phase since they do not change their annotator selection mechanism via learning. Hence, they are directly evaluated on the test set. The total number of queries is the number of test instances for the Random baseline whereas it is the number of test instances times the number of annotators for the Repeated baseline.

Figure 6.1 compares three methods on six datasets with simulated predictors. The true accuracy of each predictor in Figure 6.1 is drawn uniformly at random from within the range $[.5, 1]$. The figure reports the average classification error with respect to the total number of predictor queries issued by each method. IEThresh is the best performer in all six datasets except in svmguide data IEThresh and Random performs comparably with each

Table 6.3: Relative Performance Comparison on RTE dataset. The last column indicates the total number of queries issued to predictors by each method. IEThresh performs accurately with a moderate labeling effort as opposed to intensive labeling by Repeated.

Method	Accuracy	# Queries
IEThresh	0.92	297
Repeated	0.6	250
Random	0.64	50

Table 6.4: Relative Performance Comparison on TEMP dataset. The last column indicates the total number of queries issued to predictors by each method. IEThresh performs accurately with a moderate labeling effort as opposed to intensive labeling by Repeated.

Method	Accuracy	# Queries
IEThresh	0.92	265
Repeated	0.95	840
Random	0.71	140

other. In ringnorm and spambase datasets, IEThresh initially performs slightly worse than the other methods, indicating that predictor reliability requires more sampling in these two datasets. But, after the estimates are settled (which happens in ~ 200 queries), it outperforms the others, with especially large margins in spambase dataset. The results reported are statistically significant based on a two-sided paired t-test.

We also analyzed the effect of filtering less reliable predictors. An ideal filtering mechanism excludes the less accurate predictors early in the process and samples more from the more accurate ones. In Figure 6.2, we report the number of times each predictor is queried on image and phoneme datasets. The x-axis shows the true accuracy of each predictor. We consider the first 150 iterations of IEThresh and count the number of times each predictor is selected. Each color corresponds to a different time frame; i.e. blue, green and red correspond to $0^{th} - 10^{th}$, $10^{th} - 50^{th}$ and $50^{th} - 150^{th}$ iterations, respectively. At first, each predictor is chosen almost equally since the algorithm explores every possibility to improve its predictor accuracy estimates. Gradually, we see that less accurate predictors are sampled with decreasing frequency, as reliance shifts to the more accurate ones.

We further varied distribution of predictor accuracies to challenge IEThresh. Figures 6.3 and 6.4 show the resulting performance of each method on ringnorm and mushroom datasets. The top left figure on each graph indicates the case with 5 highly fallible predictors with accuracy level within $[.5, .7]$, and 5 reliable ones with accuracies within $[.8, 1]$ range. From left to right, top to bottom, the set of predictors becomes more skewed towards the fallible predictors. The results point out that IEThresh generalizes to work with a wide range of predictor reliability distributions. Even in the challenging case where there are

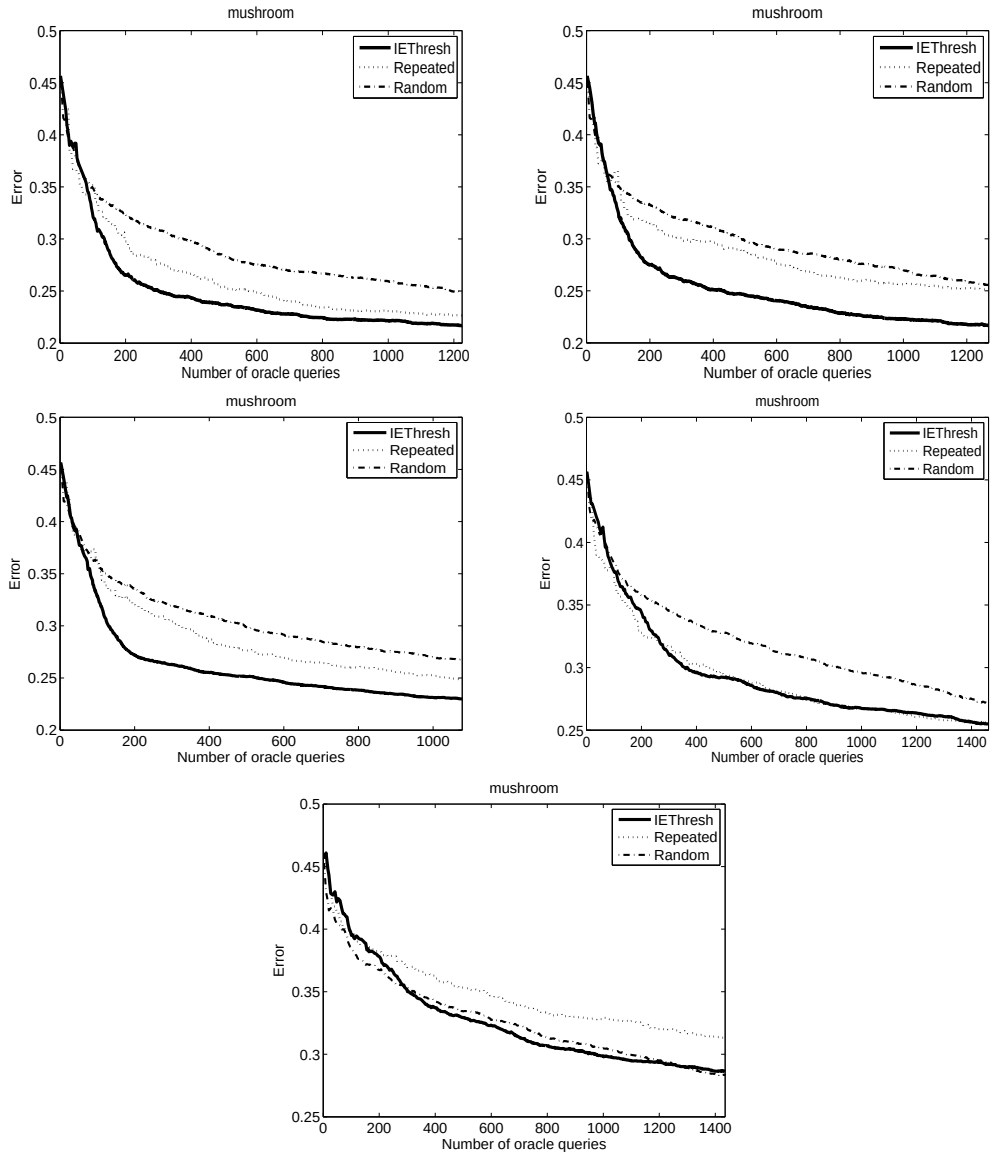


Figure 6.4: Average classification error vs. total number of predictor queries on UCI *mushroom* dataset. For the top left figure, accuracy $\in [.8, 1]$ for $k_{good} = 5$ predictors and accuracy $\in [.5, .7]$ for the remaining $k_{bad} = 5$ predictors. k_{good} decreases down to 1 and k_{bad} increases up to 9 from left to right, top to bottom.

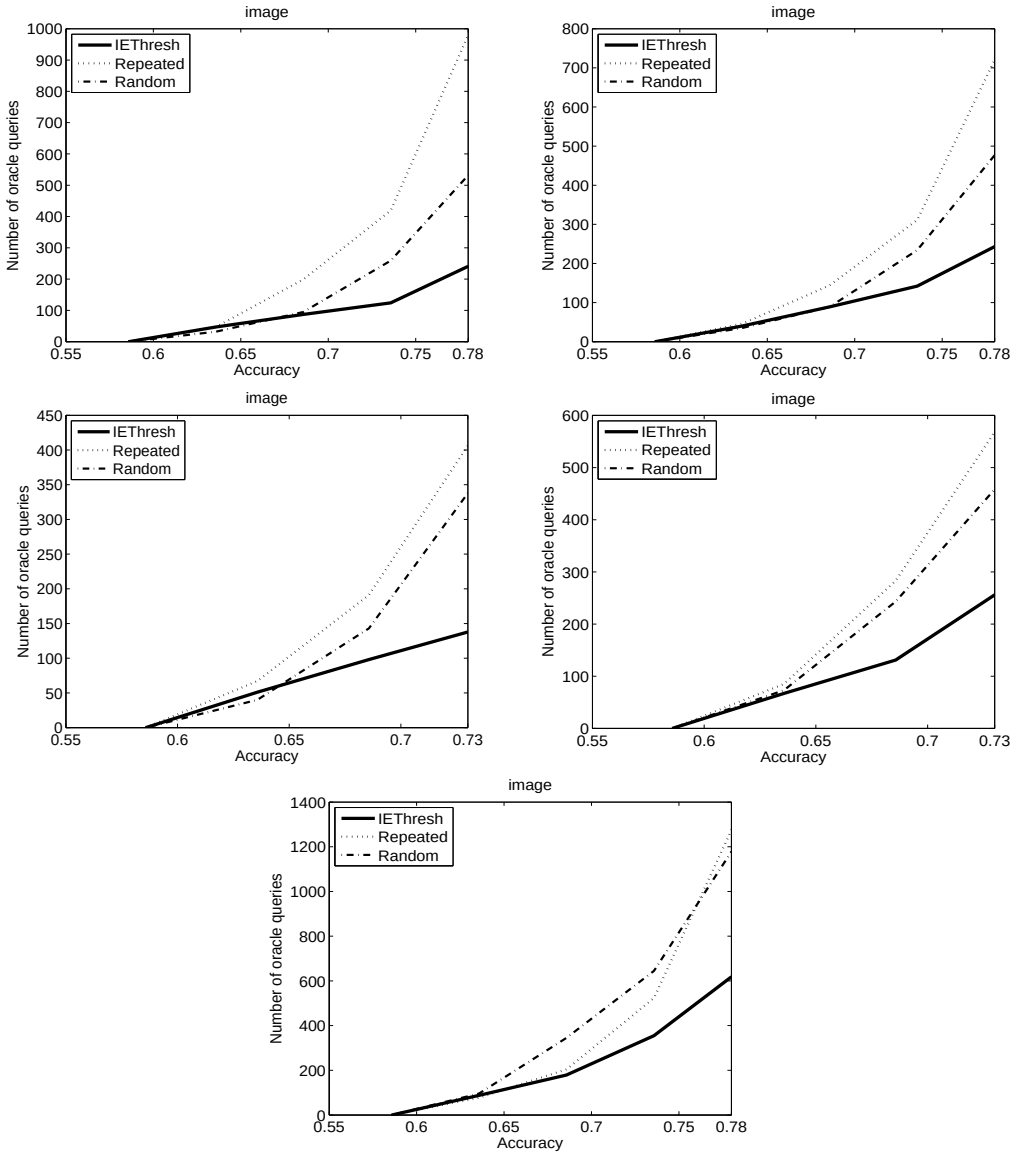


Figure 6.5: Total number of predictor queries required to reach a target accuracy is plotted on UCI *image* dataset. For the top left figure, accuracy $\in [.8, 1]$ for $k_{good} = 5$ predictors and accuracy $\in [.5, .7]$ for the remaining $k_{bad} = 5$ predictors. k_{good} decreases down to 1 and k_{bad} increases up to 9 from left to right, top to bottom.

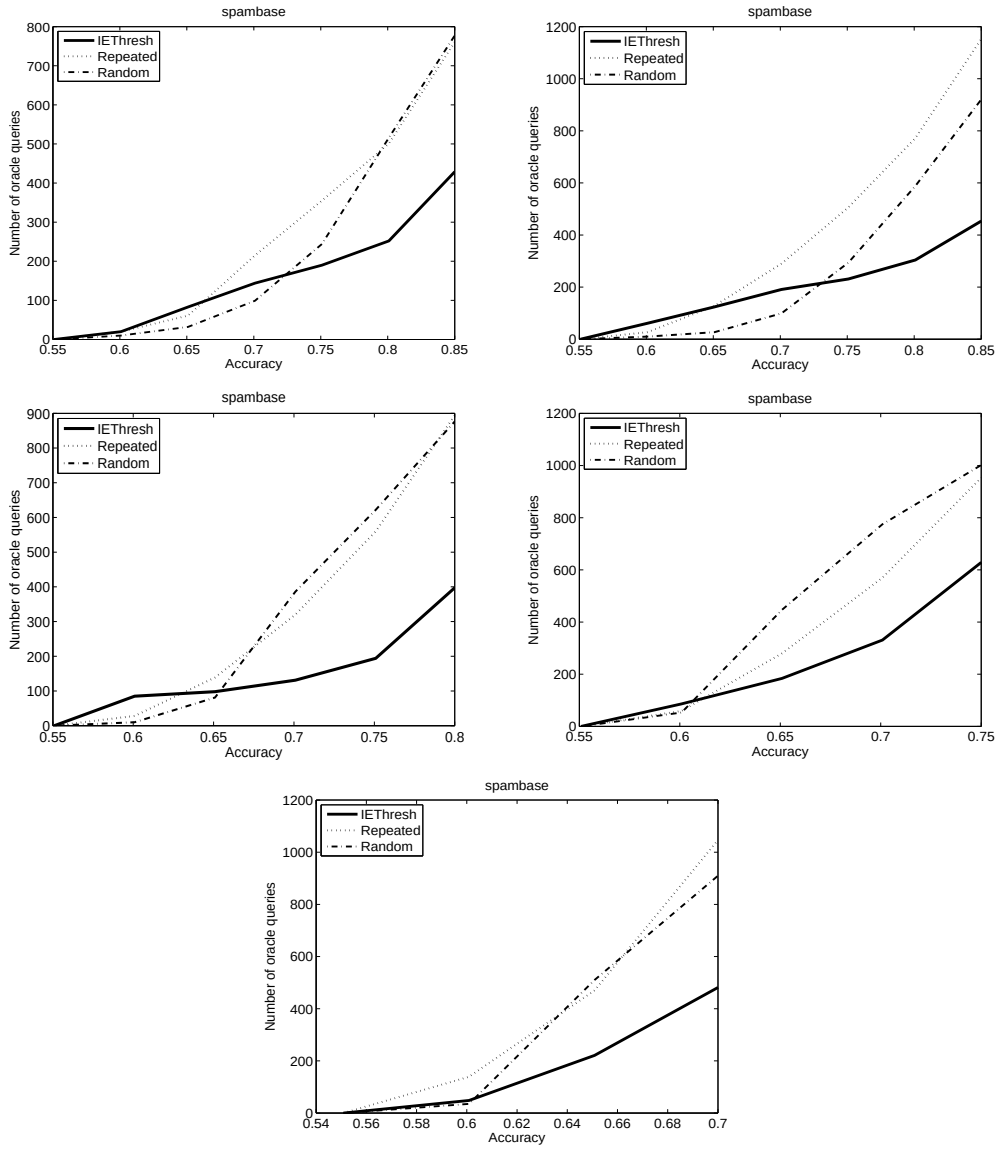


Figure 6.6: Total number of predictor queries required to reach a target accuracy is plotted on UCI *spambase* dataset. For the top left figure, accuracy $\in [.8, 1]$ for $k_{good} = 5$ predictors and accuracy $\in [.5, .7]$ for the remaining $k_{bad} = 5$ predictors. k_{good} decreases down to 1 and k_{bad} increases up to 9 from left to right, top to bottom.

only one or two reliable predictors in a given set, the algorithm is able to detect the good ones. Figures 6.5 and 6.6 report a similar set of results from a different perspective. The graphs show the total number of queries required to achieve a target classification accuracy. IEThresh requires the least number of queries for a given accuracy level for most cases. Especially when the accuracy targets are high, giving time for IEThresh to stabilize its predictor accuracy estimates, it can improve classification accuracy without the intensive labeling effort required by the baselines.

Lastly, we report the results on the RTE and TEMP datasets that have real annotations from multiple less-than-reliable predictors. Table 6.3 reports the accuracy of each method on the test set for RTE data with the corresponding number of predictor queries issued. The accuracy of IEThresh is the same as the accuracy of the single best predictor in this dataset (See Table 6.2), indicating that IEThresh managed to detect the best predictor during the training phase. The Repeated labeling and Random baselines perform poorly in this dataset due to the majority of highly unreliable predictors. Table 6.4 reports the results on the test set for TEMP data. The Repeated labeling baseline is the marginally-best performer in this dataset but at a high cost (a large number of queries). On the other hand, IEThresh has a very close performance to Repeated with much less labeling effort.

6.3 A Sequential Bayesian Estimation Approach in Non-stationary Conditions

In this section, we describe in detail our estimation and selection framework to learn with multiple predictors with time-varying accuracies. We start with the underlying particle filtering algorithm and the specific probabilistic distributions we modeled. Then, we discuss how to modify this model to estimate the varying accuracy of each predictor and simultaneously select the most accurate ones. Our framework supports a balance between exploration and exploitation to achieve estimation accuracy without extensively exploiting the predictors and incurring associated costs.

6.3.1 Sequential Bayesian Estimation

Particle filtering is a special case of a more general family of models called Bayesian Sequential Estimation [Arulampalam *et al.*, 2001a]. The Bayesian approach to estimate a system that dynamically changes is to construct the posterior probability density function (pdf) of the state of the system based on all information observed up to that point. Inference on such a system requires at least two models: a model describing the evolution of the state with time and a model governing the generation of the noisy observations. Once the state space is probabilistically modeled and the information is updated based on new

observations, we are provided with a general Bayesian framework to model the dynamics of a changing system.

The problem of estimating the time-varying accuracy of a predictor can be cast into the sequential estimation framework. The states of the system correspond to the unknown time-varying accuracy of the predictor where ϕ_t represents the accuracy of the predictor at time t. The observations are the noisy labels z_t output by the predictor according to some probability distribution governed by the corresponding labeling accuracy ϕ_t. In this problem, it is important to estimate the accuracy every time an observation is obtained. Hence, it is crucial to have an estimate of the accuracy of each predictor to infer a more accurate prediction. Or equivalently, the estimation update is required at each step to decide which predictors to query next. For problems where an estimate is required every time an observation is made, the sequential filtering approach offers a convenient solution. Such filters alternate between prediction and update stages. The prediction stage predicts the next state given all the past observations. The update stage modifies the predicted prior from the previous time step to obtain the posterior probability density of the state at the current time.

We design this dynamic system with the following probabilistic models. Let ϕ_t denote the labeling accuracy and it is assumed to change according to the following model:

$$\begin{aligned} \phi_t &= f_t(\phi_{t-1}, \Delta_{t-1}) \\ &= \phi_{t-1} + \Delta_{t-1} \end{aligned} \tag{6.5}$$

where Δ_t is a zero-mean, σ^2-variance Gaussian random variable, restricted so that ϕ_t never exceeds 1 (more on this later). f_t denotes that the accuracy at the current time step differs from the previous accuracy by some small amount drawn from a Gaussian distribution. The mean of the Gaussian is assumed to be zero not to introduce any bias towards increase or decrease in accuracy. We can easily add a bias, e.g. positive mean μ for predictors who are expected to improve with experience. However, we prefer fewer parameters and we can still track the improving sources (more details in Section 6.3.4). We assume σ or at least its upper bound to be known. We realize that the exact value of σ can be difficult to obtain in many situations. On the other hand, it is often reasonable to assume the maximum rate of change of the labeling accuracy is known. We added an analysis testing the sensitivity to the exact σ in the experiments section 6.3.4. For notational simplicity and concreteness, we focus on binary classification here; extensions to the multi-category case are straightforward generalizations to this scheme. Furthermore, the accuracy at any time t is assumed to be between 0.5 and 1, i.e. any predictor is a weak learner. Thus, the transition probability from one state to the next follows a truncated Gaussian distribution:

$$p(\phi_t \mid \phi_{t-1}, \sigma, 0.5, 1) = \frac{\frac{1}{\sigma}\beta(\frac{\phi_t - \phi_{t-1}}{\sigma})}{\Phi(\frac{1-\phi_{t-1}}{\sigma}) - \Phi(\frac{0.5-\phi_{t-1}}{\sigma})} \tag{6.6}$$

where β and Φ are the pdf and cdf of the standard Gaussian distribution, respectively.

The observed variables $z^j_{1:t}$ are the sequentially arriving noisy labels generated by predictor j at the corresponding time steps. We model the noisy label generation with a Bernoulli distribution given the true label y. In other words, the noisy label z_t is a random variable whose probability conditioned on the accuracy ϕ_t at time t and the true label y_t is given as

$$p(z^j_t \mid \phi^j_t, y_t) = {\phi^j_t}^{I(z^j_t = y_t)}(1-\phi^j_t)^{I(z^j_t \neq y_t)}. \tag{6.7}$$

(6.7) requires y_t, which is unknown. We use the wisdom-of-the-crowds trick to predict y_t. Specifically, we estimate the probability of the true label y_t conditioned on the noisy labels observed from all the other predictors.

$$p(z^j_t \mid \phi^j_t, z_t^{\acute{J}(t)}) = \sum_{y \in \mathcal{Y}} p(z^j_t \mid \phi^j_t, y_t = y) P(y_t = y \mid z_t^{\acute{J}(t)}) \tag{6.8}$$

$\acute{J}(t)$ is the set of selected predictors at time t such that $j \notin \acute{J}(t)$; hence, $z_t^{\acute{J}(t)} = \{z^s_t \mid s \in \acute{J}(t)\}$. We compute $P(y_t = y \mid z_t^{J(t)})$ using the estimated expected accuracy of the selected predictors and the estimated $\hat{P}_{t-1}(y)$ (the prior as estimated at time $t-1$) from the previous time step.

$$P(y_t = y \mid z_t^{J(t)}) = \frac{\hat{P}_{t-1}(y) \prod_{s \in \acute{J}(t)} p_{\hat{E}[\phi^s_{t-1}]}(z^s_t \mid y)}{\sum_{\acute{y} \in \mathcal{Y}} \hat{P}_{t-1}(y) \prod_{s \in \acute{J}(t)} p_{\hat{E}[\phi^s_{t-1}]}(z^s_t \mid y)} \tag{6.9}$$

We describe how to estimate $\hat{E}[\phi^s_{t-1}]$ and $\hat{P}(y)$ in the next section. Here we focus on the state space model and how it leads to an estimate of the posterior state distribution $p(\phi_t \mid z_{1:t})$ at any given time t.

The true state sequence ϕ_t is modeled as an unobserved Markov process that follows a first-order Markov assumption. In other words, the current state is independent of all earlier states given the immediately previous state.

$$p(\phi_t \mid \phi_0, \ldots, \phi_{t-1}) = p(\phi_t \mid \phi_{t-1}) \tag{6.10}$$

Similarly, the observation z_t depends only on the current state ϕ_t and is conditionally independent of all the previous states given the current state.

$$p(z_t \mid \phi_0, \ldots, \phi_t) = p(z_t \mid \phi_t) \tag{6.11}$$

Figure 6.7 shows the graphical structure of this model. As noted earlier, we are interested in constructing $p(\phi_t \mid z_{1:t})$. It is generally assumed that the initial state distribution $p(\phi_0)$ is given. However, in this work, we do not make this assumption and simply adopt a uniform

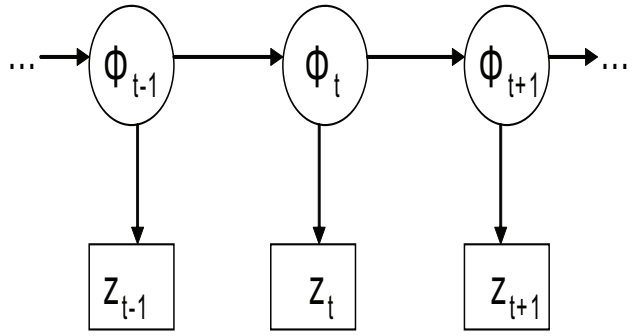


Figure 6.7: The Hidden Markov Model structure for the time-varying accuracy of a predictor. The state sequence ϕ_t is an unobserved first-order Markov process. The observation z_t is dependent only on the current state ϕ_t for all $t = 1, 2, \ldots$.

(noninformative) prior for $p(\phi_0)$ where $0.5 < \phi_0 < 1$.[2] Suppose that the state distribution $p(\phi_{t-1} \mid z_{1:t-1})$ at time $t-1$ is available. The prediction stage obtains the pdf of the state at time step t via the Chapman-Kolmogorov equation[Arulampalam *et al.*, 2001b].

$$p(\phi_t \mid z_{1:t-1}) = \int_{\phi_{t-1}} p(\phi_t \mid \phi_{t-1}) p(\phi_{t-1} \mid z_{1:t-1}) d\phi_{t-1} \tag{6.12}$$

where $p(\phi_t \mid \phi_{t-1})$ is substituted with (6.6). Once a new observation z_t is made, then (6.12) is used as a prior to obtain the posterior $p(\phi_t \mid z_{1:t})$ at the update stage via Bayes rule.

$$p(\phi_t \mid z_{1:t}) = \frac{p(z_t \mid \phi_t) p(\phi_t \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})} \tag{6.13}$$

where $p(z_t \mid \phi_t)$ is substituted with (6.8) and $p(z_t \mid z_{1:t-1}) = \int_{\phi_t} p(z_t \mid \phi_t) p(\phi_t \mid z_{1:t-1}) d\phi_t$. When the posterior (6.13) at every time step t is assumed to be Gaussian, then Kalman filters provide optimal solutions to the state density estimation. In cases like ours where the posterior density is not Gaussian, sequential particle filtering methods are appropriate for approximating the optimal solution. Next, we review the particle filtering algorithm and describe how it is modified to tackle our problem.

6.3.2 Particle Filtering for Estimating Time-Varying Predictor Accuracy

The particle filtering algorithm is a technique for implementing sequential Bayesian estimation by Monte Carlo simulations. The underlying idea is to estimate the required posterior

[2]The results show that the estimation is very effective despite the uninformative prior (See Section 6.3.4 for more details). However, in cases where such information is available, we anticipate that the results could be improved even further.

density function with a discrete approximation using a set of random samples (particles) and associated weights.

$$p(\phi_t \mid z_{1:t}) \sim \sum_{i=1}^{N} w_t^i \delta(\phi_t - \phi_t^i) \tag{6.14}$$

where N is the number of particles. As N increases, it can be shown that the above approximation approaches the true posterior density [Doucet and Crisan, 2002]. w_t^i's are called the normalized importance weights, and δ is the Dirac delta function[3]. The particle filtering algorithm estimates these weights in a sequential manner. Weights in (6.14) are approximated as

$$\begin{aligned} \tilde{w}_t^i &\approx \frac{p(\phi_{1:t}^i \mid z_{1:t})}{\pi(\phi_{1:t}^i \mid z_{1:t})} \\ w_t^i &= \frac{\tilde{w}_t^i}{\sum_{j=1}^{N} \tilde{w}_t^j} \end{aligned} \tag{6.15}$$

where π is called the importance density from which the samples ϕ^i are generated. This is because in general we cannot directly sample from the posterior $p(\phi_{1:t} \mid z_{1:t})$, but rather use an importance density that we can easily draw the samples from. The algorithm sequentially samples ϕ_t^i, $i = 1, 2, \ldots, N$, from the importance density and updates the weights w_t^i to approximate the posterior state distribution. The choice of the importance density is important and one commonly used and convenient choice is to use the state transition density $p(\phi_t \mid \phi_{t-1})$ (in our case given in (6.6)), i.e. $\pi(\phi_t \mid \phi_{1:t-1}^i, z_{1:t}) = p(\phi_t \mid \phi_{t-1})$. Then, the weight update equation simplifies to

$$\begin{aligned} \tilde{w}_t^i &\approx \frac{p(z_t \mid \phi_t^i) p(\phi_t^i \mid \phi_{t-1}^i) p(\phi_{1:t-1}^i \mid z_{1:t-1})}{\pi(\phi_t^i \mid \phi_{1:t-1}^i, z_{1:t}) \pi(\phi_{1:t-1}^i \mid z_{1:t-1})} \\ &= \frac{p(z_t \mid \phi_t^i) p(\phi_t^i \mid \phi_{t-1}^i)}{\pi(\phi_t^i \mid \phi_{1:t-1}^i, z_{1:t})} \tilde{w}_{t-1}^i \\ &= p(z_t \mid \phi_t^i) \tilde{w}_{t-1}^i \text{ for } t > 1 \end{aligned} \tag{6.16}$$

where $p(z_t \mid \phi_t^i)$ is substituted with (6.8). The pseudo-code of the basic filtering algorithm is given in Figure 6.8.

[3]

$$\delta(x) = 0, \text{ if } x \neq 0$$
$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$

1. Sample from the initial distribution $\phi_0^i \sim p(\phi_0)$ for $i = 1, \ldots, N$ and assign weights $w_0^i = \frac{1}{N}$
2. For $t > 0$, and $i = 1, \ldots, N$
   - Draw $\phi_t^i \sim p(\phi_t \mid \phi_{t-1})$ using (6.6) and update weights $\tilde{w}_t^i = p(z_t \mid \phi_t^i)\tilde{w}_{t-1}^i$.
   - Normalize the weights $w_t^i = \frac{\tilde{w}_{t-1}^i}{\sum_{j=1}^N \tilde{w}_{t-1}^j}$ and compute $\hat{N}_e = \frac{1}{\sum_{i=1}^N (w_t^i)^2}$.
   - If $\hat{N}_e < T$, then resample $\phi_t^i \sim \text{pmf}[\{w_t\}]$ and reassign $w_t^i = \frac{1}{N}$
   - Update the posterior state density $p(\phi_t \mid z_{1:t}) = \sum_{i=1}^N w_t^i \delta(\phi_t - \phi_t^i)$
3. Update $t = t + 1$ and go to step 2.

Figure 6.8: The pseudo-code for the basic filtering algorithm

The resampling step in Figure 6.8 happens only if the vast majority of the samples have negligible weights; in other words, the estimated effective sample size $\hat{N}_e$ falls below some predefined threshold. The exact value of the threshold is not crucial, the idea is to detect significantly small weights. This is called the degeneracy problem in particle filters, and is resolved by resampling using a probability mass function (pmf) over the weights. The goal of resampling is to eliminate particles with small weights and concentrate on large ones. It is shown that it is possible to implement this resampling procedure in $O(N)$ time complexity using order statistics [Ripley, 1987; Carpenter *et al.*, 1999]. We do not provide more details on this since it is out of the scope of this thesis, but we note that in our simulations the algorithm hardly needs to resample, largely because the importance density we adopted (6.6) represents the state transitions well.

6.3.3 Particle Filtering for Predictor Selection

So far, we have focused on estimating the posterior state distribution at any given time t. Our problem, however, consists of multiple predictors (annotators) where each predictor's time-varying accuracy is modeled as a separate state sequence model. Hence, at any time t we may have multiple noisy observations (labels) from multiple predictors. In this section, we describe how to select which predictors to query for the labels and how to utilize these noisy labels to predict the true label and hence estimate the marginal label distribution $P(y)$.

Our aim is to select potentially the most accurate predictors at each time step. The sequential estimation framework provides us with an effective tool to achieve this goal.

First, we approximate the prior pdf of ϕ_t^j (6.12) $\forall j$ by its discrete version

$$p(\phi_t^j \mid Z_{t-h(j)}^j) = \sum_{i=1}^{N} p(\phi_t^j \mid \phi_{t-h(j)}^{j,i}) p(\phi_{t-h(j)}^{j,i} \mid Z_{t-h(j)}^j) \tag{6.17}$$

where $t - h(j)$ denotes the last time the predictor j is selected, e.g. $t - h(j) = t - 1$ if the predictor is queried for labeling at the immediately previous time step. $Z_{t-h(j)}^j$ denotes all the observations obtained from predictor j up to time $t - h(j)$. $\{\phi_{t-h(j)}^{j,i}\}_{i=1}^N$ denotes the sampled accuracy values for the j-th predictor at time $t - h(j)$. Furthermore, we formulate the change in the labeling accuracy as a Gaussian random walk bounded between 0.5 and 1. More importantly, the true accuracy of a predictor keeps evolving according to this random walk regardless of whether it is selected by our method. For computational expediency, we approximate the state transition probability by truncating the Gaussian distribution once after $h(j)$ steps instead of after every single step for an unexplored predictor. More formally, recall (6.5) where the step size Δ_t is drawn from a Gaussian $\Delta_t \sim \mathcal{N}(0, \sigma^2)$. Hence,

$$\begin{aligned}
\phi_t^j &= \phi_{t-1}^j + \Delta_{t-1} \\
&= \phi_{t-2}^j + \Delta_{t-2} + \Delta_{t-1} \\
&\vdots \\
&= \phi_{t-h(j)}^j + \sum_{r=1}^{h(j)} \Delta_{t-r} \\
\phi_t^j &\sim \mathcal{N}(\phi_t^j \mid \phi_{t-h(j)}^j, h(j)\sigma^2, 0.5 < \phi_t^j < 1)
\end{aligned} \tag{6.18}$$

(6.18) captures our intuition that the more a predictor j goes unexplored ($h(j)$ increases), the further our belief about its accuracy diverges from the last time it was estimated (variance $h(j)\sigma^2$ increases).

