A Probabilistic Framework to Learn from Multiple Annotators with Time-Varying Accuracy

Pinar Donmez
Language Technologies Institute
Carnegie Mellon University
pinard@cs.cmu.edu

Jaime Carbonell
Language Technologies Institute
Carnegie Mellon University
jgc@cs.cmu.edu

Jeff Schneider
Robotics Institute
Carnegie Mellon University
schneide@cs.cmu.edu

Abstract

This paper addresses the challenging problem of learning from multiple annotators whose labeling accuracy (reliability) differs and varies over time. We propose a framework based on Sequential Bayesian Estimation to learn the expected accuracy at each time step while simultaneously deciding which annotators to query for a label in an incremental learning framework. We develop a variant of the particle filtering method that estimates the expected accuracy at every time step by sets of weighted samples and performs sequential Bayes updates. The estimated expected accuracies are then used to decide which annotators to be queried at the next time step. The empirical analysis shows that the proposed method is very effective at predicting the true label using only moderate labeling efforts, resulting in cleaner labels to train classifiers. The proposed method significantly outperforms a repeated labeling baseline which queries all labelers per example and takes the majority vote to predict the true label. Moreover, our method is able to track the true accuracy of an annotator quite well in the absence of gold standard labels. These results demonstrate the strength of the proposed method in terms of estimating the time-varying reliability of multiple annotators and producing cleaner, better quality labels without extensive label queries.

1 Introduction

In many data mining applications, obtaining expert labels is a time-consuming and costly process. More importantly, obtaining noise-free or low-noise labels is crucial to build reliable systems/models. Recent advances in crowdsourcing tools (e.g. Amazon Mechanical Turk) provide cheaper, faster ways of acquiring labels for large data collections. Despite crowdsourcing not providing very reliable labels, for mostly simple tasks these tools are shown to be reliable in producing quality labels on average[12]. Nevertheless, it is still unclear to what extent such tools can be trusted for tasks that require expertise higher than unvetted average annotators can handle. Furthermore, using human labelers (annotators) is not the only way to acquire labels. For instance, expensive machinery and wet lab experiments are often applied in protein structure prediction. In medical diagnosis, blood or urine tests, certain type of biopsy, etc. are used to diagnose a patient (i.e. to produce diagnosis labels). These procedures range from cheap and partially reliable to more costly and more reliable. Hence, it is essential to use multiple sources to increase confidence while maintaining a trade-off between quality and cost. This requires estimating the accuracy (reliability) of multiple prediction sources and identifying the highly accurate ones to rely on.

A dimension not yet addressed is that in many application areas, providers of information may exhibit varying accuracy over time, and it is therefore useful to track such variation in order to know when and how much to rely upon their answers. For instance, scientific equipment may lose calibration over time, and thus increase measurement error until upgraded returning to peak performance or better. Text topic labelers become careless with time due to fatigue effects, but may be refreshed for future labeling sessions or hone their skills to produce increasingly reliable results.

In this paper, we tackle the problem of learning from multiple sources whose accuracy may vary over time. To the best of our knowledge, this is the first attempt in the literature that addresses this phenomenon. Throughout the paper, we use the terms accuracy, reliability, quality interchangeably and also the terms labeler, annotator,

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

We organize the rest of the paper as follows. The next section reviews related work in the literature. Section 3 explains in detail the proposed framework, followed by the empirical evaluation in Section 4. Finally, the last section offers our conclusions and future directions.

2 Related Work

Collecting multiple annotations in the absence of gold standard labels is becoming a more common practice in the literature. Raykar et al. [9] propose an EM-based algorithm to estimate the error rate of multiple annotators assuming conditional independence of the annotator judgments given the true label. Their method iteratively estimates the gold standard, and measures the performance of multiple annotators and update the gold standard based on the performance measures. Dekel and Shamir [4] offer a solution to identify low-quality or malicious annotators. But their framework is rather limited in the sense that it is based only on Support Vector Machines (SVMs). More importantly, they assume that each annotator is either good or bad, not in a continous distribution and not time-varying. Good annotators assign labels based on the marginal distribution of the true label conditioned on the instance whereas bad annotators provide malicious answers.

There are earlier attempts that address multiple imperfect annotators, but they do not infer the performance of each individual annotator. Sheng et al. [11] and Snow et al. [12] show that it can be effective to use multiple, potentially noisy labels in the absence of gold standard. Sheng et al. [11] takes the majority voting to infer a single integrated label. Furthermore, their experimental analysis relies on the assumption that all labelers have the same labeling quality, which is very unlikely to hold in real world. Snow et al. [12] collected labeled data through Amazon Mechanical Turk for simple natural language understanding tasks. They empirically show high agreement between the existing gold-standard labels and non-expert annotations. Their analysis is carried out on fairly straightforward tasks and collecting a reasonably large number of non-experts labels. However, it is unclear if their conclusions would generalize to other tasks that require better-than-average expertise and under high label acquisition cost which restricts the total number of labelings one can afford.

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

Another point where the proposed work differs is the identification and selection of the most reliable annotators for each instance to be labeled. Donmez et al. [5] propose a solution that balances the exploration vs. exploitation trade-off, but it is only applicable when the accuracies are fixed over time. The proposed model, however, constantly monitors the potential changes in little- or non-explored annotators. It may select previously rejected annotators if there is some chance that their accuracies now exceed that of the already exploited ones. This leads to reliable estimates of changing annotator quality with very modest labeling cost.

3 Estimation Framework

In this section, we describe our estimation and selection framework in detail. We start with the underlying particle filtering algorithm and the specific probabilistic distributions we modeled. Then, we discuss how to modify this model to estimate the varying accuracy of each labeler and simultaneously select the most accu-

rate ones. Our framework supports a balance between exploration and exploitation to achieve estimation accuracy without extensively exploiting the labelers and incurring associated costs.

