Journal of Machine Learning Research 13 (2012) 491-518 Submitted 4/11; Revised 12/11; Published 2/12

Eliminating Spammers and Ranking Annotators for Crowdsourced Labeling Tasks

Vikas C. Raykar VIKAS.RAYKAR@SIEMENS.COM
Shipeng Yu SHIPENG.YU@SIEMENS.COM
Siemens Healthcare
51 Valley Stream Parkway, E51
Malvern, PA 19355, USA

Editor: Ben Taskar

Abstract

With the advent of crowdsourcing services it has become quite cheap and reasonably effective to get a data set labeled by multiple annotators in a short amount of time. Various methods have been proposed to estimate the consensus labels by correcting for the bias of annotators with different kinds of expertise. Since we do not have control over the quality of the annotators, very often the annotations can be dominated by spammers, defined as annotators who assign labels randomly without actually looking at the instance. Spammers can make the cost of acquiring labels very expensive and can potentially degrade the quality of the final consensus labels. In this paper we propose an empirical Bayesian algorithm called SpEM that iteratively eliminates the spammers and estimates the consensus labels based only on the good annotators. The algorithm is motivated by defining a spammer score that can be used to rank the annotators. Experiments on simulated and real data show that the proposed approach is better than (or as good as) the earlier approaches in terms of the accuracy and uses a significantly smaller number of annotators.

Keywords: crowdsourcing, multiple annotators, ranking annotators, spammers

1. Introduction

Annotating a data set is one of the major bottlenecks in using supervised learning to build good predictive models. Getting a data set labeled by experts can be expensive and time consuming. With the advent of crowdsourcing services (Amazon's Mechanical Turk[1] being a prime example) it has become quite easy and inexpensive to acquire labels from a large number of annotators in a short amount of time (see Sheng et al. 2008, Snow et al. 2008, and Sorokin and Forsyth 2008 for some natural language processing and computer vision case studies). For example in AMT the *requesters* are able to pose tasks known as HITs (Human Intelligence Tasks). Workers (called *providers*) can then browse among existing tasks and complete them for a small monetary payment set by the requester.

A major drawback of most crowdsourcing services is that we do not have control over the quality of the annotators. The annotators usually come from a diverse pool including genuine experts, novices, biased annotators, malicious annotators, and spammers. Hence in order to get good quality labels requestors typically get each instance labeled by multiple annotators and these multiple annotations are then consolidated either using a simple majority voting or more sophisticated methods

1. Amazon's Mechanical Turk can be found at `https://www.mturk.com`.

that model and correct for the annotator biases (Dawid and Skene, 1979; Smyth et al., 1995; Raykar et al., 2009, 2010; Yan et al., 2010) and/or task complexity (Carpenter, 2008; Whitehill et al., 2009; Welinder et al., 2010).

In this paper we are interested in the situation where the annotations are dominated by *spammers*. In our context a spammer is a low quality annotator who assigns random labels (maybe because the annotator does not understand the labeling criteria, does not look at the instances when labeling, or maybe a bot pretending to be a human annotator). Spammers can significantly *increase the cost* of acquiring annotations (since they need to be paid) and at the same time *decrease the accuracy* of the final consensus labels. A mechanism to detect and eliminate spammers is a desirable feature for any crowdsourcing market place. For example one can give monetary bonuses to good annotators and deny payments to spammers. This paper makes two novel contributions:[2]

1. ***Spammer score to rank annotators*** The first contribution of this paper is to formalize the notion of a spammer for binary and categorical labels. More specifically we define a *scalar metric* which can be used to *rank the annotators*, with the spammers having a score close to zero and the good annotators having a score close to one. We summarize the multiple parameters corresponding to each annotator into a single score indicative of how spammer like the annotator is. While this metric was implicit for binary labels in earlier works (Dawid and Skene, 1979; Smyth et al., 1995; Carpenter, 2008; Raykar et al., 2009; Donmez et al., 2009) the extension to categorical labels is novel and is quite different for the error rate computed from the confusion rate matrix. An attempt to quantify the quality of the workers based on the confusion matrix was recently made by Ipeirotis et al. (2010) where they transformed the observed labels into posterior soft labels based on the estimated confusion matrix. While we obtain somewhat similar annotator rankings, we differ from this work in that our score is directly defined in terms of the annotator parameters. Having the score defined only in terms of the annotator parameters makes it easy to specify a prior for Bayesian approaches to eliminate spammers and consolidate annotations.

2. ***Algorithm to eliminate spammers*** The second contribution is that we propose an algorithm to consolidate annotations that eliminates spammers automatically. One of the commonly used strategy is to inject some items into the annotations *with known labels* (gold standard) and use them to evaluate the annotators and thus eliminate the spammers.[3] Typically we would like to detect the spammers with as few instances as possible and eliminate them from further annotations. In this work we propose an algorithm called SpEM that eliminates the spammers *without using any gold standard* and estimates the consensus ground truth based only on the good annotators. The same algorithm can also be used if some labels are also known.

   We build on the earlier works of Dawid and Skene (1979), Smyth et al. (1995), and Raykar et al. (2009, 2010) who proposed algorithms that correct for the annotator biases by estimating the annotator accuracy and the actual true label jointly. A simple strategy would be to use these algorithms to estimate the annotator parameters, detect and eliminate the spammers (as defined by our proposed spammer score) and refit the model with only the good annotators. However this approach is not a principled approach and might be hard to control (for example, how to define spammers and how many to remove, etc). The algorithm we propose is essentially a formalization of this strategy. Our final algorithm essentially repeats

2. A preliminary version of this paper (Raykar and Yu, 2011) mainly discussed the score to rank annotators.
3. This is the strategy used by CrowdFlower (`http://crowdflower.com/docs/gold`).

this, it *iteratively* eliminates the spammers and re-estimates the labels based only on the good annotators. A crucial element of our proposed algorithm is that we eliminate spammers by thresholding on a hyperparameter of the prior (automatically estimated from the data) rather than directly thresholding on the estimated spammer score.

The rest of the paper is organized as follows. In Section 2 we model the annotators in terms of the sensitivity and specificity for binary labels and extend it to categorical labels. Based on this model the notion of a spammer is formalized in Section 3. In Section 4 we propose a Bayesian point estimate by using a prior (Section 4.2) derived from the proposed spammer score designed to favor spammer detection. This is essentially a modification of the Expectation Maximization (EM) algorithm proposed by Dawid and Skene (1979), Smyth et al. (1995), and Raykar et al. (2009, 2010). The hyperparameters of this prior are estimated via an empirical Bayesian method in Section 5 leading to the proposed SpEM algorithm (Algorithm 1) that iteratively eliminates the spammers and re-estimates the ground truth based only on the good annotators. In Section 6 we discuss this algorithm in context of other methods and also propose a few extensions. in Section 7 we extend the same ideas to categorical labels. In Section 8 we extensively validate our approach using both simulated data and real data collected using AMT and other sources from different domains.

2. Annotator Model

An annotator provides a noisy version of the true label. Let $y_i^j \in \{0,1\}$ be the label assigned to the i^{th} instance by the j^{th} annotator, and let y_i be the actual (unobserved) label. Following the approach of Raykar et al. (2009, 2010) we model the accuracy of the annotator separately on the positive and the negative examples. If the true label is one, the *sensitivity* (true positive rate) for the j^{th} annotator is defined as the probability that the annotator labels it as one.

$$\alpha^j := \Pr[y_i^j = 1 | y_i = 1].$$

On the other hand, if the true label is zero, the *specificity* (1−false positive rate) is defined as the probability that the annotator labels it as zero.

$$\beta^j := \Pr[y_i^j = 0 | y_i = 0].$$

With this model we have implicitly assumed that α^j and β^j do not depend on the instance. Extensions of this basic model have been proposed to include item level difficulty (Carpenter, 2008; Whitehill et al., 2009) and also to explicitly model the annotator performance based on the instance feature vector (Yan et al., 2010). In principle the proposed algorithm can be extended to these kind of complicated models (with more parameters), however for simplicity we use the basic model proposed in Raykar et al. (2009, 2010) in our formulation.

The same model can be extended to categorical labels. Suppose there are $C \geq 2$ categories. We introduce a multinomial parameter $\boldsymbol{\alpha}_c^j = (\alpha_{c1}^j, \ldots, \alpha_{cC}^j)$ for each annotator, where

$$\alpha_{ck}^j := \Pr[y_i^j = k | y_i = c], \qquad \sum_{k=1}^{C} \alpha_{ck}^j = 1.$$

The term α_{ck}^j denotes the probability that annotator j assigns class k to an instance given the true class is c. When $C = 2$, α_{11}^j and α_{00}^j are sensitivity and specificity, respectively.

3. Who is a Spammer? Score to Rank Annotators

Intuitively, *a spammer assigns labels randomly*, maybe because the annotator does not understand the labeling criteria, does not look at the instances when labeling, or maybe a bot pretending to be a human annotator. More precisely an annotator is a spammer if the probability of observed label y_i^j being one given the true label y_i is independent of the true label, that is,

$$\Pr[y_i^j = 1 | y_i] = \Pr[y_i^j = 1]. \tag{1}$$

This means that the annotator is assigning labels randomly by flipping a coin with bias $\Pr[y_i^j = 1]$ without actually looking at the data. Equivalently (1) can be written as

$$\begin{aligned} \Pr[y_i^j = 1 | y_i = 1] &= \Pr[y_i^j = 1 | y_i = 0], \\ \alpha^j &= 1 - \beta^j. \end{aligned} \tag{2}$$

Hence in the context of the annotator model defined in Section 2, a spammer is an annotator for whom

$$\alpha^j + \beta^j - 1 = 0.$$

This corresponds to the diagonal line on the Receiver Operating Characteristic (ROC) plot (see Figure 1).[4] If $\alpha^j + \beta^j - 1 < 0$ then the annotator lies below the diagonal line and is a malicious annotator who flips the labels. Note that a malicious annotator has discriminatory power if we can detect them and flip their labels. In fact the methods proposed in Dawid and Skene (1979) and Raykar et al. (2010) can automatically flip the labels for the malicious annotators. Hence we define the spammer score for an annotator as

$$\mathcal{S}^j = (\alpha^j + \beta^j - 1)^2. \tag{3}$$

An annotator is a spammer if $\mathcal{S}^j$ is close to zero. Good annotators have $\mathcal{S}^j > 0$ while a perfect annotator has $\mathcal{S}^j = 1$.