Next, we draw samples $\{\phi_t^{j,i}\}_{i=1}^N$ from the distribution given above in (6.18). We weight these samples by their corresponding predicted probability (6.17) given what has been observed up to time t:

$$b_t^{j,i} = p(\phi_t^j \mid Z_{t-h(j)}^j) \phi_t^{j,i} \tag{6.19}$$

For each j, we sort the weighted samples $b_t^{j,i}$ in ascending order and find their 95th percentile. This represents the value of the upper 95th confidence interval for labeler j. We then apply the IEThresh method in Section 6.2 by selecting all predictors whose 95th percentile is higher than a predefined threshold T. We denote the set of selected predictors at time t by $J(t)$. This selection allows us to take the cumulative variance into account and select the predictors with potentially high accuracies. We tuned the parameter T on a separate

dataset not reported here. It can further be adjusted according to budget allocated to label acquisition or to prevent unnecessary sampling especially when the number of predictors is large.

After the committee $J(t)$ of selected predictors is identified, we observe their outputs z_t^j and update the weights $w_t^{j,i}$ using (6.16) and update the posterior $p(\phi_t^j \mid Z_t^j)$ using (6.14). Relying on the posterior accuracy distribution, we compute the estimated expected accuracy as follows:

$$E_{p(\phi_t^j|Z_t^j)}[\phi_t^j] = \sum_{i=1}^{N} p(\phi_t^{j,i} \mid Z_t^j)\phi_t^{j,i} \tag{6.20}$$

Finally, we integrate the observed noisy labels from the selected committee to predict the true label using a map estimate of the estimated posterior distribution:

$$\begin{aligned}
\hat{y}_t^{\text{map}} &= \arg\max_{y\in\mathcal{Y}} P(y \mid Z_t^{J(t)}) \\
&= \arg\max_{y\in\mathcal{Y}} \hat{P}_{t-1}(y) \prod_{j\in J(t)} p_{E[\phi_t^j]}(z_t^j \mid y) \\
&= \arg\max_{y\in\mathcal{Y}} \hat{P}_{t-1}(y) \prod_{j\in J(t)} E[\phi_t^j]^{I(z_t^j=y)}(1-E[\phi_t^j])^{I(z_t^j\neq y)}
\end{aligned} \tag{6.21}$$

We then use the predicted labels to update the marginal label distribution:

$$\hat{P}_t(y=1) = \frac{1}{t}\sum_{s=1}^{t} \hat{y}_s^{\text{map}} \tag{6.22}$$

where $\hat{y}^{\text{map}} \in \{0,1\}$. Initially, we start with a uniform marginal label distribution.

See Figure 6.9 for an outline of our algorithm, which we name *SFilter*.

6.3.4 Experimental Evaluation

In this section, we describe the experimental evaluation and offer our observations. We have tested SFilter from many different perspectives. The predictors are simulated in such a way that they have different initial labeling accuracies but share the same σ. Furthermore, each instance that we seek to label corresponds to a single time step; hence, the accuracy transitions to the next state at every instance. These conditions are true for all experiments unless otherwise noted.

First, we conducted an evaluation to test how the proposed model behaves with respect to various degrees of change in accuracy. The degree of this change depends on the variance

1. Sample from the initial distribution $\phi_0^i \sim p(\phi_0)$ for $i = 1, \ldots, N$ and assign weights $w_0^i = \frac{1}{N}$
2. For $t = 1$, initialize $\hat{P}_1(y) = 0.5$ and run a single iteration of the basic filtering algorithm (Figure 6.8) for $j = 1, \ldots, K$.
3. Initialize $h(j) = 1$ and $Z_1^j = \{z_1^j\}\ \forall j = 1, \ldots, K$
4. For $t > 1$, and $i = 1, \ldots, N$
   - Compute $\hat{P}_t(y)$ acc. to (6.22).
   - Draw samples $\phi_t^{j,i}$ using (6.17) and estimate the expected accuracy via (6.20) for $j = 1, \ldots, K$.
   - Select the predictors $J(t)$ to be queried
   - For $j \in J(t)$
     - Update weights $\tilde{w}_t^{j,i} = p(z_t^j \mid \phi_t^{j,i}) \tilde{w}_{t-h(j)}^{j,i}$
     - Normalize the weights $w_t^{j,i} = \frac{\tilde{w}_{t-h(j)}^{j,i}}{\sum_{l=1}^{N} \tilde{w}_{t-h(j)}^{j,l}}$ and compute $\hat{N}_{ej} = \frac{1}{\sum_{i=1}^{N} (w_t^{j,i})^2}$.
     - If $\hat{N}_{ej}$ is too small, then resample $\phi_t^{j,i} \sim \text{pmf}[\{w_t^j\}]$ and reassign $w_t^{j,i} = \frac{1}{N}$.
     - Update $h(j) = t$ and $Z_t^j = Z_{t-h(j)}^j \cup \{z_t^j\}$.
     - Update the posterior state density $p(\phi_t^j \mid Z_t^j) = \sum_{i=1}^{N} w_t^{j,i} \delta(\phi_t^j - \phi_t^{j,i})$.
5. Update $t = t + 1$ and go to step 4.

Figure 6.9: Outline of the SFilter algorithm.

of the Gaussian distribution that controls the width of the step size in (6.5). We analyzed how σ affects the estimation process in terms of the overall quality of the integrated labels and the total labeling effort. We simulated 10 predictors with varying initial accuracies but with the same σ for 500 instances. The accuracy of each predictor evolves according to (6.6) and every selected predictor $j \in J_t$ outputs a noisy label z_t^j. We integrate these labels to predict the true label via (6.21). Table 6.5 shows the percentage of correct labels predicted and the number of total predictor queries with the corresponding σ. $\%$-correct represents the percentage of the correct labelings among all instances $m = 500$, i.e. $\% - \text{correct} = \frac{\sum_{t=1}^{m} I(\hat{y}_t^{\text{map}} - y_t^{\text{true}})}{m}$ where y_t^{true} is drawn from $P(y = 1) = 0.75$. Among the 10 predictors the highest average accuracy is 0.81 whereas the percentage of correct labels integrated by SFilter is significantly higher, i.e. 0.85. As the variance increases, the

Table 6.5: Performance measurement of SFilter w.r.t increasing σ values in the presence of 10 predictors.

σ	$\%$-correct	# queries	# instances
0.02	0.858	975	500
0.04	0.846	1152	500
0.06	0.834	1368	500

Table 6.6: Robustness analysis of SFilter against small perturbations in estimated vs. true σ when the true σ is equal to 0.02. The analysis is performed over 500 instances drawn from $P(y = 1) = .75$, where $y \in \{0, 1\}$

σ	$\%$-correct	# queries
0.02	0.858	975
0.022	0.854	1015
0.018	0.854	1027
0.016	0.845	1031
0.024	0.848	1082

total number of queries also increases and there is a slight decay in the percentage of correct labelings. The relatively rapid change of predictor qualities leads to more extensive exploration to catch any drifts without compromising the overall performance too much. We have also tested the robustness of the proposed algorithm against small perturbations in σ. We repeated the above analysis using $\sigma = 0.02$ to generate the accuracy drift, but executed SFilter using slightly different σ values in order to test perturbational robustness. Table 6.6 indicates that SFilter is robust to small noise in σ. The $\%$-correct remains relatively stable as we add noise to the true σ while the total amount of labeling is only slightly larger.

Next, we analyzed how the variations among the individual predictor qualities affect the estimation and selection. The first experiment uses the same committee of predictors ($k = 10$) as above whose qualities are uniformly distributed between 0.5 and 1. The second experiment uses a committee where only a few good predictors exists (only 3 predictors with initial accuracy above 0.8 and the rest below 0.65). We used a fixed $\sigma = 0.02$ for both cases. Table 6.7 shows the effectiveness of SFilter to make correct label predictions with moderate labeling effort in both cases. The last column in Table 6.7 shows the average estimated marginal label distribution $\hat{P}(y = 1)$ where the true marginal is $P(y = 1) = 0.75$. Our algorithm is very effective for estimating the unknown label distribution and also for exploiting the predictors only when they are highly reliable as shown in Figure 6.10. Figure 6.10 displays the true source accuracy over time and the times the source is selected by SFilter (denoted by red circles) and when otherwise (denoted by blue circles). We only report a representative subset out of 10 total sources. Note that SFilter selects the highly reliable sources in this skewed set at the beginning. When these sources become less reliable,

Table 6.7: Performance measurement of SFilter for various quality predictors. Uniform denotes the predictors' initial accuracies are uniformly distributed. Skewed denotes there are only a few good predictors while the majority are highly unreliable. $P(y = 1) = 0.75$.

distribution	%-correct	# queries	$\hat{P}(y = 1)$
uniform	0.858	975	0.768
skewed	0.855	918	0.773

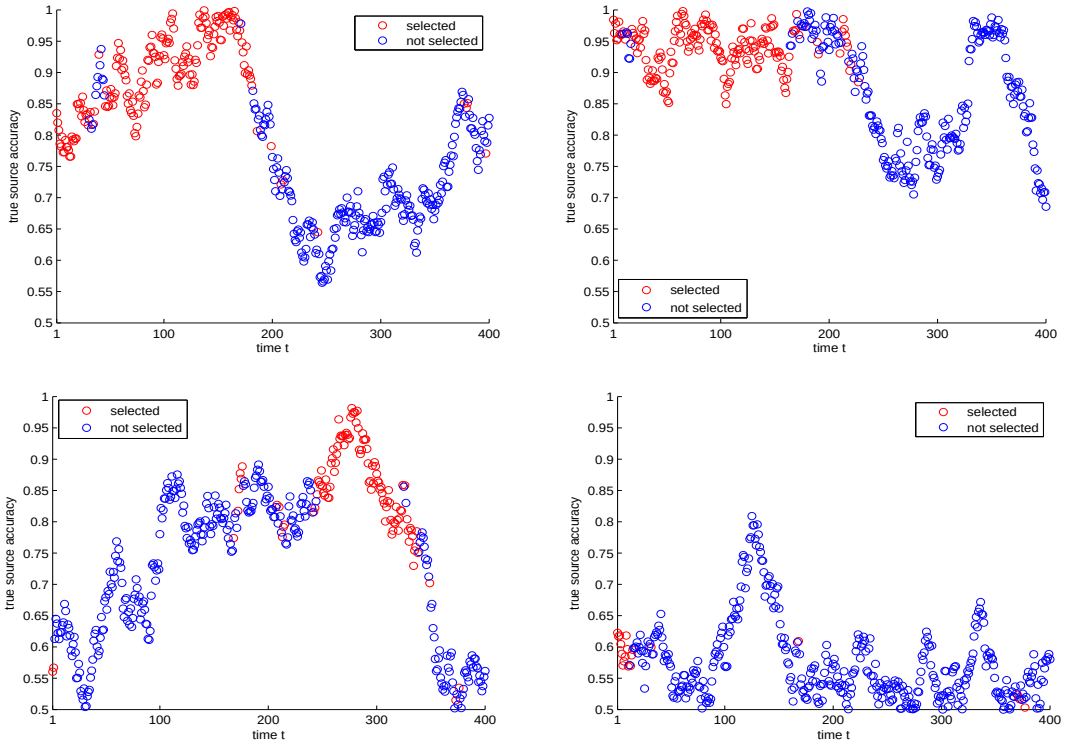


Figure 6.10: SFilter selects the sources when they are relatively highly accurate even in the skewed case where there are only a few good sources to begin with. SFilter also detects when the initially bad sources become more favorable later and begins exploiting them.

SFilter finds other sources that have become more favorable over time (i.e. lower left corner in Figure 6.10). Moreover, sources that have relatively low quality throughout the entire time are hardly selected by SFilter except a few exploration attempts. This is a desirable behaviour of the algorithm and certainly reduces the labeling effort without hurting overall performance even with skewed distributions of source accuracies.

We further challenged SFilter with a pathological example to test its robustness against

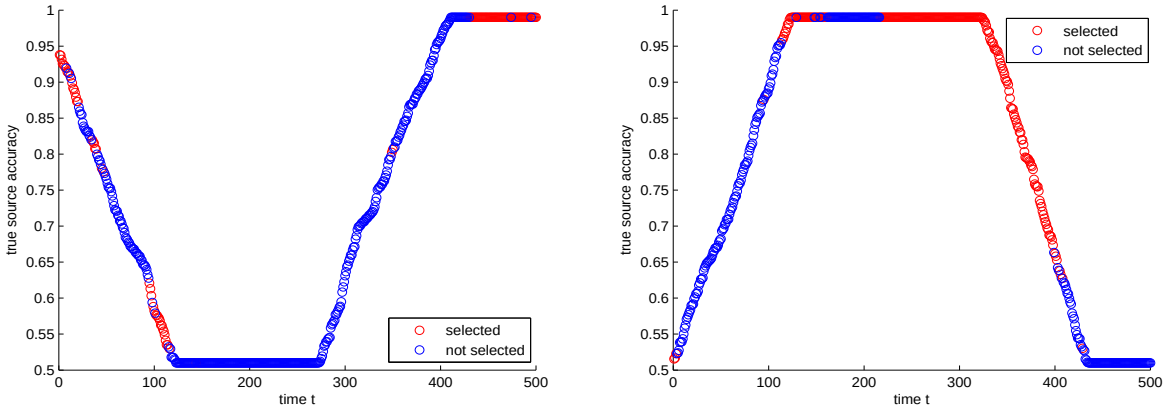


Figure 6.11: SFilter is able to detect the consistent increase or decrease in any quality and adopts quickly by choosing the predictors when they are reliable, though with occasional exploration attempts.

how the predictors' accuracies change. We simulated $k = 10$ predictors with initial accuracies ranging from 0.51 to 0.94. All the predictors except the best (0.94 accuracy) and the worst (0.51 accuracy) follow a random walk with the step size drawn from $\mathcal{N}(0, \sigma^2)$ where $\sigma = 0.02$. The best predictor consistently gets worse and the worst consistently improves, both staying within 0.5 and 1 range. As before, $P(y = 1) = 0.75$. Figure 6.11 displays the accuracy change of the two predictors with respect to time. The red dots indicate the times that the corresponding predictor is selected and the blue dots indicate otherwise. Clearly, SFilter is able to detect the consistent change in the accuracies. We also note that the most frequently chosen predictors are the ones with relatively high average accuracies. There are occasional exploration attempts at the low accuracies, but the algorithm detects this and is able to recover quickly. As a result, the algorithm chooses any undesirable predictor(s) with very low frequency. Please note that there are times when neither predictors are selected. This corresponds to the situations where the remaining predictors are selected.

Additionally, we tested the ability of SFilter to track the true accuracy. We tested SFilter using 4 predictors with the rate of change $\sigma = 0.01$ on 1000 instances drawn from a true label distribution of $P(y = 1) = 0.8$. We used 5000 particles and we constrained SFilter to choose all 4 predictors per instance (only for this experiment) to monitor the true and the estimated accuracy of each predictor at every time step. Figure 6.12 demonstrates how the estimate of the marginal label distribution $\hat{P}(y = 1)$ improves over time. Figure 6.13 compares the true and the estimated accuracy of each predictor. The dotted blue line in each graph corresponds to the estimated expected accuracy while the solid red line corresponds to the true accuracy. The black line in each graph shows the result of using maximum likelihood estimation to infer the accuracy assuming accuracy is stationary. This inference

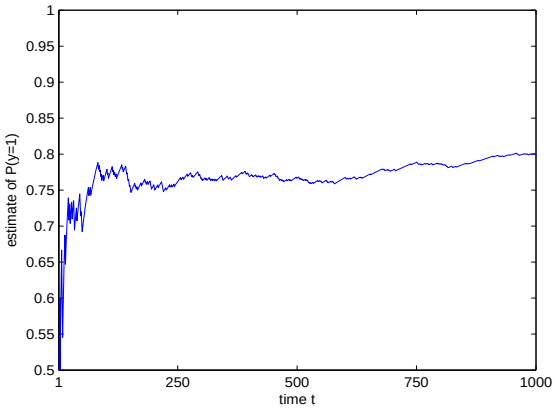


Figure 6.12: SFilter's estimate of the true marginal label distribution $P(y = 1) = 0.8$ improves over time. In fact, it converges to the true distribution with more samples.

technique proposed in [Donmez *et al.*, 2010c] (and will be described in detail in Chapter 7) shows very promising results on various test scenarios. However, it is designed to estimate stationary source accuracies. SFilter, on the other hand, manages to track the trends in the true accuracy quite well, sometimes with a lag. This is remarkable since neither the initial qualities nor the true label distribution are known in advance.

Next, we tested SFilter in terms of its effectiveness to create a high quality labeled data for training a classifier. We took 4 benchmark datasets from the UCI repository [Newman *et al.*, 1998]. We created $k = 10$ different simulated predictors with varying initial accuracies and different changing rates σ. We executed SFilter on each dataset with these predictors and integrated their output for a final labeling. Later, we trained a logistic regression classifier on this labeled set and tested the resulting classifier on a separate held-out data. The gold standard labels for the held-out set are known but used solely to test the classifier not to adjust the estimation model in any way. We compared the proposed approach with two strong baselines. One is majority voting (denoted as *Majority*) where we used all predictors per instance and assigned the majority vote as the integrated label. The other is the IEThresh technique introduced in Section 6.2. IEThresh selects a subset of predictors based on an interval estimation procedure and takes the majority vote of only the selected predictors. We report on the held-out set the accuracy of the classifier trained on the data labeled by each method. We report the classifier performance with respect to the number of queries. Hence, the corresponding size of the labeled examples differs for each method since each method uses different amount of labeling per instance, i.e. majority voting uses all annotators to label each instance. The total training size is fixed at 500 examples for

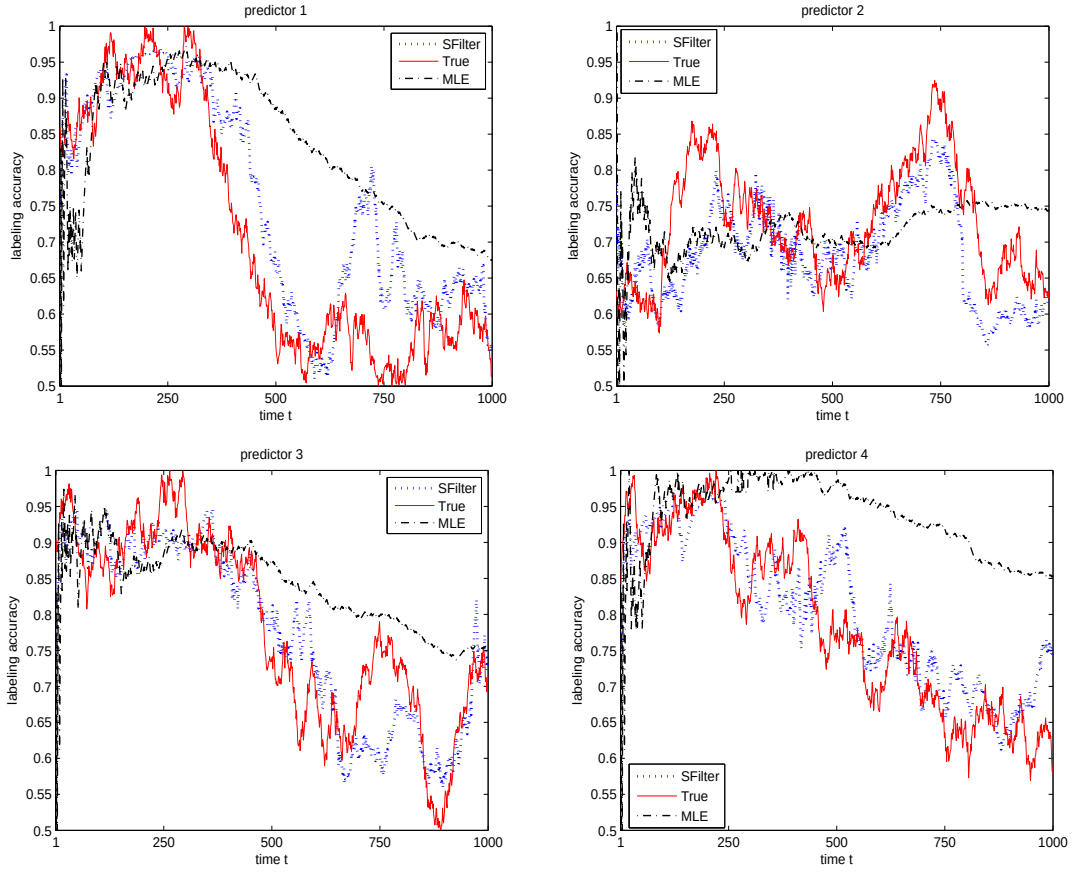


Figure 6.13: Shows how SFilter tracks the true predictor accuracy. Each graph corresponds to a different predictor. The solid red line indicates the true accuracy of the predictor whereas the dotted blue line shows the expected accuracy estimated by SFilter. This is the result of a single run, though highly typical. SFilter is able to track the tendency of the true accuracy quite well with occasional temporal lags.

each dataset. The other properties of the datasets are given in Table 6.8[4]. Figure 6.14 displays the performance of our algorithm SFilter and two baselines on four UCI datasets. The initial predictor accuracies range from as low as 0.53 to as high as 0.82. SFilter is significantly superior over the entire operating range; i.e. from small to large numbers of predictor queries. The differences are statistically significant based on a two-sided paired t-test ($p < 0.001$). The horizontal black line in each plot represents the test accuracy of a

[4]'ada' is a different version of the 'adult' dataset at UCI repository. This version is generated for the agnostic learning workshop at IJCNN '07, available at http://clopinet.com/isabelle/Projects/agnostic/index.html

Table 6.8: Properties of the UCI datasets. All are binary classification problems.

dataset	test size	dimension	+/- ratio
phoneme	3782	5	0.41
spambase	3221	57	0.65
ada	4562	48	0.33
image	1617	18	1.33

classifier trained with the gold standard labels. SFilter eventually reaches the gold standard level, suggesting that it generates a clean training data with minimal noise. Majority voting is the worst performer since it uses all predictors for every single instance; thus makes a larger number of labeling requests even for a small number of instances. IEThresh is the second best performer after SFilter. However, neither baselines take the time-varying accuracy into account and wastes labeling effort due to low quality predictors. Our method adapts to the change in predictor accuracies and filters the unreliable ones leading to a more effective estimation.

Finally, we are interested in testing the proposed framework on a different situation. It is often problematic to estimate the error rate (or accuracy) of a classifier in the absence of gold standard labels. In addition, consider classifiers that are re-trained (or modified) as additional instances become available. This situation might arise in a number of real-life scenarios. Consider spam filters. They need to be re-trained in an online fashion as new emails arrive. Web pages or blogs are other good examples of constantly growing databases. Any web page classifier needs to be modified to accomodate for the newly created web pages. Another example is autonomous agents that need to learn continuously from their environments and their performance will be changing with new discoveries of their surroundings. However, such learners might not always improve with additional data. The new stream of data might be noisy, generating confusion for the learners and causing a performance drop. Hence, capturing the time-varying quality of a learner while maintaining flexibility in terms of the direction of the change becomes crucial to adopt to this non-stationary environment.

To study this phenomenon, we evaluated the performance of SFilter on a face recognition dataset [Pham *et al.*, 2002] introduced in earlier chapters. We randomly divided the whole dataset into 3 mutually exclusive subsets. One contains 100 labeled images to train 5 classifiers. The other subset contains 500 unlabeled images to test them. The last subset is held out for validation. We trained 5 linear SVM classifiers as follows. For each classifier, we used a separate randomly chosen subset of 100 images as the initial training set. We systematically increased each training set size to 100, by adding random label noise so that the classifiers do not necessarily improve with more data. The average accuracy of classifiers are $\{0.70, 0.73, 0.78, 0.83, 0.84\}$. We executed SFilter, IEThresh and Majority voting on 500 test images to predict their labels. We then trained another classifier on the predicted labels

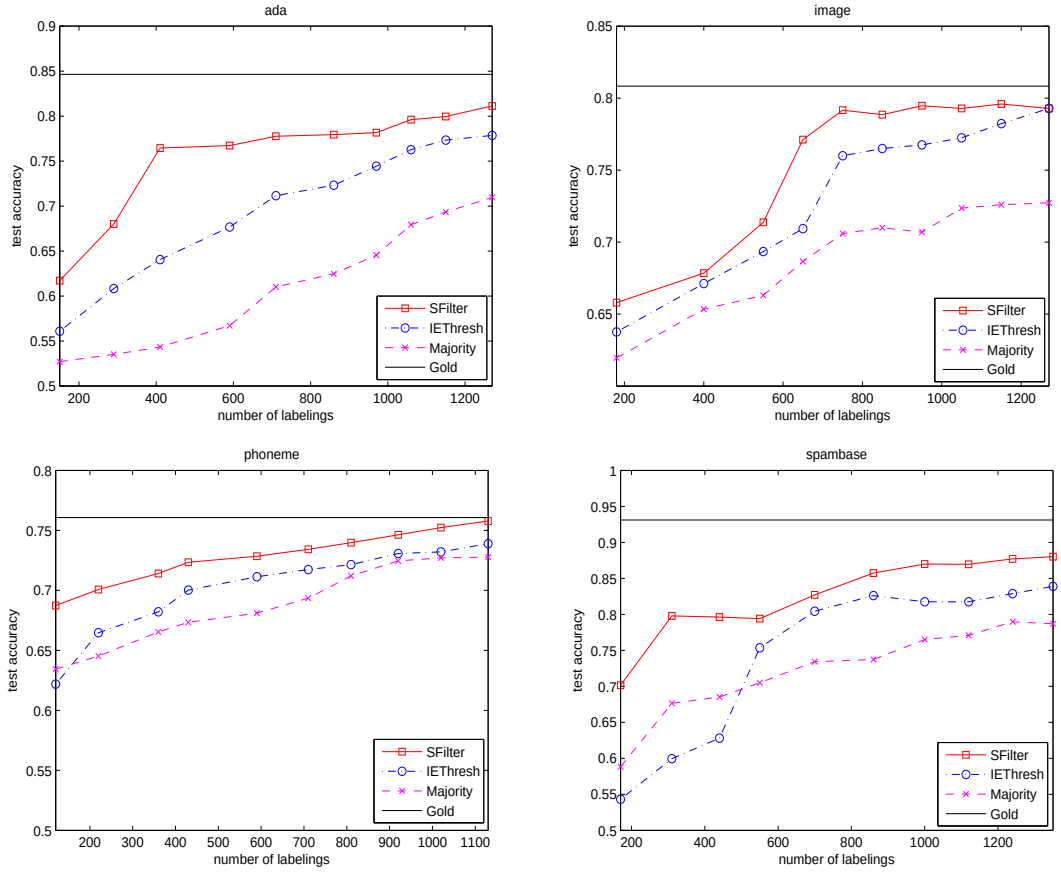


Figure 6.14: Measuring predicted label quality by training and testing a classifier on 4 UCI datasets by all three methods. The y-axis indicates the accuracy of the classifier on the test set, and the x-axis indicates the number of predictor queries. The horizontal line shows the performance of a classifier trained on the gold standard labels.

by each method. Figure 6.15 compares the performance of SFilter against IEThresh and Majority voting. SFilter significantly outperforms other baselines in this task and reaches a level of performance close to that of using gold standard labels. Again, gold labels are solely used to measure the performance of classifiers, and not for learning.

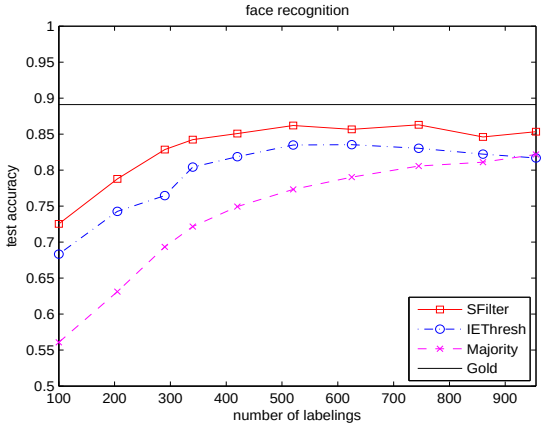


Figure 6.15: Results using 5 actual classifiers as predictors. The performance of each classifier varies over time. Their outputs on a test set are used to predict the true labels. A meta-classifier is trained using these labels and tested on a held-out set for performance evaluation. The predicted labels obtained by SFilter are significantly more effective in improving the data quality.

6.4 Chapter Conclusions

This chapter addresses the challenging task of jointly estimating predictor accuracy and predictor selection in stationary and non-stationary cases. In Section 6.2, we proposed IEThresh as an effective solution that naturally incorporates the exploration vs. exploitation tradeoff. The main advantage of IEThresh is to filter out less reliable predictors as early as possible, which boosts the performance. Our empirical evaluation indicates that IEThresh is more effective than the naive counterparts such as using all predictors or a random one, which were reported in the recent literature. Even under challenging conditions where the number of reliable predictors is low or some predictors are worse than random, IEThresh is capable of estimating the best predictor(s) through selective sampling and updating accuracy estimates.

In Section 6.3, we addressed the same problem in the non-stationary case that goes beyond consistent predictor quality. The proposed framework constantly monitors and estimates the expected behavior of each predictor while simultaneously choosing the best possible predictors for labeling each instance. The framework relies on a Hidden Markov Model (HMM) structure to represent the unknown sequence of changing accuracies and the corresponding observed predictor outputs. We adopted Bayesian particle filtering to make inference in such a dynamic system. Particle filters estimate the state distribution by sets of weighted samples and perform Bayes filter updates according to a sequential

sampling procedure. The key advantage of particle filters is their capacity to represent arbitrary probability densities whereas other Bayesian filters such as Kalman filters can only handle unimodal distributions and hence not well-suitable for our problem. Furthermore, we have proposed a variant of the basic filtering method to infer the expected accuracy of each predictor and use it as a guide to decide which ones to query at each time step. The proposed approach, *SFilter*, chooses the potentially good predictors based on their expected accuracies at each time step. In general high quality labelers are queried more frequently than the low-quality labelers. Moreover, it tracks reliably the true labeling accuracy over time.

The effectiveness of the proposed framework is demonstrated by thorough evaluation. For instance, we have shown that our data labeling process is robust to the choice of σ which governs the rate of the accuracy change. We have further shown that it is also robust to the distribution of the initial predictor accuracies. The algorithm is able to detect the most reliable predictors regardless of the reliability distribution. Additionally, our analysis demonstrates the ability of SFilter to generate high quality labels for a training data for which only multiple noisy annotations exist. This result is powerful in the sense that even though SFilter is provided with noisy labels generated from dynamically changing sources, it can label the data reliably. This is particularly useful for many practical applications where labeled data with minimal noise is essential to build reliable systems.

There are a number of potential future directions for expanding the research reported here. The rate of change σ of the state is assumed to be known (or at least its bound is known). To relax this assumption, one needs to estimate σ (which could potentially be different for different predictors). This is a harder problem but necessary to be solved to be applicable in a wider range of problems. To make the problem even more challenging, one can assume σ decreases over time, i.e. a decreasing function of time, if the predictors are assumed to learn with time. Another direction is to condition the probability of making a labeling mistake on the data instance, or at least the region of the instance space which contains the instance. Then, it is crucial to estimate this probability for a representative subset of the input space and generalize to the entire space. Another direction is to relax the assumption that the noise generation is uncorrelated. It is possible that the predictors make correlated errors as noted by [Sheng *et al.*, 2008]. This is a more challenging task since the correlation parameters need to be estimated together with the noise probabilities. Lastly, we assumed that the cost of labeling is the same for each predictor in this chapter. However, it is likely that more accurate predictors cost more than the less accurate ones. In such cases a decision-theoretic utility model would be central. These are interesting and challenging problems for future research in proactive learning to let it reach more practical application scenarios.