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

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

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

$$\begin{aligned} \phi_t &= f_t(\phi_{t-1}, \Delta_{t-1}) \\ &= \phi_{t-1} + \Delta_{t-1} \end{aligned} \tag{3.1}$$

where Δ_t is a zero-mean, σ^2-variance Gaussian random variable. f_t denotes that the accuracy at the current time step differs from the previous accuracy by some small amount drawn from a Gaussian distribution. The mean of the Gaussian is assumed to be zero not to introduce any bias towards increase or decrease in accuracy. We can easily add a bias, e.g. positive mean μ for labelers who are expected to improve with experience. However, we prefer fewer parameters and we can still track the improving sources (more details in Section 4). We assume σ or at least its upper bound to be known. We realize that the exact value of σ can be difficult to obtain in many situations. On the other hand, it is often reasonable to assume the maximum rate of change of the labeling accuracy is known. We added an analysis testing the sensitivity to the exact σ in the experiments section 4. For notational simplicity and concreteness, we focus on binary classification here; extensions to the multi-category case are straightforward generalizations to this scheme. Furthermore, the accuracy at any time t is assumed to be between 0.5 and 1, i.e. any labeler is a weak learner, which is a standard assumption. Thus, the transition probability from one state to the next follows a truncated Gaussian distribution:

$$p(\phi_t \mid \phi_{t-1}, \sigma, 0.5, 1) = \frac{\frac{1}{\sigma}\beta(\frac{\phi_t - \phi_{t-1}}{\sigma})}{\Phi(\frac{1-\phi_{t-1}}{\sigma}) - \Phi(\frac{0.5-\phi_{t-1}}{\sigma})} \tag{3.2}$$

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

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

$$p(z^j_t \mid \phi^j_t, y_t) = {\phi^j_t}^{I(z^j_t = y_t)}(1-\phi^j_t)^{I(z^j_t \neq y_t)}. \tag{3.3}$$

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

$$p(z^j_t \mid \phi^j_t, z^{\acute{J}(t)}_t) = \sum_{y \in \mathcal{Y}} p(z^j_t \mid \phi^j_t, y_t = y) P(y_t = y \mid z^{\acute{J}(t)}_t) \tag{3.4}$$

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

$$P(y_t = y \mid z^{J(t)}_t) = \frac{\hat{P}_{t-1}(y) \prod_{s \in \acute{J}(t)} p_{\hat{E}[\phi^s_{t-1}]}(z^s_t \mid y)}{\sum_{y \in \mathcal{Y}} \hat{P}_{t-1}(y) \prod_{s \in \acute{J}(t)} p_{\hat{E}[\phi^s_{t-1}]}(z^s_t \mid y)} \tag{3.5}$$

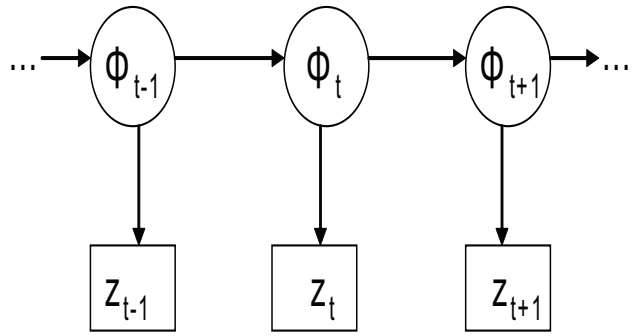

Figure 1: The Hidden Markov Model structure for the time-varying accuracy of a labeler. The state sequence ϕ_t is an unobserved first-order Markov process. The observation z_t is dependent only on the current state ϕ_t for all $t = 1, 2, \ldots$.

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

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

$$p(\phi_t \mid \phi_0, \ldots, \phi_{t-1}) = p(\phi_t \mid \phi_{t-1}) \tag{3.6}$$

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

$$p(z_t \mid \phi_0, \ldots, \phi_t) = p(z_t \mid \phi_t) \tag{3.7}$$

Figure 1 shows the graphical structure of this model. As noted earlier, we are interested in constructing $p(\phi_t \mid z_{1:t})$. It is generally assumed that the initial state distribution $p(\phi_0)$ is given. However, in this paper, we do not make this assumption and simply adopt a uniform (noninformative) prior for $p(\phi_0)$ where $0.5 < \phi_0 < 1$.[1] Suppose that the state distribution $p(\phi_{t-1} \mid z_{1:t-1})$ at time $t-1$ is available. The prediction stage obtains the pdf of the state at time step t via the Chapman-Kolmogorov equation[1].

$$p(\phi_t \mid z_{1:t-1}) = \int_{\phi_{t-1}} p(\phi_t \mid \phi_{t-1}) p(\phi_{t-1} \mid z_{1:t-1}) d\phi_{t-1} \tag{3.8}$$

[1] The results show that the estimation is very effective despite the uninformative prior (See Section 4 for more details). However, in cases where such information is available, we anticipate that the results could be improved even further.

where $p(\phi_t \mid \phi_{t-1})$ is substituted with (3.2). Once a new observation z_t is made, then (3.8) is used as a prior to obtain the posterior $p(\phi_t \mid z_{1:t})$ at the update stage via Bayes rule.

$$p(\phi_t \mid z_{1:t}) = \frac{p(z_t \mid \phi_t) p(\phi_t \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})} \tag{3.9}$$

where $p(z_t \mid \phi_t)$ is substituted with (3.4) and $p(z_t \mid z_{1:t-1}) = \int_{\phi_t} p(z_t \mid \phi_t) p(\phi_t \mid z_{1:t-1}) d\phi_t$. When the posterior (3.9) at every time step t is assumed to be Gaussian, then Kalman filters provide optimal solutions to the state density estimation. In cases like ours where the posterior density is not Gaussian, sequential particle filtering methods are appropriate for approximating the optimal solution. Next, we review the particle filtering algorithm and describe how it is modified to tackle our problem.