Another interpretation of a spammer can be seen from the log odds. Using Bayes' rule the posterior log-odds can be written as

$$\log \frac{\Pr[y_i = 1 | y_i^j]}{\Pr[y_i = 0 | y_i^j]} = \log \frac{\Pr[y_i^j | y_i = 1]}{\Pr[y_i^j | y_i = 0]} + \log \frac{p}{1-p},$$

where $p := \Pr[y_i = 1]$ is the prevalence of the positive class. If an annotator is a spammer (that is (2) holds) then

$$\log \frac{\Pr[y_i = 1 | y_i^j]}{\Pr[y_i = 0 | y_i^j]} = \log \frac{p}{1-p}.$$

Essentially the annotator provides no information in updating the posterior log-odds and hence does not contribute to the estimation of the actual true label.

4. Note that $(\alpha^j + \beta^j)/2$ is equal to the area shown in the plot and can be considered as a non-parametric approximation to the area under the ROC curve (AUC) based on one observed point $(1 - \beta^j, \alpha^j)$. It is also equal to the Balanced Classification Rate (BCR). So a spammer can also be defined as having BCR or AUC equal to 0.5. Another way to think about this is that instead of using sensitivity and specificity we can re-parameterize an annotator in terms of an accuracy parameter ($(\alpha^j + \beta^j)/2$) and a bias parameter ($(\alpha^j - \beta^j)/2$). A spammer is an annotator with accuracy equal to 0.5. The biased ($\alpha^j - \beta^j$ is large) or malicious annotators ($\alpha^j + \beta^j < 1$) (see Figure 1) are also sometimes called the spammers since they can potentially degrade the consensus labels, but in this paper we do not focus on them, since their annotations can be calibrated or reversed by the EM algorithm.

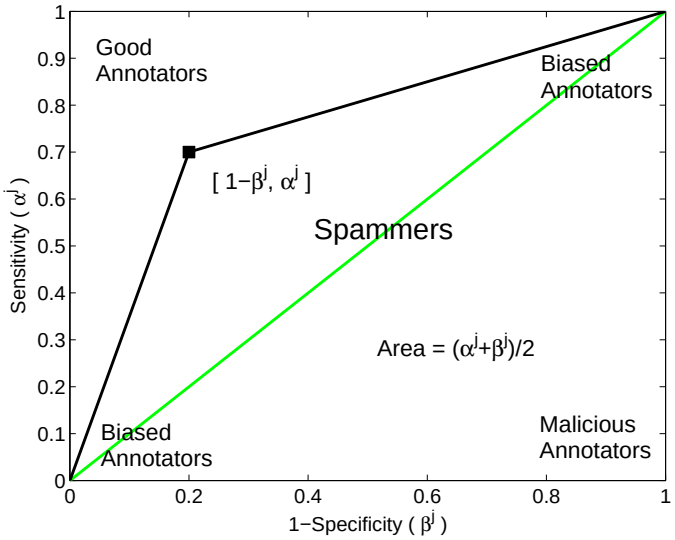


Figure 1: For binary labels each annotator is modeled by his/her sensitivity and specificity. A spammer lies on the diagonal line (that is, $\alpha^j = 1 - \beta^j$) on this ROC plot.

3.1 Accuracy

This notion of a spammer is quite different from that of the *accuracy* of an annotator. An annotator with high accuracy is a good annotator but one with low accuracy is not necessarily a spammer. The accuracy of the j^{th} annotator is computed as

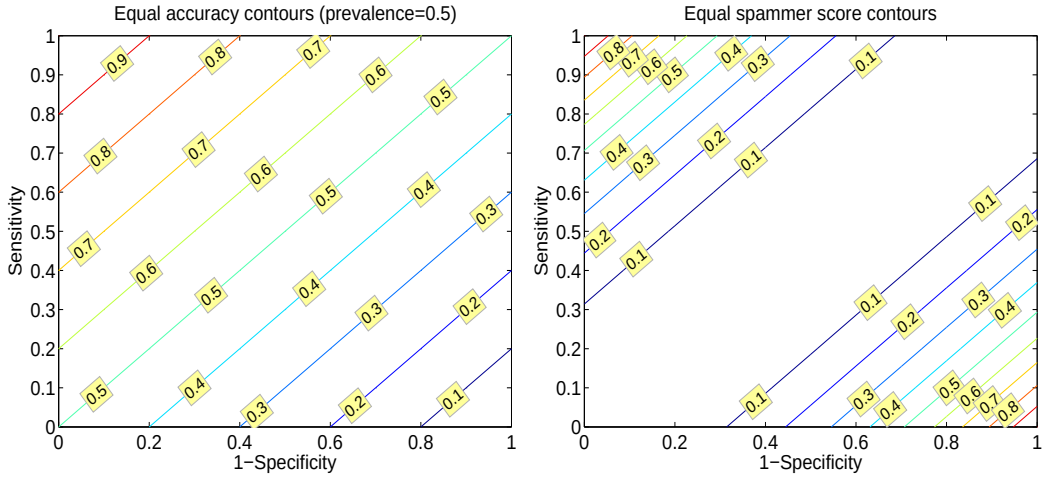
$$\text{Accuracy}^j = \Pr[y_i^j = y_i] = \sum_{k=0}^{1} \Pr[y_i^j = 1 | y_i = k] \Pr[y_i = k] = \alpha^j p + \beta^j (1-p), \tag{4}$$

where $p := \Pr[y_i = 1]$ is the prevalence of the positive class. Note that accuracy depends on prevalence. Our proposed spammer score does not depend on prevalence and essentially quantifies the annotator's inherent discriminatory power. Figure 2(a) shows the contours of equal accuracy on the ROC plot. Note that annotators below the diagonal line (malicious annotators) have low accuracy. The malicious annotators flip their labels and as such are not spammers if we can detect them and then correct for the flipping. In fact the EM algorithms (Dawid and Skene, 1979; Raykar et al., 2010) can correctly flip the labels for the malicious annotators and hence they should not be treated as spammers. Figure 2(b) also shows the contours of equal score for our proposed score and it can be seen that the malicious annotators have a high score and only annotators along the diagonal have a low score (spammers).

3.2 Categorical Labels

We now extend the notion of spammers to categorial labels. As earlier a spammer assigns labels randomly, that is,

$$\Pr[y_i^j = k | y_i] = \Pr[y_i^j = k], \forall k.$$



(a) Accuracy (b) Spammer score

Figure 2: (a) Contours of equal accuracy (4) and (b) equal spammer score (3).

This is equivalent to $\Pr[y_i^j = k | y_i = c] = \Pr[y_i^j = k | y_i = c'], \forall c, c', k = 1, \ldots, C$, which means knowing the true class label being c or c' does not change the probability of the annotator's assigned label. This indicates that the annotator j is a spammer if

$$\alpha_{ck}^j = \alpha_{c'k}^j, \forall c, c', k = 1, \ldots, C. \tag{5}$$

Let $\mathbf{A}^j$ be the $C \times C$ confusion rate matrix with entries $[\mathbf{A}^j]_{ck} = \alpha_{ck}$, a spammer would have all the rows of $\mathbf{A}^j$ equal to one another, for example, an annotator with a confusion matrix $\mathbf{A}^j = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.50 & 0.25 & 0.25 \\ 0.50 & 0.25 & 0.25 \end{bmatrix}$, is a spammer for a three class categorical annotation problem. Essentially $\mathbf{A}^j$ is a rank one matrix of the form $\mathbf{A}^j = \mathbf{e}\mathbf{v}_j^\top$, for some column vector $\mathbf{v}_j \in \mathbb{R}^C$ that satisfies $\mathbf{v}_j^\top \mathbf{e} = 1$, where $\mathbf{e}$ is column vector of ones. In the binary case we had this natural notion of spammer as an annotator for whom $\alpha^j + \beta^j - 1$ was close to zero. One natural way to summarize (5) would be in terms of the distance (Frobenius norm) of the confusion matrix to the closest rank one approximation, that is,

$$\mathcal{S}^j := \|\mathbf{A}^j - \mathbf{e}\hat{\mathbf{v}}_j^\top\|_F^2, \tag{6}$$

where $\hat{\mathbf{v}}_j$ solves

$$\hat{\mathbf{v}}_j = \arg\min_{\mathbf{v}_j} \|\mathbf{A}^j - \mathbf{e}\mathbf{v}_j^\top\|_F^2 \qquad \text{subject to} \quad \mathbf{v}_j^\top \mathbf{e} = 1. \tag{7}$$

Solving (7) yields $\hat{\mathbf{v}}_j = (1/C)\mathbf{A}^{j\top}\mathbf{e}$, which is the mean of the rows of $\mathbf{A}^j$. Then from (6) we have

$$\mathcal{S}^j = \left\| \left(\mathbf{I} - \frac{1}{C}\mathbf{e}\mathbf{e}^\top \right) \mathbf{A}^j \right\|_F^2 = \frac{1}{C} \sum_{c<c'} \sum_k (\alpha_{ck}^j - \alpha_{c'k}^j)^2.$$

This is equivalent to subtracting the mean row from each row of the confusion matrix and then summing up the squares of all the entries. So a spammer is an annotator for whom $\mathcal{S}^j$ is close to

zero. A perfect annotator has $\mathcal{S}^j = C-1$. We normalize this score to lie between 0 and 1.

$$\mathcal{S}^j = \frac{1}{C(C-1)} \sum_{c<c'} \sum_k (\alpha^j_{ck} - \alpha^j_{c'k})^2$$

When $C = 2$ this is equivalent to the score proposed earlier for binary labels.

