ACTIVE LEARNING WITH UNRELIABLE ANNOTATIONS

by

LIYUE ZHAO
B.Eng. University of Science and Technology of China, 2006
M.S. University of Central Florida, 2011

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida

Summer Term
2013

Major Professor: Gita Sukthankar

ABSTRACT

With the proliferation of social media, gathering data has became cheaper and easier than before. However, this data can not be used for supervised machine learning without labels. Asking experts to annotate sufficient data for training is both expensive and time-consuming. Current techniques provide two solutions to reducing the cost and providing sufficient labels: crowdsourcing and active learning. Crowdsourcing, which outsources tasks to a distributed group of people, can be used to provide a large quantity of labels but controlling the quality of labels is hard. Active learning, which requires experts to annotate a subset of the most informative or uncertain data, is very sensitive to the annotation errors. Though these two techniques can be used independently of one another, by using them in combination they can complement each other's weakness. In this thesis, I investigate the development of active learning Support Vector Machines (SVMs) and expand this model to sequential data. Then I discuss the weakness of combining active learning and crowdsourcing, since the active learning is very sensitive to low quality annotations which are unavoidable for labels collected from crowdsourcing. In this thesis, I propose three possible strategies, incremental relabeling, importance-weighted label prediction and active Bayesian Networks. The incremental relabeling strategy requires workers to devote more annotations to uncertain samples, compared to majority voting which allocates different samples the same number of labels. Importance-weighted label prediction employs an ensemble of classifiers to guide the label requests from a pool of unlabeled training data. An active learning version of Bayesian Networks is used to model the difficulty of samples and the expertise of workers simultaneously to evaluate the relative weight of workers' labels during the active learning process. All three strategies apply different techniques with the same expectation – identifying the optimal solution for applying an active learning model with mixed label quality to

crowdsourced data. However, the active Bayesian Networks model, which is the core element of this thesis, provides additional benefits by estimating the expertise of workers during the training phase. As an example application, I also demonstrate the utility of crowdsourcing for human activity recognition problems.

To my parents

ACKNOWLEDGMENTS

I would like to thank my committee members, Gita Sukthankar, Marshall Tappen, Michael Georgiopoulos and Rahul Sukthankar for reviewing and revising my dissertation. I really appreciate their time and effort in support of my dissertation writing.

I would also like to thank the past and present members in my research group that I have worked with: Xi Wang, Fahad Shah, Bulent Tastan, Masha Maghami, Bennie Lewis and Ken Laviers. It was my pleasure to work with this group of knowledgable and diligent people, as well as good friends.

My special appreciation goes to Yu Zhang, who has collaborated with me in developing and implementing several core algorithms in my dissertation. It was my great honor to work with Yu. I hope I will have the opportunity to work with him again with more interesting projects in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Active Learning in Crowdsourcing

My work is motivated by the recent interest in the use of crowdsourcing [37] as a source of annotations from which to train machine learning systems. Crowdsourcing transforms the problem of generating a large corpus of labeled real-world data from a monolithic labor-intensive ordeal into a manageable set of small tasks, processed by thousands of human workers in a timely and affordable manner. Human computation, organized through services such as Amazon's Mechanical Turk (MTurk), has made it possible for researchers to acquire sufficient quantities of crowdsourced labels, enabling the development of a variety of applications driven by supervised learning models. However, employing crowdsourcing to label large quantities of data remains challenging for two important reasons:

limited annotation budget: although the unit cost of obtaining each annotation is low, the overall cost grows quickly since it is proportional to the number of requested labels, which can number in the millions.

label noise: the quality of crowdsourced annotations has been found to be poor [67], with causes ranging from workers who overstate their qualifications, lack of motivation among labelers, haste and deliberate vandalism.

Active learning [66] aims to learn high-quality classifiers using only a subset of labeled data in order to reduce the overall annotation budget. Compared with traditional machine learning approaches which uniformly require labels for all training samples, active learning selects a small subset of samples that are most informative to the learning model and asks experts

for annotations. The active learner starts with a tiny subset of samples and identifies the sample with the greatest label uncertainty according to the learner as the most informative sample. Therefore, the active selection strategy converges much faster than randomly selecting samples to be labeled and efficiently reduces the annotation workload.

The majority of existing work in active learning makes the "perfect oracle" assumption [79], meaning that the requested label is guaranteed to be correct. Under such conditions, an effective strategy for active learning is to select the samples that would offer the greatest reductions to the set of hypotheses consistent with the observed training labels. Thus, "aggressive" criteria such as least confidence, smallest margin, or maximum entropy can enable active learning to obtain a high accuracy using a relatively small number of labels. Unfortunately, label noise causes aggressive active learning methods to fail because even a single incorrect label can suddenly cause the algorithm to incorrectly eliminate the wrong set of hypotheses and thus focus the search on a poor region of the version space. A second (but less well-known) concern is that active learning, by its very nature, creates a biased sampling of the data set since it favors instances that are in confusing regions of the feature space. Unless this bias is accounted for, it leads to sub-optimal learners.

A common strategy for combating annotation noise is to request multiple labels for each selected instance and then to apply majority voting. The simplest approach of requesting the same number of labels for each instance is usually not the most cost-effective since label redundancy increases the overall cost by a multiplicative factor of at least 3. Better results can be obtained by combining active learning with incremental relabeling to solicit additional labels where they are most useful [67, 95, 98].

An alternate strategy, as advocated by agnostic active learning [6], is to make more conservative use of the training data. The basic idea is to omit requesting labels for instances

only if all hypotheses under current consideration agree, sample labels from the remaining set and eliminate only those hypotheses whose lower bound on the expected error is greater than the minimum upper bound for the expected error over the set of hypotheses.

In this thesis I propose two solutions to boost active learning with noisy annotation. First, I propose a modified active learning paradigm that balances the traditionally myopic selection of instances in active learning (e.g., based on proximity of data to the decision boundary) with a term that also considers the distribution and local consistency of labels. In other words, we aim to automatically identify those unlabeled instances that are most valuable to label, but also samples that might benefit from relabeling because their initial labels seem suspect. Since acquiring multiple labels for each sample *en masse* would be impractical due to expense, our proposed method seeks to incrementally label only the most important samples. Additionally, we identify problematic samples (those for which we cannot get a sufficiently consistent set of labels) and withhold them from the training set. Thus, the judicious use of inconsistency detection and incremental relabeling significantly boost the ability of active learning to exploit crowdsourced data.

The second solution is inspired by the importance-weighted active learning algorithm (IWAL) [10], which considers each unlabeled element in a stream-based manner to determine whether it should be labeled, and if so, how to weight the data point in a manner that corrects for the effect of biased sampling. As detailed below, we advocate an importance-weighted approach in a pool-based active learning setting. Specifically, our method makes three modifications:

1. I generate and maintain an unbiased bootstrap ensemble of incrementally trained classifiers that are used to identify and reject potentially inconsistent annotations;
2. Rather than requesting labels for unlabeled instances simply based on importance sampling, I identify those instances that are (with high probability) unlikely to require

labels;

3. I present comprehensive evaluations on standard datasets that show not only the strength of our proposed approach in comparison to established active learning algorithms, but also which specific aspects of our technique lead to the greatest improvements under various noise conditions.

Applications

As an example application, in this thesis I examine the use of crowdsourcing for training classifiers for human activity recognition. Human activity recognition has become an increasingly important component of many domains such as user interfaces and video surveillance. In particular, enabling ubiquitous home user assistance systems for elder care requires rapid and robust recognition of human actions from portable sensor data. Motion trajectories, gathered from video, inertial measurement units, or mocap, are a critical cue for identifying activities that require gross body movement, such as walking, running, falling, or waving. Human motion data typically needs to be segmented into activities to be utilized by any application [19]. A common processing pipeline for motion data is:

1. segment data into short time windows;
2. recognize low-level human activities from repetitive patterns of motion executed by the human user within a time window;
3. identify a high-level intention or plan from sequences of activities.

For instance, one possible high-level household activity would be "baking pizza" which would consist of low-level activities such as "beating an egg" or "kneading dough" which could be

recognized by the motion patterns and objects manipulated.

Although domain knowledge and common-sense reasoning methods are important for reasoning about the human's high level intentions, segmentation and activity classification have been successfully addressed by a variety of data-driven approaches, including supervised classifiers, such as support vector machines, hidden Markov models, dynamic Bayes nets, and conditional random fields. In the best case, supervised learning can yield classifiers that are robust and accurate. However, two problems frequently occur in supervised learning settings.

lack of data: gathering and labeling the data is time-consuming and expensive. In some cases, the activities are highly repetitive in nature (stirring), whereas other actions are infrequent and short in duration (opening the refrigerator). To classify these short actions, learning techniques need to be sample-efficient to leverage relatively small amounts of labeled training data.

feature selection: sensors yield data that is both noisy and high-dimensional. Learning classifiers based on the raw sensor data can be problematic and applying arbitrary dimensionality reduction techniques does not always yield good results.

In this thesis, I present a case study of how I addressed these problems while performing segmentation and activity recognition of human household actions. First, I introduce an active learning method in which the classifier is initialized with training data from unsupervised segmentation and improved by soliciting unlabeled samples that lie closest to the classification hyperplane. I demonstrate that this method can be used to reduce the number of samples required to classify motion capture data using support vector machine (SVM) classifiers.

Second, I present a method to improve classification through intelligent feature selection. The signal data is converted into a set of motifs, approximately repeated symbolic subsequences, for each dimension of IMU data. These motifs leverage structure in the data and serve as the basis to generate a large candidate set of features from the multi-dimensional raw data. By measuring reductions in the conditional log-likelihood error of the training samples, I can select features and train a conditional random field (CRF) classifier to recognize human actions.

Our techniques were evaluated using the CMU Multimodal Activity database [25] which was collected to facilitate the comparison of different activity recognition techniques for recognizing household activities. The dataset contains video, audio, inertial measurement unit (IMU), and motion capture data; we demonstrate the utility of our techniques on segmenting motion capture data and recognizing inertial measurement unit (IMU) data.

CHAPTER 2: RELATED WORK

In this chapter, I give an overview of the concepts that my approach relies on 1) active learning 2) crowdsourcing 3) active learning with annotation noise and 4) applications. Table 2.1 provides an overview of this chapter's key components.

Table 2.1: Roadmap of Related Work

Field of Research	Key Papers
Active Learning	[66, 80]
Crowdsourcing	[26, 39]
Active Learning with Annotation Noise	[67, 6, 24, 10, 28]
Worker and Task Modeling	[91, 88]
Applications	[47, 82, 19]

Active Learning

Active learning can be regarded as a subfield of supervised learning in which the aim is to achieve the same classification performance with a smaller set of labeled data. [66] gives an up-to-date survey of all related active learning approaches. As shown in Figure 2.1, standard (passive) learning techniques require humans to annotate the entire set of input training data that the learner uses to build the classifier. By contrast, in active learning, the annotation of data is performed through a series of interactions between the automated learner and the human. By analyzing the unlabeled data, the active learner selects a subset of the data that has a high degree of label uncertainty to be annotated by the expert. Currently, two main issues of active learning research are how to pose queries and identify informative instances.

There are three general methods for posing queries: *membership queries*, *selective sampling* and *pool-based sampling*. Membership queries algorithms [1, 2] generate new instances in the input space and request these labels from the expert. However, such algorithms may construct unexpected instances which are difficult for human experts to label [9]. The selective sampling algorithm was introduced to overcome this problem [4, 21]. In selective sampling, the learner receives distribution information from the environment and queries the expert on parts of the domain. The learner sequentially draws individual instances and decides whether or not to request its label by evaluating the *region of uncertainty*. The *region of uncertainty* is reduced after each new instance is added; more instances are included if the uncertainty is not reduced efficiently.

Currently, the most popular query strategy in active learning is pool-based sampling, in which samples are requested from an existing closed set of instances. These algorithms share the common assumption that there is a small set of labeled data and a large pool of unlabeled data. The learner generated from the labeled set is applied to evaluate the informativeness measure of instances in the unlabeled pool. The label of the most informative instance is requested, and the instance is then added to the labeled set.

One question is how best to identify the most informative instances for future label requests. Although there are different ways to evaluate the informativeness of unlabeled instances, the main approaches are *query by uncertainty* and *query by committee*. The query by uncertainty algorithm starts by building the learner using the labeled data set. The learner is used to provide a confidence score for each unlabeled instance to probe for instances where the learner is the least certain of the currently classified labels. Those uncertain labels are requested from the experts, and the instances become the members of labeled data. This process repeats until the learner is confident about all of the unlabeled data. Query by uncertainty is probably the most straightforward approach and has been demonstrated in

several real-world applications [78, 3, 17].

An alternative query strategy is the query by committee algorithm. The essential idea behind this approach is to narrow the possible hypotheses in the *version space* (the set of classifiers consistent with the training set) [57]. A committee of classifiers is generated from labeled data to evaluate the unlabeled data. The instance with the most classifier disagreement is deemed to be the most informative instance. As shown in Figure 2.2, by querying new labels, the version space narrows to reduce the number of possible hypotheses. Therefore, the optimal learner will be reached once there are no version space among hypotheses learned with all the instances.

Active learning has been successfully applied to many classification problems. Tong et al. demonstrated the use of Support Vector Machines (SVMs) to construct pool-based active learners for both text classification [80] and image retrieval [78]. In the binary classification problem, it is assumed that the most informative instances should split the version space into two equal parts; this formulation can also be extended to multi-class classification. Several new strategies have been proposed for evaluating the informativeness of unlabeled data (e.g., [17]). Also, [89] proposed a new bootstrapping strategy for SVM active learning.

Crowdsourcing

[37] first proposed the concept of crowdsourcing as a new pool of cheap labor composed of everyday people who use their spare time to create content, and solve problems. [26] defines crowdsourcing as enlisting a crowd of humans to help solve a problem defined by the system owners. They also classify different types of crowdsourcing systems based on the nature of collaboration, the type of target problem, the degree of manual effort, and the role of human

users. For example, Wikipedia[1] lets users edit and combine textual content of a given topic within a single page. The ESP game [86] allows users to play a game of guessing common words to label images.

Amazon Mechanical Turk[2] is a web system that allows system owners to post tasks called Human Intelligence Tasks (HITs) for completion by workers who get paid between one cent and a few dollars. The commercial system, Amazon Mechanical Turk (MTurk), has become a large marketplace in crowdsourcing. There are two groups of people in MTurk. The requesters are owners who post Human Intellegent Tasks (HITs) on the system to collect responses from workers. The workers are a group of users who accept and complete those HITs and get paid by owners. [39] trace the public information of both requesters and workers according to the website[3]. They note that 25 percent of the Indian respondents and 13 percent of U.S. respondents reported that Amazon Mechanical Turk is their primary source of income. The MTurk system has been used to collect different types of data. [13] demonstrate that evaluations of machine translation quality through reading comprehension can be efficiently accomplished by humans through Mechanical Turk. The sheep market [42][4] is a web-based artwork project which collected 10000 sheep drawn for $.02 by workers on Amazon Mechanical Turk. Workers were asked to exercise their creativity to "draw a sheep facing to the left." [87] proposed a cost efficient video annotation approach which asks workers on MTurk to only locate the scales and positions of people and objects in key frames and uses computer vision models to determine their scales and positions in other frames.

[1] http://en.wikipedia.org/

[2] https://www.mturk.com/

[3] http://www.mturk-tracker.com

[4] http://users.design.ucla.edu/~akoblin/work/thesheepmarket/

Active Learning with Annotation Noise

Crowdsourcing annotation services, such as Amazon's Mechanical Turk, have became an effective way to distribute annotation tasks over multiple workers. Active learning approaches have recently become popular in this context since they provide a budget-conscious approach to data labeling [85]. However, although active learning is a mature field [20], the majority of work assumes the absence of label noise (i.e., perfect oracles). [67] observed the problem that crowdsourcing annotation may generate unreliable labels. The serious issue of label corruption in social computing has been identified both for crowdsourced data collection [38] and in the context of attacks on collaborative tagging [63]. Since the SVM active learner is greedy and myopic, these noisy annotations will lead to error accumulation when selecting the next sample.

Most active learners select the next instance to label in a greedy and myopic manner. Neglecting the global distribution of data can cause the learner to converge only to a locally optimal hypothesis. Such a strategy will lead to two issues. The first serious problem is that the uncertainty sampling is inherently "noise-seeking" [6] and may take a long time to converge. Second, the "aggressive" sampling is biased by selecting more samples close to the current hypothesis which ignores the true distribution of the whole dataset.

One set of solutions is to consider the labels of the sample's neighbors. Nguyen and Smeulders [58] improve the performance of active learning by taking into account the prior data distribution. Donmez et al. [27] proposed their dual strategy active learning which considers not only the most uncertain samples but also the most representative samples to avoid having the training process fall into local maximum. Proactive learning [28] offers one way to jointly select the optimal oracle and instance with a decision-theoretic approach. The algorithm penalizes the reluctant oracle who doesn't always give answers by reducing the

utility of remaining examples with respect to this oracle.

Another group of methods focuses on unbiasing the sample selection process. Agnostic active learning [6] is the original noise tolerant method which theoretically produces the upper and lower bounds for linear separators under the uniform distribution. They point out that the aggressive sampling strategy in active learning suffers from sampling bias which puts more weight on samples closer to the current hypothesis and use selective sampling to unbias the sampling process. Hierarchical clustering approaches such as [24] provide the learner with global information about the data and avoid the tendency of the learner to converge to local maxima. IWAL [10] proposes a more efficient sampling scheme of using importance weighting to build loss functions. UPAL [34], while designed for the noise-free scenario, also seeks unbiased sampling of unlabeled data. This bias issue is related to that of "dataset shift", where training and test distributions diverge [14]. Failing to take this into account can degrade classifier accuracy, even when label noise is not an issue.

Many previous active learning approaches lack of statistical consistency. So even with an infinite budget, those methods may not converge to an optimal solution. IWAL [10] solves this problem by using an importance weighting strategy which not only guarantees PAC-style label complexity, but also removes sampling bias. In practice, for a stream-based active learning problem, IWAL starts with a small subset of labeled samples and applies bootstrap to initialize a set of classifiers. For a new unlabeled sample, the algorithm calculates the probability p of requesting the sample. If the sample ends up being requested, the algorithm gives it an importance weight $\frac{1}{p}$ to avoid the sampling bias.

Worker and Task Modeling

Several works [29, 94, 65] propose different approaches to model the annotation accuracy of workers. They all assume there are multiple experts/annotators providing labels but no oracle exists. [29] proposes a framework to learn the expected accuracy at each time step. The estimated expected accuracies are then used to decide which annotators should be queried for a label at the next time step. [94] focuses on the multiple annotator scenario where multiple labelers with varying expertise are available for querying. This method can simultaneously answer the questions such as which data sample should be labeled next and which annotator should be queried to benefit the learning model. [65] uses a probabilistic model to evaluate the different experts and also gives an estimate of the actual hidden labels.

Whitehill et al. [91] attempt to use the probabilistic model to simultaneously infer the label of each image, the expertise of each labeler, and the difficulty of each image. Based on the labels collected from different workers on different tasks, the label of each image, the expertise of each labeler, and the difficulty of each image are represented as latent variables. In the training phase, those latent variables are inferred by iteratively running the EM algorithm given the observation data (collected labels). Moreover, for a new image, the model can estimate its label with a weighted combination of labels from different workers, rather than simply using majority voting.

One advantage of probabilistic models is that they potentially provide additional information to the active learning algorithm for selecting which sample should be labeled next. [79] proposed a way to implement active learning in Bayesian networks. Tong and Koller chose Kullback-Leibler divergence as the loss function to measure the distance between distributions. Their algorithm iteratively computes the expected change in risk and makes the query with the greatest expected change. This strategy is guaranteed to request the label of the

sample that reduces the expected risk the most.

Comparison with Other Existing Active Learning Methods on Crowdsourcing

Table 2.2 presents an overview of the differences between learning methods related to this work including

Table 2.2: Comparison of related learning techniques

Publications	Active	Noise	Multiple	Modeling	Features
[80, 79]	Yes	No	No	No	Data
[6, 24, 10, 36, 97]	Yes	Yes	No	No	Data
[67, 48, 95]	Yes	Yes	Yes	No	Data
[30, 45, 90, 94, 32]	Yes	Yes	Yes	Yes	Data
[91, 88]	No	Yes	Yes	Yes	Label
My thesis	Yes	Yes	Yes	Yes	Label

Active Learning (Active): Active Learning is defined in contrast to Passive Learning, the commonly used supervised learning algorithms in which model parameters are estimated using fully labeled data. Active Learning starts with a very small set of labels and interactively identifies the most uncertain samples from unlabeled data to annotate.

Annotation Noise (Noise): Many machine learning approaches assume annotations are collected from an oracle who always provides correct labels. Annotation Noise is introduced by incorrect labels provided by workers.

Multiple Workers (Multiple): Multiple Workers denotes approaches in which the annotation task can be annotated by more than one worker with different levels of expertise.

Worker Modeling (Modeling): Worker Modeling methods estimate the expertise of workers based on their performance in annotation tasks.

Features: Many algorithms rely on features extracted from input data. *Label* feature only considers labels collected from different workers but doesn't extract features from data.

The first group of publications is classified as active learners with data annotated by an oracle, eliminating the requirement for worker modeling. The publication include in this group standard and classical active learning methods. SVM active learner [80] proposes three different criteria to select the most uncertain unlabeled sample which reduces the version space the most. Bayesian Network active learner [79] applies relative entropy and KL-divergence as loss functions to evaluate the variances of unlabeled samples. All these classical active learning methods consider only two things, sampling strategy (membership queries, selective sampling or pool-based sampling) and selection strategy (query-by-uncertainty or query-by-committee).

When using these methods on real labels collected from non-experts, researchers find classical active learners make an important assumption which doesn't exist in real world - all labels are correct [67, 24]. The theory behind this assumption is, to reach the fastest possible error reduction, at each iteration classical active learners eliminate the version space which disagrees with the label of the selected sample. Actually many active learning strategies are "noise seeking" in practical learning problems. The situation may become worse if the active learner is built on a discriminative learning model. Imagine the situation in which a hard margin SVM is used as the learner; in this case, the active learning will fail to reach the optimal hyperplane with only one incorrect label.