3.2 Particle Filtering for Estimating Time-Varying Labeler Accuracy The particle filtering algorithm is a technique for implementing sequential Bayesian estimation by Monte Carlo simulations. The underlying idea is to estimate the required posterior density function with a discrete approximation using a set of random samples (particles) and associated weights.

$$p(\phi_t \mid z_{1:t}) \sim \sum_{i=1}^{N} w_t^i \delta(\phi_t - \phi_t^i) \tag{3.10}$$

where N is the number of particles. As N increases, it can be shown that the above approximation approaches the true posterior density [7]. w_t^i's are called the normalized importance weights, and δ is the Dirac delta function[2]. The particle filtering algorithm estimates these weights in a sequential manner. Weights in (3.10) are approximated as

$$\begin{aligned} \tilde{w}_t^i &\approx \frac{p(\phi^i_{1:t} \mid z_{1:t})}{\pi(\phi^i_{1:t} \mid z_{1:t})} \\ w_t^i &= \frac{\tilde{w}_t^i}{\sum_{j=1}^{N} \tilde{w}_t^j} \end{aligned} \tag{3.11}$$

where π is called the importance density from which the samples ϕ^i are generated. This is because in general we cannot directly sample from the posterior $p(\phi_{1:t} \mid z_{1:t})$,

[2]
$$\delta(x) = 0, \text{ if } x \neq 0$$
$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$

1. Sample from the initial distribution $\phi_0^i \sim p(\phi_0)$ for $i = 1, \ldots, N$ and assign weights $w_0^i = \frac{1}{N}$
2. For $t > 0$, and $i = 1, \ldots, N$
   - Draw $\phi_t^i \sim p(\phi_t \mid \phi^{t-1})$ using (3.2) and update weights $\tilde{w}_t^i = p(z_t \mid \phi_t^i)\tilde{w}_{t-1}^i$.
   - Normalize the weights $w_t^i = \frac{\tilde{w}_{t-1}^i}{\sum_{j=1}^N \tilde{w}_{t-1}^j}$ and compute $\hat{N}_e = \frac{1}{\sum_{i=1}^N (w_t^i)^2}$.
   - If $\hat{N}_e < T$, then resample $\phi_t^i \sim \text{pmf}[\{w_t\}]$ and reassign $w_t^i = \frac{1}{N}$
   - Update the posterior state density $p(\phi_t \mid z_{1:t}) = \sum_{i=1}^N w_t^i \delta(\phi_t - \phi_t^i)$
3. Update $t = t + 1$ and go to step 2.

Figure 2: The pseudo-code for the basic filtering algorithm

but rather use an importance density that we can easily draw the samples from. The algorithm sequentially samples ϕ_t^i, $i = 1, 2, \ldots, N$, from the importance density and updates the weights w_t^i to approximate the posterior state distribution. The choice of the importance density is important and one commonly used and convenient choice is to use the state transition density $p(\phi_t \mid \phi_{t-1})$ (in our case given in (3.2)), i.e. $\pi(\phi_t \mid \phi_{1:t-1}^i, z_{1:t}) = p(\phi_t \mid \phi_{t-1})$. Then, the weight update equation simplifies to

$$\begin{aligned}\tilde{w}_t^i &\approx \frac{p(z_t \mid \phi_t^i)p(\phi_t^i \mid \phi_{t-1}^i)p(\phi_{1:t-1}^i \mid z_{1:t-1})}{\pi(\phi_t^i \mid \phi_{1:t-1}^i, z_{1:t})\pi(\phi_{1:t-1}^i \mid z_{1:t-1})} \\ &= \frac{p(z_t \mid \phi_t^i)p(\phi_t^i \mid \phi_{t-1}^i)}{\pi(\phi_t^i \mid \phi_{1:t-1}^i, z_{1:t})}\tilde{w}_{t-1}^i \\ &= p(z_t \mid \phi_t^i)\tilde{w}_{t-1}^i \text{ for } t > 1 \qquad (3.12)\end{aligned}$$

where $p(z_t \mid \phi_t^i)$ is substituted with (3.4). The pseudo-code of the basic filtering algorithm is given in Figure 2.

The resampling step in Figure 2 happens only if the vast majority of the samples have negligible weights; in other words, the estimated effective sample size $\hat{N}_e$ falls below some predefined threshold. The exact value of the threshold is not crucial, the idea is to detect significantly small weights. This is called the degeneracy problem in particle filters, and is resolved by resampling using a probability mass function (pmf) over the weights. The goal of resampling is to eliminate particles with small weights and concentrate on large ones. It is shown that it is possible to implement this resampling procedure in $O(N)$ time complexity using order statistics [10, 3]. We do not provide more details on this since it is out of the scope of the paper, but we note that in our simulations the algorithm hardly needs to resample, largely because the importance density we adopted (3.2) represents the state transitions well.

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

Our aim is to select potentially the most accurate labelers at each time step. The sequential estimation framework provides us with an effective tool to achieve this goal. First, we approximate the prior pdf of ϕ_t^j (3.8) $\forall j$ by its discrete version

$$p(\phi_t^j \mid Z_{t-h(j)}^j) = \sum_{i=1}^N p(\phi_t^j \mid \phi_{t-h(j)}^{j,i})p(\phi_{t-h(j)}^{j,i} \mid Z_{t-h(j)}^j) \qquad (3.13)$$

where $t - h(j)$ denotes the last time the labeler j is selected, e.g. $t - h(j) = t - 1$ if the labeler is queried for labeling at the immediately previous time step. $Z_{t-h(j)}^j$ denotes all the observations obtained from labeler j up to time $t - h(j)$. $\{\phi_{t-h(j)}^{j,i}\}_{i=1}^N$ denotes the sampled accuracy values for the j-th labeler at time $t - h(j)$. Furthermore, we formulate the change in the labeling accuracy as a Gaussian random walk bounded between 0.5 and 1. More importantly, the true accuracy of a labeler keeps evolving according to this random walk regardless of whether it is selected by our method. For computational expediency, we approximate the state transition probability by truncating the Gaussian distribution once after $h(j)$ steps instead of after every single step for an unxplored labeler. More formally, recall (3.1) where the step size Δ_t is drawn from a Gaussian $\Delta_t \sim \mathcal{N}(0, \sigma^2)$. Hence,

$$\begin{aligned}\phi_t^j &= \phi_{t-1}^j + \Delta_{t-1} \\ &= \phi_{t-2}^j + \Delta_{t-2} + \Delta_{t-1} \\ &\vdots \\ &= \phi_{t-h(j)}^j + \sum_{r=1}^{h(j)} \Delta_{t-r} \\ \phi_t^j &\sim \mathcal{N}(\phi_t^j \mid \phi_{t-h(j)}^j, h(j)\sigma^2, 0.5 < \phi_t^j < 1) \qquad (3.14)\end{aligned}$$