4. Algorithm to Consolidate Multiple Annotations

Using the spammer score proposed in the earlier section to define a prior we describe an empirical Bayesian algorithm to consolidate the multiple annotations and eliminate the spammers simultaneously. For ease of exposition we first start with binary labels and later extend it to categorical labels in Section 7.

4.1 Likelihood

Let N be the number of instances and M be the number annotators. Let $\mathcal{D} = \{y_i^1, \ldots, y_i^M\}_{i=1}^N$ be the observed annotations from the M annotators, and let $p = \Pr[y_i = 1]$ be the prevalence of the positive class. Assuming the instances are independent, the likelihood of the parameters $\boldsymbol{\theta} = [\alpha^1, \beta^1, \ldots, \alpha^M, \beta^M, p]$ given the observations $\mathcal{D}$ can be factored as $\Pr[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^N \Pr[y_i^1, \ldots, y_i^M|\boldsymbol{\theta}]$. Under the assumption that the annotation labels $y_i^1, \ldots, y_i^M$ are independent given the true label y_i, the log likelihood can be written as

$$\log \Pr[\mathcal{D}|\boldsymbol{\theta}] = \sum_{i=1}^N \log \sum_{y_i=0}^1 \prod_{j=1}^M \Pr[y_i^j|y_i, \boldsymbol{\theta}] \cdot \Pr[y_i|\boldsymbol{\theta}] = \sum_{i=1}^N \log \left[a_i p + b_i(1-p)\right], \tag{8}$$

where we denote

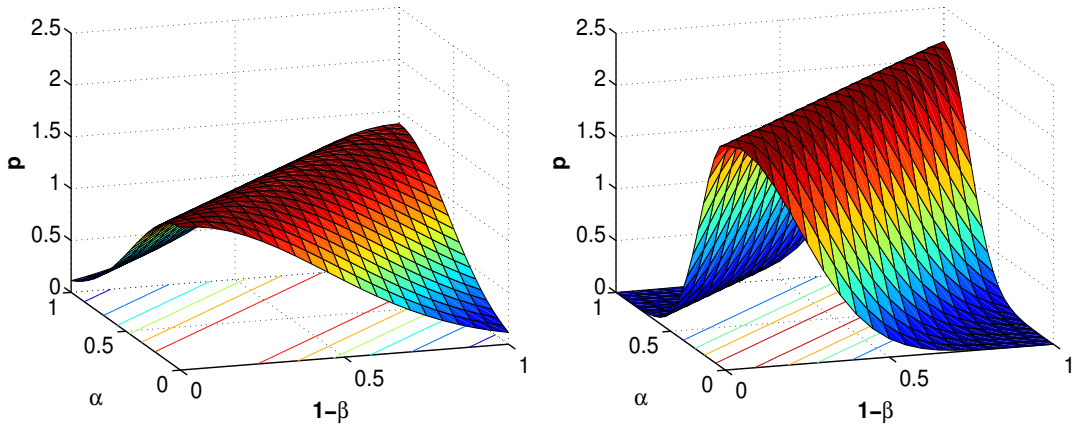
$$a_i = \prod_{j=1}^M \Pr[y_i^j|y_i = 1, \alpha^j] = \prod_{j=1}^M [\alpha^j]^{y_i^j} [1-\alpha^j]^{1-y_i^j},$$
$$b_i = \prod_{j=1}^M \Pr[y_i^j|y_i = 0, \beta^j] = \prod_{j=1}^M [\beta^j]^{1-y_i^j} [1-\beta^j]^{y_i^j}.$$

This log likelihood can be efficiently maximized by the Expectation Maximization (EM) algorithm (Dempster et al., 1977) leading to the iterative algorithm proposed in the earlier works (Dawid and Skene, 1979; Smyth et al., 1995; Raykar et al., 2010).

4.2 Automatic Spammer Detection Prior

Several authors have proposed a Bayesian approach by imposing a prior on the parameters (Raykar et al., 2009; Carpenter, 2008). For example, Raykar et al. (2009) assigned a beta prior for each α^j and β^j independently. Since we are interested in the situation when the annotations are mostly dominated by spammers, based on the score $\mathcal{S}^j$ (3) derived earlier we propose a prior called *Automatic Spammer Detection* (ASD) prior which favors the spammers. Specifically we assign the following prior to the pair $\{\alpha^j, \beta^j\}$ with a separate precision parameter $\lambda^j > 0$ (hyperparameter) for each annotator:

$$\Pr[\alpha^j, \beta^j|\lambda^j] = \frac{1}{N(\lambda^j)} \exp\left(-\frac{\lambda^j(\alpha^j + \beta^j - 1)^2}{2}\right). \tag{9}$$



(a) $\lambda = 5$ (b) $\lambda = 20$

Figure 3: The proposed Automatic Spammer Detection prior (9) for different values of λ^j.

where the normalization term N is given by (see Appendix A)

$$N(\lambda^j) \;=\; \int_0^1 \int_0^1 \exp\left(-\frac{\lambda^j(\alpha^j+\beta^j-1)^2}{2}\right) d\alpha^j d\beta^j = \sqrt{\frac{2\pi}{\lambda^j}}\left(\frac{2}{\sqrt{\lambda^j}}\int_0^{\sqrt{\lambda^j}} \Phi(t)dt - 1\right),$$

where Φ is the Gaussian cumulative distribution function. This prior is effectively a truncated Gaussian on $\alpha^j+\beta^j-1$ with mean zero and variance $1/\lambda^j$. Figure 3 illustrates the prior for two different values of the precision parameter. When λ^j is large the prior is sharply peaked along the diagonal corresponding to the spammers on the ROC plot.

We also assume that the ASD priors for each annotator are independent. For sake of completeness we further assume a beta prior for the prevalence, that is, $\text{Beta}(p|p_1,p_2)$. Denote $\boldsymbol{\lambda} = [\lambda^1,\ldots,\lambda^M,p_1,p_2]$, we have

$$\Pr[\boldsymbol{\theta}|\boldsymbol{\lambda}] = \text{Beta}(p|p_1,p_2)\prod_{j=1}^{M}\Pr[\alpha^j,\beta^j|\lambda^j]. \tag{10}$$

4.3 Maximum-a-posteriori Estimate Via EM Algorithm

Given the log likelihood (8) and the prior (10), the task is to estimate the parameters $\boldsymbol{\theta} = [\alpha^1,\beta^1,\ldots,\alpha^M,\beta^M,p]$. The maximum-a-posteriori (MAP) estimator is found by maximizing the log-posterior, that is,

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}}\{\ln\Pr[\mathcal{D}|\boldsymbol{\theta}] + \ln\Pr[\boldsymbol{\theta}]\}.$$

An EM algorithm can be derived for MAP estimation by relying on the interpretation of Neal and Hinton (1998) which is an efficient iterative procedure to compute the solution in presence of missing/hidden data. We will use the unknown hidden true label $\boldsymbol{y} = [y_1,\ldots,y_N]$ as the missing data. The complete data log-likelihood can be written as

$$\log\Pr[\mathcal{D},\boldsymbol{y}|\boldsymbol{\theta}] = \sum_{i=1}^{N}\Big[y_i\log pa_i + (1-y_i)\log(1-p)b_i\Big].$$

Each iteration of the EM algorithm consists of two steps: an Expectation(E)-step and a Maximization(M)-step. The M-step involves maximization of a lower bound on the log-posterior that is refined in each iteration by the E-step.

E-step: Given the observation $\mathcal{D}$ and the current estimate of the model parameters $\boldsymbol{\theta}$, the conditional expectation (which is a lower bound on the true likelihood) is computed as

$$\mathbb{E}\left\{\log \Pr[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\right\} = \sum_{i=1}^{N} \Big[\mu_i \log p a_i + (1-\mu_i)\log(1-p) b_i\Big],$$

where the expectation is with respect to $\Pr[\boldsymbol{y}|\mathcal{D},\boldsymbol{\theta}]$, and $\mu_i = \Pr[y_i = 1|y_i^1,\ldots,y_i^M,\boldsymbol{\theta}]$ is the expected label for y_i conditioned on the observed annotations and the model parameters. Using Bayes theorem we can compute

$$\mu_i \propto \Pr[y_i^1,\ldots,y_i^M|y_i=1,\boldsymbol{\theta}]\cdot\Pr[y_i=1|\boldsymbol{\theta}] = \frac{a_i p}{a_i p + b_i(1-p)}. \tag{11}$$

M-step: Based on the current estimate μ_i and the observations $\mathcal{D}$, we can estimate p by maximizing the lower bound on the log posterior, $\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}$, where

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta}} &= \mathbb{E}\left\{\log \Pr[\mathcal{D}, \boldsymbol{y}|\theta]\right\} + \log \Pr[\boldsymbol{\theta}|\boldsymbol{\lambda}] \\ &= \sum_{i=1}^{N}\Big[\mu_i \log p a_i + (1-\mu_i)\log(1-p) b_i\Big] + \log \text{Beta}(p|p_1,p_2) \\ &\quad - \sum_{j=1}^{M} \frac{\lambda^j}{2}(\alpha^j+\beta^j-1)^2 - \sum_{j=1}^{M}\log N(\lambda^j).\end{aligned} \tag{12}$$

Equating the derivative of $\mathcal{L}_{\boldsymbol{\theta}}$ with respect to p to zero, we estimate p as

$$p = \frac{p_1 - 1 + \sum_{i=1}^{N}\mu_i}{p_1 + p_2 - 2 + N}. \tag{13}$$

The derivative with respect to α^j and β^j can be computed as follows:

$$\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}}{\partial \alpha^j} = \frac{\sum_{i=1}^{N}\mu_i y_i^j - \alpha^j \sum_{i=1}^{N}\mu_i}{\alpha^j(1-\alpha^j)} - \lambda^j(\alpha^j+\beta^j-1), \tag{14}$$

$$\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}}{\partial \beta^j} = \frac{\sum_{i=1}^{N}(1-\mu_i)(1-y_i^j) - \beta^j \sum_{i=1}^{N}(1-\mu_i)}{\beta^j(1-\beta^j)} - \lambda^j(\alpha^j+\beta^j-1). \tag{15}$$