Many publications are aware of this flaw and get rid of the "perfect labeler" assumption by introducing annotation noise when they implement their active learners. The Agnostic Active

Learning [6] claimed to be the first method which works on annotation noise. This method adopts a more conservative strategy in eliminating the potential hypotheses in version space - tand refraims from eliminating the hypotheses in version space based on only one sample. By introducing the upper and lower bound of all hypotheses in the version space, the hypotheses that fall outside of the bounds could be safely removed. A series of derivatives [24, 10, 36, 97] have been published to either improve or extend the performance of Agnostic Active Learning.

Another group of researchers, especially those who want to apply active learning in crowdsourcing, estimate the winning label based on annotations from multiple workers. Sheng et al. [67] pointed out that using majority vote to generate the winning label may get worse if the label quality is bad. Different relabeling strategies [48, 95] have been proposed to identify the "absolute majority" as the optimal label for the task.

Laws et al. [48] proposed a combination of active learning and crowdsourcing to address two tasks, named entity recognition and sentiment detection, in natural language processing. To enhance the performance of active learning with noisy annotations, this paper applied two techniques, *adaptive voting* and *fragment recovery*. *Adaptive voting* has been applied to consolidate labels from multiple workers by incrementally requesting more labels until the absolute majority appears. If the absolute majority doesn't appear after requesting a certain number of labels, the label of the sentence will be discarded. *Fragment recovery* will salvage tokens from a discarded sentence by cutting it into several fragments and discarding only disagreed tokens.

With multiple workers in crowdsourcing, many recent works [30, 45, 90, 94, 32] step forward to model the quality of workers based on their performance on different tasks. The assumption behind this idea is, given multiple labels from different workers, we should trust the

label provided by the worker with a good performance record more than those with a bad record. This assumption has been translated into a machine learning model which gives high quality workers a higher weight to their votes.

In [30], the most informative samples have been defined as those unlabeled samples that are less likely to be incorrectly labelled but still provide high information. This work also estimates posterior class probabilities for labeled samples to discover the ones that are most likely to have been incorrectly labelled, and re-request the labels from the oracle. Since these two criteria are exclusive with each other, there is a probability of switching the label querying strategy at each iteration.

Welinder and Perona [90] came up with an idea of an online crowdsourcing algorithm which actively asks for labels only for images where the target value is still uncertain. Moreover, the algorithm also estimates annotator expertise and reliability and only assigns tasks to best annotators. Their algorithm has been tested with binary annotations, multi-valued attributes and continuous-valued annotations in experiments. Although this work also focuses on using the active learning strategy to estimate appropriate samples to be labeled, there are several differences between their work and mine. First, they applied the selective sampling strategy rather than the pool-based strategy in selecting the sample. At each iteration their algorithm adds several samples into the set and decides whether to request their labels by computing the probabilities of giving them the label z. As discussed in Section 2, selective sampling potentially suffers from the sampling bias problem. The decision about whether or not to request the label is governed by a threshold on the informative measure of giving the label z. This criterion may work well in binary classification problem but is neither robust nor easy to extend to multi-classification problems.

Kumar and Lease [45] compared two previous works ([67] and [71]) evaluating the relative merits of labeling additional unlabeled examples, or generating additional labels for labeled examples in order to reduce potential label noise. The experimental results with simulated annotators showed that incorporating knowledge of annotator accuracies into the model improves the performance of learning accuracy with annotation noise, particularly in adversarial settings.

Yan et al. [94] has published a paper on active learning for crowdsourcing which is closely related to this dissertation. In their work, they generated a graphical model with input data, labels collected from workers and true labels to estimate the expertise of workers. The paper mainly answers two questions: which task should be annotated and which worker should be asked to provide the label. For the first question, they assumed the distribution of input data belongs to either a Gaussian distribution or a Bernoulli distribution. In a binary classification problem, the sample that has the most difficult to fit in both distributions of positive class and negative class (which means $p(y = +1|\mathbf{X})$ is most close to $p(y = -1|\mathbf{X})$) should be selected as the most uncertain sample to be labeled. However, the explicit drawback of this selection strategy is that the sample with input data (feature) that is most ambiguous to the learning model is not necessarily difficult for workers to label. Moreover, the previous publication [77] has shown that many classification problems can not be easily represented with generative models. Therefore, a method [91] of estimating the performance of workers based on only labels is more appropriate practical environments such as crowdsourcing. My thesis steps forward to apply this idea in combination with the active learning technique.

Application Domain: Activity Recognition

Activity recognition using body-worn inertial sensors has been an important research topic for many years, particularly in the context of eldercare and healthcare applications; see [5] for a recent survey.

[7] and [64] recognized simple human activities from data acquired using accelerometers. [92] and [59] proposed approaches to estimate complex daily life activities based on object sequences collected using RFID tags attached to objects. To bootstrap the learning process, video data is combined with RFID data to jointly infer the most likely activity. The Carnegie Mellon University Multi-Modal Activity Database (CMU-MMAC) aims to provide a rich source of partially ground-truthed data for research in this area [25]. Promising results have been reported using different machine learning techniques on CMU-MMAC [72, 96] and this thesis uses it as a standard dataset to evaluate our proposed methods.

Conditional Random Fields

CRFs have been used for a variety of classification problems, including natural language processing [47, 22], computer vision [46, 76, 50, 60], human activity recognition [70, 51, 84] and bioinformatics [33]. In this thesis, I examine the use of CRFs for human activity recognition from motion trajectory data.

However, effective learning and inference of CRFs remain challenging problems. If the optimization cost function is nonlinear and nonconvex, learning and inference of CRFs must be implemented with sampling-based algorithms that take long time to converge. However, recent work by [50] showed the use of first order approximation to efficiently estimate conditional likelihood. [76] demonstrated the use of Gaussian CRFs by minimizing the error in

the model MAP solution to improve the learning algorithm.

More specifically, linear CRFs have been widely applied to classification and segmentation of sequential data. [47] demonstrate the use of linear CRFs to solve the label bias problem in natural language processing. The main superiority of linear CRFs is their ability to effectively take advantage of complex overlapping, non-independent features. For instance, [33] evaluated different feature sets to see if they improved the predictive power of a linear CRF; their experiments indicated that their model could effectively take advantage of discriminative features in alternate feature sets to improve the classification results, unlike their previous HMM-based classification system [53]. In an activity recognition problem, [70] applied their model on both images and motion capture sequences with overlapping observation features to demonstrate that the performance of their CRFs-based classification approach surpasses other an HMM-based model. In summary, since the linear CRF model allows non-independent features to be leveraged, the choice of features becomes a particularly important concern when constructing a model.

Feature Selection

Feature selection can make CRFs even more effective classifiers. There are three general strategies used in feature selection: *filters*, *wrappers* and *embedded* [83]. *Filters* treat feature selection as the pre-processing step, independent from the training stage. *Wrappers* consider all possible combinations of potential features to select the best feature set according to the learning performance. *Embedded* strategies perform feature selection during the training stage and iteratively grow the feature set.

Filters apply heuristic methods, such as correlation, mutual information and information gain [35], to select features prior to training. Those methods rank all potential features

by evaluating their relevance; features with high relevance scores are used as the feature set for training. It is worth noting that that such methods evaluate potential features only once; no features can be added or eliminated during training, which makes the filter strategy computationally efficient. However, the tradeoff is that the filters may select irrelevant features while ignoring highly relevant features if the heuristics do not perform well in a particular domain.

Wrappers perform an exhaustive search on all possible combination sets of potential features. Every set of features is used to train the model and evaluated with cross-validation [11]. However, the number of combination of feature sets increases exponentially with the number of potential features. Although this strategy avoids the *the weakly relevant feature* problem in filters, obviously it is computational inefficient if the feature pool is large.

The *embedded* method can be viewed as a tradeoff between filters and wrappers. This method generates a small set of features for training while leaving other features as candidates for selection. Candidate features are added to the set evaluated by the learning model. The feature set is iteratively grown through adding features with high evaluation scores. This method is more computationally efficient than wrappers and does not rely on potentially fallible heuristics. Currently, many real-world applications are based on embedded methods.

Prior work [69] has examined the relative benefits of using discriminative conditional random fields (CRFs) vs. commonly-used generative models (e.g., the Hidden Markov Models) on diverse activity recognition problems such as video recognition and analyzing Robocup team behavior. The consensus has been that many of features commonly used in activity recognition problems are based on overlapping time windows that nullify the observational independence assumptions of many generative graphical models. However, feature representation and selection does have an impact on both the performance and computational

accuracy of CRFs.

Due its resiliency to classification problems arising from non-independent features, CRFs can often leverage arbitrary and complex features derived from some basic features. Unfortunately, the strategy of adding as many features as possible to guarantee no loss of information creates certain problems. First, large feature sets can cause overfitting. Since each feature corresponds to one parameter in the CRF model, a large feature set implicitly means that there is a large set of parameters which can fit the training data perfectly while performing poorly in the test set. Convergence during learning becomes harder with large numbers of weights. [55] proposed a feature induction method for linear-chain CRFs in which features with the highest gain of the conditional likelihood are iteratively selected. Initially, the model generates a pool of feature candidates. Each feature candidate is then added to the current feature set to evaluate its contribution to the conditional likelihood of current model. Those features with highest gain are selected as new training features and the CRFs model is re-trained. Vail et al. [84, 82] applied a similar feature selection framework for robotic activity recognition. Vail's method generates a large pool of complex, overlapping features of robot activities, such as positions, velocities and distances. l_1 and l_2 regularization are applied to reduce feature quantity and speed up the selection process. However, the pool-based feature selection strategy is time consuming, since the conditional likelihood has to be re-calculated with every potential feature at each step, which is not practical if there are a large number of candidate features. Our proposed feature selection method draws from these ideas, but relies on an approximation technique to reduce the computation time.

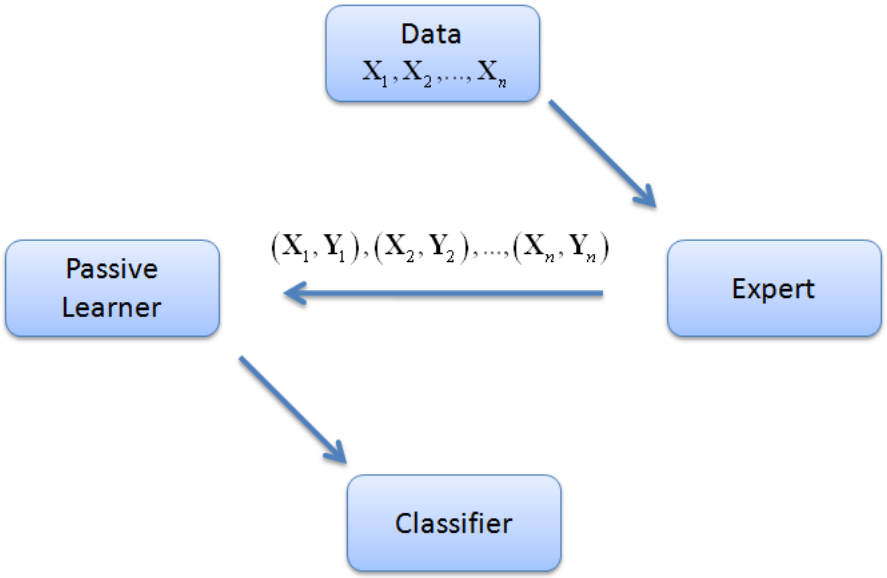
Motif Discovery

Feature selection enables the selection of the most discriminative features from an initial set of candidate features. For noisy sensor data from inertial measurement units, an open question is which features should be used in the initial candidate set. Unsupervised motif discovery has been demonstrated as a promising technique for identifying and matching imperfect repetitions in time series data; by using motifs as the basis for our CRF features, we can robustly identify subtle patterns of peaks and valleys in the IMU data.

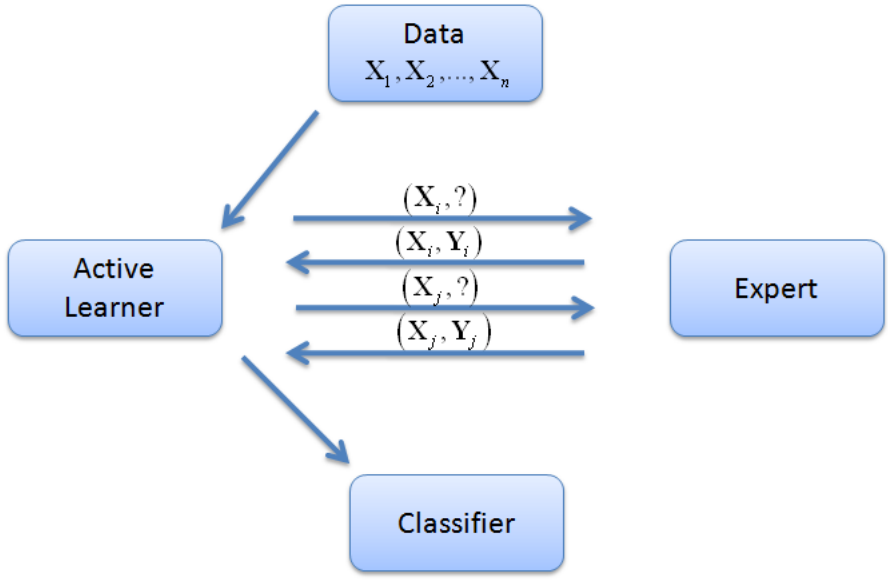
The problem of repeated subsequences in time series spans multiple research areas. Researchers have primarily addressed two problems: 1) initial motif selection criteria [75] and 2) rapid and robust matching [19]. [52] initially presented a formal definition of *motifs*, the approximately repeated subsequences in time series. Their approach for detecting motifs in time series suffers from certain limitations. A brute force algorithm requires a quadratic number of comparisons corresponding to the length of the time series. [52] proposed a triangular inequality approach to improve the time complexity of the comparisons; however their method requires the manually setting of large constant factors which makes it hard to extend to massive databases. Also this approach is sensitive to noise, as demonstrated by the example shown by [19] in which the matching of two subsequences is corrupted if the subsequence has a noisy downward spike. To solve those two issues, [19] proposes a novel motif discovery algorithm to efficiently discover motifs in which the time series is discretized as a symbolic sequences using Piecewise Aggregate Approximation (PAA) [41]. By switching to a symbolic representation, they achieve some noise reduction. In order to speed up the computation of motif matching, they apply random projection to approximate the comparison of every potential subsequence. [40] extend this method to finding arbitrarily scaled motifs in a massive time series database.

Another issue is how to generalize single dimensional motif techniques to the problem of detecting and matching motifs in multi-dimensional data [81]. [75] propose the use of a Minimum Description Length (MDL) principle to extract motifs from time series data. Principal Component Analysis (PCA) is then applied to synchronize motifs in multidimensional time series data. [81] demonstrated a graph clustering approach for grouping single dimensional motifs to solve the synchrony issue. In our work, we identify and match motifs in each dimension separately and use the conditional random field to learn the linkage between motif occurrences across different dimensions and action class labels.

Motif discovery has became a powerful tool to analyze time series data delivered by wearable sensors. [56] used motifs in conjunction with HMMs to distinguish six arm techniques from a single IMU, but the assumption was made that each action could be characterized by a single six-dimensional motif. Their work synthesizes several existing approaches to estimate the hidden location probability and motif model parameters. First, the algorithm generates a pool of candidate motifs and selects the best motifs based on their scores of an informative-theoretic criterion. Then the seed motifs are refined by splitting different motifs or merging similar motifs. Finally a Hidden Markov Model [62] is built with those seed motifs and corresponding occurrences could be detected by decoding the HMM. In contrast, our data is substantially more complicated (full body data generated from from five IMU sensors) and our action set is composed of 14 unscripted actions performed during a cooking task (e.g., "stir brownie mix", "pour mix in pan", "get fork", "break eggs", and "pour oil"). The [81] motif discovery method was used in two wearable systems, SmartCane [93] and SmartShoe [23], which include accelerometers, gyros and press sensors. [73, 74] apply their string-matched activity template to recognize continuous activities in a car assembly scenario.



(a) Passive learning



(b) Active learning

Figure 2.1: Active learning vs. passive learning.

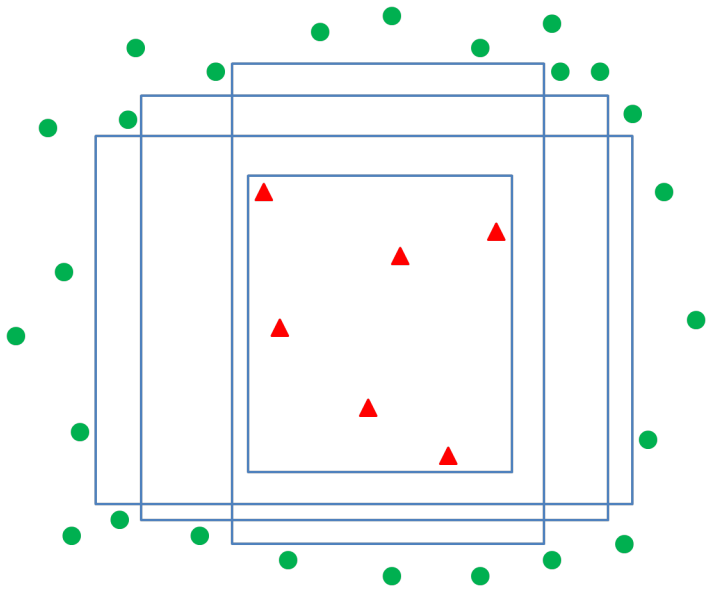


Figure 2.2: Version space.

CHAPTER 3: ACTIVE LEARNING

Human activity datasets collected under natural conditions are an important source of data. Since these contain multiple activities in unscripted sequence, temporal segmentation of multimodal datasets is an important precursor to recognition and analysis. Manual segmentation is prohibitively time consuming and unsupervised approaches for segmentation are unreliable since they fail to exploit the semantic context of the data. Gathering labels for supervised learning places a large workload on the human user since it is relatively easy to gather a mass of unlabeled data but expensive to annotate. This chapter proposes an active learning approach for segmenting large motion capture datasets with both small training sets and working sets. Support Vector Machines (SVMs) are learned using an active learning paradigm; after the classifiers are initialized with a small set of labeled data, the users are iteratively queried for labels as needed. I propose a novel method for initializing the classifiers, based on unsupervised segmentation and clustering of the dataset. By identifying and training the SVM with points from pure clusters, the proposed method improves upon a random sampling strategy for creating the query set. The active learning approach improves upon the initial unsupervised segmentation used to initialize the classifier, while requiring substantially less data than a fully supervised method; the resulting segmentation is comparable to the latter while requiring significantly less effort from the user.

Sample Efficiency through Active Learning

In the initial phase, our SVM classifiers are initialized with a small set of training data from an unsupervised clustering. During active learning, the classifiers are iteratively trained by having the users provide labels for a small set of automatically selected samples. Although the

classifiers can be initialized by having the user provide labels for randomly sampled frames, we demonstrate that we can improve on that by selectively querying and propagating labels using a clustering approach.

Data Clustering

Several methods have been proposed to cluster data in geometric space [68, 99]. Since the motion segmentation problem is based on continuous time data sequences, it is possible to base the clustering on temporal discontinuities in the data stream. We use the PCA segmentation approach [8] outlined in the previous section to provide a coarse initial segmentation of the data.

Each raw motion capture frame can be expressed as a pose vector, $\mathbf{x} \in \Re^d$, where $d = 56$. This high-dimensional vector can be approximated by the low-dimensional feature vector, $\theta \in \Re^m$, using the linear projection:

$$\theta = \mathbf{W}^T(\mathbf{x} - \mu), \tag{3.1}$$

where $\mathbf{W}$ is the principal components basis and μ is the average pose vector, $\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$. The projection matrix, $\mathbf{W}$, is learned from a training set of $N = 300$ frames of motion capture data. $\mathbf{W}$ consists of the eigenvectors corresponding to the m largest eigenvalues of the training data covariance matrix, which are extracted using singular value decomposition (SVD). Transitions are detected using the discrete derivative of reconstruction error; if this error is more than 3 standard deviations from the average of all previous data points, a motion cut is introduced.

We found that this method provides a better starting point than traditional unsupervised

clustering methods, such as k-means, which do not consider temporal information. Many of the clustering errors generated by the coarse segmentation are detected by pruning clusters based on a small set of labels solicited from the user. We ask the user to label the endpoints of the coarse segmentation and perform a consistency check on the labels; if both endpoints have the same label, the segment is potentially pure; however if the labels of the endpoints disagree, we add a new cut in the middle of the segment and query the user for the label of that point. Clusters shorter than a certain duration (1% of total sequence length) are eliminated from consideration. The remaining clusters are used to initialize the support vector machine classifiers; labels from the end points are propagated across the cluster and the data is used to initialize the SVMs. This process requires the user to label only 20–30 frames.

Active Learning

The clusters created by the coarse PCA segmentation, and refined with the user queries, are used to train a SVM classifier with both labeled and unlabeled samples. Semi-supervised support vector machines are regarded as a powerful approach to solve the classification problem with large data sets. Learning a semi-supervised SVM is a non-convex quadratic optimization problem; there is no optimization technique known to perform well on this topic [18]. However, our solution is a little different to the traditional methods based on linear or non-linear programming. Instead of searching for the global maximum solution directly, we use a simple optimization approach which may not identify the optimal margin hyperplane but will help the classifier decide which unlabeled samples should be added into the training set to improve the classification performance. We then query the user for the class labels of each of the selected samples and add them back to the training set. Suppose the labeled samples are denoted by $\mathcal{L} = \{x_1, x_2, ..., x_l\}$ and the unlabeled samples

are $\mathcal{U} = \{x_{l+1}, x_{l+2}, ..., x_n\}$, the SVM classification problem can be represented as finding the optimal hyperplane with labeled samples that satisfies the equation:

$$\begin{aligned} &\min_{\mathbf{w},b,\epsilon} \quad C\sum_{i=1}^{l}\epsilon_i + \|\mathbf{w}\|_2 \\ &s.t. \qquad\quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \quad i = 1, ..., l \end{aligned} \tag{3.2}$$

where ϵ_i is a slack term such that if $\mathbf{x_i}$ is misclassified and C is the constant of the penalty of the misclassified samples. All possible hyperplanes that could separate the training data as $f(\mathbf{x_i}) > 0$ for $y_i = 1$ and $f(\mathbf{x_i}) < 0$ for $y_i = -1$ are consistent with the version space $\mathcal{V}$. Tong and Koller [80] have shown that the best way to split the current version space into two equal parts is to find the unlabeled sample whose distance in the mapping space is close to the current hyperplane $\mathbf{w}_i$. The description of our method is detailed in Algorithm 1.