Chapter 7

Unsupervised Estimation of Classification and Regression Risks

7.1 Introduction

Thus far, we have studied joint estimation of predictor accuracies and predictor selection. In this chapter, we approach the problem from a different perspective. We formulate the problem as *risk estimation* of multiple predictors without using any labeled data and narrow our focus solely on risk estimation rather than predictor selection. The motivation for the risk estimation framework is an ambitious one. Generally, supervised machine learning algorithms aim to minimize an empirical risk estimate on the training data to obtain a predictor with a desirable generalization power. In the absence of labeled data, unsupervised risk estimation becomes crucial. If the risk can accurately be estimated using unlabeled data, then a classifier can be trained to minimize this estimate instead of a supervised alternative. With this goal in mind, we systematically tackle the problem by first dealing with a straightforward risk function, namely the 0-1 risk, and then extending this work to more general risk involving continuous loss functions such as log-loss. We propose a maximum likelihood estimator for both risks which we prove to be statistically consistent. Our empirical evaluation supports the theoretical claims and indicates the effectiveness of the proposed approach on both synthetic and real-world datasets.

In this chapter, we focus on the first framework and describe the methodology and the analysis in detail. In the following chapter, we move to the discussion of more general risk estimation and how it can be adopted to build classifiers with no labeled data. We start with introducing some preliminary concepts and notation before explaining the estimation framework in detail.

Assuming a joint distribution $p(x, y)$ and a loss function $L(y, \hat{y})$, a predictor $f : \mathcal{X} \to \mathcal{Y}$

is characterized by an expected loss or risk function

$$R(f) = \mathsf{E}_{p(x,y)}\{L(y, f(x))\}. \tag{7.1}$$

If L is chosen such that $L(y, \hat{y}) = I(y \neq \hat{y})$ where $I(A) = 1$ if A is true and 0 otherwise, and $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{1, \ldots, l\}$, the resulting risk is known as the 0-1 risk or simply the classification error rate

$$R(f) = P(f \text{ predicts the wrong class}). \tag{7.2}$$

In regression we may have $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, and $L(y, \hat{y}) = (y - \hat{y})^2$. The resulting risk is the mean squared error

$$R(f) = \mathsf{E}_{p(x,y)}(y - f(x))^2. \tag{7.3}$$

We consider the case where we are provided with k predictors $f_i : \mathcal{X} \to \mathcal{Y}$, $i = 1, \ldots, k$ ($k \geq 1$) whose risks are unknown. Our main goal is to estimate the risks $R(f_1), \ldots, R(f_k)$ without using any labeled data whatsoever. The estimation of $R(f_i)$ is rather based on an estimator $\hat{R}(f_i)$ that uses unlabeled data $x^{(1)}, \ldots, x^{(n)} \overset{\text{iid}}{\sim} p(x)$. A secondary task that we consider is obtaining effective schemes for combining k predictors $f_1, \ldots, f_k$ in a completely unsupervised manner to make predictions on the true label y.

In the absence of true labels, there is no ground truth that guides us in estimating the risks. Hence, consistent unsupervised risk estimation is a very challenging task. However, as we show in this chapter, if the marginal $p(y)$ is known it is possible in some cases to obtain a consistent estimator for the risks using only unlabeled data i.e.,

$$\lim_{n\to\infty} \hat{R}(f_i\,; x^{(1)}, \ldots, x^{(n)}) = R(f_i) \quad \text{with probability 1}, \quad i = 1, \ldots, k.$$

We explore the statistical consistency together with the asymptotic variance of the risk estimators and their dependency on n (amount of unlabeled data), k (number of predictors), and $R(f_1), \ldots, R(f_k)$ (risks). We also demonstrate that the proposed estimation technique works well in practice on both synthetic and real world data.

The assumption that $p(y)$ is known seems restrictive, but there are plenty of cases where it holds. Examples include medical diagnosis ($p(y)$ is the well known marginal disease frequency), handwriting recognition/OCR ($p(y)$ is the easily computable marginal frequencies of different English letters), regression model for life expectancy ($p(y)$ is the well known marginal life expectancy tables). In these and other examples $p(y)$ is obtained from extremely accurate histograms. In cases where it is not known, the consistency results may not hold. However, we have conducted an analysis to measure the effect of misspecifications of $p(y)$ on the estimation accuracy. Section 7.5 contains the details of the analysis, we note here that small perturbations do not affect the results significantly, especially when $p(y)$ is over-specified.

The collaborative nature of this diagnosis is especially useful for multiple predictors as the predictor ensemble $\{f_1, \ldots, f_k\}$ diagnoses itself. However, our framework is not restricted to a large k and works even for a single predictor with $k = 1$. It may further be extended to the case of active learning where classifiers are queried for specific data and hence not all predictors output an answer all the time.

7.2 Unsupervised Risk Estimation Framework

In addition to the preliminaries presented in Section 7.1, we further require that the predictors $f_1, \ldots, f_k$ are stochastic i.e. their prediction $\hat{y} = f_i(x)$ (conditioned on x) is a random variable. This is possible if the predictors are probabilistic i.e., f_i models a conditional distribution q_i and predicts y' with probability $q_i(y'|x)$.

As mentioned previously our goal is to estimate the risk associated with classification or regression models $f_1, \ldots, f_k$ based on unlabeled data $x^{(1)}, \ldots, x^{(n)} \overset{\text{iid}}{\sim} p(x)$. The testing marginal and conditional distributions $p(x), p(y|x)$ may differ from the distributions used at training time for the different predictors. In fact, each predictor may have been trained on a completely different training distribution, or may have been designed by hand with no training data whatsoever. We consider the predictors as black boxes and do not assume any knowledge of their modeling assumptions or training processes.

We define a parameter vector $\theta \in \Theta$ which characterizes the risks $R(f_1), \ldots, R(f_k)$ i.e. $R(f_j) = g_j(\theta)$ for some function $g_j : \Theta \to \mathbb{R}$, $j = 1, \ldots, k$. θ is our main parameter which is estimated from data by connecting it to the conditional probabilities $p_j(y'|y), j = 1, \ldots, k$ which are the probabilities that the j-predictor emits prediction y', conditioned on the true value being y. In other words, θ governs the distribution that generates the noisy outputs. We propose to estimate the risks using a plug-in estimate $\hat{R}(f_j) = g_j(\hat{\theta})$ where $\hat{\theta}$ is obtained by maximizing the observed likelihood over the unlabeled data. We thus obtain the following estimator

$$\hat{R}(f_j\,;\hat{y}^{(1)},\ldots,\hat{y}^{(n)}) = g_j(\hat{\theta}^{\text{mle}}(\hat{y}^{(1)},\ldots,\hat{y}^{(n)})), \quad j = 1,\ldots,k \tag{7.4}$$

$$\hat{\theta}^{\text{mle}}(\hat{y}^{(1)},\ldots,\hat{y}^{(n)}) = \arg\max_{\theta} \ell(\theta\,;\hat{y}^{(1)},\ldots,\hat{y}^{(n)}) \tag{7.5}$$

$$\begin{aligned}\ell(\theta\,;\hat{y}^{(1)},\ldots,\hat{y}^{(n)}) &= \sum_{i=1}^{n} \log p_\theta(\hat{y}_1^{(i)},\ldots,\hat{y}_k^{(i)}) \\ &= \sum_{i=1}^{n} \log \int_{\mathcal{Y}} p_\theta(\hat{y}_1^{(i)},\ldots,\hat{y}_k^{(i)}|y^{(i)}) p(y^{(i)})\, d\mu(y^{(i)})\end{aligned} \tag{7.6}$$

where $\hat{y}^{(i)} \overset{\text{def}}{=} (\hat{y}_1^{(i)},\ldots,\hat{y}_k^{(i)})$ and $\hat{y}_j^{(i)} \overset{\text{def}}{=} f_j(x^{(i)})$.

The integral in (7.6) is over the unobserved label $y^{(i)}$ associated with $x^{(i)}$. It should be a continuous integral $\int_{y^{(i)}=-\infty}^{\infty}$ for regression and a finite summation $\sum_{y^{(i)}=1}^{l}$ for classification. For notational simplicity we maintain the integral sign for both cases with the understanding that it is over a continuous or discrete measure μ, depending on the topology of $\mathcal{Y}$. Note that (7.6) and its maximizer are computable without any labeled data. All that is required are the classifiers (as black boxes), unlabeled data $x^{(1)}, \ldots, x^{(n)}$, and the marginal label distribution $p(y)$.

Besides being a diagnostic tool for the predictor accuracy, $\hat{\theta}^{\text{mle}}$ can be used to effectively aggregate $f_1, \ldots, f_j$ to predict the label of a new example x^{new}, assuming the predictors' outputs $f_1(x^{\text{new}}), \ldots, f_j(x^{\text{new}})$ are conditionally independent given the true label y:

$$\begin{aligned} \hat{y}^{\text{new}} &= \arg\max_{y \in \mathcal{Y}} p_{\hat{\theta}^{\text{mle}}}(y \mid f_1(x^{\text{new}}), \ldots, f_k(x^{\text{new}})) \\ &= \arg\max_{y \in \mathcal{Y}} p(y) \prod_{j=1}^{k} p_{\hat{\theta}^{\text{mle}}_j}(f_j(x^{\text{new}}) \mid y). \end{aligned} \tag{7.7}$$

As a result, our framework may be used to combine existing classifiers or regression models in a completely unsupervised manner.

There are three important research questions concerning the above framework. First, what are the statistical properties of $\hat{\theta}^{\text{mle}}$ and $\hat{R}$ (consistency, asymptotic variance)? Second, how can we efficiently solve the maximization problem (7.5)? And third, how does the framework work in practice? We address these three questions in Sections 7.3, 7.4, 7.5 respectively, We devote the rest of the current section to examine some important special cases of (7.5)-(7.6) and consider some generalizations in the next section.

7.2.1 Non-Collaborative Estimation of the Risks

In the non-collaborative case we estimate the risk of each one of the predictors $f_1, \ldots, f_k$ separately. This reduces the problem to that of estimating the risk of a single predictor, which is repeated k times for each one of the predictors. We thus assume in this subsection the framework (7.4)-(7.6) with $k = 1$ with no loss of generality. For simplicity we denote the single predictor by f rather than f_1 and denote $g = g_1$ and $\hat{y}^{(i)} = \hat{y}_1^{(i)}$. The corresponding simplified expressions are

$$\hat{R}(f\,; \hat{y}^{(1)}, \ldots, \hat{y}^{(n)}) = g(\hat{\theta}^{\text{mle}}(\hat{y}^{(1)}, \ldots, \hat{y}^{(n)})) \tag{7.8}$$

$$\hat{\theta}^{\text{mle}}(\hat{y}^{(1)}, \ldots, \hat{y}^{(n)}) = \arg\max_{\theta} \sum_{i=1}^{n} \log \int_{\mathcal{Y}} p_\theta(\hat{y}^{(i)} | y^{(i)}) p(y^{(i)})\, d\mu(y^{(i)}) \tag{7.9}$$

where $\hat{y}^{(i)} = f(x^{(i)})$.

We consider below several important special cases.

Classification

Assuming l labels $\mathcal{Y} = \{1, \dots, l\}$, the classifier f defines a multivariate Bernoulli distribution $p_\theta(\hat{y}|y)$ mapping the true label y to $\hat{y}$

$$p_\theta(\hat{y}|y) = \theta_{\hat{y},y}. \tag{7.10}$$

where θ is the stochastic confusion matrix or noise model corresponding to the classifier f. In this case, the relationship between the risk $R(f)$ and the parameter θ is

$$R(f) = 1 - \sum_{y \in \mathcal{Y}} \theta_{yy}\, p(y). \tag{7.11}$$

Equations (7.10)-(7.11) may be simplified by assuming a symmetric error distribution [Cover and Thomas, 2005], meaning that either the true label is output with probability θ, or any incorrect label is output with equal probability $\frac{1-\theta}{l-1}$ when there are l labels.

$$p_\theta(\hat{y}|y) = \theta^{I(\hat{y}=y)} \left(\frac{1-\theta}{l-1}\right)^{I(\hat{y}\neq y)} \tag{7.12}$$

$$R(f) = 1 - \theta \tag{7.13}$$

where I is the indicator function and $\theta \in [0, 1]$ is a scalar corresponding to the classifier accuracy. Estimating θ by maximizing (7.9), with (7.10) or (7.12) substituting p_θ completes the risk estimation task.

In the simple binary case $l = 2, \mathcal{Y} = \{1, 2\}$ with the symmetric noise model (7.12) the loglikelihood

$$\ell(\theta) = \sum_{i=1}^{n} \log \sum_{y^{(i)}=1}^{2} \theta^{I(\hat{y}^{(i)}=y^{(i)})} (1-\theta)^{I(\hat{y}^{(i)}\neq y^{(i)})} p(y^{(i)}). \tag{7.14}$$

may be shown to have the following closed form maximizer

$$\hat{\theta}^{\text{mle}} = \frac{p(y=1) - m/n}{2p(y=1) - 1}. \tag{7.15}$$

where $m \stackrel{\text{def}}{=} |\{i \in \{1, \dots, n\} : \hat{y}^{(i)} = 2\}|$. The estimator (7.15) works well in practice and is shown to be a consistent estimator in the next section (i.e., it converges to the true parameter value when $p(y)$ is non-uniform and $\theta > 0.5$). In cases where the symmetric noise model (7.12) does not hold, using (7.15) to estimate the classification risk may be misleading. For example, in some cases (7.15) may be negative. In these cases, using the more general model (7.10) instead of (7.12) should provide more accurate results. We discuss this further from theoretical and experimental perspectives in Sections 7.3, and 7.5 respectively.

Regression

Assuming a regression equation

$$y = ax + \epsilon, \qquad \epsilon \sim N(0, \tau^2)$$

and an estimated regression model or predictor $\hat{y} = a'x$ we have

$$\hat{y} = a'x = a'a^{-1}(y - \epsilon) = \theta y - \theta\epsilon$$

where $\theta = a'a^{-1}$. Thus, in the regression case the distribution $p_\theta(\hat{y}|y)$ and the relationship between the risk and the parameter $R(f) = g(\theta)$ are

$$p_\theta(\hat{y}|y) = (2\pi\theta^2\tau^2)^{-1/2} \exp\left(-\frac{(\hat{y} - \theta y)^2}{2\theta^2\tau^2}\right) \tag{7.16}$$

$$R(f|y) = \mathsf{bias}^2(f) + \mathsf{Var}(f) = (1-\theta)^2 y^2 + \theta^2\tau^2 \tag{7.17}$$

$$R(f) = \theta^2\tau^2 + (1-\theta)^2 \mathsf{E}_{p(y)}(y^2). \tag{7.18}$$

Note that we consider regression as a stochastic estimator in that it predicts the model to be $y = a'x + \epsilon$ or $y|x \sim N(a'x, \tau^2)$.

Assuming $p(y) = N(\mu_y, \sigma_y^2)$, we have

$$p_\theta(\hat{y}^{(i)}) = \int_{\mathbb{R}} p_\theta(\hat{y}^{(i)}|y)dy = (2\pi\theta^2\tau^2 2\pi\sigma_y^2)^{-1/2} \int_{\mathbb{R}} \exp\left(-\frac{(\hat{y} - \theta y)^2}{2\theta^2\tau^2} - \frac{(y - \mu_y)^2}{2\sigma_y^2}\right) dy \tag{7.19}$$

$$= \frac{1}{\theta\sqrt{2\pi(\tau^2 + \sigma_y^2)}} \exp\left(\frac{(\hat{y}^{(i)})^2}{2\theta^2\tau^2}\left(\frac{\sigma_y^2}{\sigma_y^2 + \tau^2} - 1\right) + \frac{\mu_y^2}{2\sigma_y^2}\left(\frac{\tau^2}{\sigma_y^2 + \tau^2} - 1\right) + \frac{\hat{y}^{(i)}\mu_y}{\theta\left(\tau^2 + \sigma_y^2\right)}\right) \tag{7.20}$$

where we used the following lemma in the last equation.

Lemma 1 (e.g., [Papoulis, 1984])**.**

$$\int_{-\infty}^{\infty} A\, e^{-Bx^2 + Cx + D}\, dx = A\sqrt{\frac{\pi}{B}} \exp\left(C^2/4B + D\right) \tag{7.21}$$

where A, B, C, D are constants that do not depend on x.

In this case the loglikelihood simplifies to

$$\ell(\theta) = -n\log\left(\theta\sqrt{2\pi(\tau^2 + \sigma_y^2)}\right) - \left(\frac{\sum_{i=1}^n (\hat{y}^{(i)})^2}{2(\tau^2 + \sigma_y^2)}\right)\frac{1}{\theta^2} + \left(\frac{\mu_y \sum_{i=1}^n \hat{y}^{(i)}}{\tau^2 + \sigma_y^2}\right)\frac{1}{\theta} - n\frac{\mu_y^2}{2(\sigma_y^2 + \tau^2)} \tag{7.22}$$

which can be shown to have the following closed form maximizer

$$\hat{\theta}^{\text{mle}} = -\frac{\mu_y \sum_{i=1}^n \hat{y}^{(i)}}{2n(\tau^2+\sigma_y^2)} \pm \sqrt{\frac{\left(\mu_y \sum_{i=1}^n \hat{y}^{(i)}\right)^2}{4n^2(\tau^2+\sigma_y^2)^2} + \frac{\sum_{i=1}^n (\hat{y}^{(i)})^2}{n(\tau^2+\sigma_y^2)}} \tag{7.23}$$

where the two roots correspond to the two cases where $\theta = a'/a > 0$ and $\theta = a'/a < 0$.

A straightforward generalization to the multivariate case is $y = Ax + \epsilon$ where y, x are vectors, A is the matrix of regression parameters, and $\epsilon \sim N(0, \sigma_y^2 I)$. An estimated model using a matrix B instead of A can be written as $\hat{y} = Bx = \Theta(y - \epsilon)$ with $\Theta = BA^{-1}$ which leads to $\hat{y}|y \sim N(\Theta y, \sigma_y^2 \Theta\Theta^\top)$. The task remains estimating the regression risk by estimating Θ from observed data $\hat{y}^{(1)}, \ldots, \hat{y}^{(n)}$.

Noisy Gaussian Channel

In this case our predictor f corresponds to a noisy channel taking a real valued signal y as input and yielding its noisy version $\hat{y}$ as output. The aim is to estimate the mean squared error or noise level $R(f) = \mathsf{E}\,\|y - \hat{y}\|^2$. In this case the distribution $p_\theta(\hat{y}|y)$ and the relationship between the risk and the parameter $R(f) = g(\theta)$ are

$$p_\theta(\hat{y}|y) = (2\pi\theta^2)^{-1/2} \exp\left(-\frac{(\hat{y}-y)^2}{2\theta^2}\right) \tag{7.24}$$

$$R(f|y) = E_{p_\theta(\hat{y}|y)}\|y - \hat{y}\|^2 = Var_{p_\theta(\hat{y}|y)}(\hat{y}) = \theta^2 \tag{7.25}$$

$$R(f) = \theta^2 \mathsf{E}_{p(y)}(y). \tag{7.26}$$

The loglikelihood and other details in this case are straightforward variations on the linear regression case described above. We therefore concentrate in this chapter on the classification and linear regression cases.

As mentioned above, in both classification and regression, estimating the risks for $k \geq 2$ predictors rather than a single one may proceed by repeating the optimization process described above for each predictor separately. That is $\hat{R}(f_j) = g_j(\hat{\theta}_j^{\text{mle}})$ where $\hat{\theta}_1^{\text{mle}}, \ldots, \hat{\theta}_k^{\text{mle}}$ are estimated by maximizing k different loglikelihood functions. In some cases the convergence rate to the true risks can be accelerated by jointly estimating the risks $R(f_1), \ldots, R(f_k)$ in a collaborative fashion. Such collaborative estimation is possible under some assumptions on the statistical dependency between the noise processes defining the k predictors. We describe below such an assumption followed by a description of more general cases.

7.2.2 Collaborative Estimation of the Risks: Conditionally Independent Predictors

Previously, we have analyzed estimating the risks of k predictors through maximizing (7.6) for each predictor separately. If the predictors are known to be conditionally independent given the true label i.e. $p_\theta(\hat{y}_1, \ldots, \hat{y}_k|y) = \prod_j p_{\theta_j}(\hat{y}_j|y)$ the loglikelihood (7.6) simplifies to

$$\ell(\theta) = \sum_{i=1}^{n} \log \int_{\mathcal{Y}} \prod_{j=1}^{k} p_{\theta_j}(\hat{y}_j^{(i)}|y^{(i)}) p(y^{(i)})\, d\mu(y^{(i)}), \quad \text{where} \quad \hat{y}_j^{(i)} = f_j(x^{(i)}) \tag{7.27}$$

and p_{θ_j} above is (7.10) or (7.12) for classification and (7.16) for regression. Maximizing the loglikelihood (7.27) jointly over $\theta_1, \ldots, \theta_k$ results in estimators $\hat{R}(f_1), \ldots, \hat{R}(f_k)$ that converge to the true value faster than the non-collaborative MLE (7.9) (See Section 7.5 for more details). Equation (7.27) does not have a closed form solution neither in classification nor in regression cases; hence, it requires the use of iterative optimization techniques.

We note that the conditional independence assumption may not hold in every case. We propose a more general formulation for conditionally dependent predictors in the following section. However, we argue that the conditional independence of the predictors is a much weaker condition than the independence of the predictors which is very unlikely to hold. In our case, each predictor f_j has its own stochastic noise operator $T_j(r, s) = p(\hat{y} = r|y = s)$ (regression) or matrix $[T_j]_{rs} = p_j(\hat{y} = r|y = s)$ (classification) where $T_1, \ldots, T_k$ may be arbitrarily specified. In particular, some predictors may be similar e.g., $T_i \approx T_j$, and some may be different e.g., $T_i \not\approx T_j$. The conditional independence assumption that we make in this subsection is that conditioned on the latent label y the predictions of the predictors proceed stochastically according to $T_1, \ldots, T_k$ in an independent manner.

Figure 7.1 displays the loglikelihood functions $\ell(\theta)$ for three different dataset sizes $n = 100, 250, 500$. As the size n of the unlabeled data grows the curves become steeper and $\hat{\theta}_n^{\text{mle}}$ approach θ^{true}. Figure 7.2 displays a similar figure for $k = 1$ in the case of regression.

In the case of regression (7.27) involves an integral over a product of $k + 1$ Gaussians,

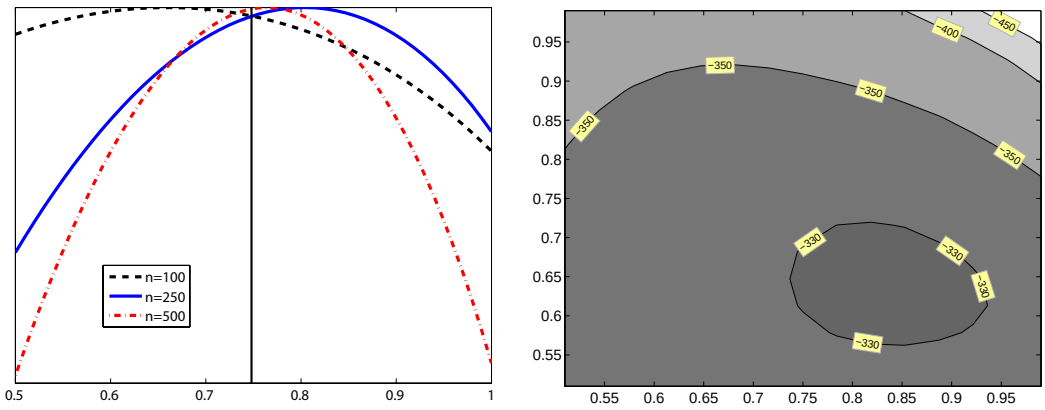


Figure 7.1: A plot of the loglikelihood functions $\ell(\theta)$ in the case of classification for $k = 1$ (left, $\theta^{\text{true}} = 0.75$) and $k = 2$ (right, $\theta^{\text{true}} = (0.8, 0.6)^\top$). The loglikelihood was constructed based on random samples of unlabeled data with sizes $n = 100, 250, 500$ (left) and $n = 250$ (right) and $p(y = 1) = 0.75$. In the left panel the y values of the curves were scaled so their maxima would be aligned. For $k = 1$ the estimators $\hat{\theta}^{\text{mle}}$ (and their errors $|\hat{\theta}^{\text{mle}} - 0.75|$) for $n = 100, 250, 500$ are 0.6633 (0.0867), 0.8061 (0.0561), 0.765 (0.0153). As additional unlabeled examples are added the loglikelihood curves become steeper and their maximizers become more accurate and closer to θ^{true}.

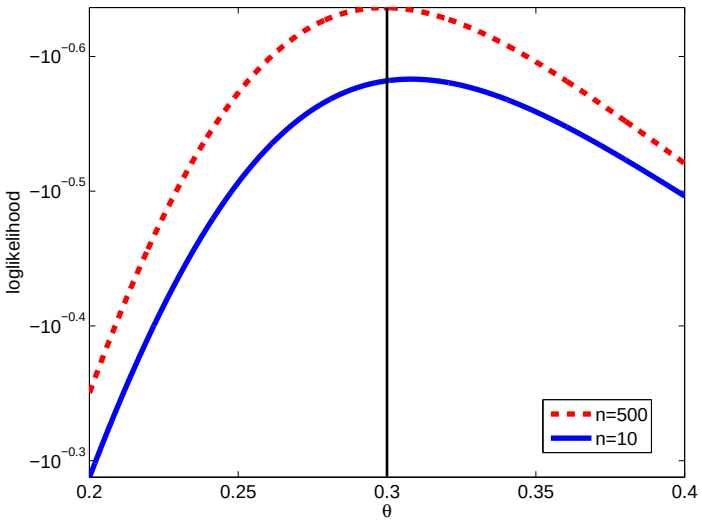


Figure 7.2: A plot of the loglikelihood function $\ell(\theta)$ in the case of regression for $k = 1$ with $\theta^{\text{true}} = 0.3$, $\tau = 1$, $\mu_y = 0$ and $\sigma_y = 0.2$. As additional unlabeled examples are added the loglikelihood curve become steeper and their maximizers get closer to the true parameter θ^{true} resulting in a more accurate risk estimate.

assuming that $y \sim N(\mu_y, \sigma_y^2)$. In this case the integral in (7.27) simplifies to

$$
\begin{aligned}
& p_\theta(\hat{y}_1^{(i)}, \ldots, \hat{y}_k^{(i)}) = \\
& \int_{-\infty}^{\infty} \left(\prod_{j=1}^{k} \frac{1}{\theta_j \tau \sqrt{2\pi}} e^{-\left(\hat{y}_j^{(i)} - \theta_j y^{(i)}\right)^2 \Big/ 2\theta_j^2 \tau^2} \right) \frac{1}{\sigma_y \sqrt{2\pi}} e^{-\left(y^{(i)} - \mu_y\right)^2 \Big/ 2\sigma_y^2} \, dy^{(i)} \\
= & \frac{1}{\tau^k (\sqrt{2\pi})^{k+1} \sigma_y \prod_{j=1}^{k} \theta_j} \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2} \left(\left(\frac{y^{(i)} - \mu_y}{\sigma_y} \right)^2 + \sum_{j=1}^{k} \left(\frac{y^{(i)}}{\tau} - \frac{\hat{y}_j^{(i)}}{\tau \theta_j} \right)^2 \right) \right] dy^{(i)} \\
= & \frac{\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2} \left(\frac{1}{\sigma_y^2} + \frac{k}{\tau^2} \right) (y^{(i)})^2 + \left(\frac{\mu_y}{\sigma_y^2} + \sum_{j=1}^{k} \frac{\hat{y}_j^{(i)}}{\tau^2 \theta_j} \right) y^{(i)} - \frac{1}{2} \left(\frac{\mu_y^2}{\sigma_y^2} + \sum_{j=1}^{k} \frac{(\hat{y}_j^{(i)})^2}{\tau^2 \theta_j^2} \right) \right)}{\tau^k (\sqrt{2\pi})^{k+1} \sigma_y \prod_{j=1}^{k} \theta_j} \\
= & \frac{\sqrt{\pi} \left[\frac{1}{2} \left(\frac{1}{\sigma_y^2} + \frac{k}{\tau^2} \right) \right]^{-1/2}}{\tau^k (\sqrt{2\pi})^{k+1} \sigma_y \prod_{j=1}^{k} \theta_j} \exp\left(\frac{\left(\frac{\mu_y}{\sigma_y^2} + \sum_{j=1}^{k} \frac{\hat{y}_j^{(i)}}{\tau^2 \theta_j} \right)^2}{2 \left(\frac{1}{\sigma_y^2} + \frac{k}{\tau^2} \right)} - \sum_{j=1}^{k} \frac{(\hat{y}_j^{(i)})^2}{2\tau^2 \theta_j^2} - \frac{\mu_y^2}{2\sigma_y^2} \right)
\end{aligned}
\tag{7.28}
$$

where the last equation was obtained using Lemma 1 concerning Gaussian integrals.

7.2.3 Collaborative Estimation of the Risks: Conditionally Correlated Predictors

In some cases the conditional independence assumption made in the previous subsection does not hold and the factorization (7.27) is violated. In this section, we discuss how to relax this assumption in the classification case. A similar approach may also be used for regression. We omit the details here due to notational clarity.

There are several ways to relax the conditional independence assumption. Most popular, perhaps, is the mechanism of hierarchical loglinear models for categorical data [Bishop *et al.*, 1975]. For example, generalizing our conditional independence assumption to second-order interaction log-linear models we have

$$
\log p(\hat{y}_1, \ldots, \hat{y}_k | y) = \alpha_y + \sum_{i=1}^{l} \beta_{i, \hat{y}_i, y} + \sum_{i<j} \gamma_{i,j,\hat{y}_i,\hat{y}_j,y} \tag{7.29}
$$

where the following ANOVA-type parameter constraints are needed [Bishop *et al.*, 1975]

$$
\begin{aligned}
0 &= \sum_{\hat{y}_i} \beta_{i,\hat{y}_i,y} \quad \forall i, y \\
0 &= \sum_{\hat{y}_i} \gamma_{i,j,\hat{y}_i,\hat{y}_j,y} = \sum_{\hat{y}_j} \gamma_{i,j,\hat{y}_i,\hat{y}_j,y} \quad \forall i, j, y.
\end{aligned}
\tag{7.30}
$$

The β parameters in (7.29) correspond to the first order interaction between the variables $\hat{y}_1, \ldots, \hat{y}_k$, conditioned on y. They correspond to the θ_i in the independent formulation (7.10)-(7.12). The γ parameters capture second order interactions which do not appear in the conditionally independent case. Indeed, setting $\gamma_{i,j,\hat{y}_i,\hat{y}_j,y} = 0$ resumes the independent models (7.10)-(7.12).

In the case of classification, the number of degrees of freedom or free unconstrained parameters in (7.29) depends on whether the number of classes is 2 or more and what additional assumptions exist on β and γ. For example, assuming that the probability of f_i, f_j making an error depends on the true class y but not on the predicted classes $\hat{y}_i, \hat{y}_j$ results in a $k + k^2$ parameters. Relaxing that assumption but assuming binary classification results in $2k + 4k^2$ parameters. The estimation and aggregation techniques described in Section 7.2.2 work as before with a slight modification of replacing (7.10)-(7.12) with variations based on (7.29) and enforcing the constraints (7.30).

Equation (7.29) captures two-way interactions but cannot model higher order interactions. However, higher order interaction models are straightforward generalizations of (7.29) culminating in the full loglinear model which does not make any assumption on the statistical dependency of the noise operators $T_1, \ldots, T_k$. However, as we weaken the assumptions underlying the loglinear models and add higher order interactions the number of parameters increases adding to the difficulty in estimating the risks $R(f_1), \ldots, R(f_k)$.