Equating these derivatives to zero we obtain two cubic equations[5] involving α^j and β^j, respectively. We can iteratively solve one cubic equation (for example, for α^j) by fixing the counterpart (for

5. The pair of cubic equations are given by

$$\begin{aligned}\lambda^j(\alpha^j)^3 + (\beta^j\lambda^j - 2\lambda^j)(\alpha^j)^2 - (\lambda^j - \beta^j\lambda^j - \sum_{i=1}^{N}\mu_i)\alpha^j + (\sum_{i=1}^{N}\mu_i y_i^j) &= 0 \\ \lambda^j(\beta^j)^3 + (\alpha^j\lambda^j - 2\lambda^j)(\beta^j)^2 - (\lambda^j - \alpha^j\lambda^j - \sum_{i=1}^{N}\mu_i)\beta^j + (\sum_{i=1}^{N}\mu_i y_i^j) &= 0\end{aligned}$$

For each equation we retain only the root that lies in the range $[0,1]$.

example, β^j) till convergence. Also note that when $\lambda^j = 0$ we get the standard EM algorithm proposed by Dawid and Skene (1979). These two steps (the E- and M-step) can be iterated till convergence. We use majority voting $\mu_i = 1/M \sum_{j=1}^{M} y_i^j$ as the initialization for μ_i to start the EM-algorithm.

5. Algorithm to Eliminate Spammers

For each annotator we imposed the Automatic Spammer Detection prior of the form $\Pr[\alpha^j, \beta^j | \lambda^j] \propto \exp\left(-\lambda^j(\alpha^j + \beta^j - 1)^2/2\right)$, parameterized by precision hyperparameter λ^j. If we know the hyperparameters $\boldsymbol{\lambda} = [\lambda^1, \ldots, \lambda^M]$ we can compute the MAP estimate efficiently via the EM algorithm as described in the previous section. However it is crucial that we use the right λ^j for each annotator for two reasons: (1) For the good annotators we want the precision term to be small so that we do not over penalize the good annotators. (2) We can use the estimated λ^j to detect spammers. Clearly, as the precision λ^j increases, that is, the variance tends to zero, thus concentrating the prior sharply around the random diagonal line in the ROC plot. Hence, regardless of the evidence of the training data, the posterior will also be sharply concentrated around $\alpha^j + \beta^j = 1$, thus that annotator will not affect the ground truth and hence, it can be effectively removed. Therefore, the discrete optimization problem corresponding to spammer detection (should each annotator be included or not?), can be more easily solved via an easier continuous optimization over hyperparameters. In this section we adopt an empirical Bayesian strategy (specifically the *type-II maximum likelihood*) to automatically learn the hyperparameters from the data itself. This is in the spirit of the commonly used automatic relevance determination (ARD) prior used for feature selection by relevance vector machine (Tipping, 2001) and Gaussian process classification (Rasmussen and Williams, 2006).

5.1 Evidence Maximization

In *type-II maximum likelihood* approach, the hyperparameters $\boldsymbol{\lambda}$ are chosen to maximize the marginal likelihood (or equivalently the log marginal likelihood), that is,

$$\widehat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}} \Pr[\mathcal{D}|\boldsymbol{\lambda}] = \arg\max_{\boldsymbol{\lambda}} \log \Pr[\mathcal{D}|\boldsymbol{\lambda}],$$

where the marginal likelihood $\Pr[\mathcal{D}|\boldsymbol{\lambda}]$ is essentially the *evidence* for $\boldsymbol{\lambda}$ with the parameters $\boldsymbol{\theta}$ marginalized or integrated out.

$$\Pr[\mathcal{D}|\boldsymbol{\lambda}] = \int_{\boldsymbol{\theta}} \Pr[\mathcal{D}|\boldsymbol{\theta}] \Pr[\boldsymbol{\theta}|\boldsymbol{\lambda}] d\boldsymbol{\theta}.$$

Since this integral is analytically intractable we use the Laplace method which involves a second order Taylor series approximation around the MAP estimate.

5.2 Laplace Approximation

The marginal likelihood can be rewritten as follows, $\Pr[\mathcal{D}|\boldsymbol{\lambda}] = \int_{\boldsymbol{\theta}} \exp[\Psi(\boldsymbol{\theta})] d\boldsymbol{\theta}$ where

$$\Psi(\boldsymbol{\theta}) = \log \Pr[\mathcal{D}|\boldsymbol{\theta}] + \log \Pr[\boldsymbol{\theta}|\boldsymbol{\lambda}].$$

We approximate Ψ using a second order Taylor series around the MAP estimate $\widehat{\boldsymbol{\theta}}_{\text{MAP}}$,

$$\Psi(\boldsymbol{\theta}) \approx \Psi(\widehat{\boldsymbol{\theta}}_{\text{MAP}}) + \frac{1}{2}(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{\text{MAP}}) \mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\lambda})(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{\text{MAP}})^{\top},$$

where $\mathbf{H}$ is the Hessian matrix. We have made use of the fact that the gradient of Ψ evaluated at the MAP estimate $\widehat{\boldsymbol{\theta}}_{\text{MAP}}$ is zero. Hence we have the following approximation to the log-marginal likelihood.

$$\log \Pr[\mathcal{D}|\boldsymbol{\lambda}] \approx \log \Pr[\mathcal{D}|\widehat{\boldsymbol{\theta}}_{\text{MAP}}] + \log \Pr[\widehat{\boldsymbol{\theta}}_{\text{MAP}}|\boldsymbol{\lambda}] - \frac{1}{2}\log\det[-\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\lambda})] + \frac{d}{2}\log 2\pi.$$

The hyperparameters $\boldsymbol{\lambda}$ are found by maximizing this approximation to the log marginal likelihood. We use a simple iterative re-estimation method by setting the first derivative to zero. The derivative can be written as (see Appendix B for more details)

$$\frac{\partial}{\partial \lambda^j}\log \Pr[\mathcal{D}|\boldsymbol{\lambda}] \approx -\frac{1}{2}(\widehat{\alpha}^j + \widehat{\beta}^j - 1)^2 + \frac{1}{2\lambda^j}\delta(\lambda^j) - \frac{1}{2}\sigma(\lambda^j),$$

where we have defined

$$\delta(\lambda^j) = 2 - \frac{\sqrt{2\pi\lambda^j}\,\text{erf}(\sqrt{\lambda^j/2})}{\sqrt{2\pi\lambda^j}\,\text{erf}(\sqrt{\lambda^j/2}) + 2\exp(-\lambda^j/2) - 2}, \tag{16}$$

in which $\text{erf}(x) = (2/\sqrt{\pi})\int_0^x \exp(-t^2)dt$ is the error function, and

$$\sigma(\lambda^j) = \text{Tr}\left[\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\lambda})\frac{\partial}{\partial \lambda^j}\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\lambda})\right].$$

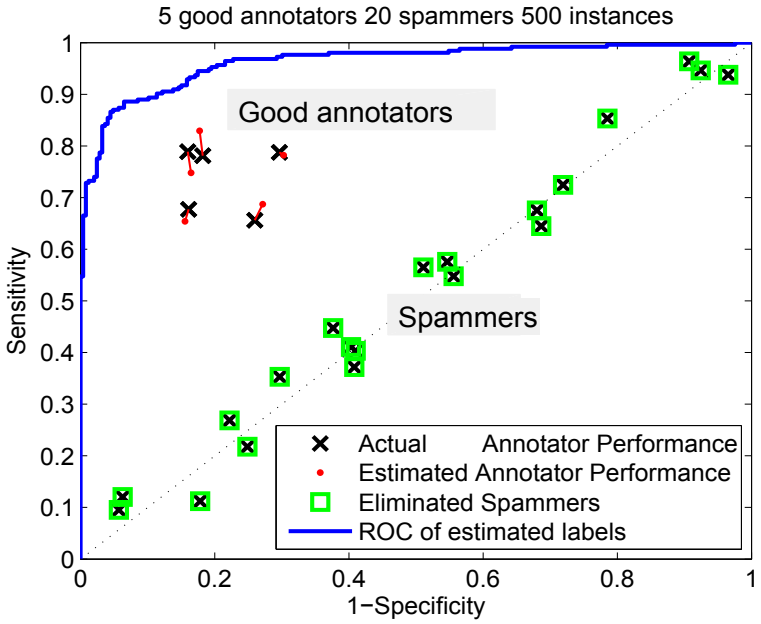
See Appendix B for more details on computation of $\sigma(\lambda^j)$. Assuming $\delta^j = \delta(\lambda^j)$ and $\sigma^j = \sigma(\lambda^j)$ does not depend on λ^j, a simple update rule for the hyperparameters can be written by equating the first derivative to zero.[6]

$$\lambda^j = \frac{\delta^j}{(\widehat{\alpha}^j + \widehat{\beta}^j - 1)^2 + \sigma^j}. \tag{17}$$

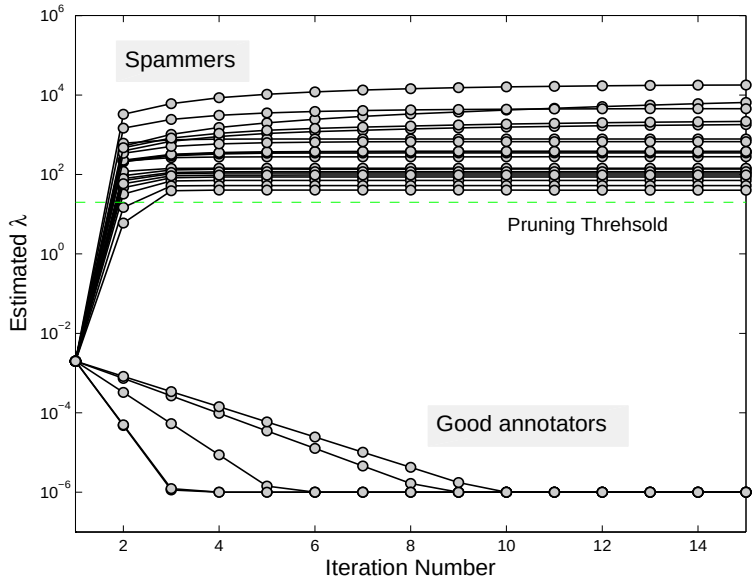
One way to think of this is that the penalization is inversely proportional to $(\widehat{\alpha}^j + \widehat{\beta}^j - 1)^2$, that is, good annotators get penalized less while the spammers suffer a large penalization. Figure 4(b) plots the estimated hyperparameter $\widehat{\lambda}^j$ for each annotator as a function of the iteration number for a simulation setup shown in Figure 4(a). The simulation has 5 good annotators and 20 spammers. It can be seen that as expected for the good annotators $\widehat{\lambda}^j$ starts decreasing[7] while for the spammers $\widehat{\lambda}^j$ starts increasing with iterations.[8] By using a suitable pruning threshold we can detect and eliminate the spammers.