Algorithm 1 Proposed active learning algorithm

Require:

Input: The complete data set with labeled set $\mathcal{T}$ and unlabeled set $\mathcal{U}$.

Output: h^*: The optimal SVMs hyperplane to separate the available data into two groups.

1: **while** the variation classification hyperplane is not stable **do**
2: Calculate the distance **d** between unlabeled set $\mathcal{U}$ and the current SVMs hyperplane $\mathbf{w_i}$
3: Query the unlabeled sample x_{l+i} with the smallest distance d_i
4: Manually label the sample x_{l+i}
5: Update the labeled set as $\mathcal{T} = \mathcal{T} \cup \{x_{l+i}\}$ and unlabeled set as $\mathcal{U} = \mathcal{U} \setminus \{x_{l+i}\}$
6: Re-train the SVM classifiers $\mathbf{w_{i+1}}$ with labeled set $\mathcal{T}$
7: **end while**
8: Return the optimal SVM hyperplane h^* trained on $\mathcal{T}$.

The traditional initialization method arbitrarily selects samples to include in the training sets. However, randomly choosing samples may lead to sampling bias which make the SVM

classifier unable to achieve the global maximum. In our approach, the labels of samples in each viable cluster are set as the majority labels of querying samples. This converts learning a semi-supervised SVM into a classical SVM optimization problem. However, the clustering strategy does not guarantee that the decision boundary is optimal since the clustering step is not reliable. It merely gives a good initial hyperplane; active learning is still required to perfect the solution.

In our experiments, the SVM classifier was implemented with the SVM-KM toolbox using a polynomial kernel [15]; multi-class classification is handled using a one vs. all voting scheme. Instead of using a *hard margin* for the SVM, our method relies on a *soft margin* restriction in classification. A hard margin forces both labeled and unlabeled data out of the margin area, whereas the soft margin allows unlabeled samples to lie on the margin with penalties. With limited training samples, we find that the hard margin restriction is so restrictive that it may force the separating hyperplane to a local maximum.

To demonstrate this approach, we used the publicly available Carnegie Mellon Motion Capture dataset (http://mocap.cs.cmu.edu/) collected with a Vicon motion capture system. Subjects wore a marker set with 43 14mm markers that is an adaptation of a standard biomechanical marker set with additional markers to facilitate distinguishing the left side of the body from the right side in an automatic fashion. The dataset contains a bunch of sequences with different human actions; to evaluate our method we selected 15 sequences that include actions such as running, swinging, jumping, and sitting. The first baseline (C) is trained using data that is sampled at random (with uniform distribution) from the activity sequence. The second (B) is initialized using a random segmentation but employs our proposed margin-based approach for generating instances for the user to label. The third (A) is our proposed approach and employs an unsupervised clustering to initialize the segmentation followed by margin-based sampling for identifying informative active learning

query instances.

We evaluate the quality of segmentation using classification accuracy. Figure 3.1 shows how this accuracy improves with additional training data for each of the methods. Clearly, adding training data in a haphazard manner (C) leads to an inefficient form of active learning. The second method (B) demonstrates the benefits of our margin-based method for selecting queries for active learning. Finally, the accuracy curve for the proposed method (A) shows the boost that we obtain through intelligent initialization using unsupervised clustering. In comparison to a fully supervised SVM trained with 100 samples, our method achieves the same 95% accuracy with only half the data (40 samples).

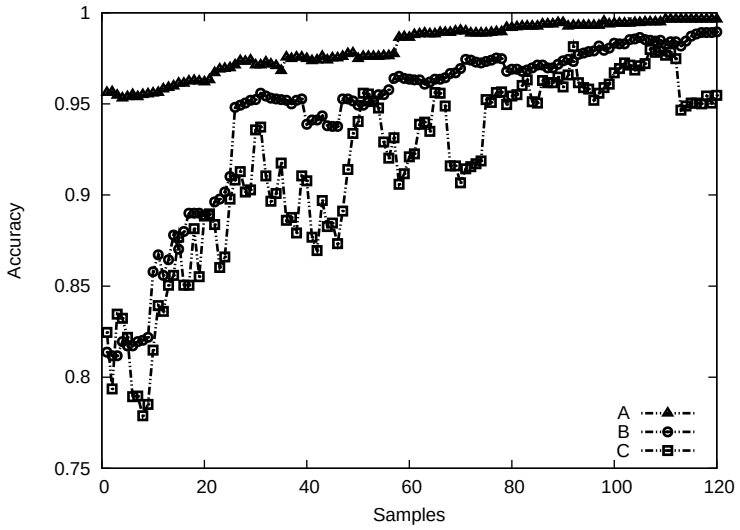


Figure 3.1: Improvement in quality of SVM segmentation as additional labels are acquired using active learning. The proposed method (A) benefits through intelligent initialization and margin-based selection of active learning queries.

CHAPTER 4: ACTIVE LEARNING IN CROWDSOURCING

Crowdsourcing has become an popular approach for annotating the large quantities of data required to train machine learning algorithms. However, obtaining labels in this manner poses two important challenges. First, naively labeling all of the data can be prohibitively expensive. Second, a significant fraction of the annotations can be incorrect due to carelessness or limited domain expertise of crowdsourced workers. Active learning provides a natural formulation to address the former issue by affordably selecting an appropriate subset of instances to label. Unfortunately, most active learning strategies are myopic and sensitive to label noise, which leads to poorly trained classifiers.

This paper presents a practical method for pool-based active learning that is robust to annotation noise. This chapter demonstrates, using several standard datasets, that the proposed approach, which employs label prediction in combination with importance-weighting, significantly improves active learning in the presence of annotation noise.

Incremental Relabeling for Active Learning

Figure 4.1 presents an overview of our activity annotation system with feature selection and active learning on Amazon's Mechanical Turk. The goal of the annotation system is to accurately label large quantities of IMU data with activity labels, from which we can train activity recognition classifiers. Given temporally-aligned video and IMU data, such as those provided in the CMU-MMAC dataset [25], we assign MTurk workers to label short, automatically-segmented video clips of cooking activities with the label(s) corresponding to the current action(s).

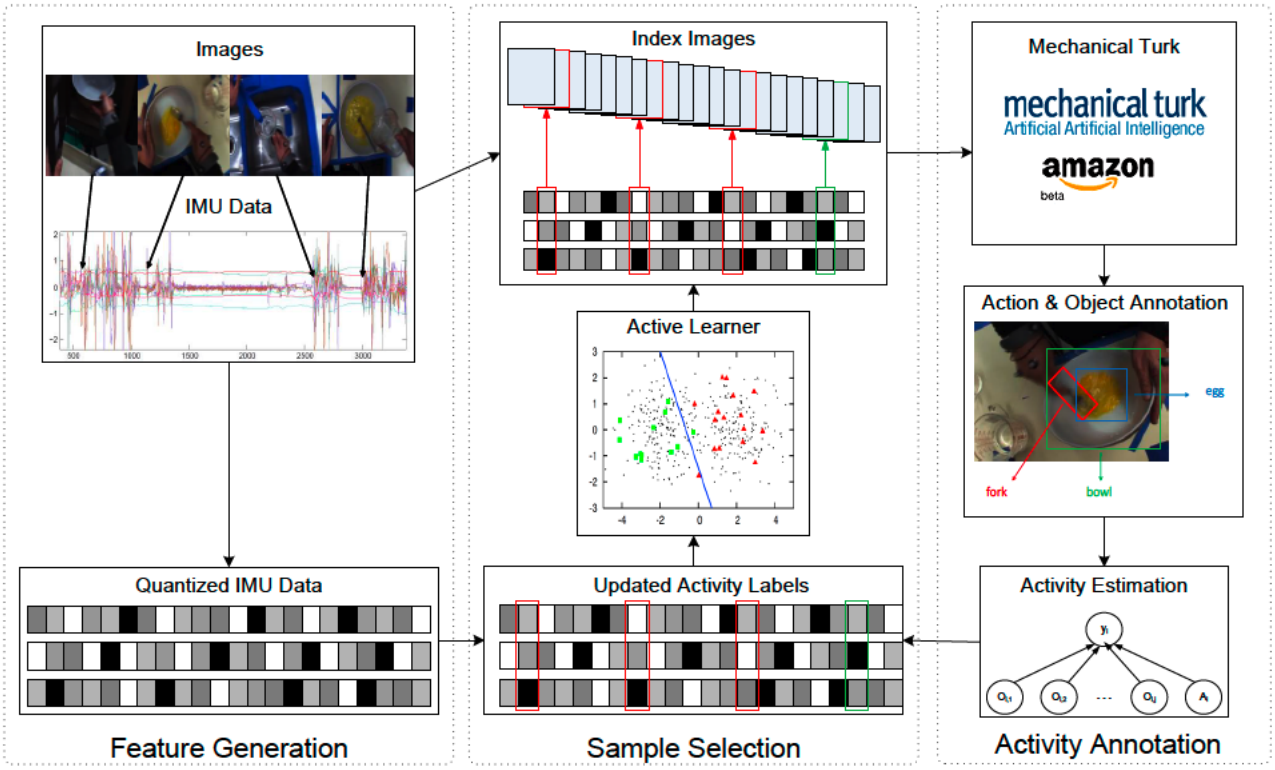


Figure 4.1: Overview of the proposed method

Since labeling IMU data is difficult for untrained users, we crowdsource short activity labels for temporally aligned video clips supplemented with object labels for still images. The object labels generate more accurate activity labels. Our modified active learning approach uses both uncertainty and inconsistency, requests relabels for important samples, and rejects problematic samples. The resulting system can train accurate activity recognition classifiers from unreliable crowdsourced data.

Unfortunately, labels predicted using the raw crowdsourced labels are unacceptably inaccurate. Inspired by ideas in multi-task learning [16], we ask each worker to solve a second related task — to identify visible objects. As detailed below, we train a Naive Bayes classifier to infer actions based on the combined annotations. We then apply this framework in an active learning context to iteratively annotate data. The key idea is to select instances according to a mixture of two criteria: 1) Maxmin [79] and 2) based on the distribution of

labels in local neighborhoods.

Inferring Actions from Visible Objects

Our experiments use the CMU-MMAC dataset [25], which consists of data collected from subjects performing unscripted recipes in an instrumented kitchen using a variety of sensors. Our goal is to perform reognition on the IMU data, but because this data is difficult to annotate directly, we also provide annotators with temporally aligned video from an egocentric camera. As discussed earlier, to compensate for the poor annotation accuracy of raw crowdsourced action labels, we ask MTurk workers to annotate which objects are visible in each scene. We use these as a secondary source from which to infer action labels, and the two sources are combined using a Naive Bayes formulation. Experiments show that while this significantly improves annotation accuracy, the label noise is still too high for traditional active learning methods.

Adding Robustness to Label Noise for Active Learning

Active learning seeks to iteratively obtain labels for data by identifying, at each iteration, the most uncertain sample (i.e., the instance, which if labeled would have the greatest impact on the classifier). Before describing our modification, we briefly review active learning in the context of a support vector machine (SVM) framework. Let $\mathcal{T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$ be a set of initially labeled instances, with corresponding labels given by $\mathcal{L} = \{y_1, y_2, \ldots, y_l\}$. We also define a set of unlabeled instances as $\mathcal{U} = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_n\}$. Our method extends Tong and Koller's *Maxmin Margin* strategy [80] for selecting samples that have the greatest impact on the loss: $\text{Loss}_{\text{uncertainty}}(\mathbf{x}_i) = \min_{\mathbf{x}_i \in \mathbf{U}}(V^+(\mathbf{x}_i), V^-(\mathbf{x}_i))$, where $V^+(\mathbf{x}_i)$ and $V^-(\mathbf{x}_i)$ denote the sizes of the version spaces resulting from labeling $\mathbf{x}_i$ as $+$ or $-$, respectively. This

formulation extends naturally from binary to multiclass settings by computing the product of the loss function for the appropriate classification hyperplanes from all classes. The version space idea also applies to SVMs with non-linear kernels; in our work, we employ the popular Gaussian kernel.

Note that this standard formulation for SVM-based active learning is potentially ill-suited to our task for two important reasons. First, it assumes that the requested labels are perfect. This can be a dangerous assumption since it makes the SVM very sensitive to label noise. Second, a given sample, once labeled can never be selected for relabeling, which means that an error, once made, tends to become "locked in".

To address the first concern, we modify the loss function, as described below, to add a term characterizing the inconsistency of each labeled sample based on labels in its neighborhood. The intuition behind this idea is that one should request labels for samples that are in unexplored regions of the feature space or those near samples that may have been incorrectly labeled. To address the second concern, we also propose relabeling such points using crowdsourcing. Let $\mathbf{x}_i \in \mathcal{U}$ be the unlabeled sample with unknown label y_i and $\mathbf{x}_j \in \mathcal{T}$ be a labeled sample with known label y_j. Now, we can estimate the probability of labeling $\mathbf{x}_i$ as class c (based on labels in its neighborhood) as

$$p(y_i = c) = \frac{1}{|\mathcal{T}|} \sum_{\mathbf{x}_j \in \mathcal{T}} h(c, y_j) f(\mathbf{x}_i, \mathbf{x}_j). \tag{4.1}$$

Here, $h(c, y_i) = 1$ if the labels agree and 0 otherwise, and $f(.,.)$ is a Gaussian function (corresponding to the SVM RBF kernel). We formulate the inconsistency term of the modified loss simply as the entropy of the label distribution $p(y_i)$:

$$\text{Loss}_{\text{inconsistency}}(\mathbf{x}_i) = -\sum_{c \in \mathcal{C}} p(y_i = c) \ln(p(y_i = c)) \tag{4.2}$$

Finally, the modified loss consists of a linear combination of these two terms. The sample that we select is obtained using:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}_i \in \mathbf{U}} \text{Loss}_{\text{uncertainty}}(\mathbf{x}_i) + \lambda \cdot \text{Loss}_{\text{inconsistency}}(\mathbf{x}_i), \tag{4.3}$$

where λ denotes the mixing weight between the two criteria and depends on the expected annotation accuracy. Given *a priori* knowledge about annotation accuracy, one could set λ appropriately — with reliable labels, a lower value for λ suffices. In practice, since the expected accuracy of labels is unknown, we employ cross-validation with a hold-out set to determine λ. As suggested in [28], we initialize our labeled set by crowdsourcing k "representative samples" taken near cluster centers of our data.

Relabeling Strategy

As discussed above, the modified active learning scheme can crowdsource additional labels for samples that might have been incorrectly labeled. Given multiple labels, the most straightforward strategy for incorporating them is a voting scheme where the most popular label is assumed to be correct, termed *majority relabeling* (MR) in our experiments below. Unfortunately, this form of voting is no better than random when too many workers disagree (when it would be better to discard the sample) and does not distinguish between a clear and slight majorities.

For these reasons, we propose an alternate scheme, termed *absolute majority relabeling* (AMR) that incrementally adds labels until the sample is either accepted (with a clear majority) or deemed problematic due to inconsistent labeling and discarded. The intuition behind our proposed scheme is straightforward: given the distribution of labels for the sam-

ple, we accept the majority label iff it has received more than 50% of the votes. Otherwise, we incrementally request additional labels; if a clear majority fails to emerge after $N = 7$ labels, we mark the sample as problematic and remove it from the training set.

$$\text{relabeling accepts label } c \text{ iff:} \quad p(Y = c) > 0.5. \tag{4.4}$$

We can make a further modification to the AMR strategy by allowing workers to provide soft votes for multiple labels rather than a hard vote for a single label (analogous to cumulative voting). Specifically, each worker's vote consists of the label distribution $p(Y|O, A)$ generated using Naive Bayes (described in Section 4). These histograms are accumulated and normalized. The decision criterion remains the same: if $\exists c \in \mathcal{C}$ s.t. $p(Y = c) > 0.5$ then we accept the label c. If no consistent labeling emerges after N workers have relabeled the sample, we discard the sample from the training set. In the experiments below, we term this modified relabeling scheme as *absolute cumulative majority relabeling* (ACMR).

Experiments

We present a selection of results from three categories of experiments. In the first set, we perform a controlled study on synthetic data by corrupting labels with known quantities of noise to evaluate the robustness of different active learning selection criteria. The second set evaluates the three proposed relabeling strategies under controlled conditions. The third set examines the performance of the different active learning sample selection criteria and the relabeling strategies on the CMU-MMAC dataset, and examines the impact of combining object and action labels to infer more accurate annotations.

Robustness to Annotation Noise

The first set of experiments examines the robustness of various active learning strategies to annotation noise. Figure 4.2, 4.3 and 4.4 compares four active learning sample selection strategies under four scenarios with increasing levels of random error ($r = 0\%$, 20%, 40%). The selection strategies are:

1. Maxmin [80] - the standard criterion that only considers uncertainty (black);
2. DWUS [58] - a global criterion that combines both uncertainty and data distributions (red);
3. Proposed - our criterion that combines both uncertainty and inconsistency criteria (blue).

In these experiments, there is no systematic error ($s = 0$).

We make several observations. First, in the noise-free case, the classifier error of all three strategies decreases sharply and converges to a low error rate. **Maxmin** and **Proposed** perform better than **DWUS** at the final phase of the learning. This is consistent with our expectations that the standard criterion, which mainly focuses on most uncertain samples is well suited for noise-free scenarios, and that our proposed criterion can match this.

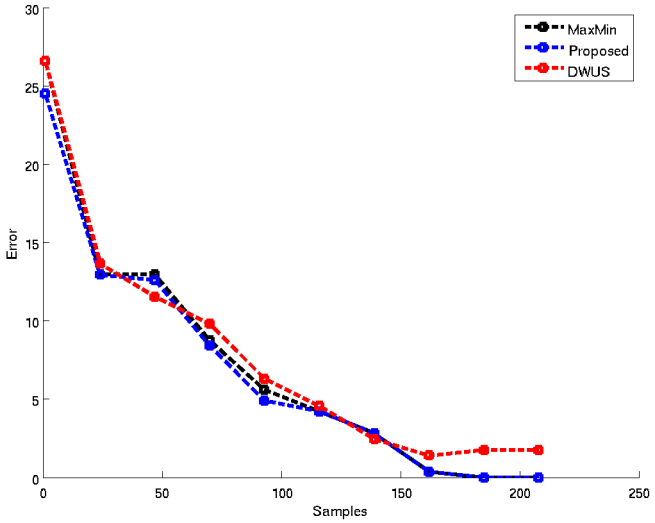


Figure 4.2: 0% noise - Impact of label noise on classifier error rates for different active learning selection criteria (lower is better). The figure key is as follows: 1) Maxmin [80] (red), 2) DWUS [58] (black) 3) Proposed (blue).

Next, in the $r = 20\%$ noise case, we see that **DWUS** decreases quickly but converges to a higher error rate since the representative term in its loss function forces the strategy select samples from high density regions. That means although **DWUS** benefits by selecting representative samples in the initial phase, it suffers in the final phase because the most uncertain samples remaining probably do not occur in the high-density regions. We observe that the **Proposed** scheme continues to have low errors in the final phase of learning since it focuses on inconsistent samples that have mislabeled neighbors.

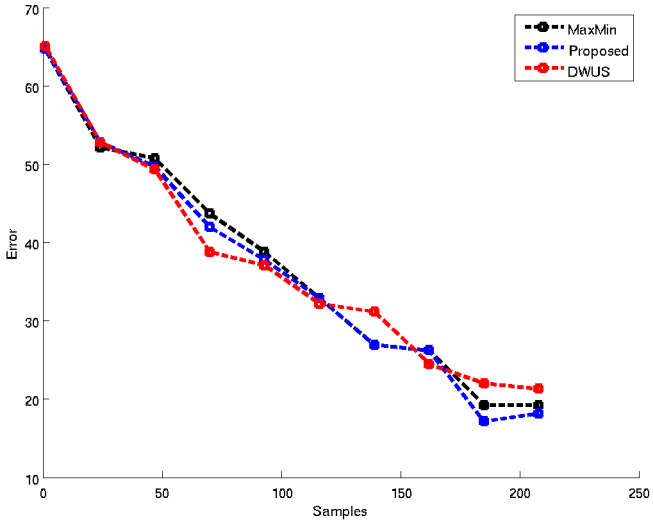


Figure 4.3: 20% noise - Impact of label noise on classifier error rates for different active learning selection criteria (lower is better). The figure key is as follows: 1) Maxmin [80] (red), 2) DWUS [58] (black) 3) Proposed (blue).

Finally, under the challenging conditions of 40% noise, none of the three criteria perform particularly well. From this, we can conclude that improving the criteria for selection in active learning can effectively counter moderate noise but is not sufficient by itself when dealing with very noisy annotations. This validates our earlier observation that crowdsourced labels may require judicious relabeling in addition to improved active learning strategies.

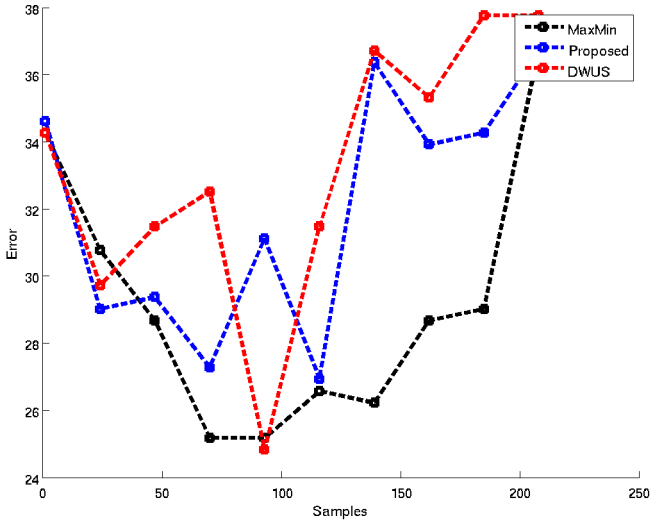


Figure 4.4: 40% noise - Impact of label noise on classifier error rates for different active learning selection criteria (lower is better). The figure key is as follows: 1) Maxmin [80] (red), 2) DWUS [58] (black) 3) Proposed (blue).