In our experiments on real world data (see Section 7.5), it is often the case that maximizing the loglikelihood under the conditionally independent assumption (7.27) provides adequate accuracy and there is no need for the more general (7.29)-(7.30). Nevertheless, we include here the case of loglinear models as it may be necessary in some situations.

7.2.4 Extensions to Missing Values

Occasionally, some predictors are unable to provide their output over specific data points. That is assuming a dataset $x^{(1)}, \ldots, x^{(n)}$ each predictor may provide output on an arbitrary subset of the data points $\{f_j(x^{(i)}) : i \in S_j\}$, where $S_j \subset \{1, \ldots, n\}$, $j = 1, \ldots, k$.

Although the reasons may vary, i.e. different parts of the unlabeled data are unavailable to all preditors but some due to privacy reasons, computational cost, expertise area, etc., this situation is commonly referred as a missing value situation. We proceed in this case by defining indicators β_{ji} denoting whether predictor j is available to emit $f_j(x^{(i)})$. The risk estimation proceeds as before with the observed likelihood modified to account for the

missing values:

$$\begin{aligned}\hat{\theta}_n^{\text{mle}} &= \arg\max_{\theta} \ell(\theta) \\ \ell(\theta) &= \sum_{i=1}^{n} \log \sum_{r:\beta_{ri}=0} \int_{\mathcal{Y}} p_\theta(\hat{y}_1^{(i)}, \ldots, \hat{y}_k^{(i)})\, d\mu(\hat{y}_r^{(i)}) \\ &= \sum_{i=1}^{n} \log \sum_{r:\beta_{ri}=0} \iint_{\mathcal{Y}^2} p_\theta(\hat{y}_1^{(i)}, \ldots, \hat{y}_k^{(i)} | y^{(i)}) p(y^{(i)})\, d\mu(\hat{y}_r^{(i)}) d\mu(y^{(i)})\end{aligned} \tag{7.31}$$

where p_θ may be further simplified using the non-collaborative approach, or using the collaborative approach with conditional independence or loglinear model assumptions. The different variations concerning missing values and non-collaborative or collaborative estimation with conditionally independent or correlated noise processes can all be combined in different ways to provide the appropriate likelihood function. This provides substantial modeling flexibility.

7.3 Statistical Analysis of $\hat{\theta}_n^{\text{mle}}$ and $\hat{R}(f_j)$

In this section we consider the statistical behavior of the estimator $\hat{\theta}_n^{\text{mle}}$ defined in (7.5) and the risk estimator $\hat{R}(f_j) = g_j(\hat{\theta}^{\text{mle}})$ defined in (7.4). The analysis is conducted under the assumption that the vectors of observed predictors outputs $\hat{y}^{(i)} = (\hat{y}_1^{(i)}, \ldots, \hat{y}_k^{(i)})$ are iid samples from the distribution

$$p_\theta(\hat{y}) = p_\theta(\hat{y}_1, \ldots, \hat{y}_k) = \int_{\mathcal{Y}} p_\theta(\hat{y}_1, \ldots, \hat{y}_k | y) p(y)\, d\mu(y).$$

7.3.1 Consistency

We start by investigating whether estimator $\hat{\theta}^{\text{mle}}$ in (7.5) converges to the true parameter value. More formally, strong consistency of the estimator $\hat{\theta}_n^{\text{mle}} = \hat{\theta}(\hat{y}^{(1)}, \ldots, \hat{y}^{(n)})$, $\hat{y}^{(1)}, \ldots, \hat{y}^{(n)} \overset{\text{iid}}{\sim} p_{\theta_0}$ is defined as strong convergence of the estimator to θ_0 as $n \to \infty$ [Ferguson, 1996]

$$\lim_{n\to\infty} \hat{\theta}_n^{\text{mle}}(\hat{y}^{(1)}, \ldots, \hat{y}^{(n)}) = \theta_0 \text{ with probability } 1. \tag{7.32}$$

In other words as the number of samples n grows, the estimator will surely converge to the true parameter θ_0 governing the data generation process.

Assuming that the risks $R(f_j) = g_j(\theta)$ are defined using continuous functions g_j, strong consistency of $\hat{\theta}^{\text{mle}}$ implies strong convergence of $\hat{R}(f_j)$ to $R(f_j)$. This is due to the

fact that continuity preserves limits. Indeed, as the g_j functions are continuous in both the classification {(7.11) and (7.13)} and regression {(7.18) and (7.26) } cases, strong consistency of the risk estimators $\hat{R}(f_j)$ reduces to strong consistency of the estimators $\hat{\theta}^{\text{mle}}$.

It is well known that the maximum likelihood estimator is often strongly consistent. Consider, for example, the following theorem.

Proposition 1 (e.g., [Ferguson, 1996])**.** *Let $\hat{y}^{(1)}, \ldots, \hat{y}^{(n)} \overset{\text{iid}}{\sim} p_{\theta_0}$, $\theta_0 \in \Theta$. If the following conditions hold*

1. *Θ is compact* *(compactness)*
2. *$p_\theta(\hat{y})$ is upper semi-continuous in θ for all $\hat{y}$* *(continuity)*
3. *There exists a function $K(\hat{y})$ such that $\mathsf{E}_{p_{\theta_0}}|K(\hat{y})| < \infty$ and $\log p_\theta(\hat{y}) - \log p_{\theta_0}(\hat{y}) \leq K(\hat{y}) \quad \forall \hat{y} \quad \forall \theta$* *(boundedness)*
4. *For all θ and sufficiently small $\rho > 0$, $\sup_{|\theta' - \theta| < \rho} p_{\theta'}(\hat{y})$ is measurable in $\hat{y}$* *(measurability)*
5. *$p_\theta \equiv p_{\theta_0} \Rightarrow \theta = \theta_0$* *(identifiability)*

then the maximum likelihood estimator is strongly consistent i.e., $\hat{\theta}^{mle} \to \theta_0$ as $n \to \infty$ with probability 1.

Note that $p_\theta(\hat{y})$ in the proposition above corresponds to $\int_{\mathcal{Y}} p_\theta(\hat{y}|y)p(y)\, d\mu(y)$ in our framework. That is the MLE operates on the observed data or predictor output $\hat{y}^{(1)}, \ldots, \hat{y}^{(n)}$ that is sampled iid from the distribution $p_{\theta_0}(\hat{y}) = \int_{\mathcal{Y}} p_{\theta_0}(\hat{y}|y)p(y)\, d\mu(y)$.

Of the five conditions above, the last condition of identifiability is the only one that is truly problematic. The first condition of compactness is trivially satisfied in the case of classification. In the case of regression it is satisfied assuming that the regression parameter and model parameter are finite and $a \neq 0$ as the estimator $\hat{\theta}^{\text{mle}}$ will eventually lie in a compact set. The second condition of continuity is trivially satisfied in both classification and regression as the function $\int_{\mathcal{Y}} p_\theta(\hat{y}|y)p(y)\, d\mu(y)$ is continuous in θ once $\hat{y}$ is fixed. The third condition is trivially satisfied for classification (finite valued y). In the case of regression conditions 1,2 (compactness and semi-continuity) allow the substitution of the quantifier $\forall\theta$ with a particular value $\theta' \in \Theta$ chosen such that the logarithm difference is maximum. Then, this maximum value with respect to the worst case θ' may be used as the bound K. The expectation of the difference of the log terms converges to the KL divergence, which is never ∞ for Gaussian distributions or its derivatives. The fourth condition of measurability follows as p_θ is specified in terms of compositions, summations, multiplications, and point-wise limits of well-known measurable functions.

The fifth condition of identifiability states that if $p_\theta(\hat{y})$ and $p_{\theta_0}(\hat{y})$ are identical as functions i.e., they are identical for every value of $\hat{y}$, then necessarily $\theta = \theta_0$. This condition does not hold in general and needs to be verified in each one of the special cases.

We start with establishing consistency in the case of classification where we rely on a

symmetric noise model (7.12). The non-symmetric case (7.10) is more complicated and is treated afterwards. We conclude the consistency discussion with an examination of the regression case.

Consistency of Classification Risk Estimation

Proposition 2. *Let $f_1, \ldots, f_k$ be classifiers $f_i : \mathcal{X} \to \mathcal{Y}$, $|\mathcal{Y}| = l$, with conditionally independent noise processes described by* (7.12)*. If the classifiers are weak learners i.e., $1/l < R(f_i) < 1$ and $p(y)$ is not uniform the unsupervised collaborative diagnosis model is identifiable.*

Corollary 1. *Let $f_1, \ldots, f_k$ be classifiers $f_i : \mathcal{X} \to \mathcal{Y}$ with $|\mathcal{Y}| = l$ and noise processes described by* (7.12)*. If the classifiers are weak learners i.e., $1/l < R(f_i) < 1$, and $p(y)$ is not uniform the unsupervised non-collaborative diagnosis model is identifiable.*

Proof: Proving identifiability in the non-collaborative case proceeds by invoking Proposition 2 (whose proof is given below) with $k = 1$ separately for each classifier. The conditional independence assumption in Proposition 2 becomes redundant in this case of a single classifier, resulting in identifiability of $p_{\theta_j}(\hat{y}_j)$ for each $j = 1, \ldots, k$ ∎

Corollary 2. *Under the assumptions of Proposition 2 or Corollary 1 the unsupervised maximum likelihood estimator is consistent i.e.,*

$$P\left(\lim_{n\to\infty} \hat{\theta}_n^{mle}(\hat{y}^{(1)}, \ldots, y^{(n)}) = (\theta_1^{true}, \ldots, \theta_k^{true})\right) = 1.$$

Consequentially, assuming that $R(f_j) = g_j(\theta), j = 1, \ldots, k$ with continuous g_j we also have

$$P\left(\lim_{n\to\infty} \hat{R}(f_j\,; y^{(1)}, \ldots, y^{(n)}) = R(f_j), \quad \forall j = 1, \ldots, k\right) = 1.$$

Proof: Proposition 2 or Corollary 1 establishes identifiability, which in conjunction with Proposition 1 proves the corollary. ∎

Proof: (**for Proposition 2**) We prove identifiability by induction on k. In the base case of $k = 1$, we have a set of l equations, corresponding to $i = 1, 2 \ldots l$,

$$\begin{aligned} p_\theta(\hat{y}_1 = i) &= p(y = i)\theta_1 + \left(\sum_{j\neq i} p(y = j)\right)\frac{(1-\theta_1)}{(l-1)} \\ &= p(y = i)\theta_1 + (1 - p(y = i))\frac{(1-\theta_1)}{(l-1)} \\ &= \frac{\theta_1(lp(y = i) - 1) + 1 - p(y = i)}{(l-1)} \end{aligned}$$

from which we can see that if $\eta \neq \theta$ and $p(y=i) \neq 1/l$ then $p_\theta(\hat{y}_1) \neq p_\eta(\hat{y}_1)$. This proves identifiability for the base case of $k=1$.

Next, we assume identifiability holds for k and prove that it holds for $k+1$. We do so by deriving a contradiction from the assumption that identifiability holds for k but not for $k+1$. We denote the parameters corresponding to the k labelers by the vectors $\theta, \eta \in [0,1]^k$ and the parameters corresponding the additional $k+1$ labeler by θ_{k+1}, η_{k+1}.

In the case of k classifiers we have

$$p_\theta(\hat{y}_1, \ldots, \hat{y}_k) = \sum_{i=1}^{l} p_\theta(\hat{y}_1, \ldots, \hat{y}_k | y=i) p(y=i) = \sum_{i=1}^{l} G(\mathcal{A}_i, \theta)$$

where

$$G(\mathcal{A}_i, \theta) \stackrel{\text{def}}{=} p(y=i) \prod_{j \in \mathcal{A}_i} \theta_j \cdot \prod_{j \notin \mathcal{A}_i} \frac{(1-\theta_j)}{(l-1)}.$$

$$\mathcal{A}_i \stackrel{\text{def}}{=} \{j \in \{1, 2..., k\} : \hat{y}_j = i\}.$$

Note that the $\mathcal{A}_1, \ldots, \mathcal{A}_l$ form a partition of $\{1, \ldots, k\}$ i.e., they are disjoint and their union is $\{1, \ldots, k\}$.

In order to have unidentifiability for the $k+1$ classifiers we need that $(\theta, \theta_{k+1}) \neq (\eta, \eta_{k+1})$ implies $p_{(\theta,\theta_{k+1})}(\hat{y}_1, \ldots, \hat{y}_{k+1}) = p_{(\eta,\eta_{k+1})}(\hat{y}_1, \ldots, \hat{y}_{k+1})\ \forall \{\hat{y}_1, \ldots, \hat{y}_{k+1}\}$. That is, the following l equations (corresponding to $\hat{y}_{k+1} = 1, 2, \ldots, l$) must hold for any $\hat{y}_1, \ldots, \hat{y}_k$ which corresponds to any partition $\mathcal{A}_1, \ldots, \mathcal{A}_l$

$$\theta_{k+1} G(\mathcal{A}_1, \theta) + \frac{(1-\theta_{k+1})}{(l-1)} \sum_{i \neq 1} G(\mathcal{A}_i, \theta) = \eta_{k+1} G(\mathcal{A}_1, \eta) + \frac{(1-\eta_{k+1})}{(l-1)} \sum_{i \neq 1} G(\mathcal{A}_i, \eta)$$

$$\theta_{k+1} G(\mathcal{A}_2, \theta) + \frac{(1-\theta_{k+1})}{(l-1)} \sum_{i \neq 2} G(\mathcal{A}_i, \theta) = \eta_{k+1} G(\mathcal{A}_2, \eta) + \frac{(1-\eta_{k+1})}{(l-1)} \sum_{i \neq 2} G(\mathcal{A}_i, \eta)$$

$$\vdots$$

$$\theta_{k+1} G(\mathcal{A}_l, \theta) + \frac{(1-\theta_{k+1})}{(l-1)} \sum_{i \neq l} G(\mathcal{A}_i, \theta) = \eta_{k+1} G(\mathcal{A}_l, \eta) + \frac{(1-\eta_{k+1})}{(l-1)} \sum_{i \neq l} G(\mathcal{A}_i, \eta). \quad (7.33)$$

We consider two cases in which $(\theta, \theta_{k+1}) \neq (\eta, \eta_{k+1})$: (a) $\theta \neq \eta$, and (b) $\theta = \eta, \theta_{k+1} \neq \eta_{k+1}$. In the case of (a) we add the l equations above which marginalizes $\hat{y}_{k+1}$ out of $p_\theta(\hat{y}_1, \ldots, \hat{y}_k, \hat{y}_{k+1})$ and $p_\eta(\hat{y}_1, \ldots, \hat{y}_k, \hat{y}_{k+1})$ to provide

$$\sum_{i=1}^{l} G(\mathcal{A}_i, \theta) = \sum_{i=1}^{l} G(\mathcal{A}_i, \eta) \quad (7.34)$$

which together with $\theta \neq \eta$ contradicts the identifiability for the case of k classifiers.

In case (b) we have from the l equations above

$$\theta_{k+1}G(\mathcal{A}_t,\theta) + \frac{1-\theta_{k+1}}{l-1}\left(\sum_{i=1}^{l} G(\mathcal{A}_i,\theta) - G(\mathcal{A}_t,\theta)\right) \\ = \eta_{k+1}G(\mathcal{A}_t,\eta) + \frac{1-\eta_{k+1}}{l-1}\left(\sum_{i=1}^{l} G(\mathcal{A}_i,\eta) - G(\mathcal{A}_t,\eta)\right)$$

for any $t \in \{1,\ldots,l\}$ which simplifies to

$$0 = (\theta_{k+1} - \eta_{k+1})\left(lG(\mathcal{A}_t,\theta) - \sum_{i=1}^{l} G(\mathcal{A}_i,\theta)\right) \tag{7.35}$$

As we assume at this point that $\theta_{k+1} \neq \eta_{k+1}$ the above equality entails

$$lG(\mathcal{A}_t,\theta) = \sum_{i=1}^{l} G(\mathcal{A}_i,\theta). \tag{7.36}$$

We show that (7.36) cannot hold by examining separately the cases $p(y=t) > 1/l$ and $p(y=t) < 1/l$. Recall that there exists a t for which $p(y=t) \neq 1/l$ since the proposition requires that $p(y)$ is not uniform.

If $p(y=t) > 1/l$ we choose $\mathcal{A}_t = \{1,\ldots,k\}$ and obtain

$$\begin{aligned} lp(y=t)\prod_{j=1}^{k}\theta_j &= \sum_{i\neq t} p(y=i)\prod_{j=1}^{k}\frac{1-\theta_j}{l-1} + p(y=t)\prod_{j=1}^{k}\theta_j \\ (l-1)p(y=t)\prod_{j=1}^{k}\theta_j &= (1-p(y=t))\prod_{j=1}^{k}\frac{1-\theta_j}{l-1} \\ p(y=t)\prod_{j=1}^{k}\theta_j &= \frac{(1-p(y=t))}{(l-1)}\prod_{j=1}^{k}\frac{1-\theta_j}{l-1} \end{aligned}$$

which cannot hold as the term on the left hand side is necessarily larger than the term on the right hand side (if $p(y=t) > 1/l$ and $\theta_j > 1/l$). In the case of $p(y=t) < 1/l$ we choose

$\mathcal{A}_s = \{1, \ldots, k\}$ where $s \neq t$ to obtain

$$lp(y=t)\prod_{j=1}^{k}\frac{1-\theta_j}{l-1} = \sum_{i\neq s} p(y=i)\prod_{j=1}^{k}\frac{1-\theta_j}{l-1} + p(y=s)\prod_{j=1}^{k}\theta_j$$

$$(lp(y=t) - p(y\neq s))\prod_{j=1}^{k}\frac{1-\theta_j}{l-1} = p(y=s)\prod_{j=1}^{k}\theta_j$$

which cannot hold as the term on the left hand side is necessarily smaller than the term on the right hand side (if $p(y=t) < 1/l$ and $\theta_j > 1/l$).

Since we derived a contradiction to the fact that we have k-identifiability but not $k+1$ identifiability, the induction step is proven which establishes identifiability for any $k \geq 1$. ■

The conditions asserted above that $p(y) \neq 1/l$ and $1/l < R(f_i) < 1$ are intuitive. Consider three classifiers f_1, f_2, f_3 with conditionally independent noise processes where f_1, f_2 are relatively accurate and f_3 is relatively inaccurate. Observing the sequence of triplets of predicted labels over the unlabeled data $\{(f_1(x^{(i)}), f_2(x^{(i)}), f_3(x^{(i)})) : i = 1, \ldots, n\}$ we notice that for each triplet the first two values tend to agree with each other while the last one tends to disagree. The straighforward conclusion is that f_1 and f_2 are relatively accurate while f_3 is not. However, it seems likely that another solution is also possible: f_1 and f_2 are relatively inaccurate while f_3 is accurate. The only way we can rule out this possibility is by assuming that the classifiers are more accurate than chance. In this case the classifiers tend to be more accurate than to make mistakes and the two symmetric alternatives can be resolved. Similarly, for a case with a uniform $p(y)$ and $k = 1$ we cannot distinguish from the data due to symmetry whether $\theta_i = \alpha$ or $\theta_i = 1 - \alpha$.

In the case of the non-collaborative estimation for binary classification with the non-symmetric noise model, the matrix θ in (7.10) is a 2×2 matrix with two degrees of freedom as each row sums to one. In particular we have $\theta_{11} = p_\theta(\hat{y} = 1|y = 1)$, $\theta_{12} = p_\theta(\hat{y} = 1|y = 2)$, $\theta_{21} = p_\theta(\hat{y} = 2|y = 1)$, $\theta_{22} = p_\theta(\hat{y} = 2|y = 2)$ with the overall risk $R(f) = 1 - \theta_{11}p(y = 1) - \theta_{22}p(y = 2)$. Unfortunately, the matrix θ is not identifiable in this case and neither is the scalar parameter $\theta_{11}p(y = 1) + \theta_{22}p(y = 2)$ that can be used to characterize the risk.

We can, however, obtain a consistent estimator for θ (and therefore for $R(f)$) by first showing that the parameter $\theta_{11}p(y = 1) - \theta_{22}p(y = 2)$ is identifiable and then taking the intersection of two such estimators.

Lemma 2. *In the case of the non-collaborative estimation for binary classification with the non-symmetric noise model and $p(y) \neq 0$, the parameter $\theta_{11}p(y = 1) - \theta_{22}p(y = 2)$ is identifiable.*

Proof: For two different parameterizations θ, η we have

$$p_\theta(\hat{y}=1) = p(y=1)\theta_{11} + (1-p(y=1))(1-\theta_{22}) \tag{7.37}$$
$$p_\theta(\hat{y}=2) = p(y=1)(1-\theta_{11}) + (1-p(y=1))\theta_{22} \tag{7.38}$$

and

$$p_\eta(\hat{y}=1) = p(y=1)\eta_{11} + (1-p(y=1))(1-\eta_{22}) \tag{7.39}$$
$$p_\eta(\hat{y}=2) = p(y=1)(1-\eta_{11}) + (1-p(y=1))\eta_{22}. \tag{7.40}$$

Equating the two Equations (7.37) and (7.39) we have

$$\begin{gathered} p(y=1)(\theta_{11}+\theta_{22}) + 1 - p(y=1) - \theta_{22} = p(y=1)(\eta_{11}+\eta_{22}) + 1 - p(y=1) - \eta_{22} \\ p(y=1)\theta_{11} - (1-p(y=1))\theta_{22} = p(y=1)\eta_{11} - (1-p(y=1))\eta_{22} \\ p(y=1)\theta_{11} - p(y=2)\theta_{22} = p(y=1)\eta_{11} - p(y=2)\eta_{22} \end{gathered}$$

Similarly, equating Equation (7.38) and Equation (7.40) also results in $p(y=1)\theta_{11} - p(y=2)\theta_{22} = p(y=1)\eta_{11} - p(y=2)\eta_{22}$. As a result, we have

$$p_\theta \equiv p_\eta \quad \Rightarrow \quad p(y=1)\theta_{11} - p(y=2)\theta_{22} = p(y=1)\eta_{11} - p(y=2)\eta_{22}.$$

■

The above lemma indicates that we can use the maximum likelihood method to obtain a consistent estimator for the parameter $\theta_{11}p(y=1) - \theta_{22}p(y=2)$. Unfortunately the parameter $\theta_{11}p(y=1) - \theta_{22}p(y=2)$ neither has a clear probabilistic interpretation nor does directly characterize the risk. As the following proposition shows we can obtain a consistent estimator for the risk $R(f)$ if we have two populations of unlabeled data drawn from distributions with two distinct marginals $p_1(y)$ and $p_2(y)$.

Proposition 3. *Consider the case of the non-collaborative estimation of binary classification risk with the non-symmetric noise model. If we have access to two unlabeled datasets drawn independently from two distributions with different marginals i.e.*

$$x^{(1)}, \ldots, x^{(n)} \overset{\text{iid}}{\sim} p_1(x) = \sum_y p(x|y)p_1(y)$$
$$x'^{(1)}, \ldots, x'^{(m)} \overset{\text{iid}}{\sim} p_2(x) = \sum_y p(x|y)p_2(y)$$

we can obtain a consistent estimator for the classification risk $R(f)$.

Proof: Operating the classifier f on both sets of unlabeled data we get two sets of observed classifier outputs $\hat{y}^{(1)}, \ldots, \hat{y}^{(n)}$, $\hat{y}'^{(1)}, \ldots, \hat{y}'^{(m)}$ where $\hat{y}^{(i)} \overset{\text{iid}}{\sim} \sum_y p_\theta(\hat{y}|y)p_1(y)$ and

$\hat{y}'^{(i)} \overset{\text{iid}}{\sim} \sum_y p_\theta(\hat{y}|y)p_2(y)$. In particular, note that the marginal distributions $p_1(y)$ and $p_2(y)$ are different but the parameter matrix θ is the same in both cases as we operate the same classifier on samples from the same class conditional distribution $p(x|y)$.

Based on Lemma 2 we construct a consistent estimator for $p_1(y=1)\theta_{11} - p_1(y=2)\theta_{22}$ by maximizing the likelihood of $\hat{y}^{(1)}, \ldots, \hat{y}^{(n)}$. Similarly, we construct a consistent estimator for $p_2(y=1)\theta_{11} - p_2(y=2)\theta_{22}$ by maximizing the likelihood of $\hat{y}'^{(1)}, \ldots, \hat{y}'^{(m)}$. Note that $p_1(y=1)\theta_{11} - p_1(y=2)\theta_{22}$ and $p_2(y=1)\theta_{11} - p_2(y=2)\theta_{22}$ describe two lines in the 2-D space $(\theta_{11}, \theta_{22})$. Since the true value of θ_{11}, θ_{22} represent a point in that 2-D space belonging to both lines, it is necessarily the intersection of both lines (the lines cannot be parallel since their linear coefficients are distributions which are assumed to be different).

As n and m increase to infinity, the two estimators converge to the true parameter values. As a result, the intersection of the two lines described by the two estimators converges to the true values of $(\theta_{11}, \theta_{22})$ thus allowing reconstruction of the matrix θ and the risk $R(f)$. ■

Clearly, the conditions for consistency in the asymmetric case are more restricted than in the symmetric case. However, situations such as in Proposition 3 are not necessarily unrealistic. In many cases it is possible to identify two unlabeled sets with different distributions. For example, if y denotes a medical condition, it may be possible to obtain two unlabeled sets from two different hospitals or two different regions with different marginal distribution corresponding to the frequency of the medical condition.

As indicated in the previous section, the risk estimation framework may be extended beyond non-collaborative estimation and collaborative conditionally independent estimation. In these extensions, the conditions for identifiability need to be determined separately, in a similar way to Corollary 1. A systematic way to do so may be obtained by noting that the identifiability equations

$$0 = p_\theta(\hat{y}_1, \ldots, \hat{y}_k) - p_\eta(\hat{y}_1, \ldots, \hat{y}_k) \quad \forall \hat{y}_1, \ldots, \hat{y}_k$$

is a system of polynomial equations in (θ, η). As a result, demonstrating lack of identifiability becomes equivalent to obtaining a solution to a system of polynomial equations. Using Hilbert's Nullstellensatz theorem we have that a solution to a polynomial system exists if the polynomial system defines a proper ideal of the ring of polynomials [Cox *et al.*, 2006]. As k increases the chance of identifiability failing decays dramatically as we have a system of l^k polynomials with $2k$ variables. Such an over-determined system with substantially more equations than variables is very unlikely to have a solution.

These observations serve as both an interesting theoretical connection to algebraic geometry as well as a practical tool due to the substantial research in computational algebraic geometry. See [Sturmfels, 2002] for a survey of computational algorithms and software associated with systems of polynomial equations.

Consistency of Regression Risk Estimation

In this section, we prove the consistency of the maximum likelihood estimator $\hat{\theta}_n^{\text{mle}}$ in the regression case. As in the classification case our proof centers on establishing identifiability.

Proposition 4. *Let $f_1, \ldots, f_k$ be regression models $f_i(x) = a_i' x$ with $y \sim N(\mu_y, \sigma_y^2)$, $y = ax + \epsilon$. Assuming that $a \neq 0$ the unsupervised collaborative estimation model assuming conditionally independent noise processes* (7.27) *is identifiable.*

Corollary 3. *Let $f_1, \ldots, f_k$ be regression models $f_i(x) = a_i' x$ with $y \sim N(\mu_y, \sigma_y^2)$, $y = ax + \epsilon$. Assuming that $a \neq 0$ the unsupervised non-collaborative estimation model* (7.27) *is identifiable.*

Proof: Proving identifiability in the non-collaborative case proceeds by invoking Proposition 4 (whose proof is given below) with $k = 1$ separately for each regression model. The conditional independence assumption in Proposition 4 becomes redundant in this case of a single predictor, resulting in identifiability of $p_{\theta_j}(\hat{y}_j)$ for each $j = 1, \ldots, k$. ■

Corollary 4. *Under the assumptions of Proposition 4 or Corollary 3 the unsupervised maximum likelihood estimator is consistent i.e.,*

$$P\left(\lim_{n\to\infty} \hat{\theta}_n^{mle}(\hat{y}^{(1)}, \ldots, y^{(n)}) = (\theta_1^{true}, \ldots, \theta_k^{true})\right) = 1.$$

Consequentially, assuming that $R(f_j) = g_j(\theta), j = 1, \ldots, k$ with continuous g_j we also have

$$P\left(\lim_{n\to\infty} \hat{R}(f_j\,; y^{(1)}, \ldots, y^{(n)}) = R(f_j), \quad \forall j = 1, \ldots, k\right) = 1.$$

Proof: Proposition 4 or Corollary 3 establish identifiability, which in conjunction with Proposition 1 completes the proof. ■

Proof: (of Proposition 4).

We will proceed, as in the case of classification, with induction on the number of predictors k. In the base case of $k = 1$ we have derived $p_{\theta_1}(\hat{y}_1)$ in Equation (7.19). Substituting in it $\hat{y}_1 = 0$ we get

$$\begin{aligned} P_{\theta_1}(\hat{y}_1 = 0) &= \frac{1}{\theta_1\sqrt{2\pi(\tau^2 + \sigma_y^2)}} \exp\left(\frac{\mu_y^2}{2\sigma_y^2}\left(\frac{\tau^2}{\sigma_y^2 + \tau^2} - 1\right)\right) \\ P_{\eta_1}(\hat{y}_1 = 0) &= \frac{1}{\eta_1\sqrt{2\pi(\tau^2 + \sigma_y^2)}} \exp\left(\frac{\mu_y^2}{2\sigma_y^2}\left(\frac{\tau^2}{\sigma_y^2 + \tau^2} - 1\right)\right). \end{aligned} \tag{7.41}$$

The above expression leads to $\theta_1 \neq \eta_1 \Rightarrow p_{\theta_1}(\hat{y}_1 = 0) \neq p_{\eta_1}(\hat{y}_1 = 0)$ which implies identifiability.

In the induction step we assume identifiability holds for k and we prove that it holds also for $k+1$ by deriving a contradiction to the assumption that it does not hold. We assume that identifiability fails in the case of $k+1$ due to differing parameter values i.e.,

$$p_{(\theta,\theta_{k+1})}(\hat{y}_1,\ldots,\hat{y}_k,\hat{y}_{k+1}) = p_{(\eta,\eta_{k+1})}(\hat{y}_1,\ldots,\hat{y}_k,\hat{y}_{k+1})\ \forall \hat{y}_j \in \mathbb{R}\ j=1,\ldots,k+1 \tag{7.42}$$

with $(\theta,\theta_{k+1}) \neq (\eta,\eta_{k+1})$ where $\theta,\eta \in \mathbb{R}^k$. There are two cases which we consider separately: (a) $\theta \neq \eta$ and (b) $\theta = \eta$.