The final algorithm has two levels of iterations (see Algorithm 1): in an outer loop we update the hyper-parameters $\widehat{\lambda}^j$ and in an inner loop we find the MAP estimator for sensitivity and specificity given the hyper-parameters. At each iteration we eliminate all the annotators for whom the estimated $\widehat{\lambda}^j$ is greater than a certain pruning threshold T.[9]

6. In practice, one can iterate (17) and (16) several times to get better estimate for λ^j.
7. For numerical stability we do not let the hyper parameter go below 10^{-6}.
8. We have different rates of convergence for the good annotators and the spammers. This is because of our assumption that δ (16) does not depend on λ. This is almost true for large λ and is not a good approximation for small λ.
9. For all our experiments for each annotator we set the pruning threshold to 0.1 times the number of instances labeled by him.



(a) Setup



(b) Estimated Hyperparameters

Figure 4: *Illustration of spammer elimination via evidence maximization* (a) The black cross plots the actual sensitivity and specificity of each annotator. The simulation has 5 good annotators and 20 spammers and 500 instances. The red dot plots the sensitivity and specificity as estimated by the SpEM algorithm. The green squares show the annotators eliminated as spammers. (b) The estimated hyperparameter λ^j for each annotator as a function of the iteration number. The pruning threshold is also shown on the plot.

Algorithm 1 SpEM

Require: Annotations $y_i^j \in \{0,1\}$, $j = 1,\ldots,M$, $i = 1,\ldots,N$ from M annotators on N instances.
Initialize $\lambda^j = 1/N$, for $j = 1,\ldots,M$.
Initialize $\mathcal{A} = \{1,\ldots,M\}$ the set of good annotators.
Initialize $\mu_i = 1/M \sum_{j=1}^{M} y_i^j$ using soft majority voting.
repeat {Outer loop with evidence maximization}
  repeat {**EM loop**}
    {M-step}
    Update p based on (13).
    Update α^j, β^j based on (14)-(15), $\forall j \in \mathcal{A}$.
    {E-step}
    Estimate μ_i using (11), $\forall i = 1,\ldots,N$.
  until Change of expected posterior (12) $< \varepsilon_1$.
  {**Evidence Maximization**}
  for all $j \in \mathcal{A}$ **do**
    Update λ^j based on (17).
    if $\lambda^j > T$ (the pruning threshold) **then**
      $\mathcal{A} \leftarrow \mathcal{A} \backslash \{j\}$
    end if
  end for
until Change of expected posterior (12) $< \varepsilon_2$.
Ensure: Detected spammers in set $\{1,\ldots,M\} \backslash \mathcal{A}$.
Ensure: Non-spammers in $\mathcal{A}$ with sensitivity α^j and specificity β^j, for $j \in \mathcal{A}$.
Ensure: Prevalence factor p and expected hidden label μ_i, $\forall i = 1,\ldots,N$.
In all our experiments we set the convergence tolerance $\varepsilon_1 = \varepsilon_2 = 10^{-3}$. The pruning threshold was set to $T = 0.1N$.

6. Discussions

1. **Can we use the EM algorithm directly to eliminate spammers?** Majority Voting and EM algorithm do not have a mechanism to explicitly detect spammers. However we could define an annotator as a spammer if the estimated $|\widehat{\alpha}^j + \widehat{\beta}^j - 1| \leq \varepsilon$. However it is not clear what is the right ε to use. Also the spammers influence the estimation of $\widehat{\alpha}^j$ and $\widehat{\beta}^j$ for the good annotators. A fix to this would be to eliminate the spammers and get an improved estimate of the ground truth. In principle this process could be repeated till convergence, which essentially boils down to a discrete version of our proposed SpEM algorithm.

2. **What is the advantage of different shrinkage for each annotator ?** We could have imposed a common shrinkage prior (that is, same $\lambda^j \equiv \lambda$ for all annotators) and then estimated one λ as shown earlier. While this is a valid approach, the advantage of our ASD prior is that the amount of shrinkage for each annotator is different and depends on how good the annotator is, that is, good annotators suffer less shrinkage while spammers suffer severe shrinkage.

3. **Missing annotations** The proposed SpEM algorithm can be easily extended to handle missing annotations (which is more realistic scenario in crowdsourcing marketplaces). Let M_i be the number of annotators labeling the i^{th} instance, and let N_j be the number of instances labeled

by the j^{th} annotator. Then in the EM loop, we just need to replace N by N_j for estimating α^j and β^j in (14) and (15), and replace M by M_i for updating μ_i in (11).

4. **Training a classifier directly** The proposed algorithm can be readily extended to learn a classifier along with the ground truth (Raykar et al., 2009). Let instance i have features $\boldsymbol{x}_i \in \mathbb{R}^d$, and define the classification problem as learning $\boldsymbol{w} \in \mathbb{R}^d$ such that $\Pr[y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}] = p_i = f(\boldsymbol{w}^\top \boldsymbol{x}_i)$, with f a mapping function (for example, logistic function). To learn $\boldsymbol{w}$ in SpEM we just need to replace (13) with a Newton-Raphson step to update $\boldsymbol{w}$, and replace p with p_i in (11).

5. **Partially known gold standard** If the actual ground truth is available for some instances, SpEM can readily incorporate them into the learning loop. The only change we need to make is to estimate μ_i in (11) only for the instances for which the ground truth is not available, and fix $\mu_i = y_i$ if the ground truth y_i is available. Therefore, the gold standard instances and unlabeled instances will be used together to estimate the sensitivity and specificity of each annotator (and also to estimate the labels).

7. Extension to Categorical Annotations

We now extend the proposed algorithm to handle categorical annotations. A simple solution for categorical outcomes is to use a one-against-all strategy and run the binary SpEM C times, each time obtaining a spammer indicator λ^j for each annotator. One might then identify an annotator j as a spammer if all of the λ^j in the C runs indicate that this is a spammer. However in this section we provide a more principled solution in line with the framework proposed for binary labels. Following the same motivation as before, we define the ASD prior as follows

$$\Pr[\mathbf{A}^j|\lambda^j] = \frac{1}{N(\lambda^j)} \exp\left(-\frac{\lambda^j}{2C} \sum_{c<c'} \sum_{k=1}^{C} \left(\alpha_{ck}^j - \alpha_{c'k}^j\right)^2 \right),$$

which gives more probability mass to a spammer. A similar EM algorithm can be developed under this prior, and evidence maximization follows naturally with Laplace approximation. Under the same assumptions as earlier, the log-likelihood of the parameters $\boldsymbol{\theta} = [\mathbf{A}^1, \ldots, \mathbf{A}^M, p_1, \ldots, p_C]$ is

$$\log \Pr[\mathcal{D}|\boldsymbol{\theta}] = \sum_{i=1}^{N} \log \left[\sum_{c=1}^{C} \Pr(y_i = c) \prod_{j=1}^{M} \Pr(y_i^j|y_i = c) \right] = \sum_{i=1}^{N} \log \left[\sum_{c=1}^{C} p_c \prod_{j=1}^{M} \prod_{k=1}^{C} (\alpha_{ck}^j)^{\delta(y_i^j,k)} \right],$$

where $p_c = \Pr(y_i = c)$ and $\delta(u,v) = 1$ if $u = v$ and 0 otherwise. If we know the missing labels y the complete log likelihood can be written as

$$\log \Pr[\mathcal{D}, y|\boldsymbol{\theta}] = \sum_{i=1}^{N} \sum_{c=1}^{C} \delta(y_i, c) \log \left[p_c \prod_{j=1}^{M} \prod_{k=1}^{C} (\alpha_{ck}^j)^{\delta(y_i^j,k)} \right].$$

In the E-step we compute the conditional expectation as

$$\mathbb{E}\left\{\log \Pr[\mathcal{D}, \boldsymbol{y}|\boldsymbol{\theta}]\right\} = \sum_{i=1}^{N} \sum_{c=1}^{C} \mu_{ic} \log \left[p_c \prod_{j=1}^{M} \prod_{k=1}^{C} (\alpha_{ck}^j)^{\delta(y_i^j,k)} \right]$$

where $\mu_{ic} = \Pr[y_i = c | y_i^1, \ldots, y_i^M, \boldsymbol{\theta}]$ and is computed as $\mu_{ic} \propto p_c \prod_{j=1}^{M} \prod_{k=1}^{C} (\alpha_{ck}^j)^{\delta(y_i^j, k)}$. Based on the current estimate μ_{ic} in the M-step we can estimate the parameters by maximizing the lower bound on the log posterior (along with the Lagrange multipliers γ), $\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}$, where