Relabeling — Synthetic Data

In this section we evaluate the performance of relabeling strategies on synthetic data with both random and systematic noise. We employ two sets of synthetic data in this experiment. The first simulates annotation with random error $r = 20\%$. The second set of data has 50% of samples correctly annotated and the other 50% with systematic error $s = 40\%$. The relabeling strategy can request up to 7 labels to annotate a given sample. We applied **Maxmin** and our proposed active learning methods to select the most informative sample to be annotated. This experiment involves two relabeling strategies:

1. Majority relabeling (MR) - a sample is relabeled with the most popular action label. If multiple labels are equally popular, a random label from that set is used.

2. Absolute majority relabeling (AMR) - proposed strategy using only the raw action labels.

Figure 4.5 indicates the results by applying different relabeling strategies on synthetic data with random error ($r = 20\%$) only. The results shows that by asking multiple workers to annotate samples, the majority relabeling (MR) (dashed lines) effectively reduces the learning error to a low rate (approximately 5%) even when the label noise is 20%. Moreover, with the absolute majority relabeling (AMR) (solid lines), which identifies unreliable labels and asks additional workers to relabel them, the learning error converges rapidly to a much lower level for both of the active learning sample selection criteria.

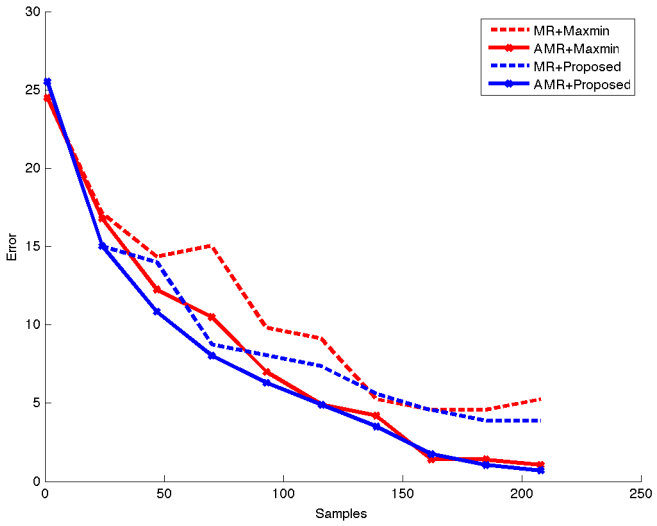


Figure 4.5: The performances of different relabeling strategies on synthetic data with random error ($r = 20\%$).

Another experiment on the synthetic data with systematic error ($s = 40\%$) is shown in Figure 4.6. This experiment also validates the benefit of relabeling. AMR (solid lines) outperforms the MR (dashed lines).

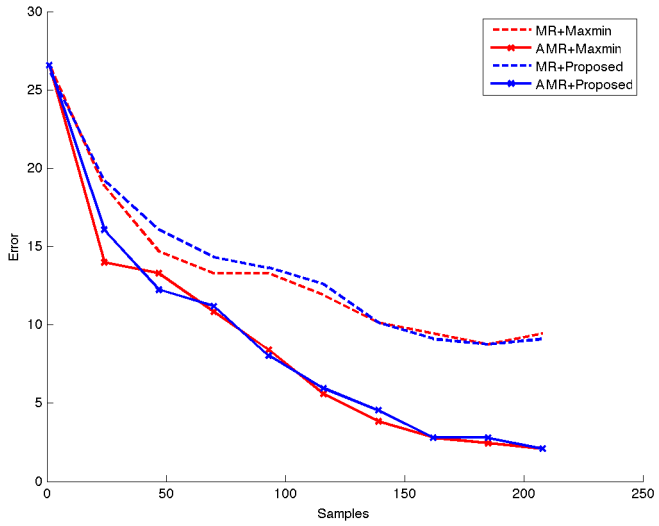


Figure 4.6: The performances of different relabeling strategies on synthetic data with systematic error ($s = 40\%$). Absolute majority (AMR) dominates majority relabeling for both active learning selection criteria.

Relabeling — CMU-MMAC Dataset

In this experiment, we compare the classifier error of different relabeling strategies (see Figure 4.7). Specifically, we examine the following conditions:

1. Majority relabeling (MR) - a sample is relabeled with the most popular action label (black dashed).

2. Absolute majority relabeling (AMR) - proposed strategy described above (black solid).

3. Majority + augmented labels (MR+Augment) - for each worker, we infer an augmented label using the Naive Bayes model based on both action and visible objects and select the most popular (red dash).

4. Absolute majority relabeling + augmented labels (AMR+Augment) - AMR applied to augmented labels (red solid).

5. Cumulative majority relabeling + augmented labels (CMR+Augment) - rather than accumulating peak votes as in MR+Augment, we take the peak of the summed distributions (blue dashed)

6. Absolute cumulative majority relabeling + augmented labels (ACMR+Augment) - we accept the majority label from CMR only if it has an absolute majority, i.e., $> 50\%$ of probability mass (blue solid).

Note that in AMR and ACMR, samples where incremental crowdsourcing fails to obtain a label that can capture an absolute majority ($> 50\%$ after up to $N = 7$ independent votes) are discarded from the training set. To facilitate direct comparisons, all of these relabeling strategies are employed in conjunction with the standard uncertainty-based active learning sample selection criterion (Maxmin [80]).

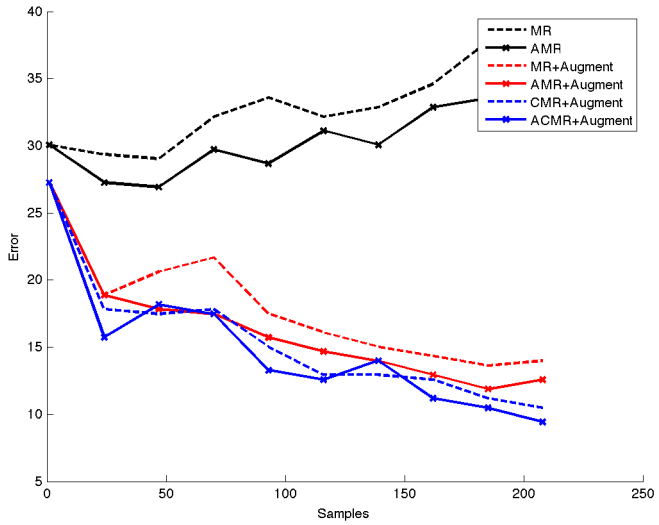


Figure 4.7: The learning error by using Maxmin active learner combined with different relabeling strategies.

As observed in Figure 4.7, using the raw action labels collected directly from MTurk (black dashed and solid) is problematic in conjunction with Maxmin because the label noise is severe enough to cause active learning to increase classifier error over time. This is true both for the straightforward majority relabeling strategy as well as our proposed AMR strategy (though the latter does better). Employing the augmented feature set, which combines action and object labels, addresses this shortcoming.

We also observe that employing cumulative voting with the augmented features (CMR+Augment, ACMR+Augment) during active learning significantly improves the classifier performance, and that the proposed absolute majority schemes are consistently better than their counterparts (AMR vs. MR, and ACMR vs. CMR), as seen by the contrast between dashed and solid lines. This validates our hypothesis that discarding inconsistently labeled samples

from the training set leads to better performance. The best performance is achieved by the proposed method (ACMR+Augment), which combines absolute majority voting over label distributions with the augmented object+action features.

The final set of experiments (see Figure 4.8) explores the impact of the different active learning sample selection criteria on the CMU-MMAC dataset. In this experiment, all of the active learning criteria used the best relabeling strategy along with the augmented feature set (ACMR+Augment), with a pool of upto $N = 7$ independent relabeling opportunities. The active learning strategies are:

1. Maximin [80], which focuses on sample uncertainty (black line);
2. DWUS [58], which considers both unertainty and representative samples using the data distribution (red line);
3. Proposed method that combines sample uncertainty with label consistency (blue line).

We see that DWUS initially improves faster than Maximin, since focusing on representative samples is an advantage. However, the strategy ultimtely converges to a higher error rate than the others because DWUS has reduced sensitivity to the uncertain samples. The proposed method performs much better than DWUS and slightly better than Maxmin. The addition of the inconsistency term initially pushes the selection criterion towards samples far from the labeled data and then gradually picks up samples that have suspicious labels (those inconsistent with their local neighborhood).

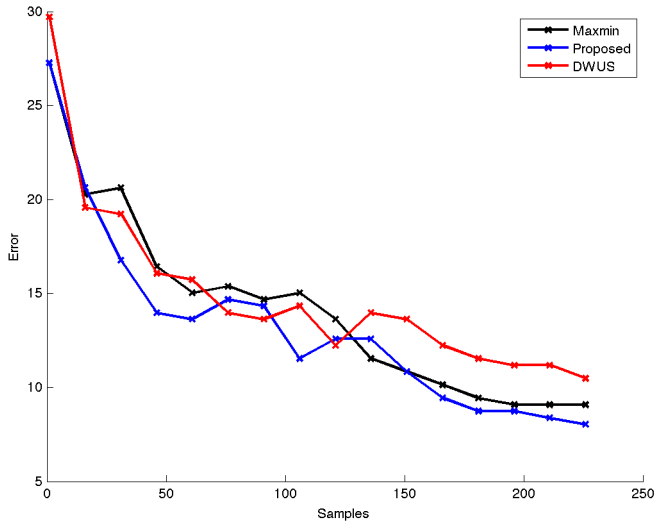


Figure 4.8: Proposed relabeling strategy with different active learning methods on real data from MTurk.

Importance-Weighted Label Prediction for Active Learning

We still formulate the active learning problem with annotation noise as follows. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denote the (initially unlabeled) pool of instances. We model the oracle as a process that, given an instance x_i, returns an incorrect label with i.i.d. probability η and the correct label otherwise. We assume that the noise rate $\eta \in (0, 0.5)$ is unknown. Following standard approaches in active learning, we initially request noisy labels for an initial set $\mathcal{L}$ of randomly-selected instances, where $|L|$ is typically 10% of $|\mathcal{X}|$. Using these, we train an ensemble of K support vector machines (SVMs) on bootstrap samples [31] of $|\mathcal{L}|$ elements, drawn from the labeled set $\mathcal{L}$. This set of K SVMs, denoted as $\mathcal{H} = \{h_1, \ldots, h_K\}$, correspond to the bootstrap estimators employed in the bootstrap variant of IWAL [10]. In

our implementation, we assume (without loss of generality) that SVM outputs are interpreted probabilistically, e.g., using Platt scaling [61].

Algorithm 2 Importance-Weighted Label Prediction for Active Learning Algorithm

Require:

Input: $\mathcal{X}$: set of (initially unlabeled) instances;

$p_{\min}$: lower bound for importance sampling;

θ: label prediction confidence threshold;

T: period over which ensemble retraining occurs;

B: labeling budget.

Output: h^*: The optimal classifier.

$\mathcal{T} \leftarrow$ randomly drawn subset of $\mathcal{X}$ (e.g., 10%);

Request (noisy) labels for $\mathcal{T}$ from oracle;

Train ensemble of probabilistic SVMs

$\mathcal{H} = \{h_1, \ldots, h_K\}$ using bootstrap sampling on $\mathcal{T}$;

while $B > 0$ **do**

  Compute $\pi(\mathbf{x})$ $\forall \mathbf{x} \in \mathcal{X}$ using Equation 4.5;

  Select a sample $\mathbf{x}_i$ from $\mathcal{X} \setminus \mathcal{T}$ using importance sampling with normalized CDF from π;

  if $s(\mathbf{x}_i, \mathcal{H}) < \theta$ **then**

    $y_i \leftarrow$ predicted label for $\mathbf{x}_i$ using $\mathcal{H}$;

  else

    Request label for $\mathbf{x}_i$ from oracle;

    $B \leftarrow B - \text{cost}(\mathbf{x}_i)$;

    After every T queries to oracle:

    (Optional:) Retrain ensemble $\mathcal{H}$ on bootstrap of $\mathcal{T}$;

  end if

  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{x}_i\}$ with weight $1/\pi(\mathbf{x}_i)$;

end while

Return SVM h^* trained on weighted set of labeled data.

Our proposed method now diverges from IWAL in several respects, as detailed in Algorithm 2. Rather than simulating a stream by drawing unlabeled samples from the unlabeled pool $\mathcal{X} \setminus \mathcal{L}$, we sample directly from the pool in a more computationally efficient manner using importance sampling. Specifically, we calculate the probability $\pi(\mathbf{x})$ of sampling an unlabeled element $\mathbf{x}$ by running it through the K classifiers in $\mathcal{H}$ using:

$$\pi(\mathbf{x}_i) = p_{\min} + (1 - p_{\min}) \left(h_{\max}(\mathbf{x}_i) - h_{\min}(\mathbf{x}_i) \right), \tag{4.5}$$

where $p_{\min}$ is a constant (typically set to 0.1) and $h_j(.)$ is the probabilistic output of an SVM classifier from $\mathcal{H}$, with $h_{\min}(.)$ and $h_{\max}(.)$ denoting $\min_{h \in \mathcal{H}} h(\mathbf{x}_i)$ and $\max_{h \in \mathcal{H}} h(\mathbf{x}_i)$, respectively. In other words, $\pi(\mathbf{x})$ is highest when there is significant disagreement between any two classifiers in the bootstrap ensemble and lowest when all of them agree. We sample from the data according to π simply by generating a cumulative probability distribution, where each instance $\mathbf{x}_i$ in the unlabeled pool $\mathcal{X} \setminus \mathcal{L}$ corresponds to an interval in (0,1] proportional to $\pi(\mathbf{x}_i)$. Then, a random number drawn in (0,1] maps to the selected instance. We emphasize that this sampling process induces a *biased* sampling that favors those instances with high $\pi(.)$, i.e., those that most merit labeling. We make two important observations. First, this bias leads the distribution of data in $\mathcal{L}$ to drift away from that in the original data set $\mathcal{X}$. Unless the bias is rectified, training on $\mathcal{L}$ may not result in classifiers that are well suited to solving the original problem. Second, it is important to sample using $\pi(.)$ rather than pursuing an aggressive strategy such as ranking unlabeled instances according to $\pi(.)$ and requesting labels only for the top few, since that would exacerbate the sampling bias.

Based on these observations, after obtaining a label for the given instance $\mathbf{x}_i$, we add it to the labeled set $\mathcal{L}$ with a weight of $1/\pi(\mathbf{x}_i)$, which unbiases the distribution of labeled data. Intuitively, one can consider this as finely sampling the uncertain data points (and adding

them with small weights) while coarsely sampling the more certain data points (and adding them with greater weights).

Unlike IWAL, we do not necessarily request a label for every point $\mathbf{x}_i$ drawn from the unlabeled pool. For some instances, including some for which $\pi(\mathbf{x}_i)$ is high due to disagreement among a pair of classifiers, it is possible that the overwhelming majority of the remaining classifiers is still in agreement. For such instances, particularly when η is low, we can short-circuit the labeling process and add $\mathbf{x}_i$ to $\mathcal{L}$ with the majority predicted label, without consulting the oracle. This form of label prediction, which can be viewed as a variant of self-training, is employed only in cases where the ensemble is confident, formalized as $s(\mathbf{x}_i, \mathcal{H}) > \theta$, where

$$s(\mathbf{x}_i, \mathcal{H}) = \frac{1}{|\mathcal{H}|} \left| \sum_{h_k \in \mathcal{H}} h_k(\mathbf{x}_i) \right|. \tag{4.6}$$

In practice, we employ a fixed threshold $\theta = 0.75$ for the experiments reported in this paper. One would expect label prediction to occur more frequently and with greater accuracy when η is low; this is confirmed in our experiments below.

Additionally, after requesting T=50 labels from the oracle, we can retrain our ensemble classifier set $\mathcal{H}$ from the current training set $\mathcal{T}$ using a fresh round of bootstrap sampling. We elect not to retrain at every iteration simply for computational reasons. Retraining the ensemble set is another major point of difference between our proposed method and bootstrap IWAL, which uses the same set throughout active learning. As our ensemble becomes a more representative sample of $\mathcal{X}$, we can dynamically adjust $p_{\min}$ according to a schedule to reflect our increased confidence. However, in the experiments below, we employ a fixed $p_{\min} = 0.1$ to make it easier for others to duplicate our results.

The final classifier is obtained by training a classifier using all of the labeled data (from both oracle and predictions) in $\mathcal{L}$. In practice, this would be done once at the termination of

active learning. However, in our experiments shown below, to show the incremental progress of the different algorithms, we train classifiers throughout the active learning process on the weighted set of labeled data to date and show performance on a held-out test set as a function of requested labels.

Experimental Results

We have conducted a comprehensive series of experiments using several standard datasets. Figure 4.9, 4.10 and 4.11 presents results on three datasets: *mushroom* and *spambase* (both from the UCI[1] Machine Learning Repository, and *MNIST*[2] digits.

To enable repeatability and a controlled evaluation testbed, we corrupt the data using different levels of i.i.d. annotation noise; results shown here employ $\eta = \{0.1, 0.2, 0.3\}$; the knowledge of η is withheld from the algorithm.

Figure 4.9, 4.10 and 4.11 summarizes our results. We compare four algorithms: (1) Tong & Koller [80], denoted as "aggressive"; (2) IWAL [10]; (3) Proposed method.

Each curve is the average computed over 50 independent runs.

[1] http://archive.ics.uci.edu/ml/

[2] http://yann.lecun.com/exdb/mnist/

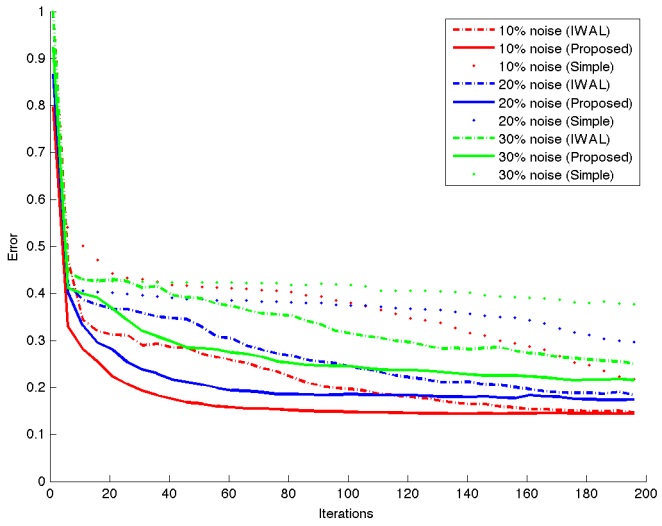


Figure 4.9: Classification error on Spambase.

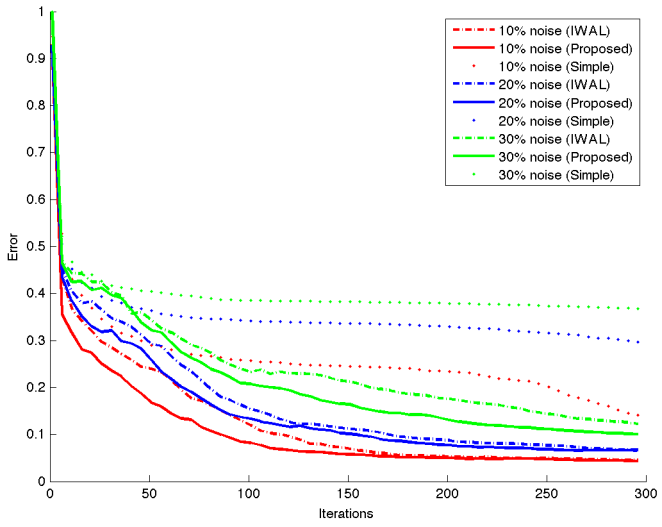


Figure 4.10: Classification error on Spambase.

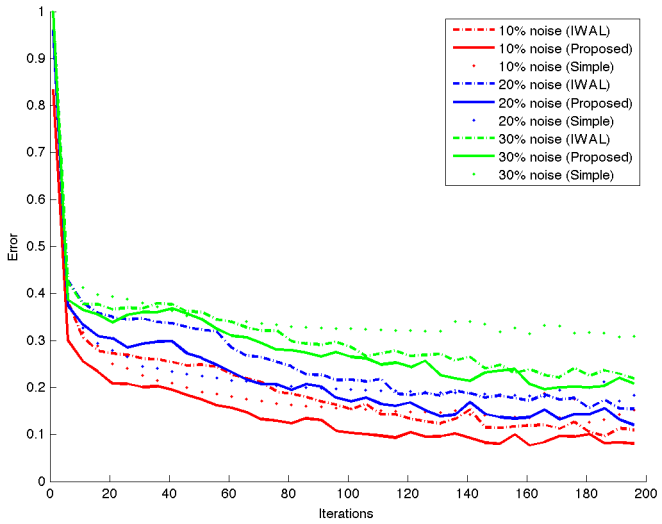


Figure 4.11: Classification error on MNIST.

To ensure a fair comparison, aggressive, IWAL and proposed were all initialized with exactly the same set of randomly-selected data ($\mathcal{T}$) in a given run. The graphs plot accuracy on the test set for a classifier trained only using the selected data to date, against the number of requested labels. We discuss the results on each dataset in greater detail below.

Figure 4.9 shows classification results on the *spambase* dataset, which consists of 4601 instances with 56 attributes, and 7% internal misclassification error. We use PCA to reduce the input dimensions from 56 to 20.

Figure 4.10 presents results on *mushroom*, which contains 8124 instances with 23 attributes.

Finally, Figure 4.11 shows results on *MNIST*, where we use the digits '3' and '5' for testing. The dataset has 1902 instances with 784 attributes. We use PCA to reduce dimensionality to 20.

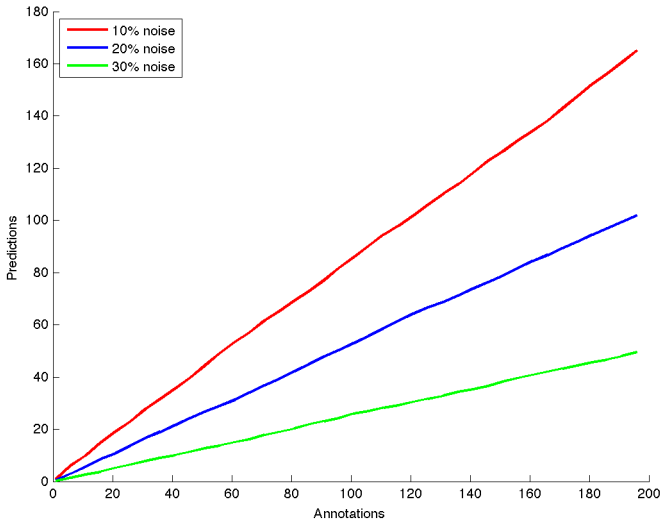


Figure 4.12: Prediction number on Spambase.