In case (a) we marginalize both sides of (7.42) with respect to $\hat{y}_{k+1}$ which leads to a contradiction to our assumption that identifiability holds for k

$$\begin{aligned}\int_{-\infty}^{\infty} p_{(\theta,\theta_{k+1})}(\hat{y}_1,\ldots,\hat{y}_k,\hat{y}_{k+1})d\hat{y}_{k+1} &= \int_{-\infty}^{\infty} p_{(\eta,\eta_{k+1})}(\hat{y}_1,\ldots,\hat{y}_k,\hat{y}_{k+1})d\hat{y}_{k+1}\\ p_\theta(\hat{y}_1,\ldots,\hat{y}_k) &= p_\eta(\hat{y}_1,\ldots,\hat{y}_k).\end{aligned} \tag{7.43}$$

In case (b) $\theta = \eta$ and $\theta_{k+1} \neq \eta_{k+1}$. Substituting $\hat{y}_1 = \cdots = \hat{y}_{k+1} = 0$ in (7.42) (see (7.28) for a derivation) we have

$$P_{(\theta,\theta_{k+1})}(\hat{y}_1 = 0,\ldots,\hat{y}_{k+1} = 0) = P_{(\eta,\eta_{k+1})}(\hat{y}_1 = 0,\ldots,\hat{y}_{k+1} = 0) \tag{7.44}$$

or

$$\begin{aligned}&\frac{\sqrt{\pi}\left[\frac{1}{2}\left(\frac{1}{\sigma_y^2}+\frac{k+1}{\tau^2}\right)\right]^{-1/2}}{\tau^{k+1}(\sqrt{2\pi})^{k+2}\sigma_y\theta_{k+1}\prod_{j=1}^k\theta_j}\ \exp\left(\frac{\left(\frac{\mu_y}{\sigma_y^2}\right)^2}{2\left(\frac{1}{\sigma_y^2}+\frac{k+1}{\tau^2}\right)}-\frac{\mu_y^2}{2\sigma_y^2}\right)\\ &\qquad = \frac{\sqrt{\pi}\left[\frac{1}{2}\left(\frac{1}{\sigma_y^2}+\frac{k+1}{\tau^2}\right)\right]^{-1/2}}{\tau^{k+1}(\sqrt{2\pi})^{k+2}\sigma_y\eta_{k+1}\prod_{j=1}^k\eta_j}\ \exp\left(\frac{\left(\frac{\mu_y}{\sigma_y^2}\right)^2}{2\left(\frac{1}{\sigma_y^2}+\frac{k+1}{\tau^2}\right)}-\frac{\mu_y^2}{2\sigma_y^2}\right)\end{aligned}$$

which cannot hold if $\theta = \eta$ but $\theta_{k+1} \neq \eta_{k+1}$. This constitutes a contradiction to the assumption that we have identifiability for k but not for $k+1$, which in turn completes the proof and establishes the identifiability for any $k \geq 1$. ■

7.3.2 Asymptotic Variance

A standard result from statistics is that the MLE has an asymptotically normal distribution with mean vector θ^{true} and variance matrix $(nJ(\theta^{\text{true}}))^{-1}$, where $J(\theta)$ is the $r \times r$ Fisher information matrix

$$J(\theta) = \mathsf{E}_{p_\theta}\{\nabla \log p_\theta(\hat{y})(\nabla \log p_\theta(\hat{y}))^\top\} \tag{7.45}$$

with $\nabla \log p_\theta(\hat{y})$ represents the $r \times 1$ gradient vector of $\log p_\theta(\hat{y})$ with respect to θ. Stated more formally, we have the following convergence in distribution as $n \to \infty$ [Ferguson, 1996]

$$\sqrt{n}\,(\hat{\theta}_n^{\text{mle}} - \theta_0) \rightsquigarrow N(0, J^{-1}(\theta^{\text{true}})). \tag{7.46}$$

It is instructive to consider the dependency of the Fisher information matrix, which corresponds to the asymptotic estimation accuracy, on $n, k, p(y), \theta^{\text{true}}$.

In the case of classification considering (7.12) with $k = 1$ and $\mathcal{Y} = \{1, 2\}$ it can be shown that

$$J(\theta) = \frac{\alpha(2\alpha-1)^2}{(\theta(2\alpha-1)-\alpha+1)^2} - \frac{(2\alpha-1)^2(\alpha-1)}{(\alpha-\theta(2\alpha-1))^2} \tag{7.47}$$

where $\alpha = P(y = 1)$. As Figure 7.3 (right) demonstrates, the asymptotic accuracy of the MLE (as indicated by J) tends to increase with the degree of non-uniformity of $p(y)$. Recall that since identifiability fails for a uniform $p(y)$ the risk estimate under a uniform $p(y)$ is not consistent. The above derivation (7.47) is a quantification of that fact reflecting the added difficulty in estimating the risk as we move closer to a uniform label distribution $\alpha \to 1/2 \implies J(\theta) \to 0 \implies J^{-1}(\theta) \to \infty$. The dependency of the asymptotic accuracy on θ^{true} is more complex, tending to favor θ^{true} values close to 1 or 0.5. Figure 7.3 (left) displays the empirical accuracy of the estimator as a function of $p(y)$ and θ^{true} and shows remarkable similarity to the contours of the Fisher information (see Section 7.5 for more details on the experiments). In particular, whenever the estimation error is high the asymptotic variance of the estimator is high (or equivalently, the Fisher information is low). For instance, the top contours in the left panel have smaller estimation error on the top right than in the top left. Similarly, the top contours in the right panel have smaller asymptotic variance on the top right than on the top left. We thus conclude that the Fisher information provides practical, as well as theoretical insight into the estimation accuracy.

Similar calculations of $J(\theta^{\text{true}})$ for collaborative classification case or for the regression case result in more complicated but straightforward derivations. It is important to realize that consistency is ensured for any identifiable $\theta^{\text{true}}, p(y)$. The value $(J(\theta^{\text{true}}))^{-1}$ is the constant dominating that consistency convergence.

A similar distributional analysis can be derived for the risk estimator. Applying Cramer's theorem [Ferguson, 1996] to $\hat{R}(f_j) = g_j(\hat{\theta}^{\text{mle}})$, $j = 1, \ldots, k$ and (7.46) we have

$$\sqrt{n}(\hat{R}(f) - R(f)) \rightsquigarrow N\left(0, \nabla g(\theta^{\text{true}}) J(\theta^{\text{true}}) \nabla g(\theta^{\text{true}})^\top\right) \tag{7.48}$$

where $R(f), \hat{R}(f)$ are the vectors of true risk and risk estimates for the different predictors $f_1, \ldots, f_k$ and $\nabla g(\theta^{\text{true}})$ is the Jacobian matrix of the mapping $g = (g_1, \ldots, g_k)$ evaluated at θ^{true}.

In the case of classification with $k = 1$ we have $R(f_j) = 1 - \theta_j$ and the Jacobian matrix is -1, leading to an identical asymptotic distribution to that of the MLE (7.46)-(7.47)

$$\sqrt{n}(\hat{R}(f) - R(f)) \rightsquigarrow N\left(0, \left(\frac{\alpha(2\alpha-1)^2}{(\theta(2\alpha-1)-\alpha+1)^2} - \frac{(2\alpha-1)^2(\alpha-1)}{(\alpha-\theta(2\alpha-1))^2}\right)^{-1}\right). \quad (7.49)$$

7.4 Optimization Algorithms

Recall that we obtained closed forms for the likelihood maximizers in the cases of non-collaborative estimation for binary classifiers and non-collaborative estimation for one dimensional regression models. The lack of closed form maximizers in the other cases necessitates iterative optimization techniques.

One class of technique for optimizing nonlinear loglikelihoods is the class of gradient based methods. Techniques in this class such as gradient descent, conjugate gradients, and quasi Newton methods often have good performance and are easy to derive. The main difficulty is the derivation of the loglikelihood and its derivatives. For example, in the case of collaborative estimation of classification ($l \geq 2$) with symmetric noise model and missing values the loglikelihood gradient is

$$\frac{\partial \ell}{\partial \theta_j} = \sum_{i=1}^{n} \frac{\sum_{y^{(i)}} p(y^{(i)}) \sum_{r:\beta_{ri}=0} \sum_{\hat{y}_r^{(i)}} \prod_{p \neq j} h_{pi} (I(\hat{y}_j^{(i)} = y^{(i)}) - \theta_j)((l-1)\theta_j)^{I(\hat{y}_j^{(i)}=y^{(i)})-1}(1-\theta_j)^{-I(\hat{y}_j^{(i)}=y^{(i)})}}{\sum_{y^{(i)}} p(y^{(i)}) \sum_{r:\beta_{ri}=0} \sum_{\hat{y}_r^{(i)}} \prod_{p=1}^{k} h_{pi}}$$

$$h_{pi} = \theta_p^{I(\hat{y}_p^{(i)}=y^{(i)})} \left(\frac{1-\theta_p}{l-1}\right)^{I(\hat{y}_p^{(i)} \neq y^{(i)})}. \quad (7.50)$$

Similar derivations may be obtained in the other cases in a straightforward manner.

An alternative iterative optimization technique for finding the MLE is expectation maximization (EM). The derivation of the EM update equations is again relatively straightforward. For example in the above case of collaborative estimation of classification ($l \geq 2$) with

symmetric noise model and missing values the EM update equations are

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_{i=1}^{n} \sum_{y^{(i)}} \sum_{r:\beta_{ri}=0} \sum_{\hat{y}_r^{(i)}} q^{(t)}(\hat{y}_r^{(i)}, y^{(i)}) \sum_{j=1}^{k} \log p_j(\hat{y}_j^{(i)}|y^{(i)}) \tag{7.51}$$

$$= \frac{1}{n} \sum_{i=1}^{n} \sum_{y^{(i)}} \sum_{r:\beta_{ri}=0} \sum_{\hat{y}_r^{(i)}} q^{(t)}(\hat{y}_r^{(i)}, y^{(i)}) I(\hat{y}_j^{(i)} = y^{(i)})$$

$$q^{(t)}(\hat{y}_r^{(i)}, y^{(i)}) = \frac{p(y^{(i)}) \prod_{j=1}^{k} p_j(\hat{y}_j^{(i)}|y^{(i)}, \theta^{(t)})}{\sum_{y^{(i)}} \sum_{r:\beta_{ri}=0} \sum_{\hat{y}_r^{(i)}} p(y^{(i)}) \prod_{j=1}^{k} p_j(\hat{y}_j^{(i)}|y^{(i)}, \theta^{(t)})}.$$

where $q^{(t)}$ is the conditional distribution defining the EM bound over the loglikelihood function.

If all the classifiers are always observed i.e., $\beta_{ri} = 1 \; \forall r, i$ Equation (7.31) reverts to (7.27), and the loglikelihood and its gradient may be efficiently computed in $O(nlk^2)$. In the case of missing classifier outputs a naive computation of the gradient or EM step is exponential in the number of missing values $R = \max_i \sum_r \beta_{ri}$. This, however, can be improved by careful dynamic programming. For example, the nested summations over the unobserved values in the gradient may be computed using a variation of the elimination algorithm in $O(nlk^2R)$ time.

7.5 Experimental Evaluation

This section is devoted to empirically analyze the proposed framework and its effectiveness on various scenarios. We start with some experiments demonstrating our framework using synthetic data. These experiments are designed to examine the behavior of the estimators in a controlled setting. We then describe some experiments using several real world datasets. In these experiments we examine the behavior of the estimators in an uncontrolled setting where some of the underlying assumptions may be violated. Throughout the section, we consider the mean absolute error (mae) or the ℓ_1 error as a metric that measures the estimation quality unless otherwise noted.

$$\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}}) = \frac{1}{k} \sum_{i=1}^{k} \left| \theta_i^{\text{true}} - \hat{\theta}_i^{\text{mle}} \right| \tag{7.52}$$

where k is the number of predictors. In the non-collaborative case (which is equivalent to the collaborative case with $k = 1$) this translates into the absolute deviation of the estimated parameter from the true parameter.

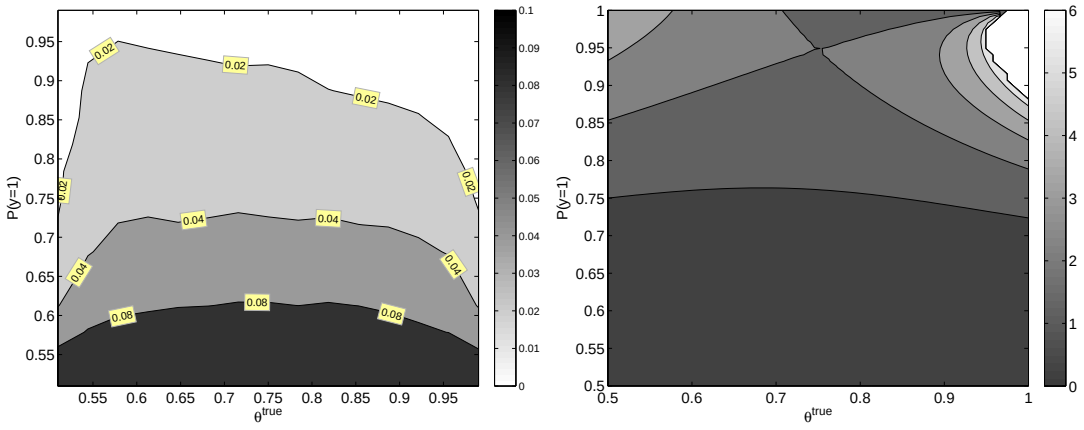


Figure 7.3: Left: Average value of $|\hat{\theta}_n^{\text{mle}} - \theta^{\text{true}}|$ as a function of θ^{true} and $p(y=1)$ for $k=1$ classifier and $n=500$ (computed over a uniform spaced grid of 15×15 points). The plot illustrates the increased accuracy obtained by a less uniform $P(y)$. Right: Fisher information $J(\theta)$ for $k=1$ as a function of θ^{true} and $P(y)$. The asymptotic variance of the estimator is $J^{-1}(\theta)$ which closely matches the experimental result in the left panel.

In Figure 7.3 (left) we display $\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ for classification with $k=1$ as a function of θ^{true} and $p(y)$ for $n=500$ simulated data points. The estimation error, while overall relatively small, decays as $p(y)$ diverges from the uniform distribution. The dependency on θ^{true} indicates that the error is worst for θ^{true} around 0.75 and it decays as $|\theta^{\text{true}} - 0.75|$ increases with a larger decay attributed to higher θ^{true}. These observations are remarkably consistent with the developed theory as Figure 7.3 (right) shows by demonstrating the value of the inverse asymptotic variance $J(\theta)$ which agrees nicely with the empirical measurement in the left panel.

Figure 7.4 (left) contains a scatter plot contrasting values of θ^{true} and $\hat{\theta}^{\text{mle}}$ for $k=1$ classifier and $p(y=1)=0.8$. The estimator was constructed based on 500 simulated data points. We observe a symmetric Gaussian-like distribution of estimated values $\hat{\theta}^{\text{mle}}$, conditioned on specific values of θ^{true}. This is in agreement with the theory predicting an asymptotic Gaussian distribution for the mle, centered around the true value θ^{true}. A similar observation is made in Figure 7.5 (left) which contains a similar scatter plot in the regression case ($k=1$, $\sigma_y=1$, $n=1000$). In both figures, the striped effect is due to selection of θ^{true} over a discrete grid with a small perturbation for increased visibility. Similar plots of larger and smaller n values (not shown) verify that the variation of $\hat{\theta}^{\text{mle}}$ around θ^{true} decreases as n increases. This agrees with the theory that indicates a $O(n^{-1})$ rate of decay for the variance of the asymptotic distribution.

Figures 7.4 and 7.5 (right) show the $\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ for various k values in classification and regression, respectively. In classification, $\hat{\theta}^{\text{mle}}$ was obtained by sampling data from

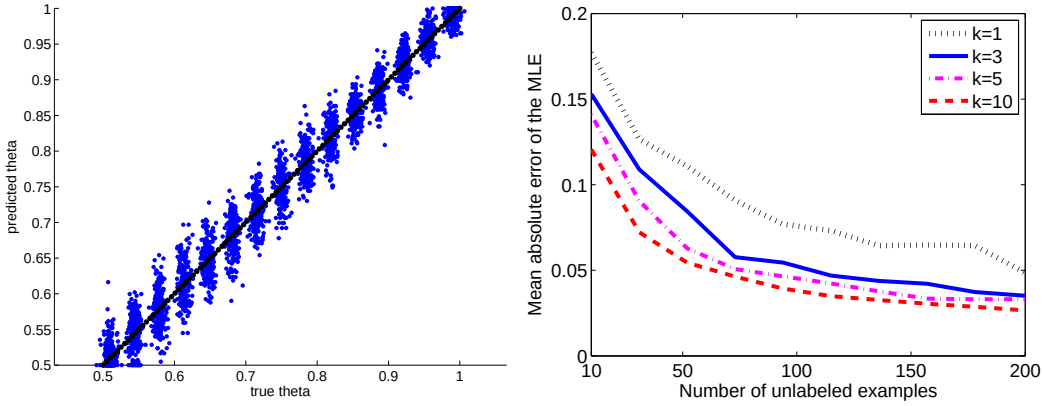


Figure 7.4: Left: Scatter plot contrasting the true and predicted values of θ in the case of a single classifier $k = 1$, $p(y = 1) = 0.8$, and $n = 500$ unlabeled examples. The displayed points were perturbed for improved visualization and the striped effect is due to empirical evaluation over a discrete grid of θ^{true} values. Right: $\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ as a function of the number of unlabeled examples for different number of classifiers ($\theta_i^{\text{true}} = p(y = 1) = 0.75$) in the collaborative case. The estimation error decreases as more classifiers are used due to the collaborative nature of the estimation process.

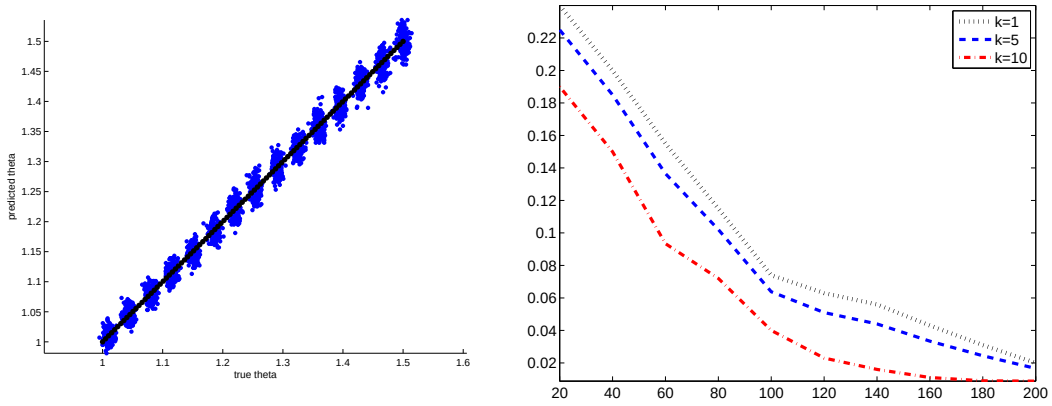


Figure 7.5: Left: Scatter plot contrasting the true and predicted values of θ in the case of a single regression model $k = 1$, $\sigma_y = 1$, and $n = 1000$ unlabeled examples. The displayed points were perturbed for improved visualization and the striped effect is due to empirical evaluation over a discrete grid of θ^{true} values. Right: $\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ as a function of the number of unlabeled examples for different number of regression models ($\theta_i^{\text{true}} = \sigma_y = 1$) in the collaborative case. The estimation error decreases as more regression models are used due to the collaborative nature of the estimation process.

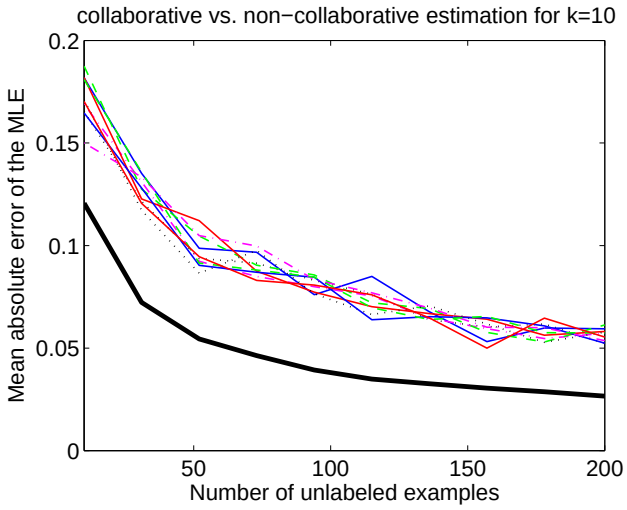


Figure 7.6: Comparison of collaborative and non-collaborative estimation for $k = 10$ classifiers. mae$(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ as a function of n is reported for $\theta_i^{\text{true}} = 0.75\ \forall k_i$ and $P(y = 1) = 0.75$. The colored lines represent the estimation error for each individual classifier and the solid black line represents the collaborative estimation for all classifiers. The estimation converges to the truth faster in the collaborative case than in the non-collaborative case.

$p(y = 1) = 0.75 = \theta_i^{\text{true}}, \forall i$. In regression, the data was sampled from the regression equation with $\theta_i^{\text{true}} = 1$ and $p(y) = N(0, 1)$. In both cases, the mae error decays with n as expected from the consistency proof and with k as a result of the collaborative estimation effect.

To further illustrate the effect of the collaboration on the estimation accuracy, we estimated the error rates individually (non-collaboratively) for 10 predictors and compared their mae to that of the collaborative estimation case in Figure 7.6. This shows that each of the classifiers have a similar mae curve when non-collaborative estimation is used. However, all of these curves are higher than the collaborative mae curve (solid black line in Figure 7.6) demonstrating the significant improvement of the collaborative estimation.

We compare in Figure 7.7 the proposed unsupervised estimation framework with supervised estimation that takes advantage of labeled information to determine the classifier accuracy. We conducted this study using equal number of examples for both supervised and unsupervised cases. Clearly, this is an unfair comparison if we assume that labeled data is unavailable or is difficult to obtain. The unsupervised estimation does not perform as well as the supervised version especially in general. Nevertheless, the unsupervised estimation accuracy improves significantly with the increasing number of classifiers and finally reaches the performance level of the supervised case due to collaborative estimation.

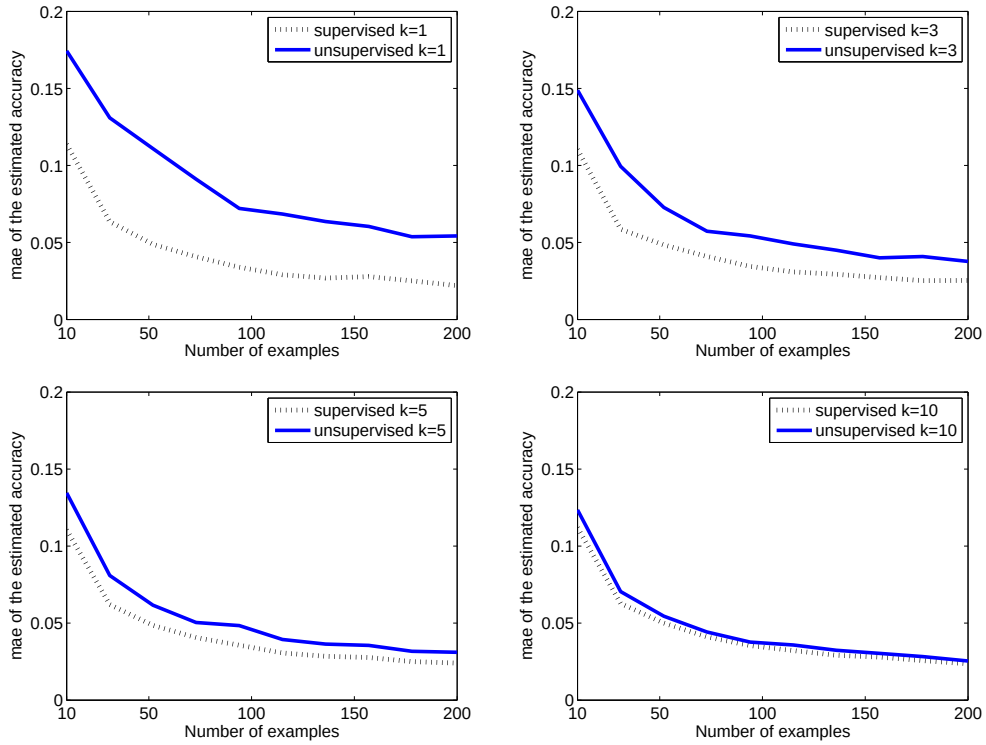


Figure 7.7: Comparison of supervised and unsupervised estimation for different values of classifiers with $k = 1, 3, 5, 10$. Supervised estimation uses the true labels to determine the accuracy of the classifiers whereas in the unsupervised case the estimation proceeds according to the collaborative estimation framework. Despite the fact that the supervised case uses labels the unsupervised framework reaches similar levels by increasing the number of classifiers.

In Figure 7.8 we report the effect of misspecification of the marginal $p(y)$ on the estimation accuracy. More specifically, we generated synthetic data using a true marginal distribution but estimated the classifier accuracy on this data assuming a misspecified marginal. Generally, the estimation framework is robust to small perturbations while over-specifying tends to hurt less than under-specifying (misspecification closer to uniform distribution).

Figure 7.9 shows the mean prediction accuracy for the unsupervised predictor combination scheme in (7.7) for synthetic data. The left panel displays classification accuracy and the right panel displays the regression accuracy as measured by $1 - \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i^{\text{new}} - y_i^{\text{new}})^2$. The graphs show that in both cases the accuracy increases with k and n in accordance with the theory and the risk estimation experiments. The parameter θ_i^{true} was chosen uniformly

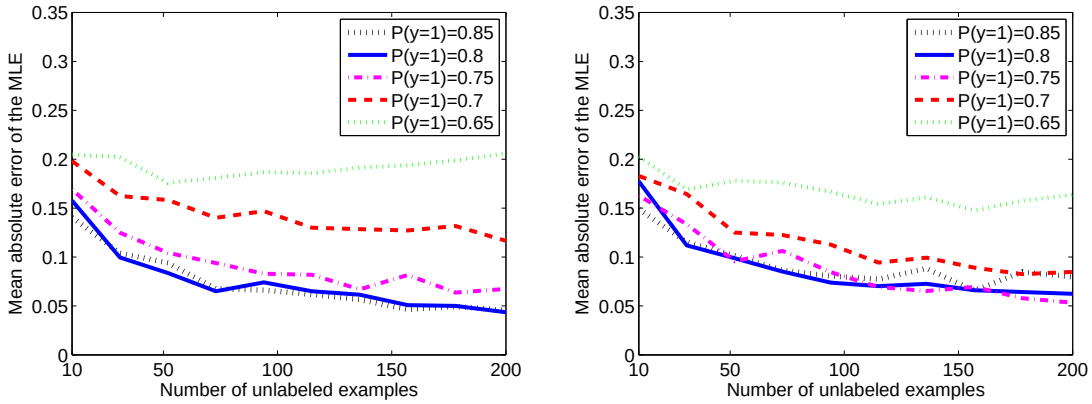


Figure 7.8: The figure compares the estimator accuracy assuming that the marginal $p(y)$ is misspecified. The plots draw mae$(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ as a function of n for $k = 1$ and $\theta^{\text{true}} = 0.75$ when $P^{\text{true}}(y = 1) = 0.8$ (left) and $P^{\text{true}}(y = 1) = 0.75$ (right). Small perturbations in $P^{\text{true}}(y)$ do not affect the results significantly; interestingly over-specifying $P^{\text{true}}(y = 1)$ leads to more accurate estimates than under-specifying (misspecification closer to uniform distribution)

in the range $(0.5, 1)$, and $P(y = 1) = 0.75$ for classification and $\theta_i^{\text{true}} = 0.3$, $p(y) = N(0, 1)$ in the case of regression.

We also experimented with the natural language understanding dataset introduced in [Snow *et al.*, 2008]. This data was created using the Amazon Mechanical Turk (AMT) for data annotation. We selected two binary tasks from this dataset which we explained in the previous chapter: the textual entailment recognition (RTE) and temporal event recognition (TEMP) tasks. For the former task, the original dataset contains 800 sentence pairs with a total of 165 annotators. For the latter task , the original dataset contains 462 pairs and 76 annotators. In both cases, most of the annotators have completed only a handful of tasks. Therefore, we selected a subset of these annotators for each task such that each annotator has completed at least 100 problems and has differing accuracies. The datasets contain ground truth labels which are used solely to calculate the annotator accuracy and not used at all during the estimation process. For efficiency, we selected only the instances for which all annotators provide an answer. This resulted in $n = 100, 190$ for RTE and TEMP, respectively.

In Figure 7.10 we display mae$(\theta^{\text{true}}, \hat{\theta}^{\text{mle}})$ for these datasets as function of n for different values of k. These plots generated from real-world data show similar trend to the synthetic experiments. The estimation errors decay to 0 as n increases and generally tend to decrease as k increases. This correspondence is remarkable since two of the labelers have worse than random accuracy and since it is not clear whether the conditional independence assumption

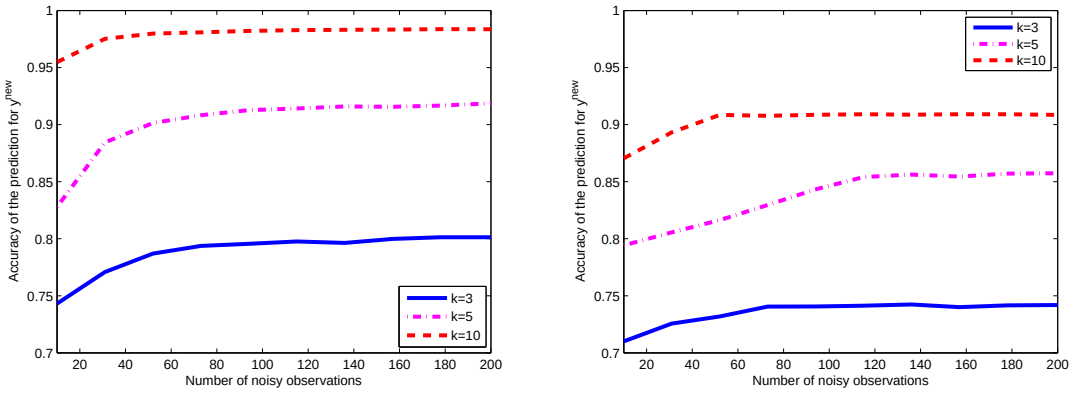


Figure 7.9: Mean prediction accuracy for the unsupervised predictor combination scheme in (7.7) for synthetic data. The left panel displays classification accuracy and the right panel displays the regression accuracy as measured by $1 - \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i^{\text{new}} - y_i^{\text{new}})^2$. The graphs show that in both cases the accuracy increases with k and n in accordance with the theory and the risk estimation experiments.

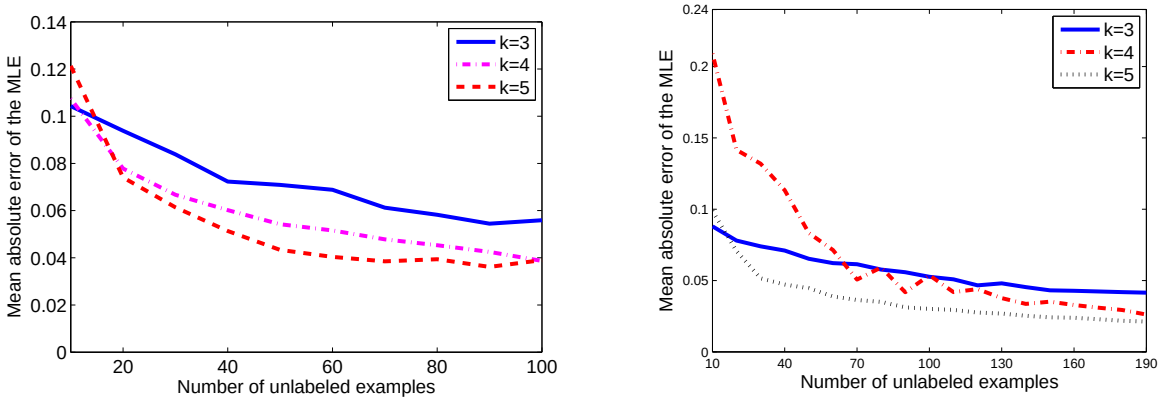


Figure 7.10: $\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ as a function of n for different number of annotators k on RTE (left) and TEMP (right) datasets. Left: $n = 100$, $P(y = 1) = 0.5$ and $\theta^{\text{true}} = \{0.85, 0.92, 0.58, 0.5, 0.51\}$. Right: $n = 190$, $P(y = 1) = 0.56$ and $\theta^{\text{true}} = \{0.93, 0.92, 0.54, 0.44, 0.92\}$. The classifiers were added in the order specified.