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N} \sum_{c=1}^{C} \mu_{ic} \left[\log p_c + \sum_{j=1}^{M} \sum_{k=1}^{C} \delta(y_i^j, k) \log \alpha_{ck}^j \right] \\ &- \sum_{j=1}^{M} \frac{\lambda^j}{2C} \sum_{c<c'} \sum_{k=1}^{C} \left(\alpha_{ck}^j - \alpha_{c'k}^j\right)^2 - \sum_{j=1}^{M} \log N(\lambda^j) + \sum_{j=1}^{M} \sum_{c=1}^{C} \gamma_c^j \left(1 - \sum_{k=1}^{C} \alpha_{ck}^j\right).\end{aligned}$$

We update the prevalence as $p_c = (1/N) \sum_{i=1}^{N} \mu_{ic}$ and for the α_{ck}^j we have

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha_{ck}^j} &= \frac{\sum_{i=1}^{N} \mu_{ic} \delta(y_i^j, k)}{\alpha_{ck}^j} - \frac{\lambda^j}{C} \sum_{c' \neq c} \left(\alpha_{ck}^j - \alpha_{c'k}^j\right) - \gamma_c^j = 0, \\ \frac{\partial \mathcal{L}}{\partial \gamma_c^j} &= \sum_{k=1}^{C} \alpha_{ck}^j - 1 = 0.\end{aligned} \tag{18}$$

The practical solution to solve[10] this for every α_{ck}^j is to fix the $\alpha_{c'k}^j$ for $c' \neq c$, solve the equation array with a fixed γ_c^j, and then update γ_c^j as

$$\gamma_c^j = \frac{1}{C} \sum_{k=1}^{C} \frac{\sum_{i=1}^{N} \mu_{ic} \delta(y_i^j, k)}{\alpha_{ck}^j},$$

which follows by summing (18) for all k. As earlier in order to determine the hyperparameters we obtain a simple iterative update by setting the derivative of the approximate log-marginal likelihood to zero.

$$\frac{\partial}{\partial \lambda^j} \log \Pr[\mathcal{D} | \boldsymbol{\lambda}] \approx -\frac{1}{2C} \sum_{c<c'} \sum_{k=1}^{C} \left(\alpha_{ck}^j - \alpha_{c'k}^j\right)^2 - \frac{1}{N(\lambda^j)} \frac{\partial}{\partial \lambda^j} N(\lambda^j) - \frac{1}{2} \sigma(\lambda^j),$$

where we have defined

$$\sigma(\lambda) = \text{Tr}\left[\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\lambda}) \frac{\partial}{\partial \lambda} \mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\lambda}) \right].$$

and

$$-\frac{1}{N(\lambda^j)} \frac{\partial}{\partial \lambda^j} N(\lambda^j) = \frac{1}{2\lambda^j} \delta(\lambda^j).$$

See Appendix C for more details on computation of σ and δ. Then the update is given by

$$\widehat{\lambda}^j = \frac{\delta(\lambda^j)}{(1/C) \sum_{c<c'} \sum_{k=1}^{C} \left(\alpha_{ck}^j - \alpha_{c'k}^j\right)^2 + \sigma(\lambda^j)}.$$

10. Keeping all terms except α_{ck}^j fixed this is a quadratic equation $A(\alpha_{ck}^j)^2 + B(\alpha_{ck}^j) + C = 0$ where $A = \frac{\lambda^j (C-1)}{C}$, $B = \gamma_c^j - \frac{\lambda^j}{C} \sum_{c' \neq c} \alpha_{c'k}^j$, and $C = -\sum_{i=1}^{N} \mu_{ic} \delta(y_i^j, k)$. We keep the root which lies between 0 and 1.

8. Experimental Validation

We first experimentally validate the proposed algorithm on simulated data. Figure 5(a) shows the simulation setup consisting of 5 good annotators (shown as red squares) and 100 spammers (shown as black crosses). The good annotators have sensitivity and specificity between 0.65 and 0.85. All the spammers lie around the diagonal. We compare our proposed SpEM algorithm against the commonly used Majority Voting and the EM algorithm (Dawid and Skene, 1979; Smyth et al., 1995; Raykar et al., 2009, 2010). All these methods estimate a probabilistic version ($[0\,1]$) of the binary ground truth ($\{0,1\}$). Since we simulate the instances we know the actual binary ground truth and hence can compute the area under the ROC curve (AUC) of the estimated probabilistic ground truth.

8.1 Effect of Increasing Spammers

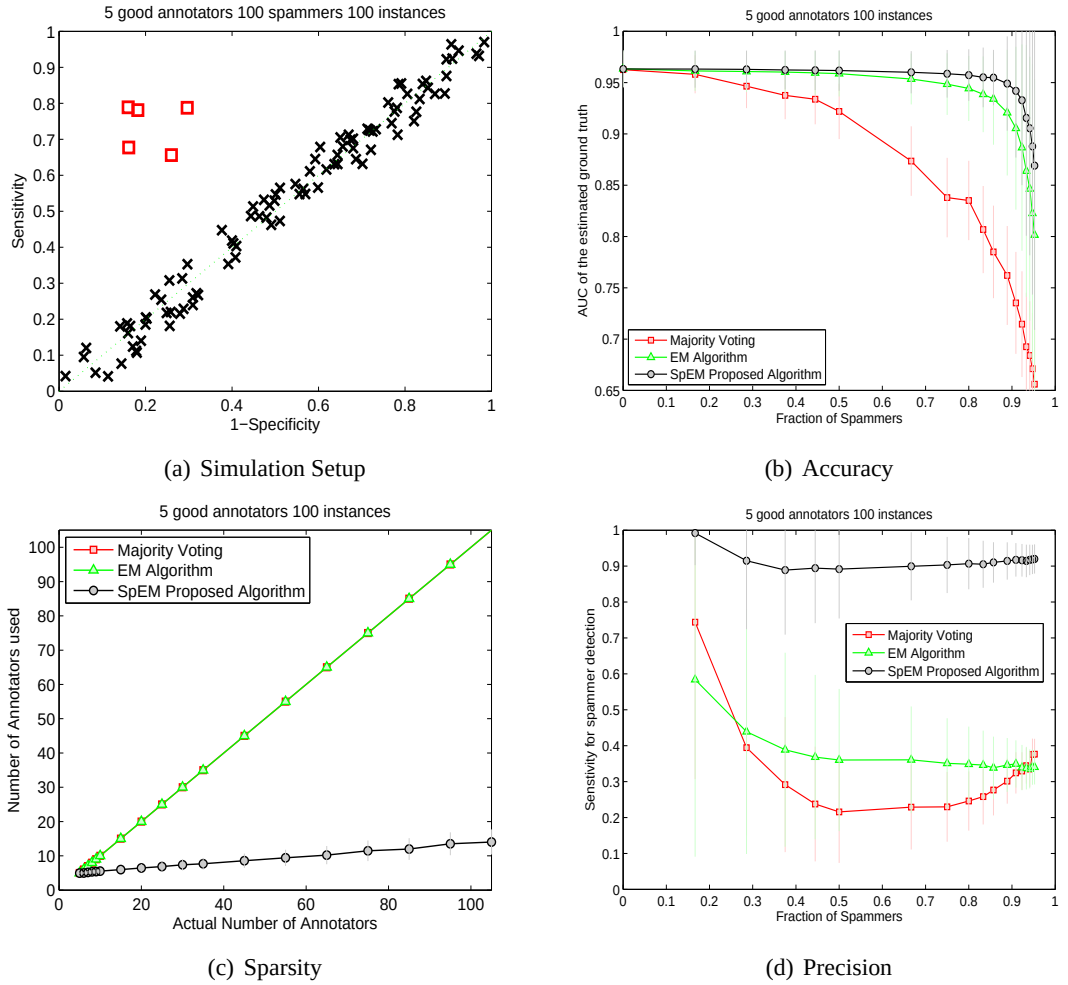
For the first experiment we deliberately choose 100 instances (with prevalence $p = 0.5$), since it is beneficial if we can detect the spammers with fewer instances. Figure 5(b) plots AUC of the estimated probabilistic ground truth as a function of the fraction of spammers (number of spammers/total number of annotators), for each point we keep all the five good annotators and keep adding more annotators from the pool of 100 spammers. All plots show the mean and one standard deviation error bars (over 100 repetitions). The pruning threshold for the SpEM algorithm was set to 20. Figure 5(d) plots the sensitivity for spammer detection which is essentially the fraction of spammers correctly detected. The following observations can be made:

1. As the fraction of spammers increases the performance of the Majority Voting degrades drastically as compared to the EM and the SpEM algorithm (refer Figure 5(b)). The proposed SpEM algorithm has a better AUC than the EM algorithm especially when the spammers dominate (when the fraction of spammers is greater than 0.7 in Figure 5(b)). The variability (one standard deviation error bars) for all the methods increases as the number of spammers increases.

2. The clear advantage of the proposed SpEM algorithm can be seen in Figure 5(d) where it can identify almost 90% of the spammers correctly as compared the EM algorithm with can identify about 40% correctly. Majority Voting and EM algorithm do not have a mechanism to explicitly detect spammers, we define an annotator as a spammer if the estimated $|\widehat{\alpha}^j + \widehat{\beta}^j - 1| \leq \varepsilon$ (We have used $\varepsilon = 0.05$ in our experiments.[11].)

3. The SpEM algorithm iteratively eliminates the spammers and then re-estimates the ground truth without the spammers. Figure 5(c) plots the actual number of annotators that were used in the final model. Note that the EM and the Majority Voting use all the annotators to estimate the model parameters while the SpEM algorithm uses only a small fraction of the annotators.

To summarize, the proposed SpEM algorithm is slightly more accurate than the EM algorithm and at the same time uses a small fraction of the annotators thus effectively eliminating most of the spammers.