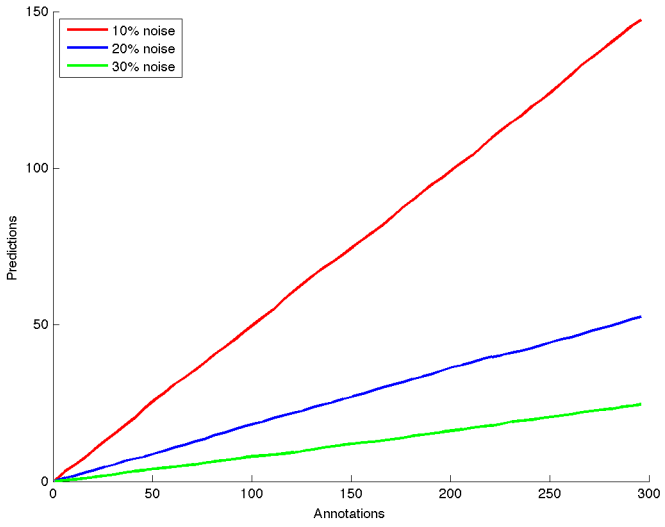


Figure 4.13: Prediction number on Mushroom.

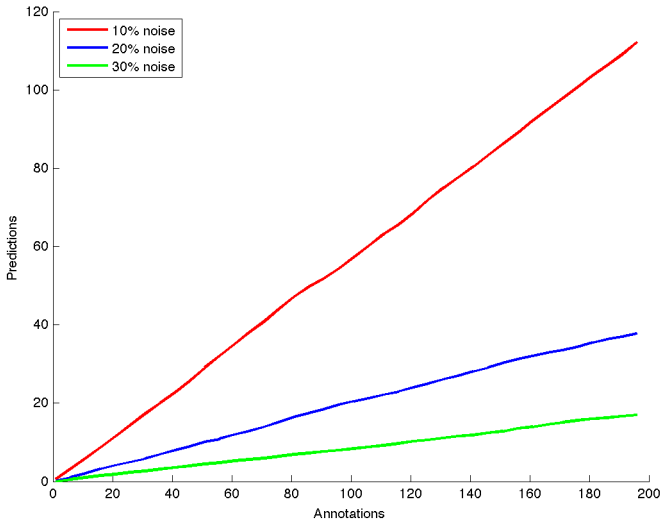


Figure 4.14: Prediction number on MNIST.

In all cases, we seed the active learning with a pool of 200 randomly-selected instances that are labeled with the same noise rate, from which K=41 bootstrap samples, each of $|\mathcal{T}|$=40 are selected.

We see that in all cases, the proposed method (solid line) outperforms both aggressive and IWAL algorithms at the given noise levels. We observe that aggressive active learning typically converges to a higher error rate, confirming our belief that such techniques are noise-seeking and can focus on only one portion of the dataset. As expected, we see that the asymptotic behavior of our method is similar to that of IWAL, but the label prediction enables the proposed method to perform better with fewer requested labels.

CHAPTER 5: FEATURE SELECTION AND ACTIVITY RECOGNITION

Conditional Random Fields

A number of supervised learning frameworks have been used to recognize low-level human activities from repetitive patterns of motion. In addition to support vector machines, we have experimented with the use of conditional random fields (CRF). Conditional random fields (CRFs) are undirected graphical models $G = (V, E)$ that represent the conditional probabilities of a label sequence $\mathbf{y} = \{y_1, ..., y_T\}$, when given the observation sequence $\mathbf{x} = \{x_1, ..., x_T\}$. When conditioned on $\mathbf{x}$, the label y_i holds the Markov property

$$p(y_i|\mathbf{y}_{G\setminus i}, \mathbf{x}) = p(y_i|\mathbf{y}_{\mathcal{N}_i}, \mathbf{x}) \tag{5.1}$$

where $\mathbf{y}_{\mathcal{N}_i}$ represents all the neighborhoods that connected to y_i. The Hammersley-Clifford Theorem tells us that this equation is equivalent to $p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(\mathbf{y},\mathbf{x})} \psi_c(\mathbf{y}_c, \mathbf{x}_c)$ if the graphical models G obey the Markov assumption, where $\psi_c(\mathbf{y}_c, \mathbf{x}_c)$ is the nonnegative potential functions of clique c and $Z(\mathbf{x})$ is a normalization constant. For our problem, the labels correspond to activities, such as "chop vegetables", while the observations consist of a set of features computed over the raw data.

Linear chain CRFs are graphical models defined with a linear chain structure such that the current state y_i only relates to the previous state y_{i-1} and the observation $\mathbf{x}_c$. Linear chain CRFs require no assumptions of independence among observations, thus $\mathbf{x}_c$ can be any part

of the observation sequence $\mathbf{x}$ and the conditional probability $p(\mathbf{y}|\mathbf{x})$ can be written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{T} \psi_i(y_{i-1}, y_i, \mathbf{x}, i) \tag{5.2}$$

where $Z(\mathbf{x}) = \sum_y \prod_{i=1}^{T} \psi_i(y_{i-1}, y_i, \mathbf{x}, i)$ and $\mathbf{x}$ represents observations over the whole sequence.

Since CRFs are log-linear models, the potential function can be written as the linear sum of feature functions as $\psi_i(y_{i-1}, y_i, \mathbf{x}, i) = \exp(\sum_j w_j f_j(y_{i-1}, y_i, \mathbf{x}, i))$, where w_j represents the weight corresponding feature $f_j(y_{i-1}, y_i, \mathbf{x}, i)$. The conditional probability can then be calculated as

$$p(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{x})} \exp(\sum_{i=1}^{T} \sum_j w_j f_j(y_{i-1}, y_i, \mathbf{x}, i)) \tag{5.3}$$

where $Z(\mathbf{x}) = \sum_y \exp(\sum_{i=1}^{T} \sum_j w_j f_j(y_{i-1}, y_i, \mathbf{x}, i))$.

Assuming a fully labeled dataset with pairs of training samples $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), ..., (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\}$ the CRF parameter vector $\mathbf{w} = \{w_1, ..., w_M\}$ can be obtained by optimizing the sum of conditional log-likelihood $L(\mathbf{Y}|\mathbf{X}; \mathbf{w})$.

$$L(\mathbf{Y}|\mathbf{X}; \mathbf{w}) = \sum_k \log p(\mathbf{y}^{(k)}|\mathbf{x}^{(k)}; \mathbf{w}) \tag{5.4}$$

$$= \sum_k (\sum_{i=1}^{T} \sum_{f_j \in \mathcal{S}} w_j f_j(y_{i-1}^{(k)}, y_i^{(k)}, \mathbf{x}^{(k)}, i) - \log Z(\mathbf{x}^{(k)})) \tag{5.5}$$

The first derivative of the log-likelihood is:

$$\frac{\partial L(\mathbf{Y}|\mathbf{X};\mathbf{w})}{\partial w_j} = \sum_k (\sum_{i=1}^{T} \sum_{f_j \in \mathcal{S}} f_j(y_{i-1}^{(k)}, y_i^{(k)}, \mathbf{x}^{(k)}, i) \tag{5.6}$$

$$- \sum_y \sum_{i=1}^{T} \sum_{f_j \in \mathcal{S}} p(\mathbf{y}|\mathbf{x}^{(k)};\mathbf{w}) f_j(y_{i-1}, y_i, \mathbf{x}^{(k)}, i)) \tag{5.7}$$

Since the CRFs log likelihood is convex with respect to the weight vector w, standard optimization methods such as conjugate gradient and limited memory BFGS [54] can be used to discover the weights. In this work, we learn both the set of features $f_j(.)$ and their associated weight parameters w_j, for multi-class classification.

Motif Discovery and Feature Selection

When classifying the IMU data, we focused on using motif discovery and feature selection to improve classification accuracy. Also, without feature selection, the time required to train the CRF on the entire motif-based feature set can become prohibitively large.

Our training approach can be described as follows. First, we discover motifs in the data collection, essentially learning a mapping to convert a given local window of IMU data from a multi-dimensional time series signal to a sequence of discrete symbols. Second, we define a series of low-level binary-valued features over motifs and pairs of motifs. From a large pool of candidate features, we select those that are most informative using an iterative approach, described below. Next, we learn a Conditional Random Field whose observations are defined over the set of selected features and whose output is over the set of action labels.

The incremental feature selection and CRF training are iterated until the training set error converges. The final CRF is then evaluated on the test set. Each of these stages is detailed below.

The first step in motif discovery is to discretize the continuous IMU signal into symbolic subsequences. Figure 5.1(b) illustrates this process. The raw data $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ (black line) is transformed into a piecewise continuous representation $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ (green line) using the Piecewise Aggregate Approximation (PAA) algorithm, which computes an average of the signal over a short window: $s_i = \frac{m}{n} \sum_{j=\frac{n}{m}(i-1)+1}^{\frac{n}{m}} t_j$. This is then mapped to a symbolic representation using "break points" (red lines) that correspond to bins; these are generated so as to separate the normalized subsequences (under a Gaussian assumption) into equalized regions. Thus, a continuous 1-D signal can be represented as a sequence of discrete symbols.

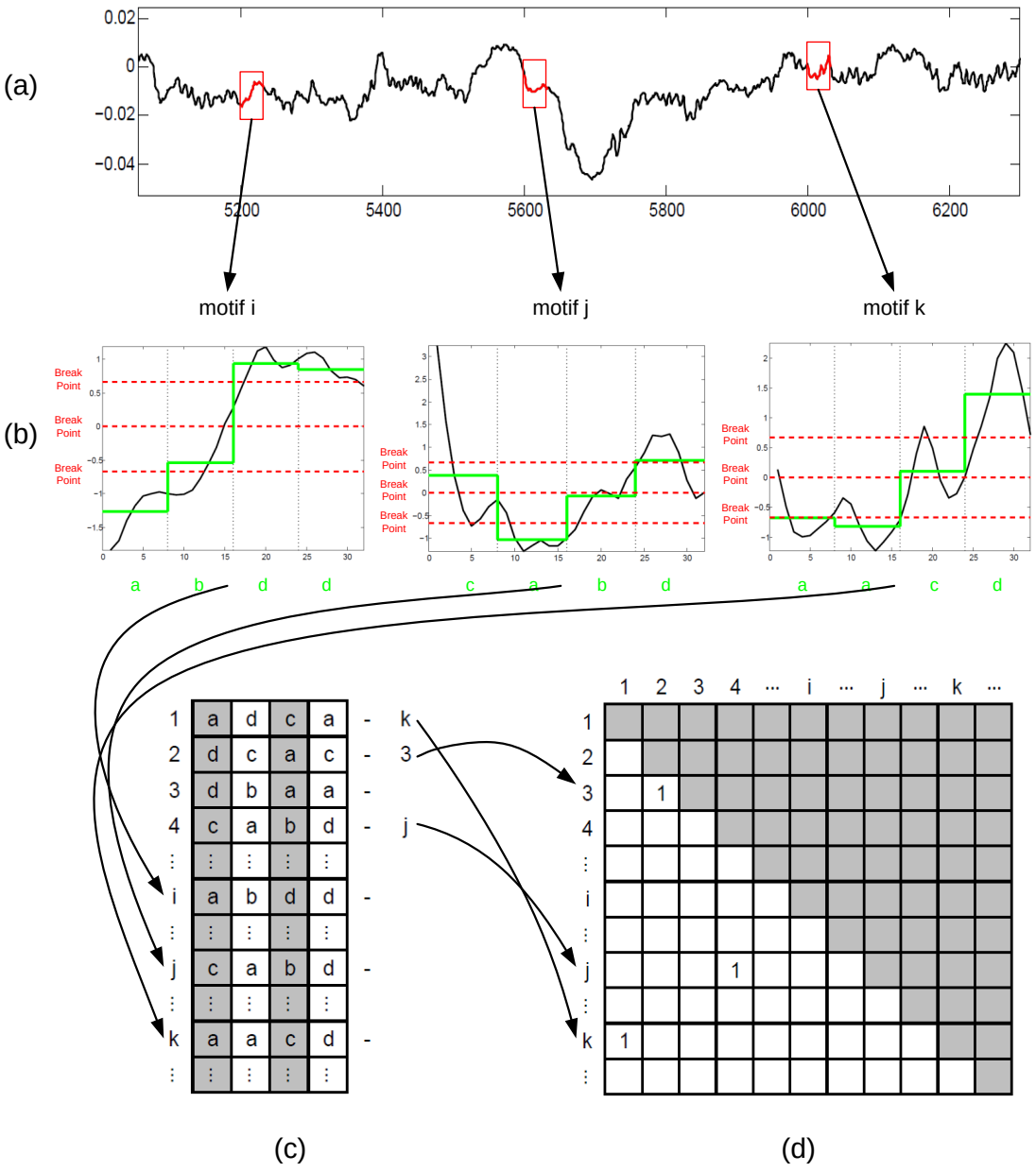


Figure 5.1: Visualization of motif discovery (illustrated in 1D). (a) Raw data; (b) Discretization using PAA and conversion to symbolic sequence; (c) Random projections (shadowed columns) used for projection; (d) Recorded collisions in the collision matrix.

Motif Matching

To compare symbolic sequences in a manner that is both computationally efficient and robust to signal noise (i.e., corresponding to symbol mismatch), we propose a matching metric that relies on random projections.

In our problem, we designate two motifs as matching if they agree on k symbol positions. Figure 5.1(c) gives an example with $k = 2$, where the randomly-selected columns 1 and 3 are used to compare motifs. In this example, motifs 1 and k, 2 and 3, and 4 and j all match. These matches can be summarized by incrementing entries in a symmetric match table (where rows and columns correspond to motifs), as shown in Figure 5.1(d). Accumulating counts in this manner using several different random projections can enable us to efficiently match long motifs in a manner that is robust to occasional symbol errors.

Feature Selection for Conditional Random Fields

A key aspect of the proposed method is that we automatically select informative features from a large pool of candidates defined over motifs. As validated in our experiments, this leads to a significant improvement over CRFs trained directly on the raw data. We define three types of binary features over our motifs to form a pool of over 4000 features, from which our goal is to select a small subset that can maximize the conditional log likelihood, without overfitting to the training data.

We adopt the following greedy forward selection procedure, where each feature in the pool is considered in turn and the best feature at each iteration is added. Specifically, we initialize the candidate set $\mathcal{C}$ with the pool of available features and the subset of selected features $\mathcal{S}$ to be empty. At each iteration, we evaluate every potential feature $f_\lambda \in \mathcal{C}$ individually by

considering a CRF with $\mathcal{S} \cup f_\lambda$ and select the feature that maximizes the gain of log-likelihood $G(\lambda, f_\lambda) = L(\mathbf{Y}|\mathbf{X}; \mathbf{w}, \lambda) - L(\mathbf{Y}|\mathbf{X}; \mathbf{w})$. This best feature is added to $\mathcal{S}$ and removed from $\mathcal{C}$. We continue selecting features until the CRF error (computed on a hold out set) ceases to improve.

Unfortunately, a straightforward implementation of this procedure is extremely time consuming since it requires an expensive computation for every potential feature at each iteration. In particular, the normalization term of the CRF, $Z(\mathbf{x}^{(k)})$ must be calculated every time the gain $G(\lambda, f_\lambda)$ is evaluated. Motivated by work on kernel CRFs [47] and image segmentation [49], we employ a first-order approximation method. Consider that the log likelihood function $L(\mathbf{y}|\mathbf{x}; \mathbf{w}, \lambda)$ could be approximated by its first-order Taylor expansion:

$$L(\mathbf{Y}|\mathbf{X}; \mathbf{w}, \lambda) = L(\mathbf{Y}|\mathbf{X}; \mathbf{w}) + \lambda \frac{\partial L(\mathbf{Y}|\mathbf{X}; \mathbf{w}, \lambda)}{\partial \lambda} \mid_{\lambda=0} .$$

In this equation, the second term can be expressed as:

$$\frac{\partial L(\mathbf{Y}|\mathbf{X}; \mathbf{w}, \lambda)}{\partial \lambda} \mid_{\lambda=0} = E[f_\lambda, \lambda] - \tilde{E}[f_\lambda, \lambda],$$

where $\tilde{E}[f_\lambda, \lambda] = \sum_k \sum_{i=1}^{T} f_\lambda(y_{i-1}^{(k)}, y_i^{(k)}, \mathbf{x}^{(k)}, i)$ represents the empirical expectation and $E[f_\lambda, \lambda] = \sum_k \sum_{i=1}^{T} \sum_{y'} p(y'|\mathbf{x}^{(k)}, \mathbf{w}, \lambda) f_\lambda(y_{i-1}, y_i, \mathbf{x}^{(k)}, i)$ is the model expectation. Employing this approximation achieves significant computational benefits in practice.

Our proposed method is agnostic to the choice of features. Motivated by Vail et al.'s work on activity recognition for robots [82], we employ the following three types of features. In our case, these are computed over motif patterns rather than the raw data, and all are two-valued features. The function $\delta(.)$ is 1 if its argument is true and 0 otherwise.

1. **Identification features:** $f(y_{i-1}, y_i, \mathbf{X}, i) = \delta(y_i = motif_k)$. These features constitute

the basic units of actions and are computed at a node level. They verify that the action label at time t corresponds to motif k.

2. **Transition features:** $f(y_{i-1}, y_i, \mathbf{X}, i) = \delta(y_{i-1} = motif_j)\delta(y_i = motif_k)$. These features capture the first-order Markov transition properties between adjacent motifs. The transitions may appear both between different actions or within the same action and are designed to overcome the lack of synchronization between motifs computed over different dimensions of a multi-dimensional signal.

3. **Observation features:** $f(y_{i-1}, y_i, \mathbf{X}, i) = \delta(y_i = motif_k)g_i(motif_k)$. In this definition, $g_t(motif_k)$ represents the magnitude average of motif k. These features make the magnitude information for a motif available to the CRF; that information is lost in a typical symbolic motif representation. Observation features recover it by returning the mean magnitude of the motif.

Experimental Results

Our experiments employ the publicly-available CMU Multi-Modal Activity Dataset (CMU-MMAC) [25]. Here we describe our classification results on the inertial measurement unit (IMU) portion of the dataset, which was collected by five MicroStrain 3DM-GX1 sensors attached to the subject's waist, wrists and ankles. The IMU is a cost-effective and unobtrusive sensor (see Figure 5.2) but generates noisier data than the richer motion capture data, making the classification problem substantially more difficult.

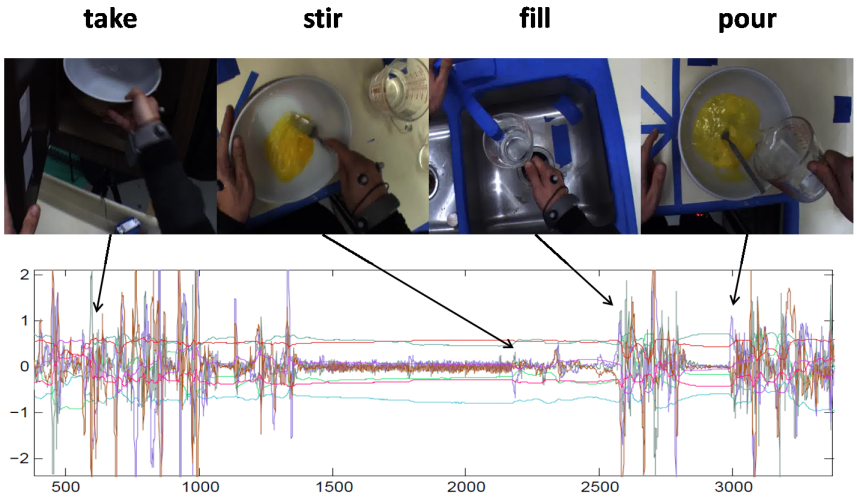


Figure 5.2: CMU-MMAC IMU dataset: example actions and corresponding data.

The dataset consists of unscripted recipes performed by several subjects in a kitchen. Thus, there is considerable variability in the manner in which the task is performed. The data corresponding to a given recipe consists of approximately 10,000 samples collected at 30 Hz over a period of about 6 minutes. Each frame includes 3-axis absolute orientation, angular velocity and instantaneous acceleration from each of the five sensors, leading to a 45-dimensional feature vector. Our experiments focus on the recipes that have been manually annotated into a series of actions (e.g., "open fridge" or 'stir brownie mix"); these correspond to the "make brownies" task. We downsample the raw data by a factor of 10.

Table **??** summarizes the classification results for 14 actions. We compare the proposed CRF method against several baselines: CRF on raw features, HMM with various parameters and kNN. Clearly, the proposed method outperforms all of these approaches on the challenging test set. Clearly, the feature set results in significant benefits in terms of improved accuracy on the test set. We compared the overall performance of our method against a set of stan-

dard classifiers (K-NN, HMMs, and Bayesian networks). Our method outperforms the best performing HMM (25.6% accuracy) and the best overall alternative (k-NN, K=13, 38.22% accuracy).

Table 5.2 shows a comparison between the CRF with intelligent feature selection and the SVMs trained with the raw features. We also evaluated combining the SVM with temporal filtering to penalize frequent class label changes. The SVM performance is comparable to the best alternative method (k-NN) but does not do as well as the CRF plus feature selection, even with the temporal filtering.

Table 5.2: Classification accuracies for our proposed approach (CRF with feature selection) against SVM and SVM plus temporal filtering. Both SVM approaches use the raw IMU features.

Seq No.	CRFs	SVMs	SVM plus filter
1	49.48%	50.21%	51.66%
2	33.86%	39.07%	38.92%
3	37.97%	44.74%	50.39%
4	53.66%	37.45%	33.47%
5	32.70%	32.80%	33.52%
6	51.78%	48.71%	48.29%
7	37.75%	8.29%	6.90%
8	43.00%	39.13%	37.38%
9	45.93%	11.13%	13.15%
10	44.90%	49.87%	50.56%
11	48.12%	32.72%	36.83%
12	46.80%	49.35%	48.58%
13	47.39%	18.54%	18.17%
14	45.17%	35.74%	34.99%
15	38.10%	50.05%	49.03%
16	41.92%	33.05%	32.84%
Ave	**44.19%**	**36.23%**	**36.32%**

Figure 5.3 illustrates the segmentation results of different methods on data sequence 1. For

the *stir* activity which appears most frequently in all 14 activities, all approaches exhibit good performance compared with the ground truth (red). Moreover, CRF performs better than other two methods on activities *none* and *pour* which account for a high percentage of the total frames in testing. However, all approaches continue to perform relatively poorly in several activities that barely appeared in the testing sequence.