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

Next, we draw samples $\{\phi_t^{j,i}\}_{i=1}^N$ from the distribution given above in (3.14). We weight these samples by their corresponding predicted probability (shown in (3.13)) given what has been observed up to time t:

$$b_t^{j,i} = p(\phi_t^j \mid Z_{t-h(j)}^j)\phi_t^{j,i} \tag{3.15}$$

For each j, we sort the weighted samples $b_t^{j,i}$ in ascending order and find their 95th percentile. This represents the value of the upper 95th confidence interval for labeler j. We then apply the IEThresh method from [5] by selecting all labelers whose 95th percentile is higher than a predefined threshold T. We denote the set of selected labelers at time t by $J(t)$. This selection allows us to take the cumulative variance into account and select the labelers with potentially high accuracies. We tuned the parameter T on a separate dataset not reported in this paper. It can be further adjusted according to the budget allocated to label acqusition or to prevent unnecessary sampling especially when the number of labelers is large.

After the committee $J(t)$ of selected labelers is identified, we observe their outputs z_t^j and update the weights $w_t^{j,i}$ using (3.12) and update the posterior $p(\phi_t^j \mid Z_t^j)$ using (3.10). Relying on the posterior accuracy distribution, we compute the estimated expected accuracy as follows:

$$E_{p(\phi_t^j|Z_t^j)}[\phi_t^j] = \sum_{i=1}^{N} p(\phi_t^{j,i} \mid Z_t^j)\phi_t^{j,i} \tag{3.16}$$

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

$$\begin{aligned}\hat{y}_t^{\text{map}} &= \arg\max_{y\in\mathcal{Y}} P(y \mid Z_t^{J(t)}) \\ &= \arg\max_{y\in\mathcal{Y}} \hat{P}_{t-1}(y) \prod_{j\in J(t)} p_{E[\phi_t^j]}(z_t^j \mid y) \\ &= \arg\max_{y\in\mathcal{Y}} \hat{P}_{t-1}(y) \prod_{j\in J(t)} \\ &\qquad E[\phi_t^j]^{I(z_t^j=y)}(1-E[\phi_t^j])^{I(z_t^j\neq y)}\end{aligned} \tag{3.17}$$

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

$$\hat{P}_t(y=1) = \frac{1}{t}\sum_{s=1}^{t} \hat{y}_s^{\text{map}} \tag{3.18}$$

1. Sample from the initial distribution $\phi_0^i \sim p(\phi_0)$ for $i = 1, \ldots, N$ and assign weights $w_0^i = \frac{1}{N}$
2. For $t = 1$, initialize $\hat{P}_1(y) = 0.5$ and run a single iteration of the basic filtering algorithm (Figure 2) for $j = 1, \ldots, K$.
3. Initialize $h(j) = 1$ and $Z_1^j = \{z_1^j\}\ \forall j = 1, \ldots, K$
4. For $t > 1$, and $i = 1, \ldots, N$
   - Compute $\hat{P}_t(y)$ acc. to (3.18).
   - Draw samples $\phi_t^{j,i}$ using (3.13) and estimate the expected accuracy via (3.16) for $j = 1, \ldots, K$.
   - Select the labelers $J(t)$ to be queried
   - For $j \in J(t)$
     - Update weights $\tilde{w}_t^{j,i} = p(z_t^j \mid \phi_t^{j,i})\tilde{w}_{t-h(j)}^{j,i}$
     - Normalize the weights $w_t^{j,i} = \frac{\tilde{w}_{t-h(j)}^{j,i}}{\sum_{l=1}^N \tilde{w}_{t-h(j)}^{j,l}}$ and compute $\hat{N}_{ej} = \frac{1}{\sum_{i=1}^N (w_t^{j,i})^2}$.
     - If $\hat{N}_{ej}$ is too small, then resample $\phi_t^{j,i} \sim \text{pmf}[\{w_t^j\}]$ and reassign $w_t^{j,i} = \frac{1}{N}$.
     - Update $h(j) = t$ and $Z_t^j = Z_{t-h(j)}^j \cup \{z_t^j\}$.
     - Update the posterior state density $p(\phi_t^j \mid Z_t^j) = \sum_{i=1}^N w_t^{j,i}\delta(\phi_t^j - \phi_t^{j,i})$.
5. Update $t = t + 1$ and go to step 4.

Figure 3: Outline of the SFilter algorithm.

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

See Figure 3 for an outline of our algorithm, which we name *SFilter*.

4 Evaluation

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

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

Table 1: Performance measurement of SFilter w.r.t increasing σ values.

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

Table 2: Robustness analysis of SFilter against small perturbations in estimated vs. true σ when the true σ is equal to 0.02. The analysis is performed over 500 instances drawn from $P(y = 1) = .75$

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

that controls the width of the step size in (3.1). We analyzed how σ affects the estimation process in terms of the overall quality of the integrated labels and the total labeling effort. We simulated 10 labelers with varying initial accuracies but with the same σ for 500 instances. The accuracy of each labeler evolves according to (3.2) and every selected labeler $j \in J_t$ outputs a noisy label z_t^j. We integrate these labels to predict the true label via (3.17). Table 1 shows the ratio of correct labels predicted and the number of total labeler queries with the corresponding σ. %-correct represents the ratio of the correct labelings among all instances $m = 500$, i.e. $\% - \text{correct} = \frac{\sum_{t=1}^{m} I(\hat{y}_t^{\text{map}} = y_t^{\text{true}})}{m}$ where y_t^{true} is drawn from $P(y = 1) = 0.75$. Among the 10 labelers the highest average accuracy is 0.81 whereas the percentage of correct labels integrated by SFilter is significantly higher, i.e. 85%. As the variance increases, the total number of queries also increases and there is a slight decay in the ratio of correct labelings. The relatively rapid change of labeler qualities leads to more extensive exploration to catch any drifts without compromising the overall performance too much. We have also tested the robustness of the proposed algorithm against small perturbations in σ. We repeated the above analysis using $\sigma = 0.02$ to generate the accuracy drift, but executed SFilter using slightly different σ values in order to test perturbational robustness. Table 2 indicates that SFilter is robust to small noise in σ. The %-correct remains relatively stable as we add noise to the true σ while the total amount of labeling is only slightly larger.