(a) Simulation Setup

(b) Accuracy

(c) Sparsity

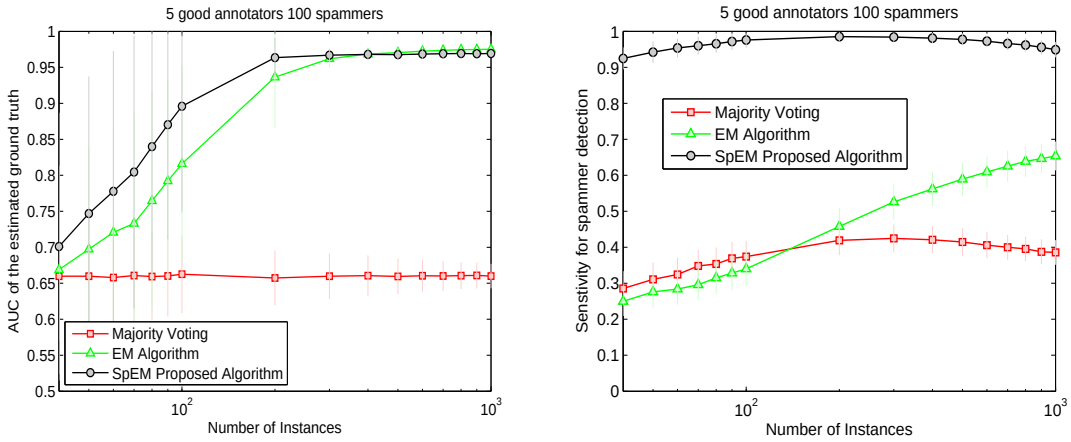
(d) Precision

Figure 5: *Effect of increasing the number of spammers* (Section 8.1) (a) The simulation setup has 5 good annotators (red squares) and 100 spammers (black crosses) and 100 instances. (b) The AUC of the estimated ground truth as a function of the fraction of spammers. (c) The actual number of annotators that were used. (d) The fraction of spammers correctly detected. All plots show mean and one standard deviation error bars (over 100 repetitions).

8.2 Effect of Increasing Instances

For the proposed algorithm to be practically useful we would like to detect the spammers with as few examples as possible so that they can be eliminated early on. Figure 6 plots the performance for the same setup as earlier as a function of the number of instances. From Figure 6(a) we see that the AUC for the proposed method is much better than the EM algorithm especially for smaller number of instances. As the number of instances increases the accuracy of the EM algorithm is as good as the proposed SpEM algorithm. The EM algorithm (and also the proposed SpEM) automatically gives

11. The 0.05 value is just a heuristic based on a band around the diagonal of the ROC plot.



(a) Accuracy (b) Sparsity

Figure 6: *Effect of increasing the number of instances* (Section 8.2) (a) The AUC of the estimated ground truth as a function of the number of instances. (b) The fraction of actual spammers that were eliminated. All plots show the mean and one standard deviation error bars (over 100 repetitions). The simulation setup has 5 good annotators and 100 spammers. The pruning threshold was set to $0.1N$ where N is the total number of instances.

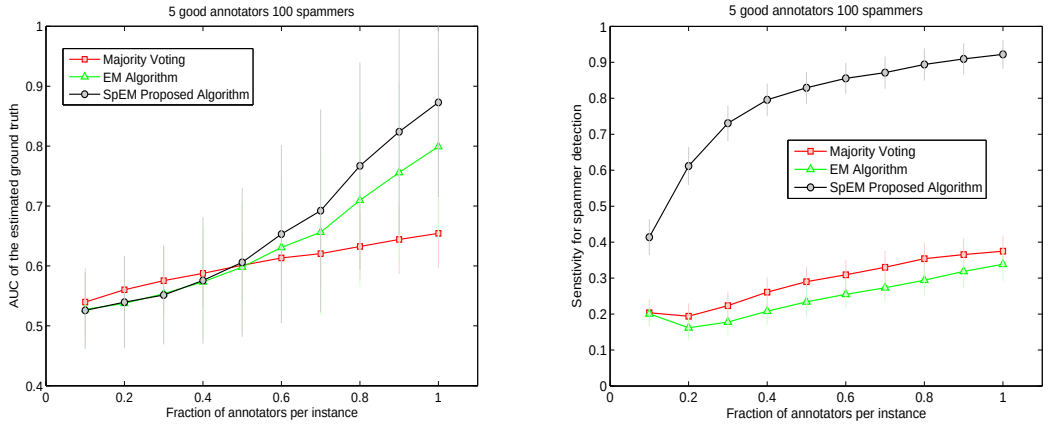
less emphasis for annotators with small $|\widehat{\alpha}^j + \widehat{\beta}^j - 1|$. The reason SpEM achieves better accuracy is that the parameters $\widehat{\alpha}^j$ and $\widehat{\beta}^j$ are better estimated because of the ASD prior we imposed. This also explains the fact that when we have a large number of instances both the EM and SpEM algorithm estimate the parameters equally well. The main benefit can be seen in Figure 6(b) where the SpEM algorithm can eliminate most of the spammers. For example with just 50 examples the SpEM algorithm can detect $> 90\%$ of the spammers and at the same time achieve a higher accuracy.

8.3 Effect of Missing Labels

In a realistic scenario an annotator does not label all the instances. Figure 7 plots the behavior of the different algorithms as a function of the fraction of annotators labeling each instance. When each annotator labels only a few instances all three algorithms achieve very similar performance in terms of the AUC. However the proposed SpEM algorithm can still eliminate more spammers then the EM algorithm.

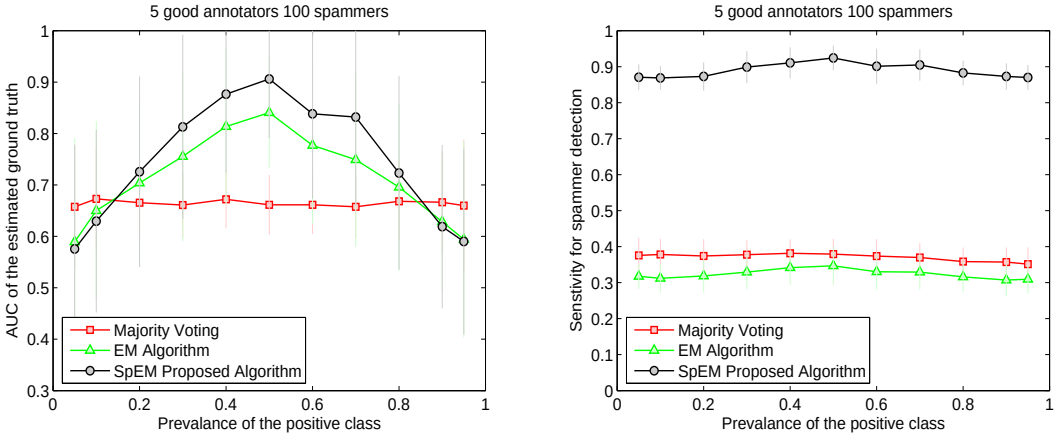
8.4 Effect of Prevalence

Figure 8 plots the behavior of the different algorithms as a function of the prevalence of the positive class. Note that when the prevalence is low the majority voting seems superior to other algorithms in terms of AUC. When the prevalence is small (or large) there are very few examples to reliably estimate the sensitivity (or specificity).



(a) Accuracy (b) Precision

Figure 7: *Effect of missing labels* (Section 8.3) (a) The AUC of the estimated labels as a function of the fraction of annotators labeling each instance. (b) The fraction of actual spammers that were eliminated. All plots show the mean and one standard deviation error bars (over 100 repetitions). The simulation setup has 5 good annotators and 50 spammers.

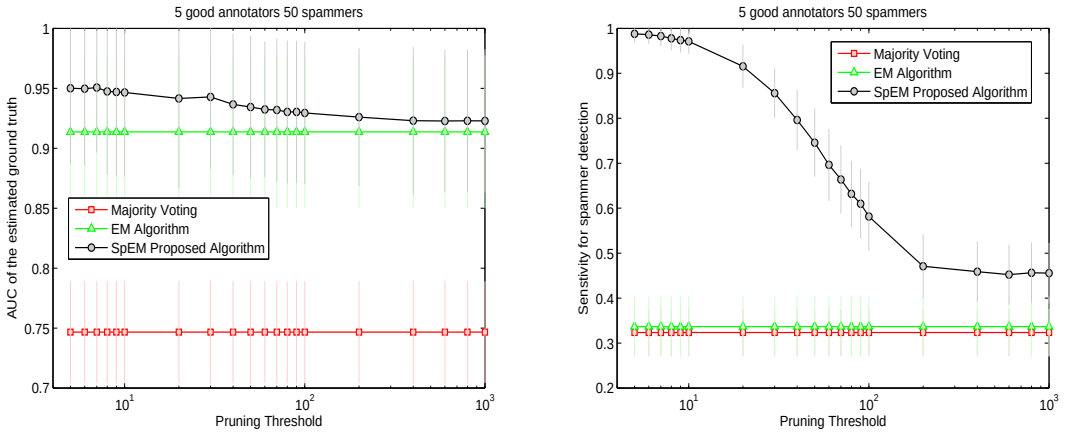


(a) Accuracy (b) Precision

Figure 8: *Effect of prevalence* (Section 8.4) (a) The AUC of the estimated labels as a function of the prevalence of the positive class. (b) The fraction of actual spammers that were eliminated. All plots show the mean and one standard deviation error bars (over 100 repetitions). The simulation setup has 5 good annotators and 50 spammers.

8.5 Effect of Pruning Threshold

The only tunable parameter of the SpEM algorithm is the pruning threshold. For all our experiments for each annotator we set the pruning threshold to 0.1 times the number of instances labeled by the



(a) Accuracy (b) Precision

Figure 9: *Effect of pruning threshold* (Section 8.5) (a) The AUC of the estimated labels as a function of the pruning threshold. (b) The fraction of actual spammers that were eliminated. All plots show the mean and one standard deviation error bars (over 100 repetitions). The simulation setup has 5 good annotators and 50 spammers.

annotator. However we can use this parameter to control the number of annotators we want to use. Figure 9 plots the performance for the same setup as earlier for different pruning thresholds. From Figure 9(b) we see that as the pruning threshold decreases the sensitivity for spammer elimination increases thereby using less annotators. Interestingly the accuracy also increases. If we had imposed a common shrinkage prior (that is, same λ^j for all annotators) then we would expect a drop in accuracy as the model becomes more sparse. The advantage of our ASD prior is that the amount of shrinkage for each annotator is different and depends on how accurate the annotator is, more accurate annotators suffer less shrinkage while spammers suffer severe shrinkage.