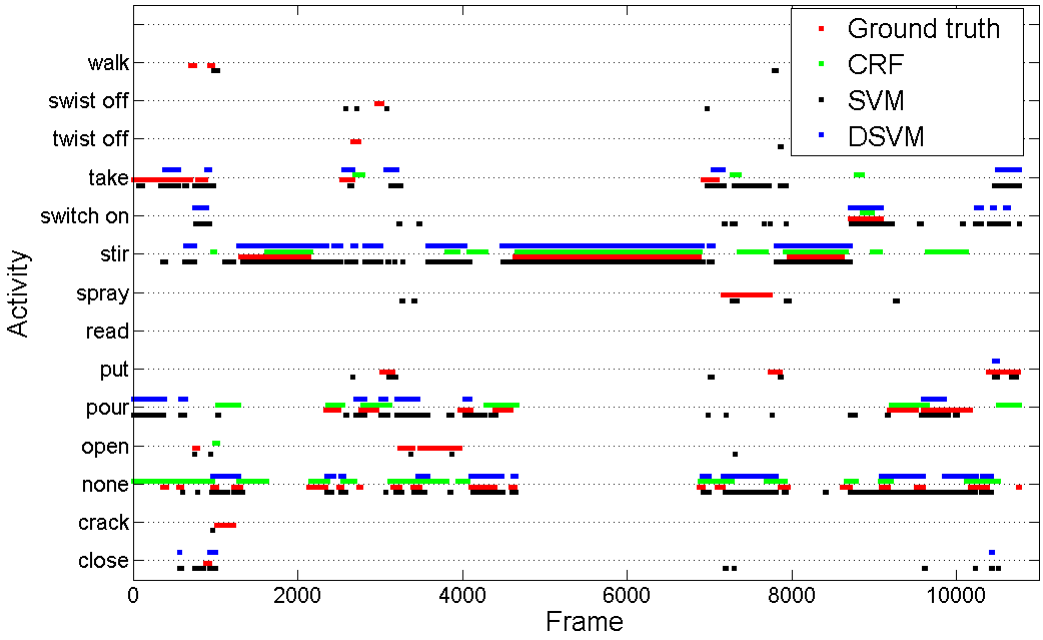


Figure 5.3: Segmentation results on CMU-MMAC dataset

Segmentation results for testing sequence 1 with all three approaches. The x axis shows the 14 classes and the y axis the frame number of the sequence.

CHAPTER 6: JOINTLY ESTIMATING WORKER PERFORMANCE AND ANNOTATION RELIABILITY WITH ACTIVE LEARNING

Services such as Amazon's Mechanical Turk (MTurk) have made it possible for researchers to acquire sufficient quantities of labels, enabling the development of a variety of applications driven by supervised learning models. However, employing crowdsourcing to label large quantities of data remains challenging for two important reasons: limited annotation budget and label noise. First, although the unit cost of obtaining each annotation is low, the overall cost grows quickly since it is proportional to the number of requested labels, which can number in the millions. This has stimulated the use of approaches such as active learning [66] that aim to minimize the amount of data required to learn a high-quality classifier. Second, the quality of crowdsourced annotations has been found to be poor [67], with causes ranging from workers who overstate their qualifications, lack of motivation among labelers, haste and deliberate vandalism. Unfortunately, the majority of popular active learning algorithms, while robust to noise in the input features, can be very sensitive to label noise, necessitating the development of approaches specifically designed for noisy annotations. In this chapter, I focus on the causal factors behind noisy crowdsourced annotations, worker quality and task difficulty.

A common simplifying assumption is that of *identicality*: all labelers provide annotations with the same accuracy and all annotation tasks pose the same difficulty to all workers. Under this assumption, a good way to combat annotation noise is to request multiple labels for each selected task and then to apply majority voting. The simplest approach of requesting the same number of labels for each instance is not usually the most cost-effective since label

redundancy increases the overall cost by a multiplicative factor of at least three.

Better results can be obtained by applying weighted voting which assigns different weights to labelers based on their previous performance [67, 95, 98]. In this chapter, I use the Generative model of Labels, Abilities, and Difficulties (GLAD) [91] in which labeler expertise and the task difficulty are simultaneously estimated using EM (Expectation-Maximization) to learn the parameters of a probabilistic graphical model which represents the relationship between labelers, tasks and annotation predictions. Rather than using previous performance to assign weights, the estimated labeler expertise is used to allocate weights to annotator votes. However, the GLAD model proposed by Whitehill et al. requires all workers to provide labels to all annotation tasks. Requesting complete labels is neither necessary nor practical in real world. Many annotation tasks are simple enough that most workers can provide correct labels. On the other hand, not all workers are willing to provide labels to all tasks. This chapter focuses on the problem where, if there is a limited budget and it is impossible for us to collect full labels for all tasks; the proposed approach identifies tasks are more valuable for the GLAD model to have labeled and which workers should provide the labels.

Theoretically the most straightforward and effective strategy for active learning is to select samples that offer the greatest reductions to the risk function. Thus, "aggressive" criteria such as least confidence, smallest margin, or maximum entropy can enable active learning to obtain high accuracy using a relatively small number of labels to set the decision boundary. Unfortunately, the existence of label noise can trigger failures in aggressive active learning methods because even a single incorrect label can cause the algorithm to eliminate the wrong set of hypotheses, thus focusing the search on a poor region of the version space. In contrast, the proposed combination of the probabilistic graphical model and active learning, avoids explicitly eliminating any set of hypotheses inconsistent with the label provided by

annotators.

Specifically, this chapter makes two contributions:

1. I propose a new sampling strategy which iteratively selects the *combination* of worker and task which offers the greatest risk reduction between the current labeling risk and the expected posterior risk. The strategy aims to focus on sampling reliable labelers and uncertain tasks to train the Bayesian network.

2. I present comprehensive evaluations on both simulation and real world datasets that show not only the strength of the proposed approach in significantly reducing the quantity of labels required for training the model, but also the scalability of the approach at solving practical crowdsourcing tasks which suffer large amounts of annotation noise.

Moreover, beside proposing the new active learning strategy on task selection, I also investigate the performance of different worker selection strategies. The most straightforward way is to pick the available worker with the highest expertise. This aggressive strategy assumes the current estimation of the worker expertise is correct and ignores to improve the evaluation of other workers. In this chapter I also test and discuss the performance of other possible strategies, such as ϵ-greedy selection and weighted selection.

Method

In this section, I describe the proposed active learning approach for jointly estimating worker performance and annotation reliability. The first part of the section defines the probabilistic graphical model for estimating the expertise of labelers and the difficulty of annotation tasks before describing how EM is used to estimate the model parameters. The second subsection

introduces the proposed active learning approach for sampling workers and tasks using an entropy-based risk function.

Probabilistic Graphical Model

My work utilizes the generative model proposed by [91]. The structure of the graphical model is shown in Figure 6.1. The same model is used to estimate the expertise of workers, the difficulty of annotation tasks, and the true labels. The expertise of worker i is defined as $\alpha_i \in (-\infty, +\infty)$, which corresponds to the worker's level of annotation accuracy. As α_i approaches $+\infty$, worker i becomes an increasingly capable oracle who almost always gives the correct label and as α_i approaches $-\infty$ the worker almost always provides the wrong label. $\alpha_i = 0$ means the worker has no capability to distinguish the difference between the two classes and just randomly guesses a label. The difficulty of task j is parameterized by β_j where $\frac{1}{\beta_j} \in (0, +\infty)$. For easy tasks, $\frac{1}{\beta_j}$ approaches zero, and it is assumed that practically every worker can give the correct label for the task. As $\frac{1}{\beta_j}$ approaches $+\infty$, the task becomes so difficult that almost no one is able to provide the right answer. In this chapter, I only consider binary classification problems which assume that both the true label Z_j and the annotation l_{ij} provided by the worker i are binary labels. $l_{ij} \in \{-1, +1\}$ is defined as the label of task j provided by annotator i. This is the only observable variable in the graphical model. Since in most crowdsourcing platforms, it is unlikely that the same labelers will be responsible for annotating all tasks in the dataset, this model also works well when the observation is incomplete. The true label z_j of task j is the variable that I am going to estimate to evaluate the labeling performance of the model.

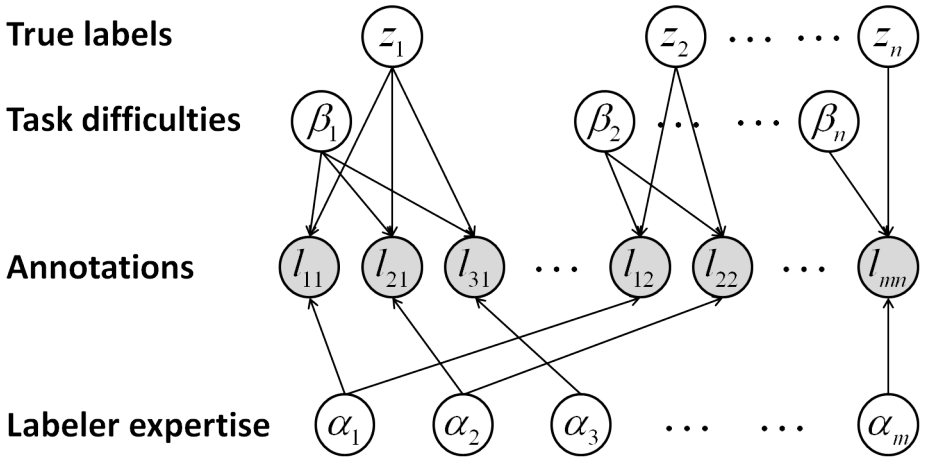


Figure 6.1: The structure of the Bayesian networks used to estimate the true labels, worker expertise, and annotation task difficulty.

Given the definitions above, the probability of annotator i providing the correct label for task j is defined in Equation 6.1. For a more skilled labeler with a larger α, or an easier task with a larger β, the probability of providing the correct label should be larger. However, if the task is too difficult ($\frac{1}{\beta} \to +\infty$) or the labeler has no background in performing the task ($\alpha = 0$), the labeler can only give a random guess ($p = 0.5$) about the task label.

$$P(l_{ij} = z_j | \alpha_i, \beta_j) = \frac{1}{1 + e^{-\alpha_i \beta_j}} \tag{6.1}$$

The set of parameters of the graphical model $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_i, ..., \alpha_m\}$ and $\boldsymbol{\beta} = \{\beta_1, ..., \beta_j, ..., \beta_n\}$ are represented as $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$. $\mathbf{L}$ and $\mathbf{z}$ are defined as the set of labels provided by the workers (observed) and the true labels (not known).

EM is an iterative algorithm for maximum likelihood estimation in a graphical model with

incomplete data. In this annotation task, the learning procedure starts with the set of unreliable annotations $\mathbf{L}$. The EM algorithm iteratively estimates the unknown parameters $\boldsymbol{\theta}$ with the current observations $\mathbf{L}$ and then updates the belief of the true labels $\mathbf{z}$ using these estimated parameters. The description of the EM algorithm as applied to this model is given below.

E-Step: At the expectation step, the parameters $\boldsymbol{\theta}$ estimated in the maximization step are fixed. The posterior probability of $z_j \in \{0, 1\}$ given $\boldsymbol{\theta}$ is computed as:

$$p(z_i|\mathbf{L}, \boldsymbol{\theta}) \propto p(z_j) \prod_i p(l_{ij}, \alpha_i, \beta_j) \tag{6.2}$$

M-Step: Given the posterior probability of z_j, the maximization step uses the cost function $Q(\boldsymbol{\theta})$ to estimate a locally optimal solution of parameters $\boldsymbol{\theta}$:

$$Q(\boldsymbol{\theta}) = \sum_j E[\ln p(z_j)] + \sum_{ij} E[\ln p(l_{ij}|z_j, \alpha_i, \beta_j)] \tag{6.3}$$

The aim of the EM algorithm is to return the parameters $\boldsymbol{\theta}^*$ which maximize the function $Q(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}) \tag{6.4}$$

$\boldsymbol{\theta}^*$ represents the optimal estimate of the expertise of labelers and the difficulty of tasks with the current set of annotations.

Query by Uncertainty

The essential question in active learning is which criterion performs the best at identifying the most valuable unlabeled sample under the present parameters. In this chapter, I employ the entropy of the label distribution to represent the risk of a particular label assignment. I assume the class of labels is defined as $\mathcal{C} = \{-1, +1\}$, and the risk function is

$$Risk(z_j) = -\sum_{|\mathcal{C}|} \frac{p(z_j|\mathbf{L}, \boldsymbol{\theta}^*)}{|\mathcal{C}|} \log \frac{p(z_j|\mathbf{L}, \boldsymbol{\theta}^*)}{|\mathcal{C}|} \tag{6.5}$$

where z_j is the true label and $p(z_j|\mathbf{L}, \boldsymbol{\theta}^*)$ represents the probability of annotating task j with label z_j under the present parameters $\boldsymbol{\theta}^*$. This risk function evaluates the risk of assigning sample j label z_j. The proposed active learning algorithm preferentially selects the sample that maximizes this risk function to be annotated.

The algorithm for using active learning within the Bayesian network is shown in Algorithm 3. The algorithm begins with the data to be annotated and a partially completed label matrix $\mathbf{L}$. The set of parameters $\boldsymbol{\theta}$ is initialized at this stage, and the labeling budget for the crowdsourcing is allocated. The goal of this algorithm is to learn the set of optimal parameters $\boldsymbol{\theta}^*$ with a given label budget. At each iteration, the proposed algorithm selects the sample that maximizes the risk function defined in Equation 6.5 with the parameters $\boldsymbol{\theta}^*$ estimated using the EM algorithm. The sample is then annotated by requesting label l_{ij} from the labeler i with the maximum current α_i who had not previously contributed a label to that task.

Algorithm 3 Active Learning Algorithm - Query by Uncertainty

Require:

Input:

A set of data along with a matrix of partial labels $\mathbf{L}$

The initial set of parameters $\boldsymbol{\theta}$

B: labeling budget.

Output:

$\boldsymbol{\theta}^* = \{\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*\}$: The set of parameters representing the expertise of labelers and the difficulty of tasks.

1: **while** $B > 0$ **do**

2: Use the EM algorithm to update the parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ using Equations 6.2 and 6.3;

3: Find parameters $\boldsymbol{\theta}^*$ that maximize the function $Q(\boldsymbol{\theta})$ using Equation 6.4;

4: Calculate the risk of assigning label z_j to task j with Equation 6.5;

5: Query the label l_{ij} by assigning task j with the maximum risk to the worker i with the highest α_i and $l_{ij} \neq 0$;

6: $B \leftarrow B - 1$;

7: **end while**

8: Return the optimal parameters $\boldsymbol{\theta}^*$.

Query by Committee

Although query-by-uncertainty is a straightforward approach to identify the most uncertain sample to be labeled, its greedy inherence ignores the prior distribution of parameters as a important information in selecting the unlabeled sample. In this section I propose a query-by-committee strategy which generates a committee of parameter variances to approximate the classifier parameter distribution. The committee is derived from the optimal parameter estimated by running the EM algorithm on the labeled dataset. The active learning strategy

tends to investigate the unlabeled dataset to select the sample which will significantly reduce the learning error if its label is required. This criterion is implemented with the query-by-committee which selects the sample that the committee strongly disagree with its label. Algorithm 4 describes the implementation details of the committee method which mainly focuses on solving two issues: how to generate the committee and how to evaluate unlabeled data.

Committee - Bootstrap

Generating the appropriate committee to approximate the classifier parameter distribution is the essential task of a query-by-committee active learning strategy. In my work I apply the *bagging* [12], also known as bootstrap aggregating, to derive the committee from the classifier parameter. The fundamental idea of bagging is to generate multiple classifiers from bootstrap replicates of the labeled dataset and using these to get an aggregated classifier.

Assuming the labeled dataset is $\mathbf{T}$, the unlabeled dataset is $\mathbf{U}$ and the classifier parameter of Bayesian Network is $\boldsymbol{\theta}$, the algorithm generates K sets of labels $\{\mathbf{T}^1, \mathbf{T}^2, ..., \mathbf{T}^K\}$ by randomly resampling from $\mathbf{L}$ with replacement. The bootstrap resampling requests the set $\mathbf{T}^k$ has the same number of elements as the original set $\mathbf{T}$. For each set $\mathbf{T}^k$, the EM algorithm is applied to learn the classifier parameter $\boldsymbol{\theta}^k$ which maximizes the function $Q(\boldsymbol{\theta})$ in Equation 6.3.

Criteria - Entropy Vote

After getting K committee members from bagging, I apply two risk funcitons, *Entropy Vote* and *KL-divergence to the mean*, to evaluate the committee disagreement. Assume there are $C = \{1, 2, ..., C\}$ optional classes in the annotation task and $K = 1, 2, ..., K$ members in the

committee generated from bagging, each committee member comes up with a probability of giving the task a label c which is represented as $p_k(z_j = c|\mathbf{T}^k, \boldsymbol{\theta}^k)$.

The most straightforward disagreement criterion is *Entropy Vote* which counts the votes for different classes and find the unlabeled sample with the maximum entropy. In *Entropy Vote*, the criterion only considers which class the committee member votes rather than the probability of the committee member to give this vote. So the indicator function has been applied to the probability as $V_k(z_j = c) = 1$ if $p_k(z_j = c|\mathbf{T}^k, \boldsymbol{\theta}^k) > \frac{1}{K}$, otherwise $V_k(z_j = c) = 0$. Therefore the number of vote for the class c providing by the committee could be easily obtained by $V(z_j) = \sum_k V_k(z_j = c)$.

In this document, I consider the binary classification problem with $\mathcal{C} = \{-1, +1\}$, and the risk function with *Entropy Vote* is represented as

$$Risk(z_j) = -\sum_{|\mathcal{C}|} \frac{V(z_j)}{K} \log \frac{V(z_j)}{K} \tag{6.6}$$

The risk function above selects the unlabeled task with maximum entropy of votes. The most suspicious task is the one that the committee has equal number of vote on each class. Although this strategy closely follow the concept of "committee disagreement", the probability of giving the task a label, with respect to the confidence of each committee member, is ignored by the voting criterion. Therefore, this document also considers an alternative task selection criterion, *KL-divergence to the mean*, to evaluate the informativeness of unlabeled tasks.

Criteria - KL-Divergence

In information theory, the Kullback-Leibler divergence (KL-divergence) [44] is a non-symmetric measure of the difference between two probability distributions P and Q. Specifically, the KL-divergence of Q from P written in Equation 6.7 represents the information lost in using distribution Q to approximate distribution P. Generally P represents the true distribution that Q is going to approximate. The KL-divergence measures the expected number of extra bits required to code samples for P when using the code samples of Q.

$$KL(P||Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right) \tag{6.7}$$

The committee strategy uses the KL-divergence as an alternative risk function for evaluating the variance of the committee. The target function is to calculate the information lost in using the mean of probability distributions $p_{mean}(z_j|\mathbf{T}, \boldsymbol{\theta})$ to each distribution $p_k(z_j|\mathbf{T}^k, \boldsymbol{\theta}^k)$. Compared to the Vote Entropy, the risk function shown in Equation 6.8 considers the differences of the class distribution for each committee member.

$$Risk(z_j) = \frac{1}{K} \sum_{k=1}^{K} KL(p_k(z_j|\mathbf{T}^k, \boldsymbol{\theta}^k)||p_{mean}(z_j|\mathbf{T}, \boldsymbol{\theta})) \tag{6.8}$$

$p_{mean}(z_j|\mathbf{T}, \boldsymbol{\theta}) = \frac{1}{K} \sum_k p_k(z_j|\mathbf{T}^k, \boldsymbol{\theta}^k)$ represents the mean of all K class distributions for the committee.

Algorithm

The algorithm for using the query by committee strategy within the Bayesian network is shown in Algorithm 4. The algorithm begins with a partially labeled dataset $\mathbf{L} = \{\mathbf{T}, \mathbf{U}\}$, where $\mathbf{T}$ represents the set of tasks j that has been annotated by worker i and $\mathbf{U}$ represents the set of tasks j has not been annotated by worker i. The set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ is initialized at this stage, and the labeling budget B for the crowdsourcing is allocated. The goal of this algorithm is to learn the set of optimal parameters $\boldsymbol{\theta^*}$ with a given label budget. At each iteration, the propsed algorithm resamples K sets of data from $\mathbf{T}$ to generate the bootstrap $\mathcal{T} = \{\mathbf{T}^1, \mathbf{T}^2, ..., \mathbf{T}^K\}$ with $|\mathbf{T}^k| = |\mathbf{T}|$. By running EM algorithm on each set $\mathbf{T}^k$, the committee has been built for evaluating the informative of unlabeled tasks. The proposed algorithm selects the task that maximizes the risk function defined in Equation 6.6 or 6.8 with the set of parameters $\boldsymbol{\Theta}$ estimated using the EM algorithm. The selected unlabeled task is then annotated by requesting label l_{ij} from the labeler i with the maximum current α_i who had not previously contributed a label to that task.

Algorithm 4 Active Learning Algorithm - Query by Committee

Require:

Input:

A set of data along with a matrix of partial labels $\mathbf{L} = \{\mathbf{T}, \mathbf{U}\}$, where $\mathbf{T}$ represents the set of tasks j annotated by worker i and $\mathbf{U}$ represents the set of tasks j not annotated by worker i;

The initial set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$;

B: labeling budget.