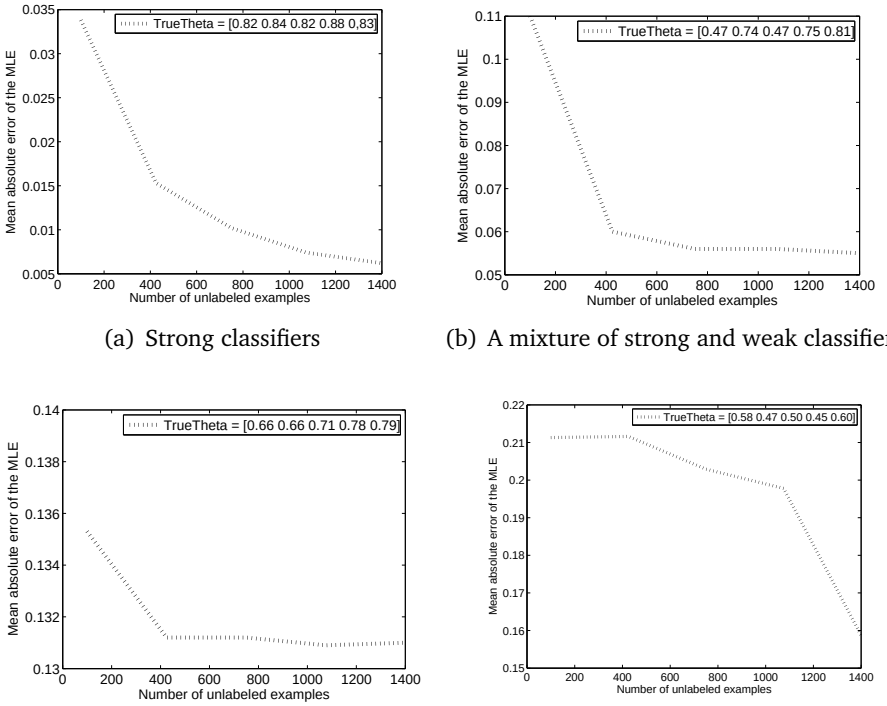
actually holds in reality for these datasets. Nevertheless, the collaborative estimation error behaves in accordance with the synthetic data experiments and the theory. This shows that the estimation framework is robust to the breakdown of the assumption that the classifier accuracy must be higher than random choice. Also, whether the conditional independence assumption holds or not is not crucial in this case.

We further experimented with classifiers trained on different representations of the same dataset and estimated their error rates. We adopted the Ringnorm dataset generated by [Breiman, 1996]. Ringnorm is a 2-class artificial dataset with 20 dimensions where each class is drawn from a multivariate normal distribution. One class has zero mean and a covariance $\Sigma = 4I$ where I is the identity matrix. The other class has unit covariance and a mean $\mu = (\frac{2}{\sqrt{20}}, \frac{2}{\sqrt{20}}, \ldots, \frac{2}{\sqrt{20}})$. The total size is 7400. We created 5 different representations of the data by projecting it onto mutually exclusive sets of principal components obtained by Principal Component Analysis (PCA). We trained an SVM classifier (with 2-degree polynomial kernel) [Vapnik, 2000; Joachims, 1999] on samples from each representation while holding out 1400 examples as the test set resulting in a total of 5 classifiers. We tested each of the 5 classifiers on the test set and used their outputs to estimate the corresponding parameters. The true labels of the test set examples were used as ground truth to calculate the mae of the mle estimators.

The mae curves for this dataset appear in Figure 7.11 as a function of the number n of unlabeled examples. When all classifiers are highly accurate (upper left panel), the collaborative unsupervised estimator is reliable, see Figure 7.11(a). With a mixture of weak and strong classifiers (upper right panel), the collaborative unsupervised estimator is also reliable. This is despite the fact that some of the weak classifiers in Figure 7.11(b) have worse than random accuracy which violates the assumptions in the consistency proposition. This shows again that the estimation framework is robust to occasional deviations from the requirement concerning better than random classification accuracies. On the other hand, as most of the classifiers become worse (bottom row), the accuracy of the unsupervised estimator decreases, in accordance with the theory developed in Sections 7.3.2 (recall the Fisher information contour plot).

Our experiments thus far assumed the symmetric noise model (7.12). Despite it not being always applicable for real world data and classifiers, it did result in good estimation accuracy in some of the cases described thus far. However, in some cases this assumption is grossly violated and the more general noise model is needed (7.10). For this reason, we conducted two experiments using real world data assuming the more general (7.10).

The first experiment concerned domain adaptation [Blitzer *et al.*, 2007] for Amazon's product reviews in four different product domains: books, DVDs, electronics and kitchen appliances. Each domain consists of positive ($y = 1$) and negative ($y = 2$) reviews with $p(y = 1) = 0.75$. The task was to estimate the error rates of classifiers (linear SVM [Vapnik, 2000; Joachims, 1999]) that are trained on 300 examples from one domain but tested on other domains. The mae values for the classification risks are displayed in Figure 7.12 with the columns indicating the test domain. In this case, the unsupervised non-collaborative estimator outperforms the collaborative estimator due to violation of the conditional independence assumption. Both unsupervised estimators perform substantially better than the baseline estimator that uses the training error on one domain to predict



(a) Strong classifiers (b) A mixture of strong and weak classifiers

(c) Mostly weak classifiers (d) Very weak classifiers

Figure 7.11: $\text{mae}(\theta^{\text{true}}, \hat{\theta}^{\text{mle}})$ as a function of the test set size on the Ringnorm dataset. $p(y = 1) = 0.47$, and θ^{true} is indicated in the legend in each plot. The four panels represent mostly strong classifiers (upper left), a mixture of strong and weak classifiers (upper right), mostly weak classifiers (bottom left), and mostly very weak classifiers (bottom right). The figure shows that the framework is robust to occasional deviations from the assumption regarding better than random guess classification accuracy (upper right panel). However, as most of the classifiers become weak or very weak, the collaborative unsupervised estimation framework results in worse estimation error.

testing error on another domain.

In the second experiment using (7.10) we estimated the risk (non-collaboratively) of 20 one vs. all classifiers (trained to predict one class) on the 20 newsgroup data [Lang, 1995]. The train set size was 1000 and the unlabeled data size was 15000. In this case the unsupervised non-collaborative estimator returned extremely accurate risk estimators. As a comparison, the risk estimates obtained from the training error are four times larger than the unsupervised MLE estimator (See Figure 7.12).

	book	dvd	kitchen	electronics	20newsgroup
training error	0.22	0.23	0.26	0.30	0.028
non-collaborative	**0.04**	**0.04**	**0.08**	**0.06**	**0.006**
collaborative	0.10	0.10	0.09	0.08	n/a

Figure 7.12: $\text{mae}(\hat{\theta}^{\text{mle}}, \theta^{\text{true}})$ for the domain adaptation ($n = 1000$, $p(y = 1) = 0.75$) and 20 newsgroup ($n = 15,000$, $p(y = 1) = 0.05$ for each one-vs-all data). The unsupervised non-collaborative estimator outperforms the collaborative estimator due to violation of the conditional independence assumption. Both unsupervised estimators perform substantially better than the baseline training error rate estimator. In both cases the results were averaged over 50 random train test splits.

7.6 Chapter Conclusions

We have demonstrated a framework to estimate classification and regression risks for multiple predictors. Our approach uses exclusively unlabeled data and knowledge of $p(y)$ whereas vast majority of risk estimation methods are supervised; i.e., cross validation [Duda *et al.*, 2001], bootstrap[Efron and Tibshirani, 1997], and others [Hand, 1986]. We prove statistical consistency in the unsupervised case and derive the asymptotic variance. Our experiments on synthetic data demonstrate the effectiveness of the framework and verify the theoretical results. Experiments on real world data show robustness to underlying assumptions. The framework may be applied to estimate additional quantities in an unsupervised manner, including noise level in noisy communication channels [Cover and Thomas, 2005] and error rates in structured prediction problems.

The difference of the work described here and the work described at the previous chapter is that the error estimator $\hat{\theta}^{\text{mle}}$ can be used to effectively aggregate predictors $f_1, \ldots, f_j$ to predict the label of unlabeled instances. Although IEThresh and SFilter have unsupervised ways to combine multiple noisy output for a final prediction, they lack the theoretical guarantees of the maximum likelihood estimation. Nevertheless, consistent estimators require knowledge of $p(y)$. For the cases when this information is unavailable or that the predictors have time-varying accuracies IEThresh or SFilter provide novel solutions.

Chapter 8

Unsupervised Margin-Based Risk Estimation

8.1 Introduction

Thus far, we have focused on accuracy and risk estimation and predictor selection. These estimation techniques yield other benefits such as predicting the true labels and so on. However, we are also interested in a more fundamental research question: Is it possible to partition the data according to the classes without any labeled data? This is a very ambitious goal as well as a very profound one. Assuming this partitioning leads to very good correlation with the true classes, it will be extremely useful in practice (predict the labels without any training data) and will also potentially attract many theoretical contributions from the field. This last chapter of the thesis is devoted to the proposal of an unsupervised way to achieve this goal without using a single labeled example whatsoever. This challenging task is possible through estimating a margin-based risk function (instead of 0-1 risk discussed in the previous chapter) using exclusively unlabeled data, assuming $p(y)$ is known and the class-conditional classifier margin $f_\theta(X) \mid y$ is normally distributed. We prove that the proposed technique is statistically consistent for high-dimensional linear classifiers and demonstrate it on synthetic and real-world data. Label scarcity is a serious and well-known problem which we have been discussing throughout the entire thesis. Active learning, semi-supervised learning and their variations are different approaches to overcome the labeling burden. The techniques we develop in this chapter introduces a new paradigm that goes beyond these alternatives in requiring no labels and therefore renders possible many applications suffering from label scarcity.

Many popular linear classifiers, such as logistic regression, boosting, or SVM, are trained by optimizing a margin-based risk function. For standard linear classifiers $\hat{Y} = \text{sign} \sum \theta_j X_j$

with $Y \in \{-1, +1\}, \quad X \in \mathbb{R}^d$ the margin is defined as

$$Y f_\theta(X) \quad \text{where} \quad f_\theta(X) \stackrel{\text{def}}{=} \sum_{j=1}^{d} \theta_j X_j. \tag{8.1}$$

Training such classifiers involves choosing a particular value of θ. This is done by minimizing the risk or expected loss

$$R(\theta) = \mathsf{E}_{p(X,Y)} L(Y, f_\theta(X)). \tag{8.2}$$

Three popular examples of the loss L are

$$L_1(Y, f_\theta(X)) = \exp\left(-Y f_\theta(X)\right) \tag{8.3}$$

$$L_2(Y, f_\theta(X)) = \log\left(1 + \exp\left(-Y f_\theta(X)\right)\right) \tag{8.4}$$

$$L_3(Y, f_\theta(X)) = \left(1 - Y f_\theta(X)\right)_+. \tag{8.5}$$

L_1 above corresponds to boosting (exponential loss), L_2 corresponds to logistic regression (logloss), and L_3 corresponds to SVM (hinge loss).

Since the risk $R(\theta)$ depends on the unknown distribution p, it is usually replaced during training with its empirical counterpart based on a labeled training set

$$(X^{(1)}, Y^{(1)}), \ldots, (X^{(n)}, Y^{(n)}) \stackrel{\text{iid}}{\sim} p \tag{8.6}$$

leading to the following estimator

$$\hat{\theta}_n = \arg\min_\theta R_n(\theta) \quad \text{where}$$

$$R_n(\theta) = \mathsf{E}_{\tilde{p}(X,Y)} L(Y, f_\theta(X)) \tag{8.7}$$

$$= \frac{1}{n} \sum_{i=1}^{n} L(Y^{(i)}, f_\theta(X^{(i)})). \tag{8.8}$$

Note, however, that evaluating and minimizing R_n requires labeled data (8.6). As we have been discussing throughout the entire thesis, obtaining labeled data in certain situations is a difficult, costly and time-consuming task.

In this chapter we construct an estimator for $R(\theta)$ using only unlabeled data, that is using

$$X^{(1)}, \ldots, X^{(n)} \stackrel{\text{iid}}{\sim} p \tag{8.9}$$

instead of (8.6). Our estimator is based on the following observations. When the data is high dimensional ($d \to \infty$) the quantities

$$f_\theta(X)|Y = y, \quad y \in \{-1, +1\} \tag{8.10}$$

are often normally distributed ($f_\theta(X) = \langle \theta, X \rangle$ as in (8.1)). This phenomenon is supported by empirical evidence and may also be derived using non-iid central limit theorems. We then observe that the limit distributions of (8.10) may be estimated from unlabeled data (8.9) and that these distributions may be used to measure margin-based losses such as (8.3)-(8.5).

We examine two novel unsupervised applications: (i) estimating margin-based losses in transfer learning and (ii) unsupervised class partitioning. We investigate these applications theoretically and also provide empirical results on synthetic and real-world data. Our empirical evaluation shows the effectiveness of the proposed framework in risk estimation and class partition without any labeled data.

The consequences of estimating $R(\theta)$ without labels are indeed profound. Label scarcity is a well known problem which has lead to the emergence of semisupervised learning (learning using a few labeled examples and many unlabeled ones), active learning and proactive learning. The techniques we develop lead to a new paradigm that goes beyond these in requiring no labels whatsoever to partition the data according to the classes; hence to make accurate predictions.

8.2 Unsupervised Margin-Based Risk Estimation

In this section we describe in detail the proposed estimation framework and discuss its theoretical properties. Specifically, we construct an estimator for $R(\theta)$ (8.2) using unlabeled data (8.9).

Our estimation is based on two assumptions. The first assumption is that the label marginals $p(Y)$ are known and that $p(Y = 1) \neq p(Y = -1)$. While this assumption may seem restrictive at first, there are many cases where it holds. See Chapter 7 for a full list of situations where it is known. We just recapitulate here that there are many examples where $p(Y)$ is known with great accuracy even if labeled data is unavailable. Nevertheless, we conducted an analysis to investigate the robustness of our estimation framework to misspecifications in $p(Y)$. Generally, small deviations result in a small degradation of the risk estimation quality while the degradation increases with larger deviations. See Section 8.3 for more details.

The second assumption is that the quantity $f_\theta(X)|Y$ follows a normal distribution. As $f_\theta(X)$ is a linear combination of random variables, it is frequently normal when X is high dimensional. This assumption holds empirically for many high dimensional data (see Section 8.2.1). From a theoretical perspective this assumption is motivated by the central limit theorem (CLT). The classical CLT states that $f_\theta(X) = \sum_{i=1}^{d} \theta_i X_i$ is approximately normal for large d if the data components $X_1, \ldots, X_d$ are iid. A more general CLT state that $f_\theta(X)|Y$ is asymptotically normal if $X_1, \ldots, X_d$ are independent (not necessary identically distributed).

Even more general CLTs state that $f_\theta(X)|Y$ is asymptotically normal if $X_1, \ldots, X_d$ are not independent but their dependency is limited in some way. We examine this issue in Section 8.2.1.

To derive the estimator we rewrite (8.2) by taking expectation with respect to Y and $\alpha = f_\theta(X)$

$$\begin{aligned} R(\theta) &= \mathsf{E}_{p(f_\theta(X),Y)} L(Y, f_\theta(X)) \\ &= \sum_{y\in\{-1,+1\}} p(y) \int_{\mathbb{R}} p(f_\theta(X) = \alpha|y) L(y,\alpha)\, d\alpha \\ &= p(y=1) \quad \int_{\mathbb{R}} p(f_\theta(X) = \alpha|y=1) L(1,\alpha)\, d\alpha \\ &+ p(y=-1) \int_{\mathbb{R}} p(f_\theta(X) = \alpha|y=-1) L(-1,\alpha)\, d\alpha. \end{aligned} \tag{8.11}$$

Equation (8.11) involves three terms $L(y,\alpha)$, $p(y)$ and $p(f_\theta(X) = \alpha|y)$. The loss function L is known and poses no difficulty. The second term $p(y)$ is assumed to be known (see discussion above). The third term is normal (assuming a CLT holds cf. Section 8.2.1)

$$f_\theta(X) \,|\, y = \sum_i \theta_i X_i \,|\, y \sim N(\mu_y, \sigma_y)$$

with parameters μ_y, σ_y, $y \in \{-1, 1\}$ that are generally unknown. Note that although we do not denote it explicitly, μ_y and σ_y are functions of θ.

We conclude with estimating $\mu = (\mu_1, \mu_{-1})$ and $\sigma = (\sigma_1, \sigma_{-1})$ by maximizing the likelihood of the unlabeled data (8.9)

$$\begin{aligned} (\hat{\mu}^{(n)}, \hat{\sigma}^{(n)}) &= \arg\max_{\mu,\sigma} \ell_n(\mu,\sigma) \quad \text{where} \\ \ell_n(\mu,\sigma) &= \sum_{i=1}^n \log p(f_\theta(X^{(i)})) \\ &= \sum_{i=1}^n \log \sum_{y^{(i)}} p(f_\theta(X^{(i)}), y^{(i)}) \\ &= \sum_{i=1}^n \log \sum_{y^{(i)}} p(y^{(i)}) p_{\mu_{y^{(i)}},\sigma_{y^{(i)}}}(f_\theta(X^{(i)})|y^{(i)}). \end{aligned} \tag{8.12}$$

Note that the loglikelihood (8.12) does not use labeled data (the label $y^{(i)}$ is marginalized over as it is unknown). Also, the loglikelihood (8.12) parameter is $\mu = (\mu_1, \mu_{-1})$ and $\sigma = (\sigma_1, \sigma_{-1})$, rather than the parameter θ associated with the classifier. We consider the latter one as a fixed constant at this point.

The estimation problem (8.12) is equivalent to the problem of estimating the means and variances of a Gaussian mixture model where the label marginals are assumed to be known. As we show in Section 8.2.2 the estimator (8.12) is consistent, that is $\lim_n(\hat{\mu}^{(n)}, \hat{\sigma}^{(n)}) = (\mu, \sigma)$ as $n \to \infty$, leading to the convergence of the plug-in estimate (see Section 8.2.2 for a proof)

$$P\Big(\lim_{n\to\infty} \hat{R}_n(\theta) = R(\theta)\Big) = 1 \quad \text{where} \tag{8.13}$$

$$\hat{R}_n(\theta) = \sum_{y\in\{-1,+1\}} p(y) \times \int_{\mathbb{R}} p_{\hat{\mu}_y^{(n)}, \hat{\sigma}_y^{(n)}}(f_\theta(X) = \alpha|y) L(y, \alpha)\, d\alpha. \tag{8.14}$$

8.2.1 Asymptotic Normality of $f_\theta(X)|Y$

The quantity $f_\theta(X)|Y$ is essentially a sum of d random variables which for large d is likely to be normally distributed. We examine below three progressingly more general central limit theorems which explore conditions for the normality of $f_\theta(X)|Y$. We also discuss whether these theorems are likely to hold in practice for high dimensional data and show empirically that it is indeed the case for some text and image data.

The original central limit theorem states that $\sum_{i=1}^d Z_i$ is approximately normal for large d if Z_i are iid.

Proposition 5 (de-Moivre). *If $Z_i, i \in \mathbb{N}$ are iid with expectation μ and variance σ^2 and $\bar{Z}_d = d^{-1}\sum_{i=1}^d Z_i$ then we have the following convergence in distribution*

$$\sqrt{d}(\bar{Z}_d - \mu)/\sigma \rightsquigarrow N(0,1) \quad \textit{as } d \to \infty.$$

As a result, the quantity $\sum_{i=1}^d Z_i$ (which is a linear transformation of $\sqrt{d}(\bar{Z}_d - \mu)/\sigma$) is approximately normal for large d. This relatively restricted theorem is unlikely to hold in most practical cases as $X_1, \ldots, X_d$ are often not iid. Moreover, even if $X_1, \ldots, X_d$ are iid, the summands $Z_i = \theta_i X_i$ are not iid.

A more general CLT by Lindberg does not require that the summands Z_i be identically distributed.

Proposition 6 (Lindberg). *For $Z_i, i \in \mathbb{N}$ independent with expectation μ_i and variance σ_i^2, and denoting $s_d^2 = \sum_{i=1}^d \sigma_i^2$, we have the following convergence in distribution as $d \to \infty$*

$$s_d^{-1} \sum_{i=1}^d (Z_i - \mu_i) \rightsquigarrow N(0,1)$$

if the following condition holds for every $\epsilon > 0$

$$\lim_{d\to\infty} s_d^{-2} \sum_{i=1}^d \mathsf{E}(Z_i - \mu_i)^2 1_{\{|X_i - \mu_i| > \epsilon s_d\}} = 0. \tag{8.15}$$

This CLT is more general as it only requires that the data dimensions be independent. The condition (8.15) is relatively mild and specifies that contributions of each of the Z_i to the variance s_d should not dominate it. Nevertheless, the Lindberg CLT is still not satisfactory as in many cases the data dimensions are dependent.

More general CLTs replace the condition that $Z_i, i \in \mathbb{N}$ be independent with the notion of $m(k)$-dependence.

Definition 1. *The random variables $Z_i, i \in \mathbb{N}$ are said to be $m(k)$-dependent if whenever $s - r > m(k)$ the two sets $\{Z_1, \ldots, Z_r\}$, $\{Z_s, \ldots, Z_k\}$ are independent.*

An early CLT for $m(k)$-dependent RVs is [Hoeffding and Robbins, 1948]. Below is a slightly weakned version of the CLT in [Berk, 1973].

Proposition 7 (Berk)**.** *For each $k \in \mathbb{N}$ let $d(k)$ and $m(k)$ be increasing sequences and suppose that $Z_1^{(k)}, \ldots, Z_{d(k)}^{(k)}$ is an $m(k)$-dependent sequence of random variables. If*

1. *$E|Z_i^{(k)}|^2 \leq M$ for all i and k*
2. *$Var(Z_{i+1}^{(k)} + \ldots + Z_j^{(k)}) \leq (j-i)K$ for all i, j, k*
3. *$\lim_{k\to\infty} Var(Z_1^{(k)} + \ldots + Z_{d(k)}^{(k)})/d(k)$ exists and is non-zero*
4. *$\lim_{k\to\infty} m^2(k)/d(k) = 0$*

then $\frac{\sum_{i=1}^{d(k)} Z_i^{(k)}}{\sqrt{d(k)}}$ is asymptotically normal as $k \to \infty$.

Proposition 7 states that under mild conditions the sum of $m(k)$-dependent RVs is asymptotically normal. If $m(k)$ is a constant i.e., $m(k) = m$, $m(k)$-dependence implies that a Z_i may only depend on its neighboring dimensions. Or in other words, dimensions that are removed from each other are independent. The full power of Proposition 7 is invoked when $m(k)$ grows with k relaxing the independence restriction as the dimensionality grows. Intuitively, the dependency of the summands is not fixed to a certain order, but it cannot grow too rapidly.

The asymptotic normality of $f_\theta(X)|Y$ depends on the specific context and should be determined empirically. In many cases the dimensionality d is high. For example, in the case of text documents (X_i is the relative number of times word i appeared in the document) d corresponds to the vocabulary size which is typically a large number in the range $10^3 - 10^5$. Similarly, in the case of image classification (X_i denotes the brightness of the i-pixel) the dimensionality corresponds to the image size and is on the order of $10^2 - 10^4$.

The question of whether such data is $m(k)$-dependent and whether $f_\theta(X)|Y$ is normal can be answered practically. Figure 8.1 answers this question in the affirmative for three separate datasets containing text and image data. Specifically, the variable $f_\theta(X)|Y$ is approximately normal for RCV1 text categorization data [Lewis *et al.*, 2004], MNIST handwritten digit images (http://yann.lecun.com/exdb/mnist/), and face detection images described in [Pham *et al.*, 2002]. This holds broadly both for randomly generated θ and for θ estimated using Fisher's LDA and logistic regression (top 3 rows). We further observe that normality holds for θ obtained using regularized logistic regression with a broad range of regularization parameters governing the amount of sparsity. The distribution of $f_\theta(X)|Y$ deviates from normal for radically sparse θ, as evidenced by the histograms of l_1 regularized logistic regression (last row).

Encouraged by this empirical observation and by the theoretical motivations we proceed in the next section to prove identifiability and unsupervised consistency of the risk estimator, assuming normality of $f_\theta(X)|Y$.

8.2.2 Statistical Consistency

Under the assumptions specified above, in particular that $p(y)$ is known and that $f_\theta(X)|Y$ is normal, the plug-in estimator (8.13) is consistent in the unsupervised sense. In other words, the risk estimator $\hat{R}_n$ converges to the true risk as the amount of unlabeled data increases.

We start with proving identifiability of the maximum likelihood estimator (MLE) for a mixture of two Gaussians with known mixture proportions. Invoking classical consistency results in conjunction with identifiability we show consistency of the MLE estimator for (μ, σ) parameterizing the distribution of $f_\theta(X)|Y$. Consistency of the estimator $\hat{R}_n$ follows.

Definition 2. *A parametric family $\{p_\alpha : \alpha \in A\}$ is identifiable when $p_\alpha(x) = p_{\alpha'}(x), \forall x$ implies $\alpha = \alpha'$.*

Proposition 8. *Assuming known label marginals with $p(y = 1) \neq p(y = -1)$, the Gaussian mixture family*

$$p_{\mu,\sigma}(x) = p(y = 1)N(x\,; \mu_1, \sigma_1^2) + p(y = -1)N(x\,; \mu_{-1}, \sigma_{-1}^2) \tag{8.16}$$

is identifiable.

Proof: It can be shown that the family of Gaussian mixture model with unknown label marginals (that is $p(y)$ in (8.16) is also a parameter) is identifiable up to a permutation of the labels y [Teicher, 1963].

We proceed by assuming with no loss of generality that $p(y = 1) > p(y = -1)$. The alternative case $p(y = 1) < p(y = -1)$ may be handled in the same manner. Using the result of [Teicher, 1963] we have that if $p_{\mu,\sigma}(x) = p_{\mu',\sigma'}(x)$ for all x, then $(p(y), \mu, \sigma) =$

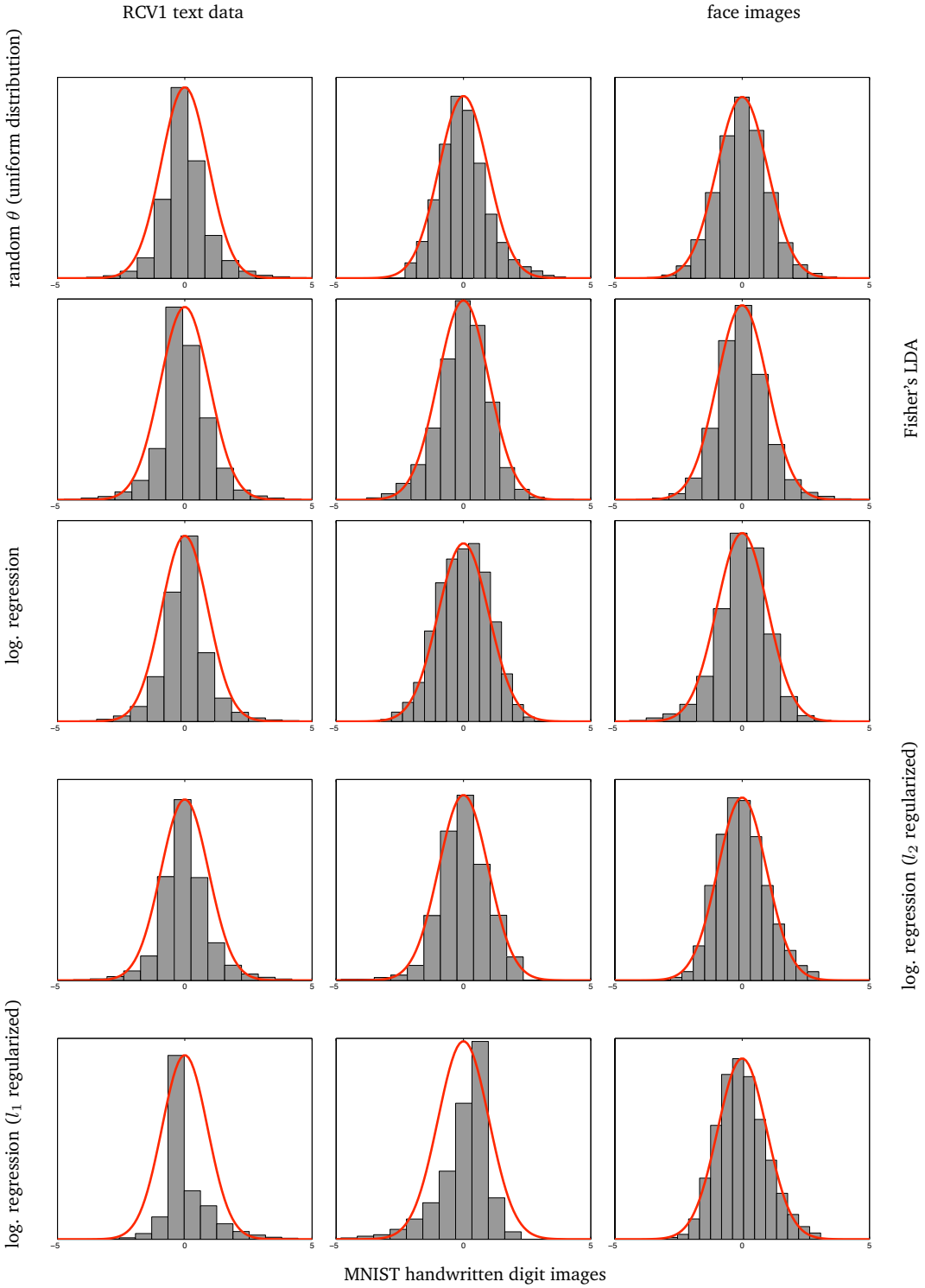


Figure 8.1: Centered histograms of $f_\theta(X)|Y = 1$ overlayed with the pdf of a fitted Gaussian for multiple θ vectors (five rows: random $\theta_i \sim U(-1/2, 1/2)$, Fisher's LDA, logistic regression, l_2 regularized logistic regression, and l_1 regularized logistic regression-all regularization parameters were selected by cross validation) and datasets (columns: Reuters RCV1 text data, MNIST digit images, and face images). The fifteen panels show that even in moderate dimensionality (RCV1: 1000 top words, MNIST digits: 784 pixels, face images: 400 pixels) the assumption that $f_\theta(X)|Y$ is normal holds well (except perhaps for l_1 regularization in the last row which promotes sparse θ).

$(p(y), \mu', \sigma')$ up to a permutation of the labels. Since permuting the labels violates our assumption $p(y=1) > p(y=-1)$ we establish $(\mu, \sigma) = (\mu', \sigma')$ which proves identifiability. ■

Proposition 9. *Under the assumptions of Proposition 8 maximizing* (8.12) *as a function of* $(\mu, \sigma) = (\mu_1, \mu_{-1}, \sigma_1, \sigma_{-1})$ *provides a consistent estimator for the distributions of* $f_\theta(X)|Y=1$ *and* $f_\theta(X)|Y=-1$*. In other words, the sequence of MLE estimators* $(\hat{\mu}_1^{(n)}, \hat{\mu}_{-1}^{(n)}, \hat{\sigma}_1^{(n)}, \hat{\sigma}_{-1}^{(n)})$ *converge as* $n \to \infty$ *to the true parameters values with probability 1.*

Proof: The loglikelihood (8.12) is identical to that of a binary Gaussian mixture with known label marginals which we prove to be identifiable in Proposition 8. Consistency thus follows from classical MLE theory e.g., chapter 17 of [Ferguson, 1996]. (Recall the full list of assumptions including identifiability in Chapter 7.) ■

Proposition 10. *Under the assumptions of Proposition 8 and assuming the loss* L *is given by one of* (8.3)-(8.5)*, the plug-in risk estimate* (8.13) *is consistent i.e.,* $\hat{R}_n(\theta) \to R(\theta)$ *with probability 1.*

Proof: The plug-in risk estimate $\hat{R}_n$ in (8.13) is a continuous function (when L is given by (8.3), (8.4) or (8.5)) of $\hat{\mu}_1^{(n)}, \hat{\mu}_{-1}^{(n)}, \hat{\sigma}_1^{(n)}, \hat{\sigma}_{-1}^{(n)}$ (note that μ_y and σ_y are functions of θ), which we denote $\hat{R}_n(\theta) = h(\hat{\mu}_1^{(n)}, \hat{\mu}_{-1}^{(n)}, \hat{\sigma}_1^{(n)}, \hat{\sigma}_{-1}^{(n)})$.