Next, we analyzed how the variations among the individual labeler qualities affect the estimation and selection. The first experiment uses the same committee of labelers ($k = 10$) as above whose qualities are uniformly distributed between 0.5 and 1. The second experiment uses a committee where only a few good labelers exists (only 3 labelers with initial accuracy above 0.8 and the rest below 0.65). We used a fixed $\sigma = 0.02$ for both cases. Table 3 shows the effectiveness of SFilter to make correct label predictions with moderate labeling effort in both cases. The last column in Table 3 shows the average estimated marginal label distribution $\hat{P}(y = 1)$ where the true marginal is $P(y = 1) = 0.75$. Our algorithm is very effective for estimating the unknown label distribution and also for exploiting the labelers only when they are highly reliable as shown in Figure 4. Figure 4 displays the true source accuracy over time and the times the source is selected by SFilter (denoted by red circles) and when otherwise (denoted by blue circles). We only report a representative subset out of 10 total sources. Note that SFilter selects the highly reliable sources in this skewed set at the beginning. When these sources become less reliable, SFilter finds other sources that have become more favorable over time (i.e. lower left corner in Figure 4). Moreover, sources that have relatively low quality throughout the entire time are hardly selected by SFilter except a few exploration attempts. This is a desirable behaviour of the algorithm and certainly reduces the labeling effort without hurting overall performance even with skewed distributions of source accuracies.

Table 3: Performance measurement of SFilter for various quality labelers. Uniform denotes the labelers' initial accuracies are uniformly distributed. Skewed denotes there are only a few good labelers while the majority are highly unreliable.

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

We further challenged SFilter with a pathological example to test its robustness against how the labelers' accuracies change. We simulated $k = 10$ labelers with initial accuracies ranging from 0.51 to 0.94. All the labelers except the best (0.94 accuracy) and the worst (0.51 accuracy) follow a random walk with the step size drawn from $\mathcal{N}(0, \sigma^2)$ where $\sigma = 0.02$. The best labeler consistently gets worse and the worst consistently improves, both staying within 0.5 and 1 range. As before, $P(y = 1) = 0.75$. Figure 5 displays the accuracy change of the two labelers with respect to time. The red dots indicate the times that the corresponding labeler is se-

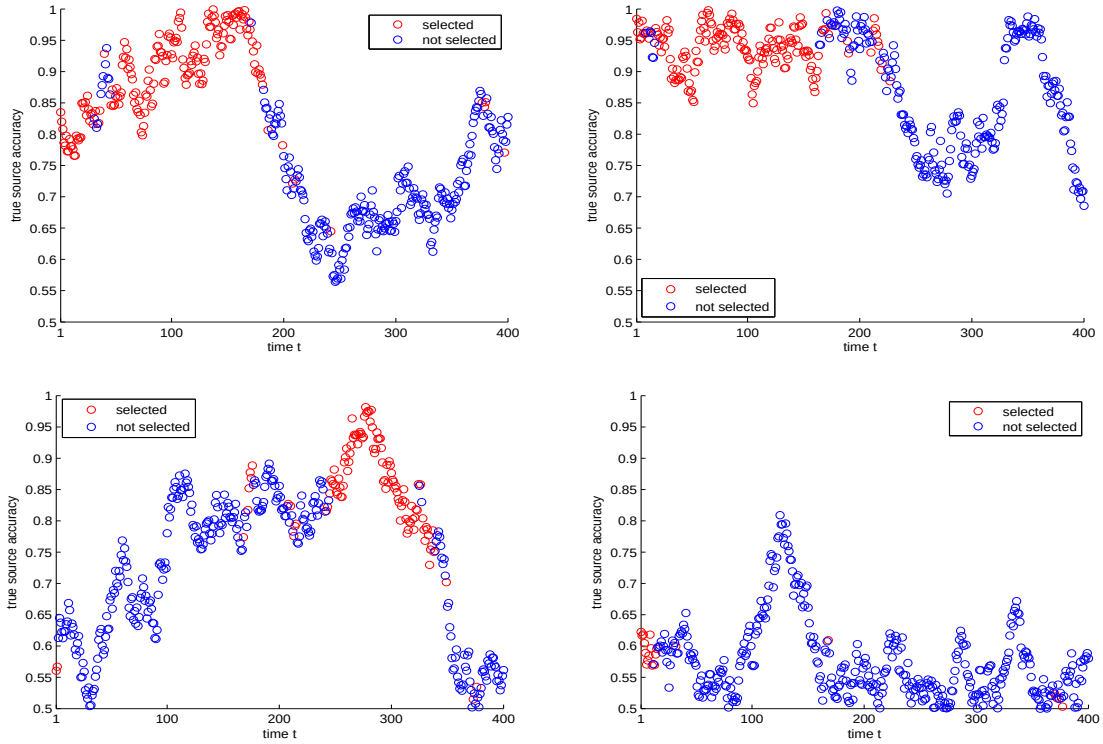

Figure 4: SFilter selects the sources when they are relatively highly accurate even in the skewed case where there are only a few good sources to begin with. SFilter also detects when the initially bad sources become more favorable later and begins exploiting them.

lected and the blue dots indicate otherwise. Clearly, SFilter is able to detect the consistent change in the accuracies. We also note that the most frequently chosen labelers are the ones with relatively high average accuracies. There are occasional exploration attempts at the low accuracies, but the algorithm detects this and is able to recover quickly. As a result, the algorithm chooses any undesirable labeler(s) with very low frequency. Please note that there are times when neither 4 of labelers is selected. This corresponds to the situations where the remaining labelers are selected.