Output:

$\boldsymbol{\theta}^* = \{\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*\}$: The set of parameters representing the expertise of labelers and the difficulty of tasks.

while $B > 0$ **do**
  Use the bagging algorithm to generate K sets of labels $\mathcal{T} = \{\mathbf{T}^1, \mathbf{T}^2, ..., \mathbf{T}^K\}$;
  Apply the EM algorithm on $\mathbf{T}^k$ to compute parameters $\boldsymbol{\theta}^k = \{\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k\}$;
  Find the set of parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, ..., \boldsymbol{\theta}^K\}$ that maximize the function $Q(\boldsymbol{\theta}^k)$ using Eqn 6.4;
  while $j \in \mathcal{U}$ **do**
    Compute $p(z_j|\mathbf{L}^k, \boldsymbol{\theta})$ with the set of parameters $\times$;
    Calculate the risk function $Risk(z_j)$ by using either Entropy Vote (Eqn 6.6) or KL-divergence to the mean (Eqn 6.8);
  end while
  Identify the most informative task $j^* = argmax_j Risk(\mathbf{z})$;
  Query the label l_{ij} by assigning task j^* to the best worker i with the highest α_i and $l_{ij} \in \emptyset$;
  $B \leftarrow B - 1$;
end while
Return the optimal parameters $\boldsymbol{\theta}^*$.

Worker Selection

Selecting the task to be labeled could be easily solved by applying active learning strategies to identify the most uncertain or informative task to workers. However, how to select the right worker to annotate the task has became another issue which is out of the scope of active learning. Selecting the most "uncertain" worker may polish the estimation of the worker expertise, but simultaneously reduce the speed to improve the training accuracy, since such strategy inevitably picks workers who the system is most unfamiliar with, rather than the best workers.

The straightforward way I have used in Algorithm 4 is to ask the worker i with the highest expertise estimation α_i^* to provide the label. The experiment results shows this strategy works well. However, there exist some arguments that challenge the strategy since 1) α_i^* is only the estimation of α_i, which means picking the worker with largest α^* may ignore the real "best worker" and 2) the evaluation of other workers are ignored since they will never be selected. The goal of this section is to investigate the performance of different worker selection strategies.

Beside always sampling the best worker, I evaluate two other options: weighted selection and ϵ-greedy selection. For the weighted selection strategy, the probability of selecting worker i is proportional to the expertise α_i^*.

$$p(i) = \frac{\alpha_i^*}{\sum_i \alpha_i^*} \tag{6.9}$$

An alternative worker selection strategy is the ϵ-greedy selection algorithm, which has been used in studying the exploration-exploitation tradeoff in reinforcement learning [43]. This

algorithm selects the most possible worker i with the highest expertise α_i with probability $1-\epsilon$. and selects other workers with probability ϵ.

$$p(i) = \begin{cases} 1-\epsilon+\frac{\epsilon}{m} & i = \max \alpha_i^* \\ \frac{\epsilon}{m} & otherwise \end{cases} \tag{6.10}$$

Experiments

In the following experiments, I test the performance of the proposed active learning algorithms on 1) a pool of simulated workers and tasks and 2) a standard image dataset crowdsourced by workers on Mechanical Turk. The algorithm is evaluated using the annotation accuracy of the predicted labels $\mathbf{z}$ and by comparing the actual ($\boldsymbol{\alpha}'$) and predicted ($\boldsymbol{\alpha}$) labeler expertise using both Spearman rank and Pearson correlation statistics. I omit results for evaluating the model's predictions of task difficulty, since its main importance is its contribution to label noise which I am measuring directly through annotation accuracy.

The experiments use the same evaluation protocol and dataset as [91] while attempting to reduce the labeling budget required to reach the desired accuracy level using four different active learning strategies:

QBCKL: selects the most uncertain task using Algorithm 4 with KL-Divergence;

QBCE: selects the most uncertain task using Algorithm 4 with Entropy Vote;

QBU: selects the most uncertain task using Algorithm 3;

RND: randomly selects tasks.

The experiments also attempt to identify appropriate workers by using different worker selection strategies:

BEST: the worker i with highest parameter α_i will be selected;

E0.1: the selection criterion follows Equation 6.10, where $\epsilon = 0.1$;

RND: randomly selects the worker.

At the initialization phase of the active learning, all tasks are annotated by exactly two labelers who are randomly selected for each task to seed the training pool. Using this pool, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are estimated. At each iteration, one sample is selected from the pool of unlabeled pairs (labeler and task). The selected sample is then added to the training pool, and the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{z}$ are updated using the extended training pool.

Since it is difficult to definitively determine the skill level of real human labelers, I ran an initial set of experiments using a simulated worker pool. In the first set of experiments, I evaluate a simple population that consists of workers with only two skill levels and tasks with two difficulty levels. The pool of simulated workers consists of 24 labelers (8 good, 16 bad) annotating 750 tasks (500 easy, 250 hard). The probabilities for each type of labeler correctly annotating different types of tasks are shown in Table 6.1.

Table 6.1: Annotation accuracy of simulated workers

Type of Labeler	Hard Task	Easy Task
Good	0.95	1
Bad	0.54	1

At the initialization stage, each task is labeled by 2 randomly selected labelers, which produces a pool of 1500 labels before all sample selection strategies start running. Each strategy runs for 4000 iterations. Figure 6.2 shows a comparison of the training accuracy of the different sampling strategies (random, query-by-uncertainty, query-by-committee with entropy and query-by-committee with KL-divergence). Since the binary case is relatively simple, the training accuracy starts from a high level (80%) with the 1500 initial samples. The QBU+BEST strategy improves rapidly and converges to almost 100% training accuracy in about 3300 (18.3%) labels. The RND+BEST strategy reaches 98% level after querying 4000 (30.6%) labels. Both QBC strategies perform not well on this binary classification dataset. After selecting 4000 labels, they reach the accuracies lower than 95% training accuracy.

Figure 6.2 also shows a comparison of the training accuracy of the different worker strategies. Since the binary case is relatively simple, the best worker strategy and the ϵ-greedy algorithm with $\epsilon = 0.1$ always work better in improving training accuracy. The random worker selection is unable to reach the same level of training accuracy even with more annotations.

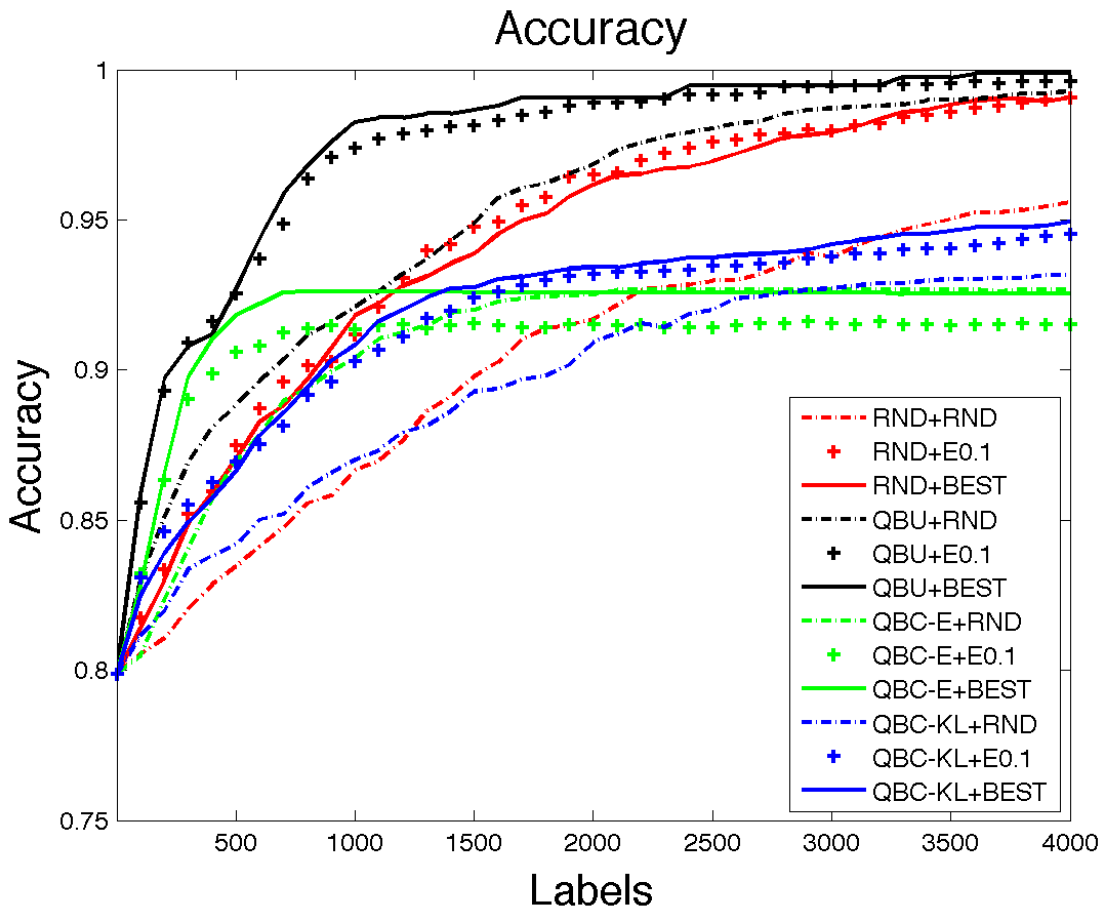


Figure 6.2: Comparison of the annotation accuracy of different task and worker selection strategies using the simulated worker pool with binary worker performance levels.

However, in the real world there can be many different levels of labeler expertise and task difficulty which makes the estimation problem more challenging. To model this, in the second experiment, the worker pool is simulated using α' and β' values that are generated from a Gaussian distribution to simulate a more diverse population of labelers and tasks. I create 50 labelers and 1000 tasks in this experiment. The expertise of labeler i is determined by $\alpha'_i \sim \mathcal{N}(1,1)$, and the difficulty of task j is determined by $\beta'_j \sim \mathcal{N}(1,1)$. At the initialization

stage, each task was labeled by two randomly selected labelers, which yields 2000 labels before all strategies start running. Each strategy runs for 10000 iterations.

Figure 6.3 shows the training accuracy of different combinations of sampling strategies and worker selection strategies. The QBU+BEST strategy still performs strongly at estimating the training accuracy which reaches to almost 98% with 5000 labels. The RND+BEST and QBC-KL+BEST strategies perform similar and reach a slightly lower accuracy rate of around 96% with 5000 labels. The QBC-E strategy still remains the worst. After selecting 5000 labels, it reaches a 90% training accuracy.

Both Pearson correlation and Spearman rank correlation measure the strength of a relationship between two sets of variables. A perfect correlation of 1 represents two sets of variables that are perfectly correlated with each other. Pearson correlation measures the correlation between the actual estimates of worker performance, whereas the Spearman rank correlation compares the similarity of the relative rankings. For practical purposes, having a good Spearman rank correlation is sufficient for correctly selecting the best labeler from the pool.

Figure 6.4 and Figure 6.5 show the result of using Pearson and Spearman rank correlations to evaluate the estimate of α. The results show that, the QBC-E+BEST approach converges to a high rank correlation score (0.95). However, it doesn't necessarily mean that the QBU+BEST algorithm will perform worse at selecting good labelers since the key is having an accurate estimate of the top labelers. Unfortunately, the aggressive sampling approach does not do a good job of evaluating those bad labelers whose α value is also an important contributor in the rank correlation score. I will discuss this phenomenon in more depth during the discussion.

Similar to the binary classification case, both best worker and the ϵ-greedy algorithm with $\epsilon = 0.1$ show similar performance in improving training accuracy. strategies finally reach

the same level of training accuracy but two aggressive strategies converge to that level faster that other two conservative ones. However, their performances change in worker evaluation. The results of using Pearson and Spearman rank show the QBC-E+BEST makes a faster convergence to a higher correlation score (0.95). QBU+RND increases slow at the early stage but finally reaches a good score (0.94). However, no worker selection strategy shows a clear advantage than other methods.

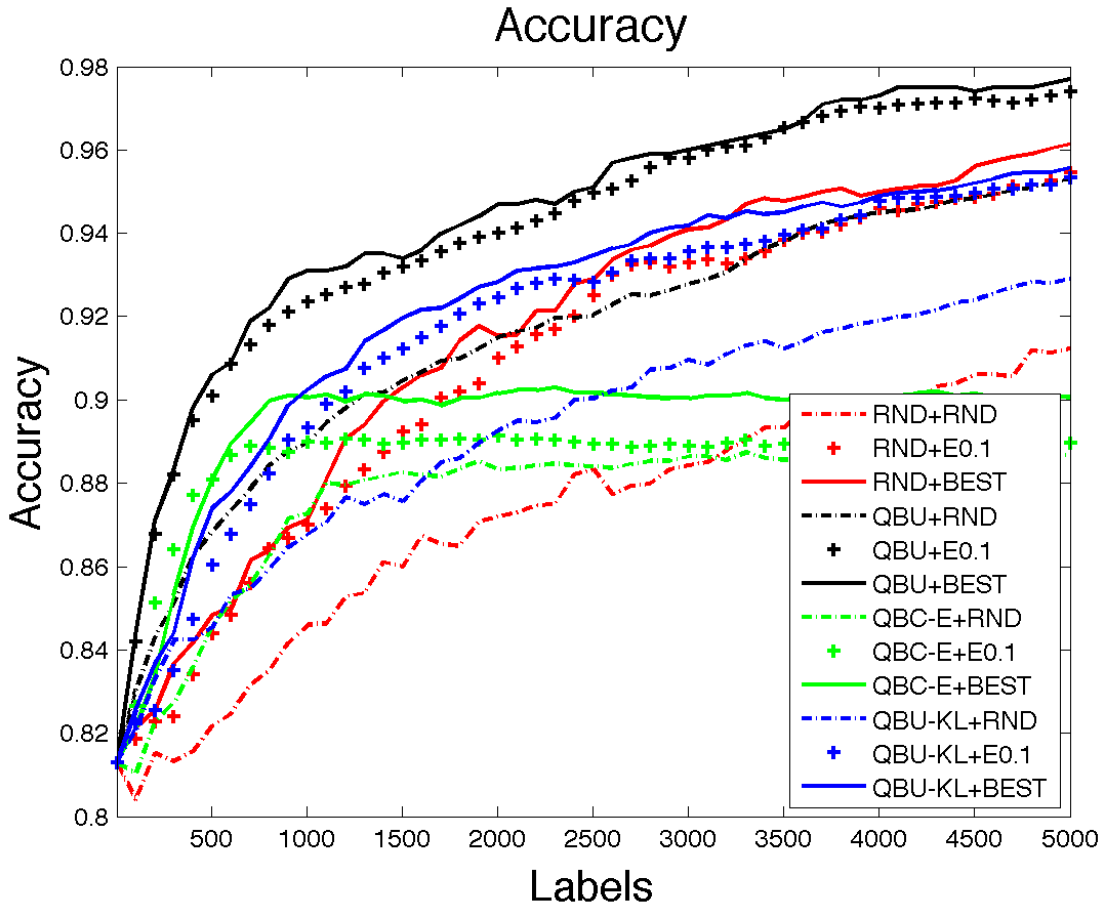


Figure 6.3: Comparison of the annotation accuracy of different sampling strategies using the simulated worker pool with Gaussian worker performance levels.

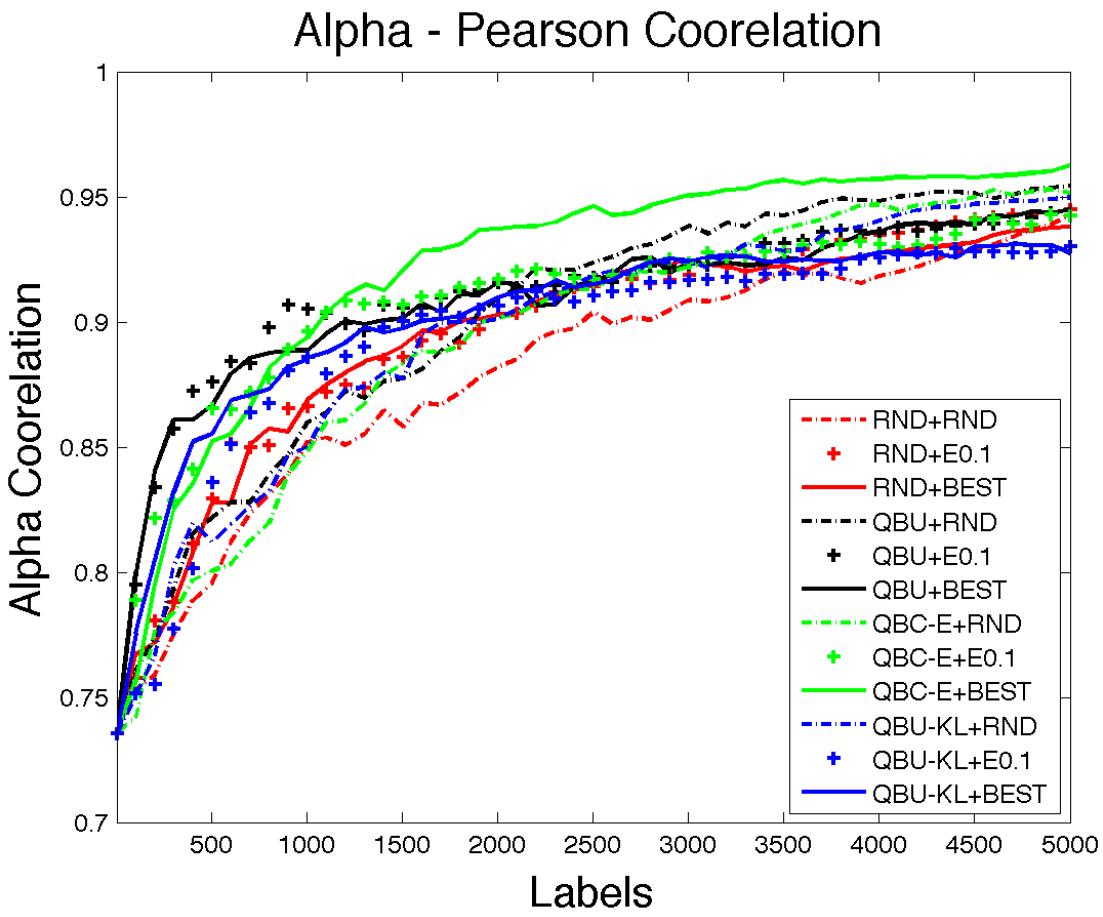


Figure 6.4: Comparison of the different sampling strategies at estimating worker performance in the simulated pool with Gaussian worker performance levels using Pearson correlation.

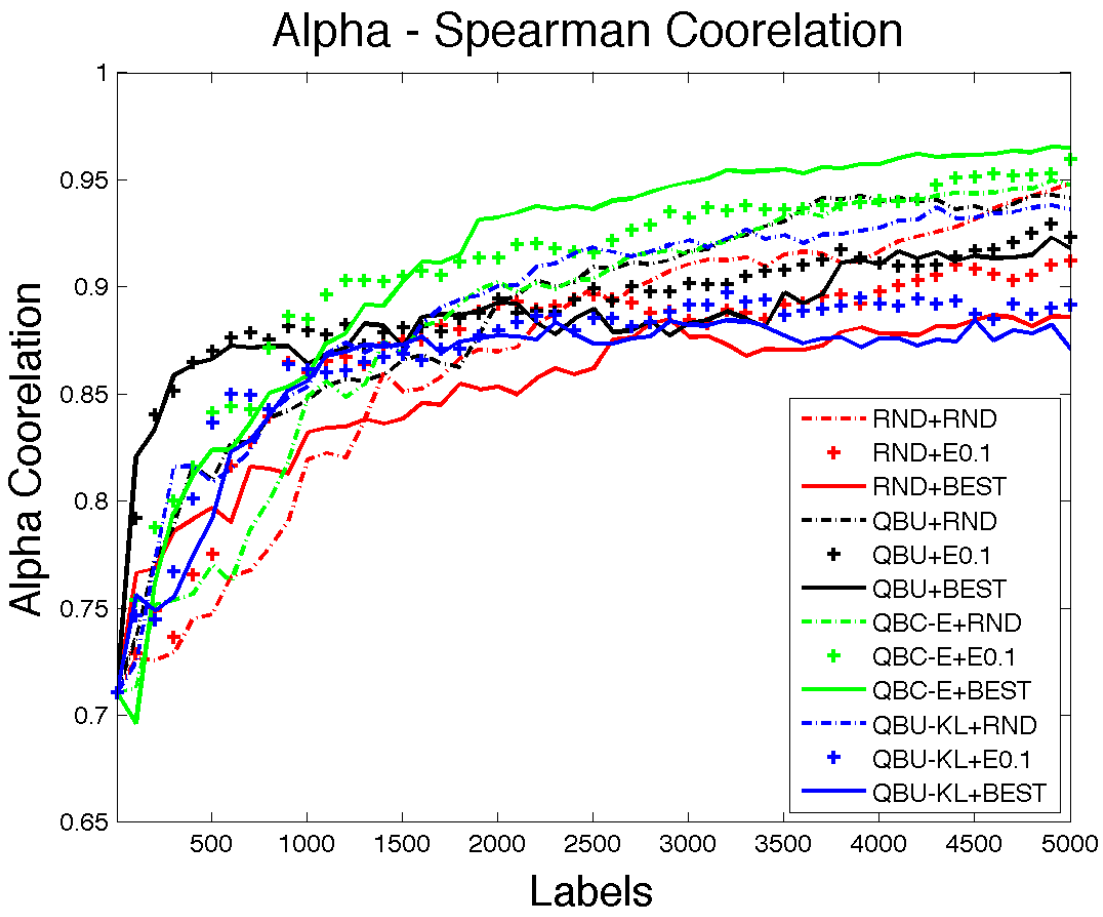


Figure 6.5: Comparison of different sampling strategies at estimated worker performance in the simulated pool with Gaussian worker performance levels using Spearman correlation.

To evaluate the proposed active learning strategy on a standard binary classification task benchmark, I used a facial image dataset crowdsourced with human labelers on Amazon's Mechanical Turk. Whitehill et al. [91] asked 20 real human workers on Mechanical Turk to annotate 160 facial images by labeling them as either Duchenne or Non-Duchenne. A Duchenne smile (enjoyment smile) is distinguished from a Non-Duchenne (social smile) through the activation of the Orbicularis Oculi muscle around the eyes, which the former exhibits and the

latter does not. The dataset consists of 3572 labels. The real worker may provide more than one label with opposite results to the same task. To evaluate the performance of these workers, these images were also annotated by two certified experts in the Facial Action Coding System. According to the expert labels, 58 out of 160 images contained Duchenne smiles.