Using Proposition 9 we have that

$$\lim_{n\to\infty} (\hat{\mu}_1^{(n)}, \hat{\mu}_{-1}^{(n)}, \hat{\sigma}_1^{(n)}, \hat{\sigma}_{-1}^{(n)}) = (\mu_1^{\text{true}}, \mu_{-1}^{\text{true}}, \sigma_1^{\text{true}}, \sigma_{-1}^{\text{true}})$$

with probability 1. Since continuous functions preserve limits we have

$$\lim_{n\to\infty} h(\hat{\mu}_1, \hat{\mu}_2, \hat{\sigma}_1, \hat{\sigma}_2) = h(\mu_1^{\text{true}}, \mu_2^{\text{true}}, \sigma_1^{\text{true}}, \sigma_2^{\text{true}})$$

with probability 1 which implies convergence $\lim_{n\to\infty} \hat{R}_n(\theta) = R(\theta)$ with probability 1. ■

8.3 Experimental Evaluation

8.3.1 Application 1: Estimating Risk in Transfer Learning

We consider applying our estimation framework in two ways. The first application, which we describe in this section, is estimating margin-based risks in transfer learning where classifiers are trained on one domain but tested on a somewhat different domain. The transfer learning assumption that labeled data exists for the training domain but not for the test domain motivates the use of our unsupervised risk estimation. The second application, which we

describe in the next section, is more ambitious. It is concerned with class partitioning without labeled data whatsoever.

In evaluating our framework we consider both synthetic and real-world data. In the synthetic experiments we generate high dimensional data from two uniform distributions $X|Y=1$ and $X|Y=-1$ with independent dimensions and certain perscribed $p(Y)$ and classification difficulty. This controlled setting allows us to examine the accuracy of our unsupervised logloss risk estimator as a function of n, $p(Y)$, and the classifier accuracy.

Figure 8.2 shows that the relative error in estimating the logloss and hingeloss decreases with n achieving accuracy of greater than 99% for $n > 1000$. Interestingly, the figure shows that the estimation error decreases as the classifiers become more accurate and as $p(Y)$ becomes less uniform. We found these trends to hold in other experiments as well. In the case of exponential loss, however, the estimator performed substantially worse. This is likely due to the exponential dependency of the loss on $Yf_\theta(X)$ which makes it very sensitive to outliers.

Figure 8.3 shows the accuracy of logloss estimation for a real world transfer learning experiment based on the 20-newsgroup data. Following the experimental setup of [Dai *et al.*, 2007; Nigam, 2001] we trained a classifier (logistic regression) on one 20 newsgroup classification problem and tested it on a related problem. Specifically, we used the hierarchical category structure to generate train and test sets with different distributions (see Figure 8.3 and [Dai *et al.*, 2007] for more detail). Log-loss estimation was quite effective with relative accuracy greater than 96% and absolute error of up to only 0.02.

8.3.2 Application 2: Unsupervised Class Partition with Known Class Prior

Our second application is class partition of the unlabeled data using only $p(Y)$. This partitioning is used to make label predictions. We measure the performance of the predictor as a function of the unsupervised train set size n, in terms of $\hat{R}_n$ (8.13) and in terms of the supervised logloss estimate R_n (8.8) (labels were used only in evaluation).

More specifically, we consider two algorithms (see Algorithms 9-10) that start with an initial $\theta^{(0)}$ and iteratively construct a sequence $\theta^{(1)}, \ldots, \theta^{(T)}$ which steadily improve the unsupervised logloss estimate (8.13) $\hat{R}_n(\theta^{(t)}) \leq \hat{R}_n(\theta^{(t-1)})$ $t = 1, \ldots, T$, as computed based on an unlabeled training set of size n. Although we focus on minimizing unsupervised logloss estimate, the same techniques may be generalized to other margin-based risk functions such as hinge loss.

Algorithm 9 adopts a gradient descent-based optimization. At each iteration t, it approximates the gradient vector $\nabla\hat{R}_n(\theta^{(t)})$ numerically using a finite difference approximation. Algorithm 10 proceeds by constructing a grid search along every dimension of $\theta^{(t)}$ and set $[\theta^{(t)}]_i$ to the grid value that minimizes $\hat{R}_n$.

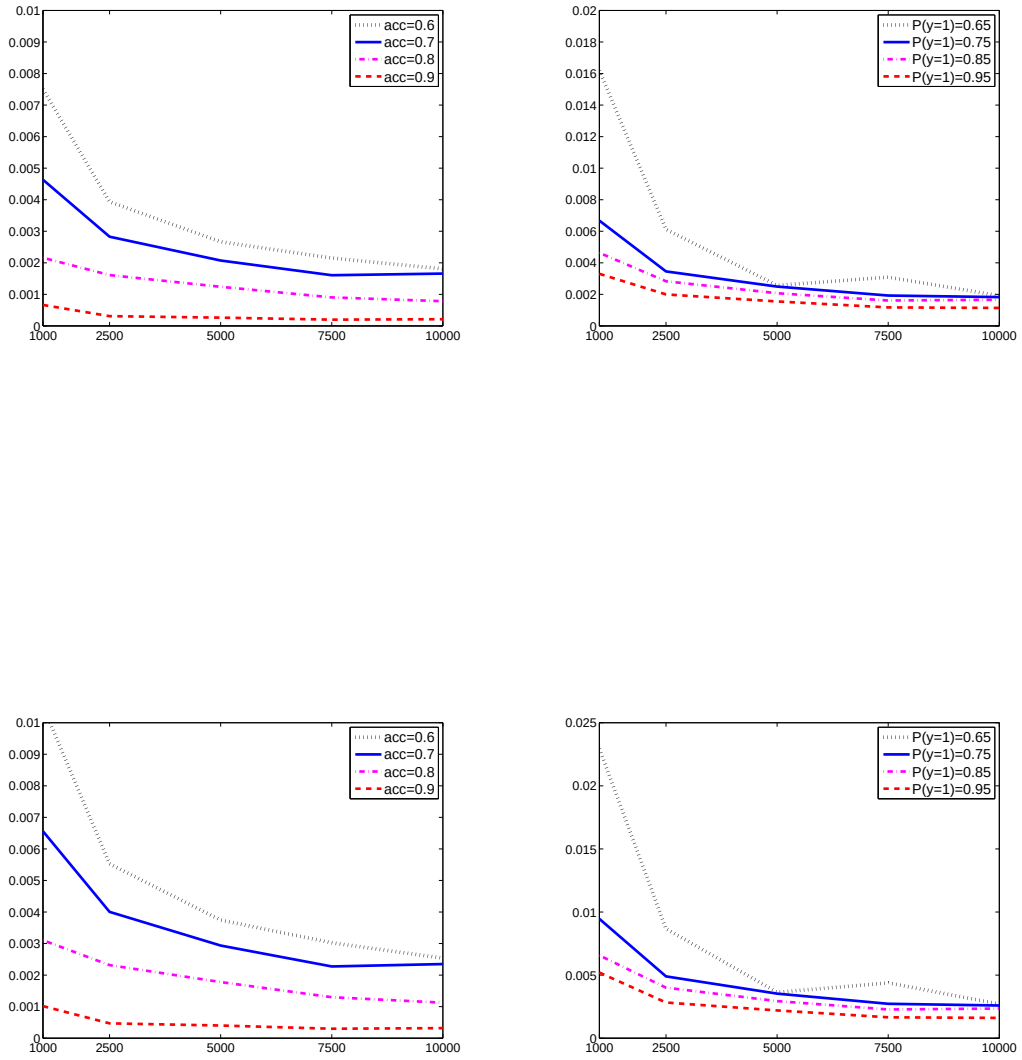


Figure 8.2: The dependence of $|\hat{R}_n - R_n|/R_n$ for logoss (top) and hingeloss (bottom), based on synthetic data, on the number of unlabeled examples n and how it changes with the classifier accuracy (acc) and the label marginal $p(Y)$. The logloss estimation generally decreases nicely with n (approaching 1% relative error at $n = 1000$ and decaying further). The estimation error decreases with the accuracy of the classifier (left) and with non-uniformity of $p(Y)$.

Data	R_n	$\lvert R_n - \hat{R}_n \rvert$	$\lvert R_n - \hat{R}_n \rvert / R_n$	n	$p(Y=1)$
sci vs. comp	0.7088	0.0093	0.013	3590	0.8257
sci vs. rec	0.641	0.0141	0.022	3958	0.7484
talk vs. rec	0.5933	0.0159	0.026	3476	0.7126
talk vs. comp	0.4678	0.0119	0.025	3459	0.7161
talk vs. sci	0.5442	0.0241	0.044	3464	0.7151
comp vs. rec	0.4851	0.0049	0.010	4927	0.7972

Figure 8.3: Error in estimating logloss for logistic regression classifiers trained on one 20-newsgroup classification task and tested on another. We followed the transfer learning setup which may be referred to for more detail. The train and test sets contained samples from two top categories in the topic hierarchy but with different subcategory proportions. As a result, the train and test distributions are similar but not identical. The first column indicates the top category classification task. The second column indicates the empirical log-loss R_n calculated using the true labels of the test set (8.8). The third and forth columns indicate the absolute and the relative errors of the unsupervised logloss estimates. The fifth column n is the test set size and the last column is the label marginal $p(y=1)$.

Algorithm 9 Unsupervised Gradient Descent

Input: $X^{(1)}, \ldots, X^{(n)} \in \mathbb{R}^d$, $p(Y)$, step size α
Initialize $t=0$, $\theta^{(t)} = \theta^0 \in \mathbb{R}^d$
repeat
  Compute $f_{\theta^{(t)}}(X^{(j)}) = \langle \theta^{(t)}, X^{(j)} \rangle \; \forall j = 1, \ldots, n$
  Estimate $(\hat{\mu}_1, \hat{\mu}_{-1}, \hat{\sigma}_1, \hat{\sigma}_{-1})$ by maximizing (8.12)
  for $i = 1$ **to** d **do**
    Plug-in the estimates into (8.13) to approximate

$$\frac{\partial \hat{R}_n(\theta^{(t)})}{\partial \theta_i} = \frac{\hat{R}_n(\theta^{(t)} + h_i e_i) - \hat{R}_n(\theta^{(t)} - h_i e_i)}{2h_i} \tag{8.17}$$

(e_i is an all zero vector except for $[e_i]_i = 1$)

  end for
  Form $\nabla \hat{R}_n(\theta^{(t)}) = (\frac{\partial \hat{R}_n(\theta^{(t)})}{\partial \theta_1}, \ldots, \frac{\partial \hat{R}_n(\theta^{(t)})}{\partial \theta_d})$
  Update $\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla \hat{R}_n(\theta^{(t)})$, $t = t+1$
until convergence
Output: $\theta^{\text{final}} = \theta^{(t)}$

We tested the two algorithms on two real-world datasets: Reuters RCV1 text categorization and MNIST digit recognition datasets. In the case of RCV1 we discarded all but the most frequent 504 words (after stop-word removal) and represented documents using their tfidf scores. We experimented on the binary classification task of distinguishing the top category

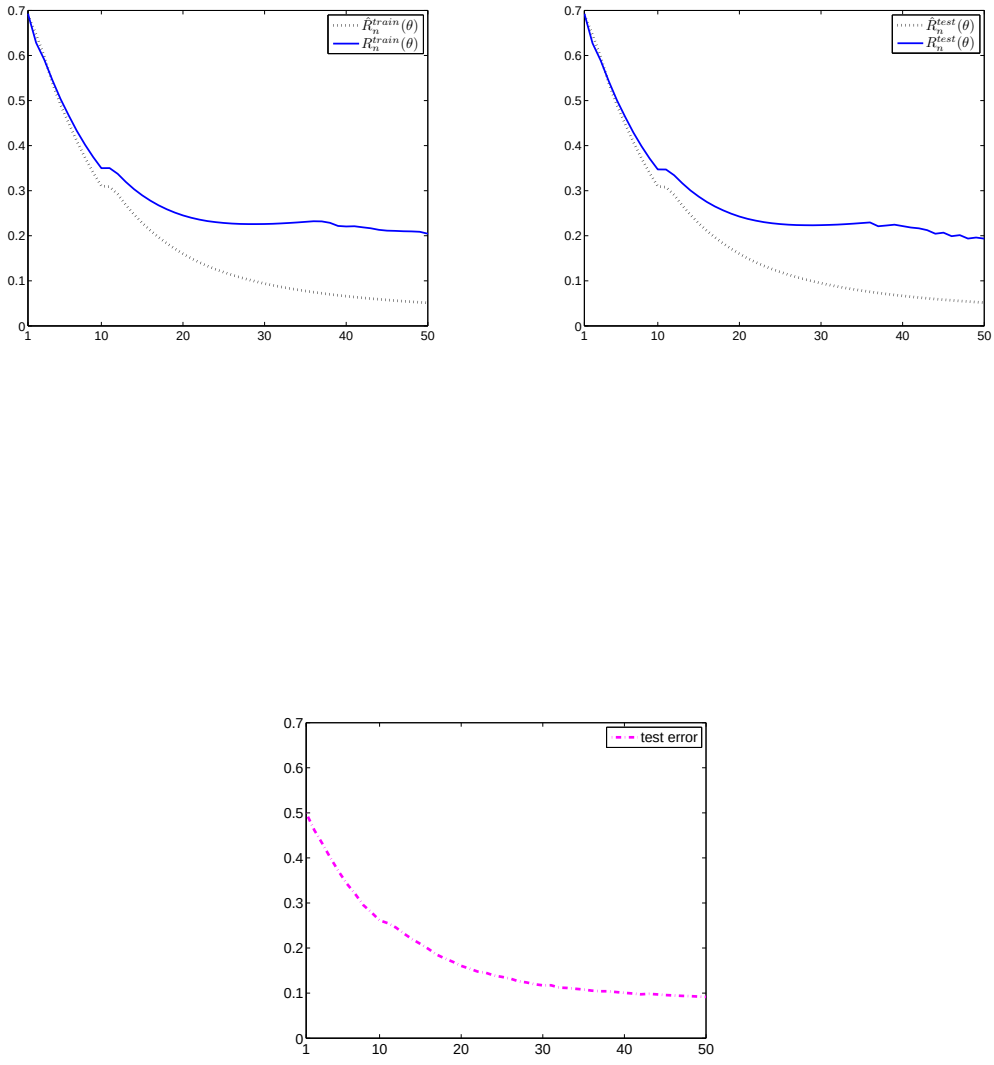


Figure 8.4: Estimation accuracy of classifiers learned by minimizing the unsupervised logloss estimate $\hat{R}_n$ (8.13) on RCV1 data. The panels display the performance of the learned classifier in terms of the unsupervised $\hat{R}_n$ and the supervised R_n logloss estimates based on the training set (left), based on the test set (middle) and the test classification error rate (right). The performance criteria are plotted as a function of the iteration number of Algorithm 9 (gradient descent). The figure shows that the algorithm obtains a relatively accurate classifier (test set error rate 0.1, and $\hat{R}_n$ decaying similarly to R_n) without the use of a single labeled example. See text for more detail.

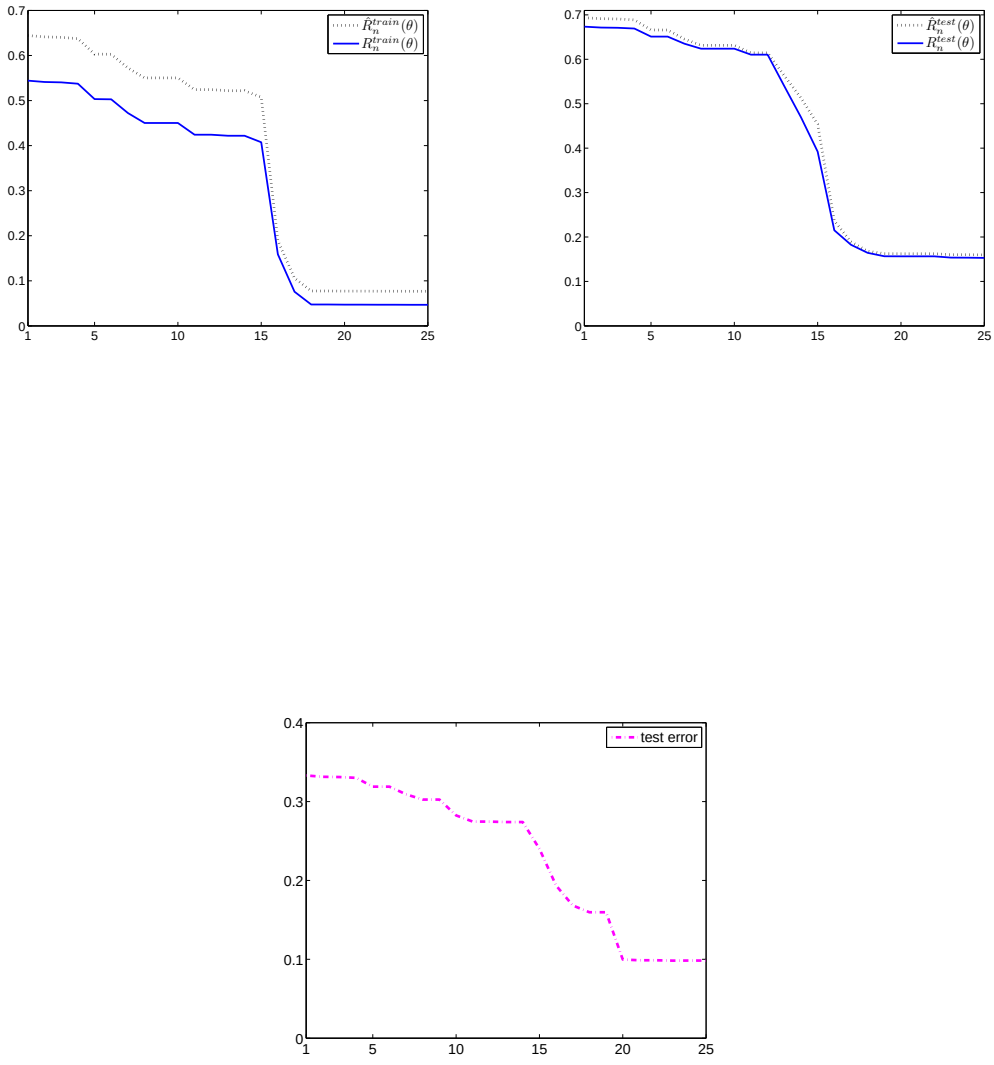


Figure 8.5: Estimation accuracy of classifiers learned by minimizing the unsupervised logloss estimate $\hat{R}_n$ (8.13) on RCV1 data. The panels display the performance of the learned classifier in terms of the unsupervised $\hat{R}_n$ and the supervised R_n logloss estimates based on the training set (left), based on the test set (middle) and the test classification error rate (right). The performance criteria are plotted as a function of the iteration number of Algorithm 10 (grid search). The figure shows that the algorithm obtains a relatively accurate classifier (test set error rate 0.1, and $\hat{R}_n$ decaying similarly to R_n) without the use of a single labeled example. See text for more detail.

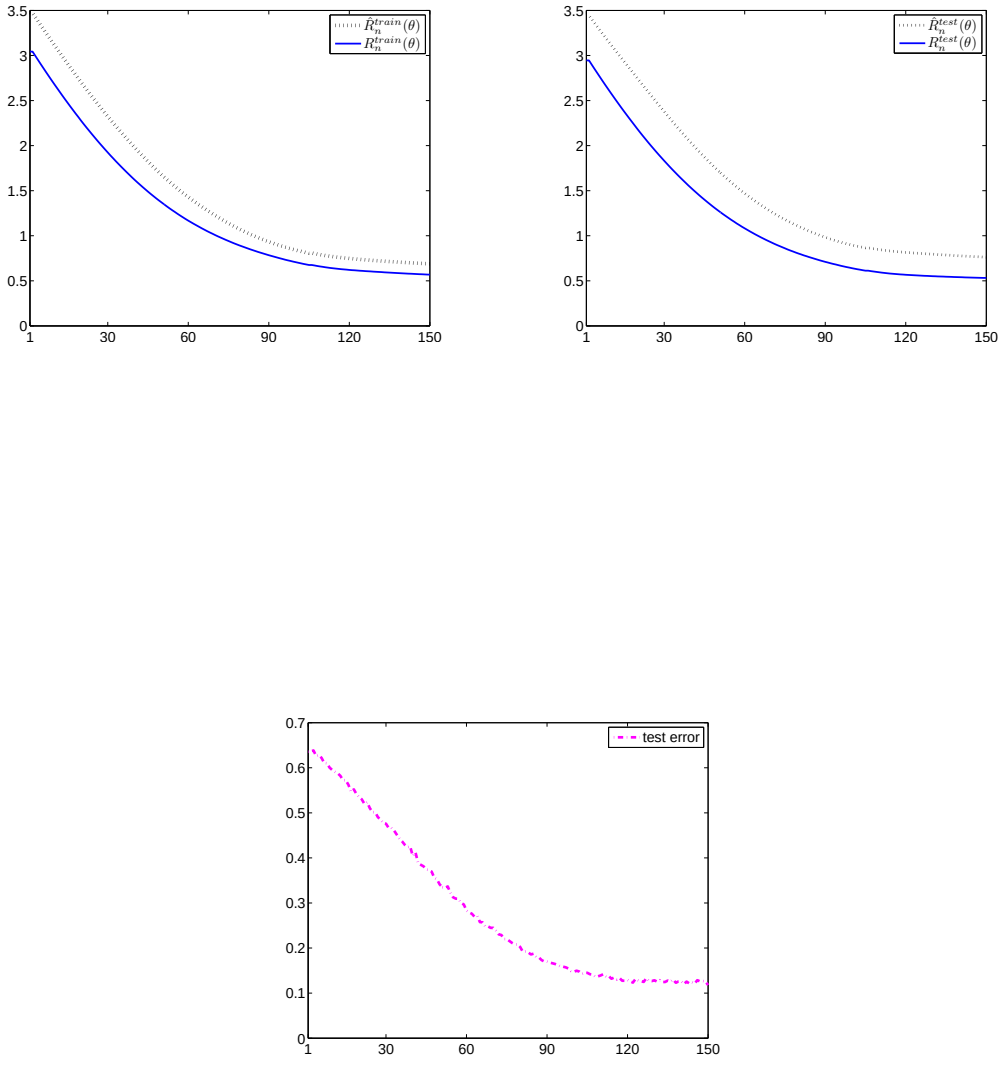


Figure 8.6: Estimation accuracy of classifiers learned by minimizing the unsupervised logloss estimate $\hat{R}_n$ (8.13) on the MNIST data. The panels display the performance of the learned classifier in terms of the unsupervised $\hat{R}_n$ and the supervised R_n logloss estimates based on the training set (left), based on the test set (middle) and the test classification error rate (right). The performance criteria are plotted as a function of the iteration number of Algorithm 9 (gradient descent). The figure shows that the algorithm obtains a relatively accurate classifier (test set error rate 0.1, and $\hat{R}_n$ decaying similarly to R_n) without the use of a single labeled example. See text for more detail.

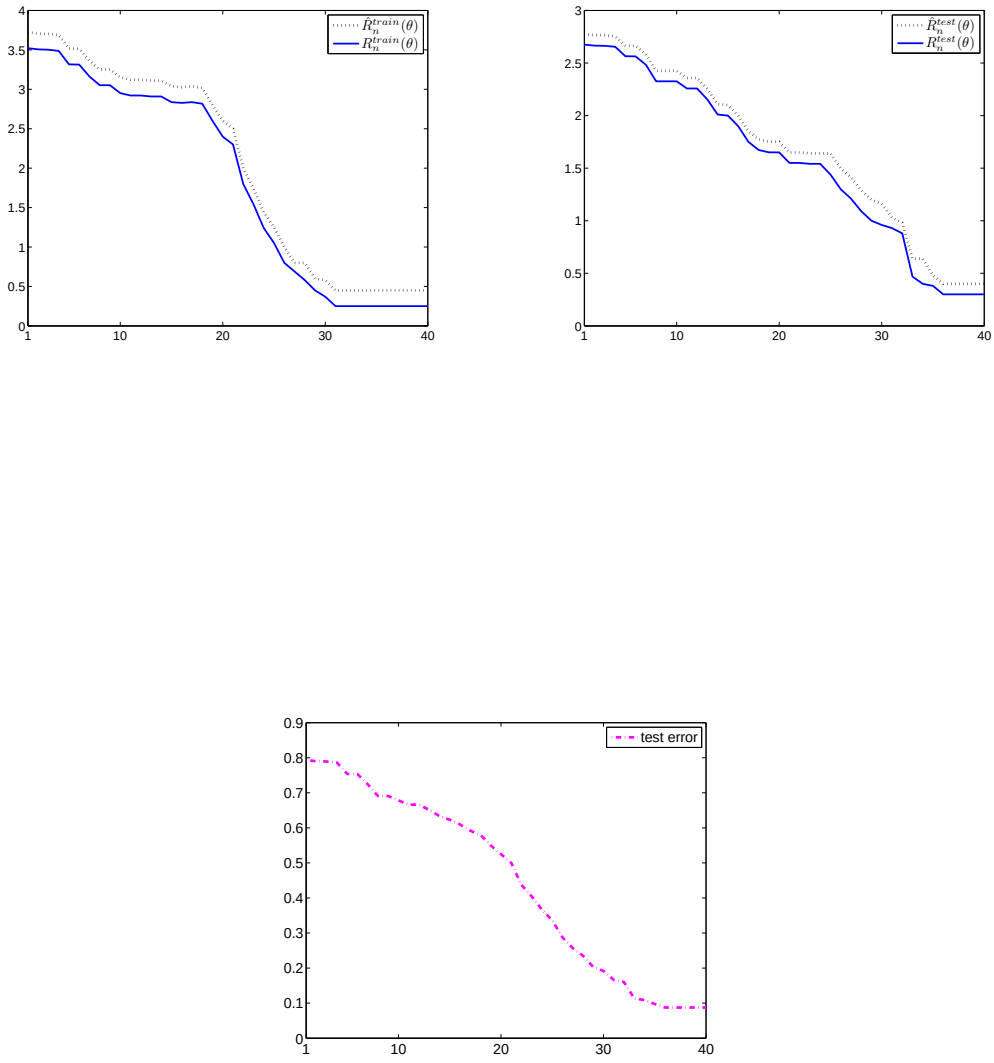


Figure 8.7: Estimation accuracy of classifiers learned by minimizing the unsupervised logloss estimate $\hat{R}_n$ (8.13) on MNIST data. The panels display the performance of the learned classifier in terms of the unsupervised $\hat{R}_n$ and the supervised R_n logloss estimates based on the training set (left), based on the test set (middle) and the test classification error rate (right). The performance criteria are plotted as a function of the iteration number of Algorithm 10 (grid search). The figure shows that the algorithm obtains a relatively accurate classifier (test set error rate 0.1, and $\hat{R}_n$ decaying similarly to R_n) without the use of a single labeled example. See text for more detail.

Algorithm 10 Unsupervised Grid Search

Input: $X^{(1)}, \ldots, X^{(n)} \in \mathbb{R}^d$, $p(Y)$, grid-size τ
Initialize $t = 0$, $\theta_i^{(0)} \sim U(-2, 2)$ for all i
repeat
  for $i = 1$ **to** d **do**
    Construct τ points grid in the range $[\theta_i^{(t)} - 4\tau, \theta_i^{(t)} + 4\tau]$
    Compute the risk estimate (8.13) where all dimensions of $\theta^{(t)}$ are fixed except for $[\theta^{(t)}]_i$ which is evaluated at each grid point.
    Set $[\theta^{(t+1)}]_i$ to the grid value that minimized (8.13)
  end for
  Update $t = t + 1$
until convergence
Output: $\theta^{\text{final}} = \theta^{(t)}$

(positive) from the next 4 top categories (negative) which resulted in $p(y = 1) = 0.3$ and 199328 samples.

70% of the data was chosen as (unlabeled) training set and the rest was held-out as a test-set. Figures 8.4-8.5 display the logloss estimates (both the unsupervised $\hat{R}_n$ and the supervised R_n) on the training and test sets as well as the test set error rate on RCV1 data. The class partitions were constructed by Algorithm 9 in the case of Figure 8.4 and by Algorithm 10 in the case of Figure 8.5.

The results indicate that minimizing the unsupervised logloss estimate is quite effective that it correlates very well with the partition according to the true classes. Both algorithms reached test set error rate of 0.1 after 50 iterations (for gradient descent) and 25 iterations (for grid search). Furthermore the two lines corresponding to the unsupervised $\hat{R}_n$ and the supervised R_n decrease with the number of iterations t for both the train and test set. The improvement in accuracy and logloss was smoother for the gradient descent than for the grid search. The gap between the two lines in the case of the gradient descent is a result of several outliers which affected R_n more than it affected $\hat{R}_n$ due to the Gaussian fitting.

In the case of MNIST data, we normalized each of the $28 \times 28 = 784$ pixels to have 0 mean and unit variance. Our classification task was to distinguish images of the digit one (positive) from the digit 2 (negative) resulting in 14867 samples and $p(Y = 1) = 0.53$. We randomly choose 70% of the data as a training set and and kept the rest as a test set.

Figures 8.6 and 8.7 show the performance of the class partition on the MNIST dataset using the gradient descent and the grid search algorithms, respectively. The results are similar to those obtained on the RCV1 dataset. The predicted labels due to partitioning have test-set classification error rate of 0.1 and the decay of the train-set and test-set estimate $\hat{R}_n$ as a function of the iteration number t closely mirrorred the behavior of the supervised

criterion R_n.

8.3.3 Inaccurate Specification of $p(Y)$

Our estimation framework assumes that the class prior $p(Y)$ is known. In some cases it is not possible to know $p(Y)$, but rather only an inaccurate estimate of $p(Y)$ may be available. It is instructive to consider how the performance degrades with the inaccuracy of the assumed $p(Y)$.

Figure 8.8 displays the performance of the learned classifier for RCV1 data as a function of the assumed value of $p(Y = 1)$ (correct value is $p(Y = 1) = 0.3$). We conclude that knowledge of $p(Y)$ is an important component in our framework but precise knowledge is not crucial. Small deviations of the assumed $p(Y)$ from the true $p(Y)$ result in a small degradation of logloss estimation quality and test set error rate. Naturally, large deviation of the assumed $p(Y)$ from the true $p(Y)$ renders the estimation framework ineffective.

8.4 Chapter Conclusions

We proposed a novel framework for estimating margin-based risks using only unlabeled data. We showed that it performs well in practice on several different datasets. We derived a theoretical basis for it by casting it as a maximum likelihood problem for Gaussian mixture model followed by plug-in estimation. Remarkably, the theory states that assuming normality of $f_\theta(X)$ and a known $p(Y)$ we are able to estimate the risk $R(\theta)$ without a single labeled example. Moreover, if we can accurately estimate $R(\theta)$ for all θ without labels then we can also estimate the risk minimizer $\arg\min_\theta R(\theta)$ which leads to unsupervised training of logistic regression and SVM.

The potential benefits of the unsupervised class partition are extensive. Costly and noisy labeling is a serious burden. Hence, researchers have focused on developing methods to learn with fewer labeled instances, i.e., semi-supervised learning, active and proactive learning. Unsupervised learning, on the other hand, generally deals with clustering and grouping the data in the absence of labels. This new paradigm we have introduced contributes to unsupervised learning by allowing unsupervised estimate of margin-based risk functions which leads to effective class partiton without any single labeled example.

The normality of $f_\theta(X) \mid y$ assumption is quite essential in our unsupervised risk estimation framework. It is required to establish consistent estimators of the risk together with the knowledge of $p(Y)$. Theoretically, the Central Limit Theorem establishes normality when the dimensionality $d \to \infty$ under progressively less restrictive independence assumptions. In practice, Figure 8.1 shows whether normality assumption holds regarding different choices of θ. Luckily, it shows that the normality assumption holds for various different cases except

for very sparse θ resulting from l_1 regularization. In practice, normality needs to be checked within a context, depending both on the data (X, y) and θ. If this assumption is violated, our consistency result is no longer valid. Both proposed algorithms (Algorithms 9-10) compute the risk estimate at each step by maximizing (8.12). Since the risk is estimated iteratively, the normality assumption should hold for θ found at each step. In other words, $f_{\theta^{(t)}}(X) \mid y$ should be normally distributed for every $\theta^{(t)}$ in the search path. If this assumption is violated at any step, the risk estimation is not guaranteed to be consistent. Hence, the risk estimator might be inaccurate resulting an inaccurate class partition. We have observed that one of the reasons for deviation from normality is having very sparse θ.