Additionally, we tested the ability of SFilter to track the true accuracy. We tested SFilter using 4 labelers with the rate of change $\sigma = 0.01$ on 1000 instances drawn from a true label distribution of $P(y = 1) = 0.8$. We used 5000 particles and we constrained SFilter to choose all 4 labelers per instance (only for this experiment) to monitor the true and the estimated accuracy of each labeler at every time step. Figure 6 demonstrates how the estimate of the marginal label distribution $\hat{P}(y = 1)$ improves over time. Figure 7 compares the true and the estimated accuracy of each labeler. The dotted blue line in each graph corresponds to the

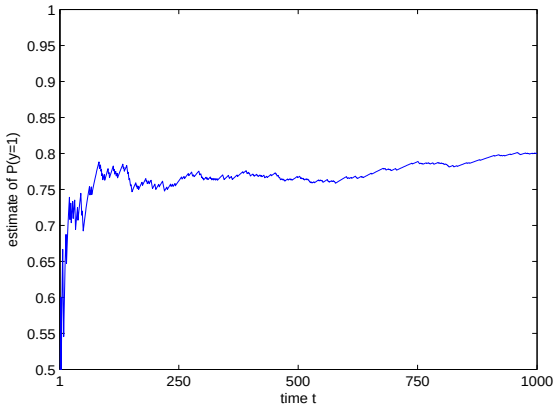

Figure 6: SFilter's estimate of the true marginal label distribution $P(y = 1) = 0.8$ improves over time. In fact, it converges to the true distribution with more samples.

estimated expected accuracy while the solid red line corresponds to the true accuracy. The black line in each graph shows the result of using maximum likelihood estimation to infer the accuracy assuming accuracy is sta-

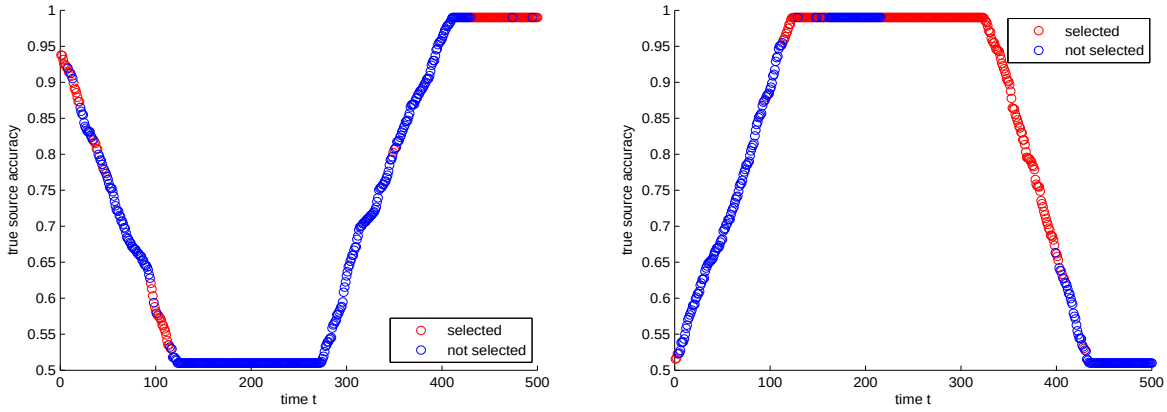

Figure 5: SFilter is able to detect the consistent increase or decrease in any quality and adopts quickly by choosing the labelers when they are reliable, though with occasional exploration attempts.

tionary. This inference technique proposed in [6] shows very promising results on various test scenarios. However, it is designed to estimate stationary source accuracies. SFilter, on the other hand, manages to track the trends in the true accuracy quite well, sometimes with a lag. This is remarkable since neither the initial qualities nor the true label distribution are known in advance.

Next, we tested SFilter in terms of its effectiveness to create a high quality labeled data for training a classifier. We took 4 benchmark datasets from the UCI repository[2]. We simulated $k = 10$ different labelers and executed SFilter on each dataset with these labelers and integrated their output for a final labeling. Later, we trained a logistic regression classifier on this labeled set and tested the resulting classifier on a separate held-out data. The gold standard labels for the held-out set are known but used solely to test the classifier not to adjust the estimation model in any way. We compared the proposed approach with two strong baselines. One is majority voting (denoted as *Majority*) where we used all labelers per instance and assigned the majority vote as the integrated label. The other is a technique proposed in [5] (denoted as *IEThresh*). IEThresh selects a subset of labelers based on an interval estimation procedure and takes the majority vote of only the selected labelers. We report on the held-out set the accuracy of the classifier trained on the data labeled by each method. We report the classifier performance with respect to the number of queries. Hence, the corresponding size of the labeled examples differs for each method since each method uses different amount of labeling per instance, i.e. majority voting uses all annotators to label each instance. The total training size is fixed at 500 examples for each dataset. The other properties of the datasets are given in Table 4[3].

Table 4: Properties of the UCI datasets. All are binary classification problems.

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

Figure 8 displays the performance of our algorithm SFilter and two baselines on four UCI datasets. The initial labeler accuracies range from as low as 0.53 to as high as 0.82. SFilter is significantly superior over the entire operating range; i.e. from small to large numbers of labeler queries. The differences are statistically significant based on a two-sided paired t-test ($p < 0.001$). The horizontal black line in each plot represents the test accuracy of a classifier trained with the gold standard labels. SFilter eventually reaches the gold standard level, suggesting that it generates a clean training data with minimal noise. Majority voting is the worst performer since it uses all labelers for every single instance; thus makes a larger number of labeling requests even for a small number of instances. IEThresh is the second best performer after SFilter. However, neither baselines take the time-varying accuracy into account and wastes labeling effort due to low quality labelers. Our method adapts to the change in labeler accuracies and filters the unreliable ones leading to a