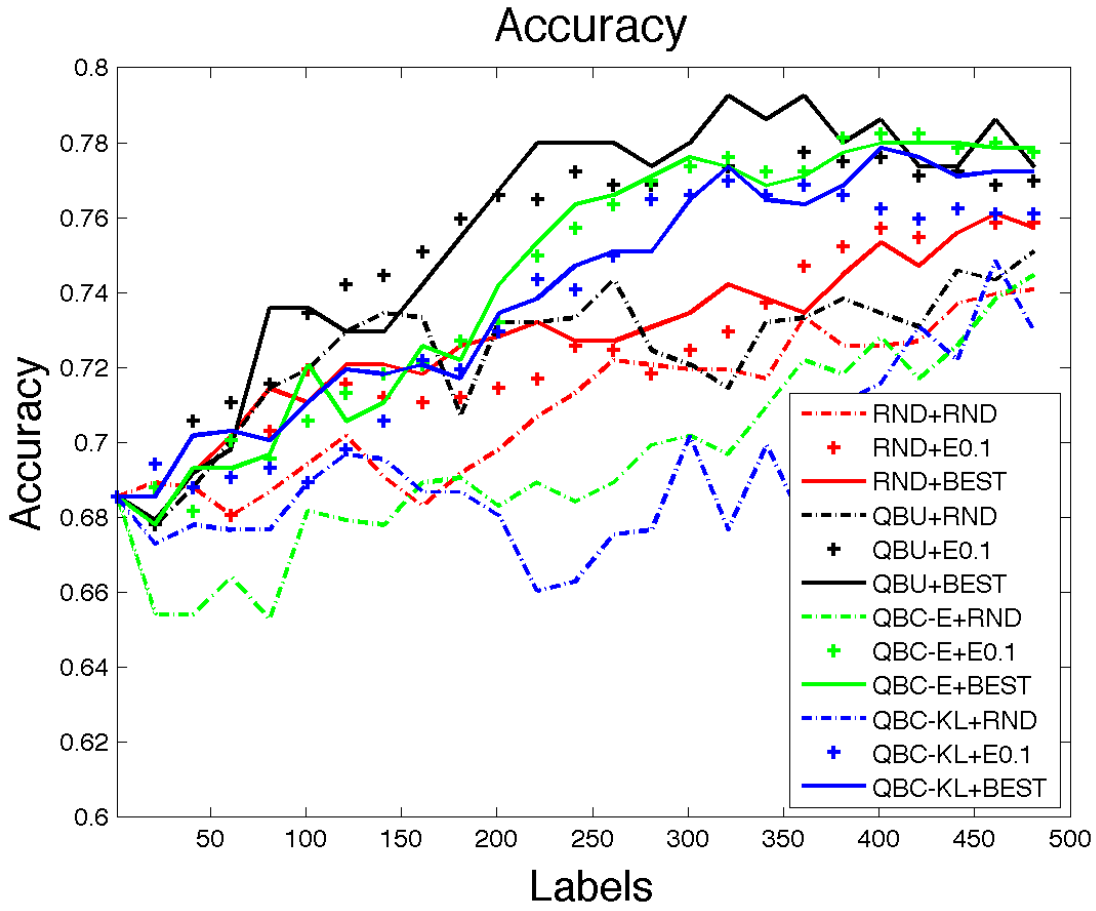


Figure 6.6: Comparison of annotation accuracy with different sampling strategies on the facial image dataset crowdsourced on Mechanical Turk.

In this experiment with labels from real MTurk workers, I also initialized the experiment by asking two people to annotate each task, which means there are $160 \times 2 = 320(10\%)$

labels before all strategies start running. Each strategy runs for 500 (15.6%) iterations. Figure 6.6 shows the training accuracy with different sampling strategies. The QBU+BEST method rises quickly and converges to 78% training accuracy after soliciting 250 (7.0%) labels in total. Both QBC-E-BEST and QBC-KL+BEST strategies converge to the same level with 500 labels, which would result in twice the labeling cost of the QBU+BEST method. RND+BEST fails to converge at the same accuracy level after 500 iterations.

Figure 6.6 also shows the training accuracy with different worker selection strategies. The experimental results indicate that the performance of using QBU+E0.1 which rises quickly and converges to 77% training accuracy after soliciting 250 (7%) labels in total is comparable good as the QBU+BEST strategy. However, the experiments were run 5 times for each strategy, the performance of random worker selection strategy is very unstable and doesn't converge at all after requiring 500 labels with real crowdsourced annotations.

Running time analysis

All experiments were run on a laptop with a 2.26GHz Core 2 Duo processor and a 4GB 1067MHz DDR3 memory. The active learning algorithms are written by MATLAB. The most time-consuming part of this algorithm is that the algorithms have to rerun the EM algorithm each time a new label has been added into the label set. In order to speed up the program and reduce the running time, my program called the EM algorithm implemented in C++[1]. Table 6.2 shows the running time of applying the QBE+RND strategy on three datasets. Table 6.3 indicates the running time of different active/passive learning strategies with random worker selection on dataset 2.

The running time is mainly decided by two variables, the number of samples (# of tasks ×

[1]http://mplab.ucsd.edu/ jake/

# of workers) and the number of iterations. The number of samples decides the size of the Bayesian Network. The bigger the BN, the longer the time spent on each EM cycle. Moreover, the number of iterations decides the number of times required to run EM algorithm, which is the most time-consuming part in the active learning algorithm.

Table 6.2 shows that the running time of the same QBC-E+RND algorithm is almost linear to the size of the Bayesian Network. Table 6.3 indicates that, as the the size of the BN increases, it takes the computer more time to come up with the results. Moreover, using the bagging strategy to generate the committee significantly increases the running time since the program need to repeat the EM algorithm n times at each iteration if we assume there are n committee members.

Table 6.2: Running time for QBC-E+RND algorithm on different datasets

Dataset	# of samples	# of iterations	Running time	Seconds/iteration
1	18,000	4,000	2hours 40minutes	2.4
2	50,000	5,000	5hours 4minutes	3.65
3	3,200	500	6minutes	0.72

Table 6.3: Running time for different sampling algorithms with random worker selection on dataset 2.

Algorithm	# of samples	# of iterations	Running time
RND	no active searching	5,000	26minutes
QBU	50,000	5,000	28minutes
QBCE	50,000	7×5,000	5hours 4minutes
QBCKL	50,000	7×5,000	5hours 0minutes

Discussion

In real-world crowdsourcing applications, annotation accuracy and budget limitations are obviously the most important immediate criteria for evaluating the performance of a learning model. However, identifying knowledgeable and reliable workers is potentially useful because these workers could be employed in future annotation tasks. The difficulty of tasks mainly serves as a discriminant for distinguishing between good and bad labelers, rather than a evaluation score for how well the learning model performs. Especially in active learning, aggressive sampling is unable to perform well on all criteria so task difficulty should be sacrificed for performance gains on the other metrics.

The proposed algorithm aggressively opts to use the best workers possible which yields labeling improvements but makes it difficult to accurately assess the relative rank of the poor labelers. I believe that a simple analysis of the rank correlation may not the best way to evaluate the model's estimate of $\boldsymbol{\alpha}'$ since in most real-world applications labelers under

a certain performance level should be eliminated early. The pool of potential workers that can be reached using MTurk is very large so devoting annotation budget to working with poor labelers is unnecessary. The proposed method is good at dividing labelers into two groups (good and bad) which enables it to perform well in the first simulation experiment while failing to make the subtle discriminations between relatively poor labelers required for assessing worker performance in the second experiment. However, in practical crowdsourcing tasks, filtering out bad labelers is enough for collecting reliable labels in future tasks, and the bottom ranked workers are largely irrelevant to the overall performance of the crowdsourcing pipeline.

In the experiments I also implement and test four worker selection strategies. The best worker strategy is the most aggressive one which only requests the worker with the highest α and ignores improving the evaluation of other workers. Therefore, the best worker strategy is easily outperforms others by converging to the optimal training accuracy faster. However, other conservative strategies, such as random selection which balances the selection of both the best worker and other workers, converge slower but to higher Pearson and Spearman rank correlations, especially in a more diverse population of labelers and tasks. Our experimental results indicate that no worker selection strategy shows consistent advantage on all sampling strategies. The performance really depends on the combination of the task and worker selection strategy and also the dataset used.

The essential difference between the proposed algorithms and other workers modeling papers [30, 45, 94, 32] is that it relies solely on annotations. Omitting feature extraction can be an advantage for crowdsourcing tasks in which it is not possible or economical to build good feature distribution models. Also, the most informative or uncertain sample in the feature space is not necessarily the one that is most difficult for workers to come up with a correct label. For example, Yan et al. [94] assume workers are better at providing correct labels if

they have seen similar tasks before. However, this assumption is not true for bad workers.

CHAPTER 7: CONCLUSION

Active Learning with Worker and Task Modeling

Several studies in different research domains show that active learning approaches developed for noise-free annotations do not perform well with crowdsourced data. This thesis presents a practical approach for using active learning in conjunction with Bayesian networks to model both the expertise of unknown labelers and the difficulty of annotation tasks.

There are main contributions of this research. First, I propose an original and efficient sampling criteria which iteratively assigns the most reliable labelers to the tasks with the highest labeling risk. Second, I present comprehensive evaluations on both simulated and real-world datasets that show the strength of our proposed approach in significantly reducing the quantity of labels required for training the model. The experiments using crowdsourced data from Mechanical Turk confirm that the proposed approach improves active learning in noisy real-world conditions. The experimental results indicate that the query-by-uncertainty task selection strategy always shows overwhelming advantages on training accuracy in both simulated and real dataset. However, no worker selection strategy consistently dominates over other methods.

Worker selection strategies have different performances when employing different task evaluation criteria. More aggressive worker selection strategies, e.g. selecting the best worker or using ϵ-greedy selection with $\epsilon = 0.1$, perform better in improving the training accuracy. However, these two strategies focus more on selecting workers who perform well in previous iterations, which means those "bad workers" are ignored in the learning process. So more conservative selection strategies perform better at evaluating the overall ranking of the

worker performances.

Future Work

Although my thesis investigates active learning strategies with different selection criteria to solve the problem of which annotation tasks should be labeled by whom, there are still several open questions worhty of discussion.

Many assumptions used in this thesis to simplify the experimental settings may not exist in real world. We assume that all workers are willing to provide labels immediately when they receive annotation requests. However, it is impossible for workers to wait online 24/7 for answering the incoming requests. Therefore, an interesting research topic would be to investigate balancing request urgency with annotation accuracy. Should we assign the annotation task to the best available worker or wait for a worker who qualifies for the task with an high enough expertise score?

Another question is, if different workers charge different costs for the same task, should we spend the money on a high quality worker who charges more, or allocate the money to more low quality workers? In the experiment it is always good to select best workers since all labels have the same price. Obviously the answer varies if the price of acquiring a label from workers are different. Our simulation model assumes that both high and low quality workers have similar probabilities of providing correct labels for tasks with a low difficulty score.

LIST OF REFERENCES

[1] D. Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[2] D. Angluin. Queries Revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

[3] O. Arikan, D.A. Forsyth, and J.F. O'Brien. Motion synthesis from annotations. In *ACM Transactions on Graphics*, volume 22, pages 402–408, 2003.

[4] L. Atlas, D. Cohn, R. Ladner, MA El-Sharkawi, and II Marks. Training Connectionist Networks with Queries and Selective Sampling. In *Advances in Neural Information Processing Systems*, pages 566–573, 1990.

[5] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *International Conference on Architecture of Computing Systems*, 2010.

[6] M. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of International Conference on Machine Learning*, 2006.

[7] L. Bao and S.S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[8] J. Barbic, A. Safonova, J-Y. Pan, C. Faloutsos, J. Hodgins, and N. Pollard. Segmenting Motion Capture Data into Distinct Behaviors. In *Proceedings of Graphics Interface*, pages 185–194, 2004.

[9] E.B. Baum and K. Lang. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, 1992.

[10] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of International Conference of Machine Learning*, 2009.

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

[12] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[13] C. Callison-Burch. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 286–295, 2009.

[14] J. Q. Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. MIT Press, 2009.

[15] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and Kernel Methods Matlab Toolbox. *Perception Systmes et Information, INSA de Rouen, Rouen, France*, 2, 2005.

[16] R. Caruana. Multitask learning. *Machine Learning*, 28, 1997.

[17] E.Y. Chang, S. Tong, K. Goh, and C. Chang. Support Vector Machine Concept-Dependent Active Learning for Image Retrieval. *IEEE Transactions on Multimedia*, 2, 2005.

[18] O. Chapelle, V. Sindhwani, and S. Keerthi. Optimization Techniques for Semi-Supervised Support Vector Machines. *Journal of Machine Learning Research*, 9:203–233, 2008.

[19] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic Discovery of Time Series Motifs. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–498, 2003.

[20] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *JAIR*, 4:129–145, 1996.

[21] D.A. Cohn. Neural Network Exploration using Optimal Experiment Design. *Neural Networks*, 9(6):1071–1083, 1996.

[22] A. Culotta and A. McCallum. Confidence Estimation for Information Extraction. In *Proceedings of HLT-NAACL*, pages 109–112, 2004.

[23] F. Dabiri, A. Vahdatpour, H. Noshadi, H. Hagopian, and M. Sarrafzadeh. Ubiquitous Personal Assistive System for Neuropathy. In *Proceedings of the Second International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, pages 17–22, 2008.

[24] S. Dasgupta and D. Hsu. Hierarchical Sampling for Active Learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 208–215, 2008.

[25] F. De la Torre, J. Hodgins, A. Bargtell, X. Artal, J. Macey, A. Castellis, and J. Beltran. Guide to the CMU Multimodal Activity Database. Technical Report RI-08-22, CMU, 2008.

[26] A. Doan, R. Ramakrishnan, and A.Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, 2011.

[27] P. Donmez, J. Carbonell, and P. Bennett. Dual strategy active learning. In *in Proceeding of the European Conference on Machine Learning*, pages 116–127, 2007.

[28] P. Donmez and J.G. Carbonell. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management*, 2008.

[29] P. Donmez, J.G. Carbonell, and J. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SIAM International Conference on Data Mining (SDM)*, pages 826–837, 2010.

[30] Jun Du and Charles X Ling. Active learning with human-like noisy oracle. In *IEEE 10th International Conference on Data Mining*, pages 797–802, 2010.

[31] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, 1994.

[32] Meng Fang, Xingquan Zhu, Bin Li, Wei Ding, and Xindong Wu. Self-taught active learning from crowds. In *IEEE 12th International Conference on Data Mining*, pages 858–863, 2012.

[33] W. Fu, P. Ray, and E.P. Xing. DISCOVER: a feature-based discriminative method for motif search in complex genomes. *Bioinformatics*, 25(12):321–329, 2009.

[34] Ravi Ganti and Alexander Gray. Upal: Unbiased pool based active learning. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 2012.

[35] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[36] Steve Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th international conference on Machine learning*, pages 353–360, 2007.

[37] J. Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6), 2006.

[38] P. Ipeirotis. Crowdsourcing using Mechanical Turk: Quality management and scalability. In *Proceedings of Workshop on Crowdsourcing for Search and Data Mining*, 2011.

[39] P.G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.

[40] E. Keogh. Efficiently Finding Arbitrarily Scaled Patterns in Massive Time Series Databases. In *Proceedings of Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 253–265. Springer Verlag, 2002.

[41] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[42] A.M. Koblin. The sheep market. In *Proceeding of the seventh ACM conference on Creativity and cognition*, pages 451–452, 2009.

[43] Volodymyr Kuleshov and Doina Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning Research*, 2010.

[44] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[45] Abhimanu Kumar and Matthew Lease. Modeling annotator accuracies for supervised learning. *Crowdsourcing for Search and Data Mining*, pages 19 – 22, 2011.

[46] S. Kumar and M. Hebert. Discriminative Fields for Modeling Spatial Dependencies in Natural Images. In *Advances in Neural Information Processing Systems*, 2003.

[47] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmentation and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning*, pages 282 – 289, 2001.

[48] Florian Laws, Christian Scheible, and Hinrich Schütze. Active learning with amazon mechanical turk. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, 2011.

[49] A. Levin and T. Weiss. Learning to combine bottom-up and top-down segmentation. In *Proceedings of European Conference on Computer Vision*, 2006.

[50] A. Levin and T. Weiss. Learning to combine bottom-up and top-down segmentation. *International Journal of Computer Vision*, 81(1):105–118, 2009.

[51] L. Liao, D. Fox, and H. Kautz. Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields. *International Journal of Robotics Research*, 26(1):119–134, 2007.

[52] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding Motifs in Time Series. In *Workshop on 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53 – 68, 2002.

[53] T. Lin, P. Ray, G.K. Sandve, S. Uguroglu, and E.P. Xing. BayCis: A Bayesian hierarchical HMM for cis-regulatory module decoding in metazoan genomes. In *Proceedings of the 12th annual International Conference on Research in Computational Molecular Biology*, pages 66–81, 2008.

[54] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[55] A. McCallum. Efficiently Inducing Features of Conditional Random Fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003.

[56] D. Minnen, T. Starner, I. Essa, and C. Isbell. Discovering Characteristic Actions from On-Body Sensor Data. In *Proceedings of the 10th International Symposium on Wearable Computers*, pages 11–18, 2006.

[57] T.M. Mitchell. Generalization as Search. *Artificial intelligence*, 18(2):203–226, 1982.

[58] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of International Conference of Machine Learning*, pages 79–86, 2004.

[59] W. Pentney, A. Popescu, S. Wang, H. Kautz, and M. Philipose. Sensor-based understanding of daily life via large-scale use of common sense. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[60] N. Plath, M. Toussaint, and S. Nakajima. Multi-class Image Segmentation using Conditional Random Fields and Global Classification. In *Proceedings of the 26th International Conference on Machine Learning*, pages 817–824, 2009.

[61] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 1999.

[62] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[63] M. Ramezani, J. Sandvig, T. Schimoler, J. Gemmell, B. Mobasher, and R. Burke. Evaluating the impact of attacks in collaborative tagging environments. In *Proceedings of Social Computing*, 2009.

[64] N. Ravi, N. Dandekar, P. Mysore, and M.L. Littman. Activity recognition from accelerometer data. In *Proceedings of the National Conference on Artificial Intelligence*, 2005.

[65] V.C. Raykar, S. Yu, L.H. Zhao, A. Jerebko, C. Florin, G.H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 889–896, 2009.

[66] B. Settles. Active learning literature survey. Technical report, University of Wisconsin, Madison, 2010.

[67] V.S. Sheng, F. Provost, and P.G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2008.

[68] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: frame transductive to semi-supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 824–831, 2005.

[69] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional model for contextual human motion recognition. In *CVPR*, 2005.

[70] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional Random Fields for Contextual Human Motion Recognition. In *Proceedings of the 10th IEEE International Conference on Computer Vision*, volume 2, pages 1808–1815, 2005.

[71] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

[72] E. Spriggs, F. De la Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 17–24, 2009.

[73] T. Stiefmeier, D. Roggen, and G. Troster. Fusion of String-Matched Templates for Continuous Activity Recognition. In *Proceedings of the 11th IEEE International Symposium on Wearable Computers*, pages 41–44, 2007.

[74] Stiefmeier, T. and Roggen, D. and Ogris, G. and Lukowicz, P. and TrOster, G. Wearable Activity Tracking in Car Manufacturing. *IEEE Pervasive Computing*, 7(2):42–50, 2008.

[75] Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. *Machine Learning*, 58(2):269 – 300, 2005.

[76] MF Tappen, C. Liu, EH Adelson, and WT Freeman. Learning Gaussian Conditional Random Fields for Low-level Vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[77] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[78] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM International Conference on Multimedia*, 2001.

[79] S. Tong and D. Koller. Active learning for parameter estimation in Bayesian networks. In *Proceedings of the Advances in Neural Information Processing Systems*, 2001.

[80] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, 2:45 – 66, 2002.

[81] A. Vahdatpour, N. Amini, and M. Sarrafzadeh. Toward Unsupervised Activity Discovery Using Multi-Dimensional Motif Detection in Time Series. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1261 – 1266, 2009.

[82] D. Vail et al. Feature Selection for Activity Recognition in Multi-Robot Domains. In *Proceedings of the National Conference on Artificial Intelligence*, volume 3, pages 1415 – 1420, 2008.

[83] D.L. Vail. *Conditional random fields for activity recognition.* PhD thesis, Carnegie Mellon University, 2008.

[84] D.L. Vail, M.M. Veloso, and J.D. Lafferty. Conditional Random Fields for Activity Recognition. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, 2007.

[85] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[86] L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, 2004.

[87] C. Vondrick, D. Ramanan, and D. Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. In *Proceedings of the European Conference on Computer Vision*, 2010.

[88] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. Managing crowdsourcing workers. In *The 2011 Winter Conference on Business Intelligence*, pages 10–12, 2011.

[89] L. Wang, K.L. Chan, and Z. Zhang. Bootstrapping SVM active learning by incorporating unlabelled images for image retrieval. In *Proceedings of IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition, volume 1, pages 629 – 634, 2003.

[90] Peter Welinder and Pietro Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32, 2010.

[91] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22:2035–2043, 2009.

[92] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J.M. Rehg. A scalable approach to activity recognition based on object use. In *Proceedings of the IEEE Eleventh International Conference on Computer Vision*, 2007.

[93] W. Wu, L. Au, B. Jordan, T. Stathopoulos, M. Batalin, W. Kaiser, A. Vahdatpour, M. Sarrafzadeh, M. Fang, and J. Chodosh. The Smartcane System: an Assistive Device for Geriatrics. In *Proceedings of the ICST 3rd International Conference on Body Area Networks*, pages 1–4, 2008.

[94] Y. Yan, R. Rosales, G. Fung, and J. Dy. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning (ICML), Bellevue, Washington*, 2011.

[95] L. Zhao, G. Sukthankar, and R. Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. In *IEEE SocialCom*, 2011.

[96] L. Zhao, X. Wang, and G. Sukthankar. Recognizing household activities from human motion data using active learning and feature selection. *Technology and Disability*, 22(1–2):17–26, 2010.

[97] Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. Importance-weighted label prediction for active learning with noisy annotations. In *The 21st International Conference on Pattern Recognition*, pages 3476–3479, 2012.

[98] Y. Zheng, S. Scott, and K. Deng. Active learning from multiple noisy labelers with varied costs. 2010.

[99] F. Zhou, F. De la Torre, and J. Hodgins. Aligned Cluster Analysis for Temporal Segmentation of Human Motion. In *Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition*, pages 1 – 7, 2008.