What we have pointed out above is a general condition that violates our assumptions and hence our system might fail to provide the desired class partition. A similar concern arises due to multiple target partitions with equi-class marginals. Note that the proposed unsupervised training algorithms are deterministic; i.e., given the same input ($X^{(1)}, \ldots, X^{(n)} \in \mathbb{R}^d$ and $p(Y)$) they will return the same θ^{final}. Then, one might wonder about the case where there are alternate partitions of the data with equivalent class priors. In other words, there might be multiple target functions with the same marginal $p(Y)$ but corresponding to different partitions of the same data. For instance, assume one wishes to classify his emails according to two different target functions: $f_{\theta_1} : X \rightarrow \{spam, non-spam\}$ and $f_{\theta_2} : X \rightarrow \{academic, personal\}$ for any $X \in \mathbb{R}^d$. Assume that the corresponding marginals are equal; i.e. $p_1(y_1) = p_2(y_2)$ where $y_1 = spam$ and $y_2 = academic$ and vice versa. If the task is to estimate the margin-based risk of $f_{\theta_1}(X)$ and $f_{\theta_2}(X)$, then our framework yields consistent estimators given our assumptions hold. If the task is unsupervised training and the same initial θ^0 is adopted for both problems then our system would yield the same solution θ^{final} in two cases since the input data to the unsupervised learner is the same. Hence, the system cannot distinguish between which target function it learns. Algorithmically, this problem may be addressed if coupled with semi-supervised learning or active learning. Minimal labeled data (determined apriori or by active learning) from each target distribution can be used to establish an initial guess θ^0 for two problems. This could then be used to initialize Algorithms 9-10, which iteratively update it minimizing the risk estimate. Hence, the resulting class partitions are more likely to resemble the target solution.

Apart from these important considerations, our approach also points at novel questions for future extensions. What benefit do labels provide over unsupervised training? Can our framework be extended to semi-supervised learning where a few labels do exist? Can it be extended to the multiclass (and also multi-label) case and to non-classification scenarios such as margin based regression or margin based structured prediction? When are the assumptions likely to hold and how can we make our framework even less resistant to deviations from them? What would be the benefits of using minimal labeled data to enhance the learned models, if any? What could we theoretically conclude about the convergence of our algorithms in the case of multi-label (multiple target functions) scenarios? These questions and others form new and exciting open research directions for future work.

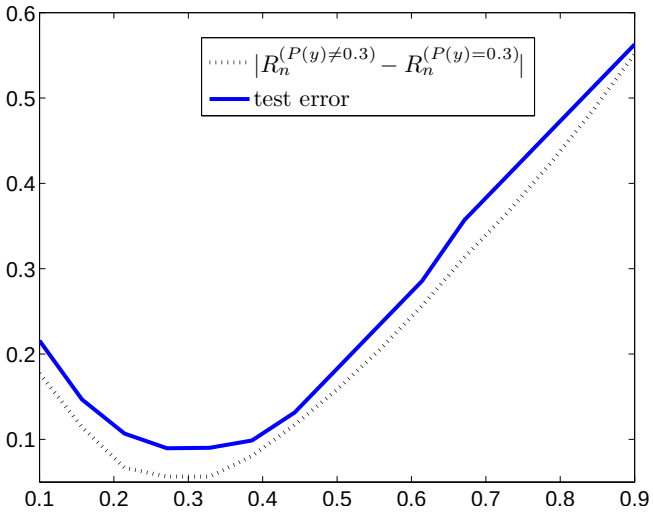


Figure 8.8: Performance of unsupervised classifier training on RCV1 data (top class vs. classes 2-5) for misspecified $p(Y)$. The performance of the estimated classifier (in terms of train set empirical logloss R_n (8.8) and test set error rate measured using held-out labels) as a function of the amount of deviation between the assumed and true $p(Y = 1)$ (true $p(Y = 1) = 0.3$)). The classifier performance is extremely good when the assumed $p(Y)$ is close to the truth and degrades relatively gracefully with the amount of misspecification.

Chapter 9

Conclusions and Future Directions

9.1 Summary

In this thesis, we presented novel machine learning techniques to address active learning and proactive learning. In particular, we focused on traditional active learning for classification and rank learning problems as well as learning with multiple imperfect predictors without any apriori information about their qualities such as reliability or reluctance. More specifically, we focused on estimating the accuracy or the risk of multiple noisy predictors in the absence of gold standard labels and consequently selecting the most reliable one(s). The proposed framework is also useful to predict true labels given the accuracy (risk) estimates and the noisy output of the predictors. Figure 9.1 provides a summary of the techniques developed during this thesis work, their main functionality (e.g. instance selection, predictor accuracy estimation, etc.), and the corresponding assumptions made. We hope that it will give the reader a useful reference for a quick review of the thesis work.

In this thesis, we first introduce two new ensemble methods to actively learn which instances to label in classification problems. While the objective is to minimize the empirical risk of the supervised learning algorithm, there are two commonly used approaches to achieve this goal. One is referred as the uncertainty sampling, which selects the most ambigious instances for the learner to discriminate. The other is the density-based sampling which focuses on selecting the most representative samples to boost the learning rate. First, we proposed DUAL, a dynamic ensemble approach which learns to switch from density-based sampling to an uncertainty-density ensemble as the classification decision boundary emerges with increasing accuracy. Several experiments demonstrated that DUAL outperforms previous state-of-the-art fixed-strategy methods and even fixed ensemble methods such as [Baram *et al.*, 2003]. Second, we presented another ensemble method for active learning that combines density-based and uncertainty sampling into a utility function

Method	Major functionality	Assumptions
DUAL	instance selection for classification	Gaussian mixture model is used to model data density.
Paired Sampling	instance selection for classification	Decision boundary should lie in low density regions and hence should not cut clusters.
AUC Optimization	instance selection for ranking	The true ranking function maximizes score difference between the lowest ranked relevant and the highest ranked non-relevant instances.
Loss Differential	instance selection for ranking	The change in the current hypothesis is correlated with the loss of the learner.
Decision-Theoretic Proactive Learning	instance and predictor selection	Restricted to 2 predictors: one perfectly reliable (expensive) and one unreliable (e.g. fallible, reluctant but cheap). A pre-defined and fixed budget is assumed. Unreliability of a predictor is a function of the uncertainty to classify an instance. Predictors behave similarly for similar instances.
IEThresh	predictor accuracy estimation and predictor selection	Individual predictor accuracy is better than random guess and predictors make uncorrelated errors.
SFilter	predictor accuracy estimation and predictor selection in non-stationary conditions	Individual predictor accuracy is better than random guess and predictors make uncorrelated errors at every time step. The rate of change σ of the predictor accuracy is known or at least its upper bound is known. A first-order HMM process models the changing predictor accuracy.
MLE for $0-1$ risk	$0-1$ risk estimation	$p(y)$ is known and the predictors output the noisy label according to some probability distribution. The risk of the predictor is a continuous function of θ, the parameter that governs the generation of the noisy labels.
MLE for margin-based risk	margin-based risk estimation and unsupervised class partition	$p(y)$ is known. Restricted to margin-based linear predictors. The predictor margin $f_\theta(X) \mid Y$ is normally distributed with unknown mean and variance.

Figure 9.1: A table summary of all the techniques described in this thesis. The left column indicates the name of the algorithm followed by its major functionality on the middle column. The last column indicates the assumptions made for the corresponding technique.

aiming to straddle the decision boundary. The utility function serves as a scoring function to sample instances that are in the dense regions of the input space as well as have high uncertainty. Moreover, it favors instances that are likely to have opposite labels to speed up the learning process. The likelihood of the two data points to have opposite class labels is measured as a function of their pairwise distance according to a non-Euclidean distance metric exploiting the natural grouping of the data. The empirical results indicated that this density-sensitive strategy is quite effective in significantly outperforming the popular active sampling methods such as uncertainty sampling, representative sampling, and density-based sampling.

Direct application of the active sampling techniques to rank learning is troublesome due to a number of factors. First of all, active sampling methods for classification try to minimize the classification error; hence, do not take into account the rank order which is crucial in ranking applications. Errors at the top ranks must be penalized more than those at the bottom. On the other hand, most ranking datasets are drawn from very skewed distributions where the more relevant items constitute a small minority. Owing to such factors, it is at best difficult to apply active sampling methods designed for classification to rank learning problems. We address these issues by proposing two different active rankers. One of them focuses on minimizing the hinge loss of an SVM-based rank learner, relying on the fact that minimizing the hinge rank loss (rank version of standard hinge loss) is an accurate approximation to maximize a ranking quality measure (AUC) [Steck, 2007]. Hence, sampling instances that have the biggest contribution to the hinge rank loss is a step in the right direction to reduce the hinge loss and hence to increase the AUC. The second technique we proposed for an active ranker aims to maximize the estimated loss differential over unlabeled data. The motivation to consider the loss differential is based on efficiently estimating the change in the expected loss over the test set. The naive implementation to compute the expected loss of a learner trained on an augmented set with additional labeled examples is almost intractable for large-scale datasets, typical in ranking applications. Thus, we propose a very efficient alternative that considers the likelihood of an instance to change the current hypothesis significantly. This results in a significant change in the version space, which in turn implies a greater chance to learn the true hypothesis faster. We proposed two alternative methods to perform loss differential: one for RankSVM and the other for RankBoost, which are both state-of-the-art rank learning algorithms. The experimental evaluation on real-life corpora was significantly in favor of both sampling strategies over strong baselines.

For the remainder of the thesis, we focused on proactive learning, first considering predictors with differing properties such as reluctance, fallibility and so on, and later dealing with a more specific case of learning with larger numbers of fallible (noisy) predictors. In the first case, we presented a decision-theoretic framework which relies on a utility function to select predictor-instance pairs. The utility is a function measuring the information value of labeling a particular instance by a particular predictor at unit cost where the

cost is the labeling fee of that predictor. The fee can represent a monetary value if the predictors are hired human annotators, or it can be the computational time in the case of algorithmic learners. Our framework is general enough to incorporate any cost function given the costs are normalized into the same range across the predictors. Additionally, our framework assumes the quality of an answer from an imperfect predictor depends on the difficulty of the classification task. We estimated the response characteristics through belief propagation during a discovery phase, relying on a comparison of the answers with the ground truth obtained from a perfectly reliable, but more costly predictor. Results were positive: significant improvement over random predictor selection and over any single predictor in terms of classification error versus the total labeling cost.

Then, we focused on learning with multiple predictors which make random labeling mistakes in the absence of gold standard labels. First, we studied estimating the reliability of the predictors while selecting the best ones along the way. Naturally, this requires a trade-off between exploration vs. exploitation. We analyzed this problem separately in two cases: predictors with stationary and time-varying accuracies. For the first condition, we proposed a simple yet effective approach relying on interval estimation from the reinforcement learning and the multi-armed bandit literature. The proposed method called IEThresh allowed us to filter out less reliable predictors early in the learning phase; thus saving substantial labeling effort and increasing the quality of the labels. A thorough analysis demonstrated the strength of this technique to identify the best predictors in a highly cost-effective manner. To solve the problem for non-stationary predictors, we presented a novel method based on Sequential Bayesian estimation. In particular, we relied on a variant of the particle filtering algorithm to estimate the expected accuracy of each predictor at each time step via maintaining a set of weighted samples and updating these weights using Bayes updates. The estimated accuracies were then used to decide which predictors to query next. Our empirical analysis showed that our method was capable of tracking the true accuracies reliably over time while simultaneously selecting the highest quality ones for labeling. Good-quality labeled data can thus be obtained without extensive labeling queries.

Finally, we introduced an alternative approach to predictor quality estimation based on a risk estimation framework. We approached the problem using two different risks: 0-1 risk (aka the classification error) and a more general risk involving margin-based classifiers and associated continuous loss functions such as logloss. For the first case, we formulated the problem such that the noisy labels are generated from a distribution governed by a specific parameter for each predictor. Then, the risk was represented as a continuous function of this parameter. The risk parameter was estimated by maximizing the likelihood of the observed data (noisy labels obtained from the predictors) marginalized over the true label y. We assumed that the marginal label distribution $p(y)$ to be known. Thus, we proved the maximum likelihood estimator is statistically consistent under mild conditions such as requiring weak predictors. That is, the estimator converges to the true value as the number of samples increases. This is a very desirable statistical property since in many applications

enormous amount of unlabeled data is available with little cost. For instance, huge amounts of unlabeled text are readily available through online data sources such as web pages, news articles, emails. Thorough empirical evaluation supported the theoretical guarantees and showed the robustness of the proposed framework against the assumptions.

The other risk estimation framework considers estimating the margin-based risk using exclusively unlabeled data. The most profound benefit of such a risk estimation is to perform class partitioning through minimizing the risk estimate obtained over only unlabeled data. Unlabeled data thus far has been used in semi-supervised learning, active and proactive learning. Unlabeled data in these tasks serves as a source of information that contains evidence about hidden class labels. However, we proposed a more significant way unlabeled data can be adopted, namely to partition the data according to classes requiring no labeled data. Our estimation framework only requires the label distribution $p(y)$ and the normality of the margin $f_\theta(X) \mid y$, which is necessary to obtain consistent risk estimators. The normality assumption has been shown to hold in real-world text and image datasets with high dimensions, and possible to hold in various other domains. Several experiments on both synthetic and real-world datasets indicated the effectiveness of the proposed framework. The unsupervised class partition reached $\sim 90\%$ test accuracy on both popular RCV1 and MNIST datasets. The test accuracy is calculated on a held-out set for which the labels are used solely for evaluation. Obtaining such accurate predictions is a remarkable result which a lot of practical applications suffering from label scarcity can benefit.

9.2 Future Directions

This thesis introduced novel active sampling methods and the new paradigm of proactive learning to extend active learning to more complex and challenging yet more realistic tasks. There are many directions for future extensions several of which we examine below.

In terms of active learning, one of the main future directions to pursue is to extend dynamic ensemble methods. Recall that our method DUAL, while powerful, still suffers from failure to detect reliably the correct switching point which caused a significant performance drop. An ideal ensemble method should be able to recognize the right operating range to apply the right sampling method and adjust appropriately depending on the context. This requires estimation of the future test error of each method and selecting the one with the lowest error. One of the main challenges is accurate estimation of this error with limited labeled data. We hypothesize that the machine learning techniques we developed for unsupervised risk estimation can be adopted for this goal. However, assumptions such as better-than-random predictors might be difficult to satisfy or evaluate. Questions such as these and the frequency of the method switch are important and should be examined as future research.

Beyond active learning, there are possible venues for application and extension of the multi-predictor framework. We now discuss these extensions. As discussed earlier, proactive learning deals with multiple predictors with differing characteristics when no apriori information is available. We investigated three cases (fallibility, reluctance and variable-cost) through simple scenarios that involves a perfect oracle. We have not fully studied all predictor properties; for example, predictors with a specific area of expertise and how they behave when confronted with problems out of their specialty. Furthermore, we have not addressed predictors having a combination of these features. Ideally, it is important to develop a unified framework that can detect the unknown features of each predictor in a multi-predictor setting and estimate the degree (level) of these features and/or parameters characterizing them. A divide-and-conquer approach might be particularly useful in this case. That is, dividing the objective into several smaller and tractable problems and then merging the solutions with a unified system. In fact, we have developed novel machine learning methods in the case of fallible predictors to estimate their reliability and make predictions on the new test examples. Similar techniques could be developed to infer other predictor features. For instance, reluctance might be represented as a probabilistic function with a parameter controlling the degree of reluctance. Then, the problem reduces to estimating this parameter. The situation with multiple features occuring in the same predictor is more challenging. However, a vector of parameters, each controlling a specific feature, can be defined and estimated. This idea bears similarity to Bennett's learning of indicator functions [Bennett *et al.*, 2005]. A framework that allows sparse representations, meaning allowing certain features to be inactive, might even be more desirable. Such a framework will also be flexible to incorporate additional features later on. This could be a very interesting and exciting direction to pursue further research, both in theory and practice.

There are further opportunities for future work concerning the noisy predictors. Throughout the thesis, we have relied on the assumption that the predictors' outputs are conditionally independent from each other given the truth, except for the non-collaborative risk estimation introduced in Chapter 7. The conditional independence assumption is crucial in IEThresh and the particle filtering algorithms described in Chapter 6 since they rely on some kind of majority voting scheme to evaluate the predictor's output. As discussed previously, relaxing this assumption introduces additional parameters corresponding to the correlation between the predictors that need to be estimated. A similar concern is related to the assumption that requires the predictors to be better than random chance. Our experimental results show that the proposed framework is robust to the violation of this assumption as long as it is restricted with a small subset of predictors (See Chapter 7). However, we have not fully examined to what extent the violation of these assumptions would affect the estimation accuracy or the quality of the predicted labels. Future extensions that address these concerns will yield great opportunities to apply in more real-life scenarios.

Although we have analyzed the predictors whose response characteristics are functions

of the input space, we have done so in the context of simple scenarios involving a perfectly reliable oracle to evaluate the unreliable predictors. Addressing this problem in a completely unsupervised manner with multiple predictors is still an open question. How to represent the fallibility as a function of the unlabeled data? How to combine the context-dependent fallibility estimation with active learning to decide the regions of the input space that need to be exploited to gain the maximum benefit? How to transfer the knowledge learned across different data distributions? These questions and possibly many others pose great challenges but also great advantages such as incorporating any predictor with no restrictions into the picture.

Another important future direction is to consider an ensemble of predictors as referral networks. That is, when an annotator or expert is queried he has two options: he can provide an answer for some fee or he can direct the question to someone else who is likely to be the expert on the subject requiring a small referral fee. In such a system, the goal is to reduce the total number of referrals and find the right expert directly. A good example to this situation is medical diagnosis. Doctors might refer the patients to specialists if they cannot diagnose them or if they need a second opinion. Visiting multiple doctors with different specialties is likely to increase the chance of a more reliable diagnosis, but at the cost of spending a lot of money, time and resources. Hospitals could significantly benefit from a system that automatically identifies the right expert given the patient's present symptoms and medical history, instead of the patient going from one doctor to the other. This problem could be addressed by modeling the expertise area of each predictor and also modeling the expertise needed to answer a question and then finding the right match between the query and the predictor. It would be interesting as future work to see a thorough investigation of this direction from both theoretical and practical perspectives.

Bibliography

[agn, 2007] Agnostic learning vs. prior knowledge challenge and data representation discovery workshop, 2007. IJCNN '07. 71

[Ambati *et al.*, 2010] V. Ambati, S. Vogel, and J. Carbonell. Active learning and crowd-sourcing for machine translation. In *LREC '10*, 2010. 2

[Amini *et al.*, 2006] M. Amini, N. Usunier, F. Laviolette, A. Lacasse, and P. Gallinari. A selective sampling strategy for label ranking. *ECML '06*, pages 18–29, 2006. 2, 10, 48

[Arulampalam *et al.*, 2001a] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001. 92

[Arulampalam *et al.*, 2001b] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001. 95

[Baram *et al.*, 2003] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *ICML '03*, pages 19–26, 2003. 9, 23, 38, 39, 165

[Bennett *et al.*, 2005] P. Bennett, S. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2005. 170

[Berk, 1973] K. N. Berk. A central limit theorem for m-dependent random variables with unbounded m. *The Annals of Probability*, 1(2):352–354, 1973. 150

[Bie and Cristianini, 2003] T. De Bie and N. Cristianini. Convex methods for transduction. In *NIPS '03*, 2003. 13

[Bishop *et al.*, 1975] Y. Bishop, S. Fienberg, and P. Holland. *Discrete multivariate analysis: theory and practice*. MIT press, 1975. 120

[Blitzer *et al.*, 2007] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL '07*, 2007. 141

[Blum and Mitchell, 1998] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *COLT '98*, pages 92–100, 1998. 9

[Breiman, 1996] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics department, University of California, 1996. 141

[Brinker, 2004] K. Brinker. Active learning of label ranking functions. *ICML '04*, pages 17–24, 2004. 2, 9, 10

[Burges and Crisp, 2000] C.J.C. Burges and D.J. Crisp. Uniqueness of the svm solution. In *NIPS '00*, pages 223–229, 2000. 52

[Campbell *et al.*, 2000] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *ICML '00*, 2000. 30

[Cao *et al.*, 2006] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. *Proceedings of the international ACM SIGIR Conference on Research and Development in information retrieval (SIGIR'06)*, pages 186–193, 2006. 41, 44, 53

[Carpenter *et al.*, 1999] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for non-linear problems. In *IEEE Proceedings on Radar and Sonar Navigation*, 1999. 97

[Cauwenberghs and Poggio, 2000] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS '00*, pages 409–415, 2000. 51

[Chapelle, 2005] O. Chapelle. Active learning for parzen window classifier. *AISTATS '05*, pages 49–56, 2005. 26, 27, 28

[Cohn *et al.*, 1995] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Advances in Neural Information Processing Systems*, 7:705–712, 1995. 7, 32

[Cover and Thomas, 2005] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, second edition, 2005. 115, 143

[Cox and Cox, 1994] T.F. Cox and M.A. Cox. *Multidimensional Scaling*. Chapman & Hall, 1994. 27

[Cox *et al.*, 2006] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2006. 129

[Craswell and Hawking, 2004] N. Craswell and D. Hawking. Overview of the trec 2004 web track. *Text Retrieval Conference (TREC'04)*, 2004. 47

[Craswell *et al.*, 2003] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the trec 2003 web track. *Text Retrieval Conference (TREC'03)*, 2003. 47

[Dai *et al.*, 2007] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proc. of International Conference on Machine Learning*, 2007. 154

[Daumé III, 2007] H. Daumé III. Frustratingly easy domain adaptation. In *ACL '07*, 2007. 5

[Dekel and Shamir, 2009] O. Dekel and O. Shamir. Good learners for evil teachers. In *Proceedings of the 26th International Conference on Machine Learning*, 2009. 12

[Donmez and Carbonell, 2008a] P. Donmez and J. G. Carbonell. Paired sampling in density-sensitive active learning. *International Symposium on Artificial Intelligence and Mathematics*, 2008a. 2, 8

[Donmez and Carbonell, 2008b] P. Donmez and J. G. Carbonell. Optimizing estimated loss reduction for active sampling in rank learning. *International Conference on Machine Learning, ICML '08*, 2008b. 1, 2, 7, 10, 11

[Donmez and Carbonell, 2008c] P. Donmez and J. G. Carbonell. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. *ACM Conference on Knowledge and Information Management, CIKM '08*, 2008c. 11

[Donmez and Carbonell, 2009] P. Donmez and J. G. Carbonell. Active sampling for rank learning via optimizing the area under the roc curve. *European Conference on Information Retrieval, ECIR '09*, 2009. 2, 10

[Donmez *et al.*, 2007] P. Donmez, J.G. Carbonell, and P.N. Bennett. Dual strategy active learning. *Proceedings of the European Conference on Machine Learning*, pages 116–127, 2007. 1, 2, 9, 32, 44, 47

[Donmez *et al.*, 2009] P. Donmez, J. G. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '09)*, pages 259–268, 2009. 4, 12

[Donmez *et al.*, 2010a] P. Donmez, J. G. Carbonell, and J. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *Proceedings of SIAM Conference on Data Mining (SDM '10)*, 2010. 4, 12

[Donmez *et al.*, 2010b] P. Donmez, G. Lebanon, and K. Balasubramanian. Margin-based classification without labels. *Submitted to (ICML '10)*, 2010. 5, 13

[Donmez *et al.*, 2010c] P. Donmez, G. Lebanon, and K. Balasubramanian. Unsupervised estimation of classification and regression error rates. *Journal of Machine Learning Research*, 2010. 5, 13, 104

[Doucet and Crisan, 2002] A. Doucet and D. Crisan. A survey of convergence results on particle filtering for practitioners. *IEEE Transactions on Signal Processing*, 2002. 96

[Duda *et al.*, 2001] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley New York, 2001. 143

[Efron and Tibshirani, 1997] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1997. 143

[Ferguson, 1996] T. S. Ferguson. *A Course in Large Sample Theory*. Chapman & Hall, 1996. 122, 123, 132, 153

[Fischer *et al.*, 2004] B. Fischer, V. Roth, and J.M. Buhmann. Clustering with the connectivity kernel. In *NIPS '04*, volume 16, 2004. 27

[Freund *et al.*, 1997] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the Query by Committee algorithm. *Machine Learning*, 28:133–168, 1997. 7

[Freund *et al.*, 2003] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003. 41, 42, 48, 54, 55, 56

[Guo and Greiner, 2007] Y. Guo and R. Greiner. Optimistic active learning using mutual information. In *IJCAI '07*, pages 823–829, 2007. 23

[Hand and Till, 2001] D.J. Hand and R.J. Till. A simple generalization of the area under the roc curve for multiple class classification problems. *Machine Learning*, pages 171–186, 2001. 43

[Hand, 1986] D. J. Hand. Recent advances in error rate estimation. *Pattern Recognition Letters*, 4(5):335–346, 1986. 143

[Hartigan and Wong, 1979] J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979. 64

[Hoeffding and Robbins, 1948] W. Hoeffding and H. Robbins. The central limit theorem for dependent random variables. *Duke Mathematical Journal*, 15:773–780, 1948. 150

[Joachims, 1999] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999. 48, 141

[Joachims, 2002] T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, 2002. 41, 42

[Joachims, 2005] T. Joachims. A support vector method for multivariate performance measures. *Proceedings of the International Conference on Machine Learning (ICML'05)*, pages 377–384, 2005. 44

[Kaelbling, 1990] L. P. Kaelbling. *Learning in Embedded Systems*. PhD thesis, Department of Computer Science, Stanford University, 1990. 80, 81

[Lang, 1995] K. Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, 1995. 142

[Lewis and Gale, 1994] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. *SIGIR '94*, pages 3–12, 1994. 8, 30, 82

[Lewis *et al.*, 2004] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004. 151

[Liere and Tadepalli, 2007] R. Liere and P. Tadepalli. Active learning with committees for text categorization. *AAAI '97*, pages 591–596, 2007. 7

[Liu *et al.*, 2007] T.Y. Liu, T. Qin, J. Xu, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. *LR4IR 2007, in conjunction with SIGIR 2007*, 2007. 47

[Mann and Whitney, 1947] H.B. Mann and D.R. Whitney. On a test whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, pages 50–60, 1947. 43

[McCallum and Nigam, 1998] A. McCallum and K. Nigam. Employing em and pool-based active learning for text classification. *ICML '98*, pages 359–367, 1998. 8, 44

[Melville and Mooney, 2003] P. Melville and R. J. Mooney. Constructing diverse classifier ensembles using artificial training examples. *IJCAI '03*, pages 505–510, 2003. 9, 39

[Melville and Mooney, 2004] P. Melville and R.J. Mooney. Diverse ensembles for active learning. In *ICML '04*, pages 584–591, 2004. 23, 39

[Melville *et al.*, 2005] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Economical active feature-value acquisition through expected utility estimation. *KDD '05 Workshop on Utility-based data mining*, 2005. 11, 64

[Moore and Schneider, 1995] A. Moore and J. Schneider. Memory-based stochastic optimization. In *Neural Information Processing Systems 8*, 1995. 80, 81

[Muslea *et al.*, 2000] I. Muslea, S. Minton, and C. Knoblock. Selective sampling with naive co-testing: preliminary results. *The ECAI-2000 workshop on Machine Learning for information extraction*, 2000. 9

[Newman *et al.*, 1998] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. University of California, Irvine, Dept. of Information and Computer Sciences. 22, 71, 85, 104

[Nguyen and Smeulders, 2004] H.T. Nguyen and A. Smeulders. Active learning with pre-clustering. *ICML '04*, pages 623–630, 2004. 1, 8, 11, 16, 17, 18, 19, 22, 32, 35, 44

[Nigam, 2001] K. Nigam. *Using Unlabeled Data for Text Classification*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2001. 154

[Papoulis, 1984] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1984. 116

[Pham *et al.*, 2002] T. Pham, M. Worring, and A. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23, 2002. 71, 106, 151

[Platt, 1999] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74, 1999. 56

[Rajaram *et al.*, 2007] S. Rajaram, C.K. Dagli, N. Petrovic, and T.S. Huang. Diverse active ranking for multimedia search. *Computer Vision and Pattern Recognition (CVPR '07)*, 2007. 2, 43, 48

[Rätsch *et al.*, 2001] G. Rätsch, T. Onoda, and K. R. Muller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001. 22, 85

[Raykar *et al.*, 2009] V. C. Raykar, S. Yu, L. Zhao, A. Jerebko, C. Florin, G. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *Proceedings of the 26th International Conference on Machine Learning*, pages 889–896, 2009. 12

[Ripley, 1987] B. Ripley. *Stochastic Simulation*. Wiley, New York, 1987. 97

[Roy and McCallum, 2001] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. *ICML '01*, pages 441–448, 2001. 1, 7, 11, 23, 50, 51

[Schein and Ungar, 2005] A.I. Schein and L.H. Ungar. Active learning for multi-class logistic regression. *Learning*, 2005. 35

[Schohn and Cohn, 2000] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. *Proceedings of the International Conference on Machine Learning (ICML'00)*, pages 839–846, 2000. 30

[Seung *et al.*, 1992] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 287–294, 1992. 7

[Shen and Zhai, 2005] X. Shen and C. Zhai. Active feedback in ad hoc information retrieval. *SIGIR '05*, pages 59–66, 2005. 8

[Sheng *et al.*, 2008] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. *KDD '08*, pages 614–622, 2008. 12, 85, 109

[Smyth *et al.*, 1994] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of venus images. In *NIPS '94*, pages 1085–1092, 1994. 12

[Smyth *et al.*, 1995] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Learning with probabilistic supervision. *Computational Learning Theory and Natural Learning Systems*, 3, 1995. 12

[Snow *et al.*, 2008] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. *EMNLP '08*, 2008. 2, 12, 81, 85, 87, 139

[Steck, 2007] H. Steck. Hinge rank loss and the area under the roc curve. *Proceedings of the European Conference on Machine Learning (ECML'07)*, pages 347–358, 2007. 10, 42, 43, 167

[Struyf *et al.*, 1997] A. Struyf, M. Hubert, and P. Rousseeuw. Integrating robust clustering techniques in s-plus. *Computational Statistics and Data Analysis*, 26:17–37, 1997. 18

[Sturmfels, 2002] B. Sturmfels. *Solving Systems of Polynomial Equations*. American Mathematical Society, 2002. 129

[Tang *et al.*, 2002] M. Tang, X. Luo, and S. Roukos. Active learning for statistical natural language parsing. *ACL '02*, 2002. 8

[Teicher, 1963] H. Teicher. Identifiability of finite mixtures. *The Annals of Mathematical Statistics*, 34(4):1265–1269, 1963. 151

[Tong and Koller, 2000] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Proceedings of International Conference on Machine Learning*, pages 999–1006, 2000. 1, 8, 30, 47

[Vapnik, 2000] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, 2000. 141

[Wilcoxon, 1945] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics 1*, pages 80–83, 1945. 43

[Xu and Schuurmans, 2005] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI '05*, 2005. 13

[Xu *et al.*, 2003] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang. Representative sampling for text classification using support vector machines. *Proceedings of the European Conference on Information Retrieval*, 2003. 8, 16, 22, 32, 35, 38, 47

[Yu, 2005] H. Yu. Svm selective sampling for ranking with application to data retrieval. *SIGKDD '05*, pages 354–363, 2005. 2, 9, 10, 48

[Zhang and Chen, 2002] C. Zhang and T. Chen. An active learning framework for content-based information retrieval. *IEEE Trans. on Multimedia*, 4:260–268, 2002. 8, 32