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

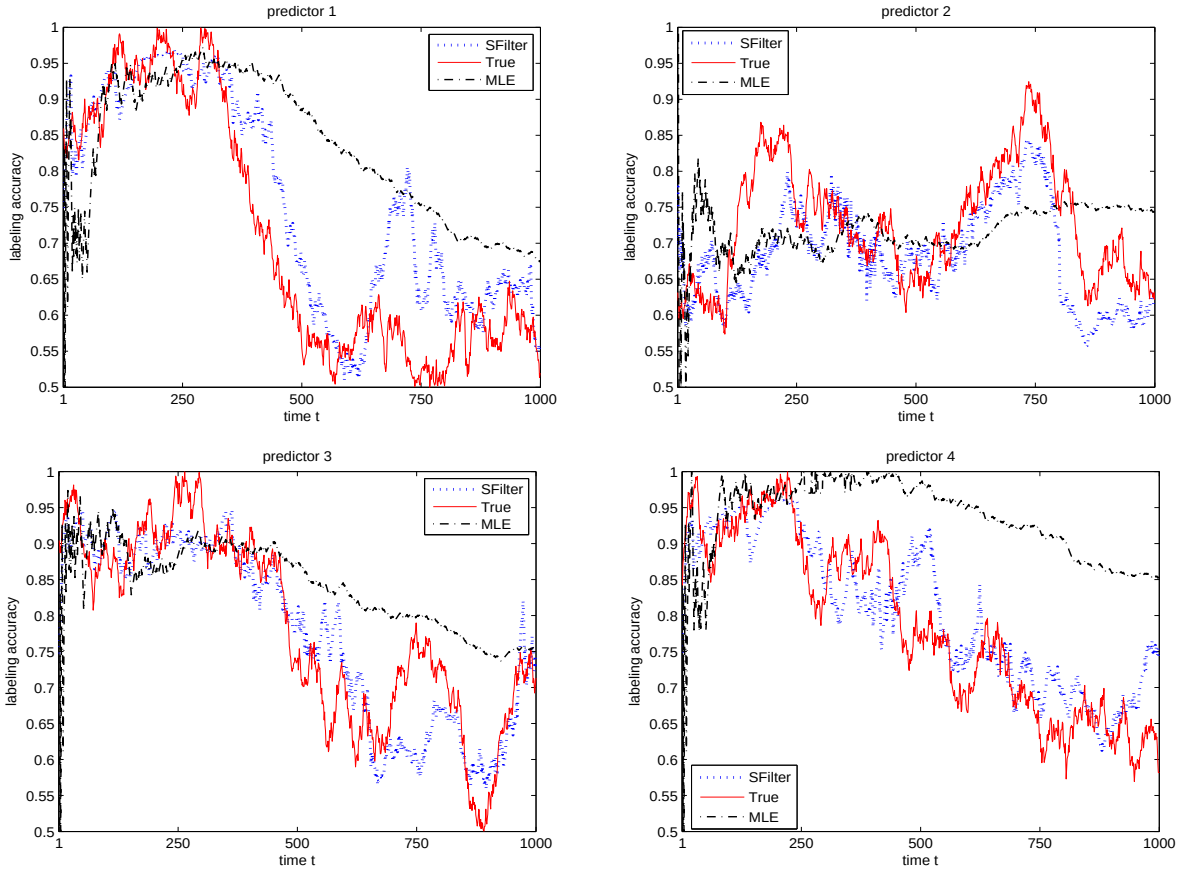

Figure 7: Shows how SFilter tracks the true labeler accuracy. Each graph corresponds to a different labeler. The solid red line indicates the true accuracy of the labeler whereas the dotted blue line shows the expected accuracy estimated by SFilter. This is the result of a single run, though highly typical. SFilter is able to track the tendency of the true accuracy quite well with occasional temporal lags.

more effective estimation.

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

To study this phenomenon, we evaluated the performance of SFilter on a face recognition dataset introduced in [8]. The total size of the face data is 2500 with 400 dimensions and $P(y = face) = 0.66$. We randomly divided the whole dataset into 3 mutually exclusive subsets. One contains 100 labeled images to train the classifier. The other subset contains 500 unlabeled images to test the classifiers. The last subset is held out for validation. We trained 5 linear SVM classifiers as follows. For each classifier, we used a separate randomly chosen subset of 100 images as the initial training set. We systematically increased each training set size to 100, by adding random label noise so that the classifiers do not necessarily improve with more data. The average accuracy of classifiers are $\{0.70, 0.73, 0.78, 0.83, 0.84\}$. We

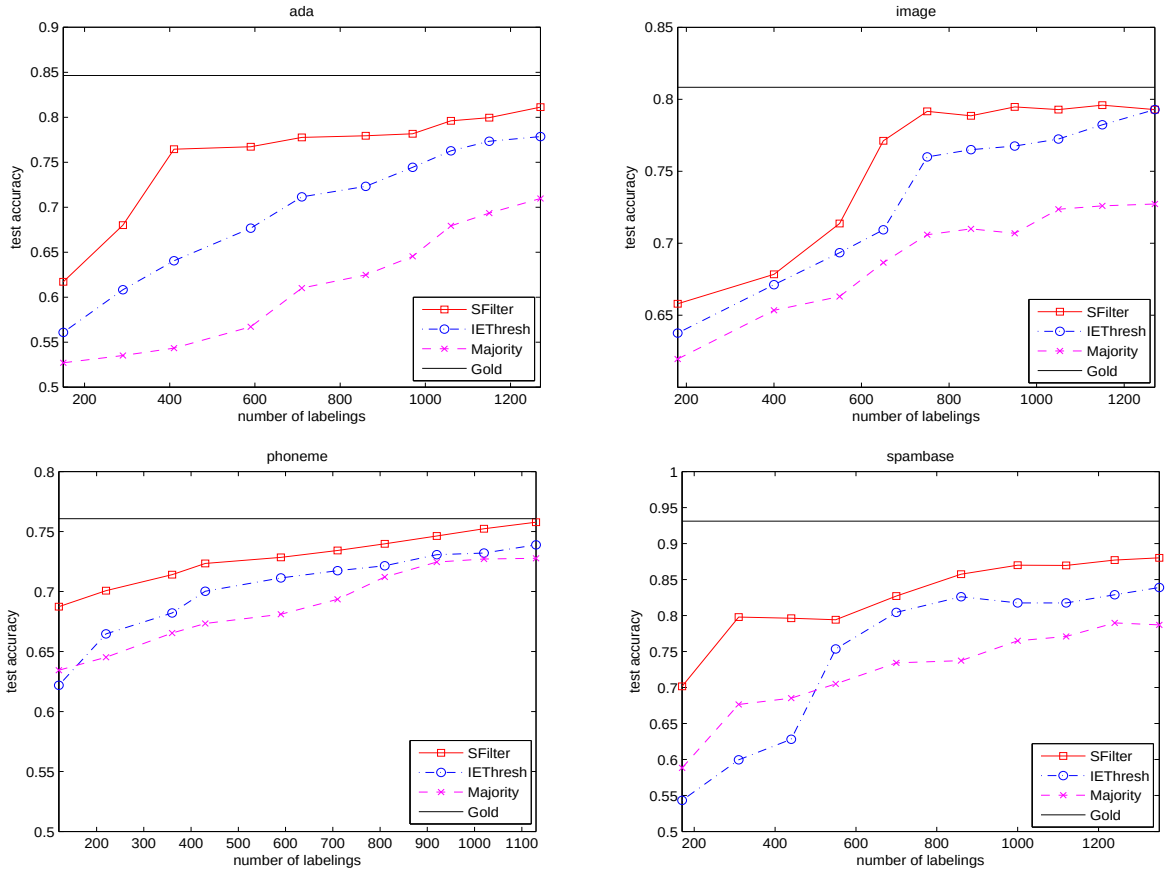

Figure 8: Measuring predicted label quality by training and testing a classifier on 4 UCI datasets by all three methods. The y-axis indicates the accuracy of the classifier on the test set, and the x-axis indicates the number of labeler queries. The horizontal line shows the performance of a classifier trained on the gold standard labels.

executed SFilter, IEThresh and Majority voting on 500 test images to predict their labels. We then trained another classifier on the predicted labels by each method. Figure 9 compares the performance of SFilter against Majority voting. SFilter significantly outperforms other baselines in this task and reaches a level of performance close to that of using gold standard labels. Again, gold labels are solely used to measure the performance of classifiers, and not for learning.

5 Discussion and Future Work

This paper addresses the challenging problem of estimating time-varying accuracy of multiple sources and selecting the most reliable ones for querying. The proposed framework constantly monitors and estimates the expected behaviour of each predictor while simultaneously choosing the best possible predictors for labeling each instance. The framework relies on a Hidden Markov Model (HMM) structure to represent the unknown sequence of changing accuracies and the corresponding observed predictor outputs. We adopted the Bayesian particle filtering to make inference in such a dynamic system. Particle filters estimate the state distribution by sets of weighted samples and perform Bayes filter updates according to a sequential sampling procedure. The key advantage of particle filters is their capacity to represent arbitrary probability densities whereas other Bayesian filters such as Kalman filters can only handle unimodal distributions and hence not well-suitable for our problem. Furthermore, we have proposed a variant of the basic filtering method to infer the expected accuracy of each predictor and use it as a guide to decide which ones to query at each time step. The proposed approach, *SFilter*, chooses the potentially good predictors based on their expected accuracies at each time step. In general high quality labelers are queried more frequently than the low-quality labelers. Moreover, it tracks reliably the true labeling accuracy over time.

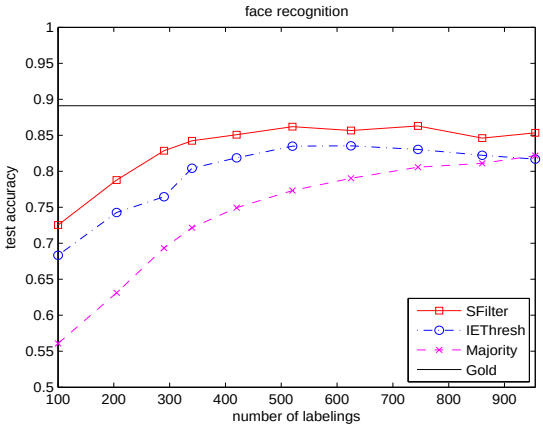

Figure 9: Results using 5 actual classifiers as labelers. The performance of each classifier varies over time. Their outputs on a test set are used to predict the true labels. A meta-classifier is trained using these labels and tested on a held-out set for performance evaluation. The predicted labels obtained by SFilter are significantly more effective in improving the data quality.

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

There are a number of potential future directions that open interesting new challenges. In this work, we assumed the rate of change (σ) of the state (or at least its bound) is known and it is the same for all predictors. It is likely that this assumption may not hold for some cases. Then, one should also estimate σ for each predictor to estimate the expected state at each time step. This is a harder problem but definitely a necessary one to solve to make the method applicable to a wider range of data mining scenarios. Another point is that in this work we deal with predictors whose accuracies change based on external factors such as fatigue, learning, etc. The existing framework can be formulated in a way to allow the accuracy to shift based on the instance such as the difficulty, ambigiuity, out-of-expertise, and so on. This introduces further challenges in terms of formulating the state transition density and the observation likelihood. We plan to investigate these extensions in the future.

References

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[2] Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[3] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for non-linear problems. In *IEEE Proceedings on Radar and Sonar Navigation*, 1999.

[4] O. Dekel and O. Shamir. Good learners for evil teachers. In *Proceedings of the 26th International Conference on Machine Learning*, June 2009.

[5] P. Donmez, J. G. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '09)*, pages 259–268, 2009.

[6] P. Donmez, G. Lebanon, and K. Balasubramanian. Unsupervised estimation of classification and regression error rates. Technical Report CMU-LTI-09-15, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA.

[7] A. Doucet and D. Crisan. A survey of convergence results on particle filtering for practitioners. *IEEE Transactions on Signal Processing*, 2002.

[8] T. Pham, M. Worring, and A. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23.

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

[10] B. Ripley. *Stochastic Simulation*. Wiley, New York, 1987.

[11] V. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pages 614–622, 2008.

[12] R. Snow, O'Connor, D. Jurafsky, and A. Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.