8.6 Experiments On Crowdsourced Data

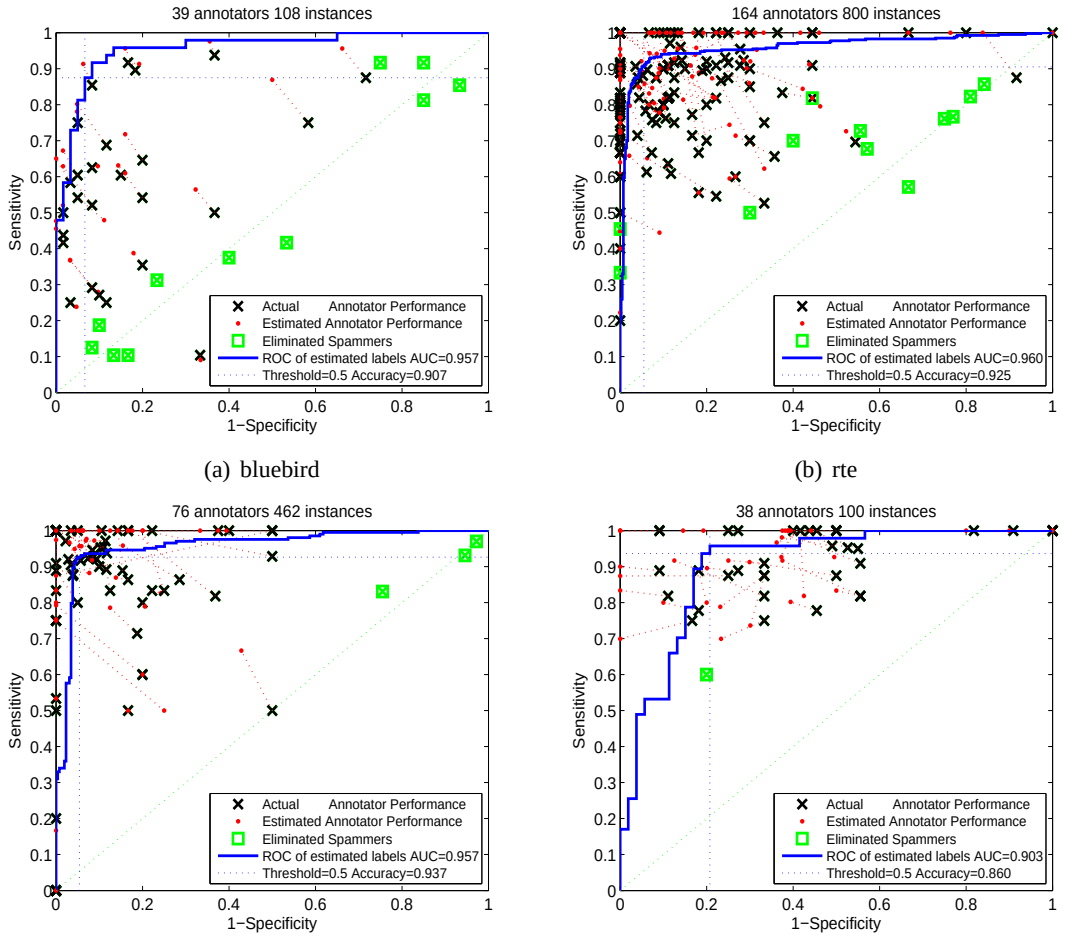
We report results on some publicly available linguistic and image annotation data collected using the Amazon Mechanical Turk and other sources. Table 1 summarizes the data sets along with a brief description of the tasks. Table 2 summarizes the results for the binary data sets with known ground truth. We compare the proposed SpEM, EM (Dawid and Skene, 1979; Raykar et al., 2010), and the Majority Voting (MV) algorithm in terms of AUC and accuracy. To compute the accuracy we use a threshold of 0.5 on the estimated probabilities. In terms of the AUC all three algorithms have similar performance. In terms of accuracy the SpEM and EM were better than the MV algorithm. The table also shows the number of annotators eliminated as spammers by the proposed algorithm. Figure 10 plots the actual and the estimated annotator performance for the SpEM algorithm for binary data sets with known ground truth.

Data Set	Type	N	M	M^*	N^*	Brief Description
bluebird	binary	108	39	39/39	108/108	bird identification (Welinder et al., 2010) The annotator had to identify whether there was a Indigo Bunting or Blue Grosbeak in the image.
rte	binary	800	164	10/10	49/20	recognizing textual entailment (Snow et al., 2008) The annotator is presented with two sentences and given a binary choice of whether the second hypothesis sentence can be inferred from the first.
temp	binary	462	76	10/10	61/16	event annotation (Snow et al., 2008) Annotators are presented with a dialogue and a pair of verbs from the dialogue, and need to label whether the event described by the first verb occurs before or after the second.
localview×	binary	832	97	5/5	43/14	word sense disambiguation (Parent and Eskenazi, 2010) Workers were asked to indicate if two definitions of a word were related to the same meaning or different meanings.
valence	ordinal	100	38	10/10	26/20	affect recognition (Snow et al., 2008) Each annotator is presented with a short headline and asked to rate the overall positive or negative valence of the emotional content of the headline.
sentiment×	categorical/3	1660	33	6/6	291/175	Irish economic sentiment analysis (Brew et al., 2010) Articles from three Irish online news sources were annotated by a group of 33 volunteer users, who were encouraged to label the articles as positive, negative, or irrelevant.

Table 1: *Data Sets.* N is the number of instances and M is the number of annotators. M^* is the mean/median number of annotators per instance. N^* is the mean/median number of instances labeled by each annotator. All the data sets except those marked $\times$ have a known gold standard. Except sentiment data set all others were collected using Amazons's Mechanical Turk. The valence data set was converted to a binary scale in our experiments.

Data	Spammers	AUC			Accuracy		
	S	SpEM	EM	MV	SpEM	EM	MV
bluebird	11/39	.96	.95	.88	.91	.90	.76
rte	12/164	.96	.96	.96	.93	.93	.92
temp	3/76	.96	.96	.97	.94	.94	.94
valence	1/38	.90	.91	.94	.86	.86	.80
localview×	12/97	-	-	-	-	-	-
sentiment×	1/33	-	-	-	-	-	-

Table 2: *Comparison of the various methods for the data sets in Table 1.* SpEM is the proposed algorithm, EM is the algorithm proposed in Dawid and Skene (1979) and Raykar et al. (2010), and MV is the soft majority voting algorithm. S is the number of annotators eliminated as spammers by the proposed algorithm. The accuracy and AUC are shown only for data sets with known gold standard.



(a) bluebird (b) rte

(c) temp (d) valence

Figure 10: *SpEM results for binary datsets shown in Table 2* The black cross plots the actual sensitivity and specificity of each annotator. The red dot plots the sensitivity and specificity estimated by the SpEM algorithm. The green squares show the annotators eliminated as spammers. We plot the ROC for the estimated ground truth and the operating point corresponding to a threshold of 0.5.

8.7 Ranking Annotators

The proposed spammer score can be used to rank the annotators. Figure 11 plots the spammer scores and rankings obtained for four data sets. The mean and the 95% CI obtained via bootstrapping are also shown. The number at the top of the CI bar shows the number of instances annotated by that annotator. The rankings are based on the lower limit of the 95% CI which factors the number of instances labeled by the annotator into the ranking. An annotator who labels only a few instances will have very wide CI. Some annotators who label only a few instances may have a high mean spammer score but the CI will be wide and hence ranked lower. Ideally we would like to have annotators with a high score and at the same time label a lot of instances so that we can reliably identify them. The authors (Brew et al., 2010) for the sentiment data set shared with us some of the qualitative observations regarding the annotators and they somewhat agree with our rankings. For example the authors made the following comments about Annotator 7 *"Quirky annotator - had a lot of debate about what was the meaning of the annotation question. I'd say he changed his labeling strategy at least once during the process"*. Our proposed score gave a low rank to this annotator.

9. Conclusion

In this paper we formalized the notion of a spammer for binary and categorical annotations. Using the score to define a prior we proposed an empirical Bayesian algorithm called SpEM that simultaneously estimates the consensus ground truth and also eliminates the spammers. Experiments on simulated and real data show that the proposed approach is better than (or as good as) the earlier approaches in terms of the accuracy and uses a significantly smaller number of annotators.

Appendix A. ASD Prior Normalization

In this appendix we analytically derive the normalization term for the proposed ASD prior. The normalization term $N(\lambda^j)$ can be computed as

$$\begin{aligned} N(\lambda^j) &= \int_0^1 \int_0^1 \exp\left(-\frac{\lambda^j(\alpha^j+\beta^j-1)^2}{2}\right) d\alpha^j d\beta^j \\ &= \int_0^1 \left[\int_0^1 \sqrt{\frac{2\pi}{\lambda^j}} \mathcal{N}\left(\beta^j; 1-\alpha^j, \frac{1}{\lambda^j}\right) d\beta^j\right] d\alpha^j \\ &= \sqrt{\frac{2\pi}{\lambda^j}} \left[\int_0^1 \Phi(\sqrt{\lambda^j}\alpha^j) d\alpha^j - \int_0^1 \Phi[\sqrt{\lambda^j}(\alpha^j-1)] d\alpha^j\right], \end{aligned}$$

where $\Phi(x) = (1/\sqrt{2\pi}) \int_{-\infty}^{x} \exp(-t^2/2) dt$ is the Gaussian cumulative distribution function and $\mathcal{N}(x;u,v)$ the Gaussian distribution of x with mean u and variance v. Using the fact that $\int \Phi(x) dx = x\Phi(x) + \phi(x)$, where ϕ is the standard normal and $\Phi(x) = (1/2)[1 + \mathrm{erf}(t/\sqrt{2})]$ the normalization

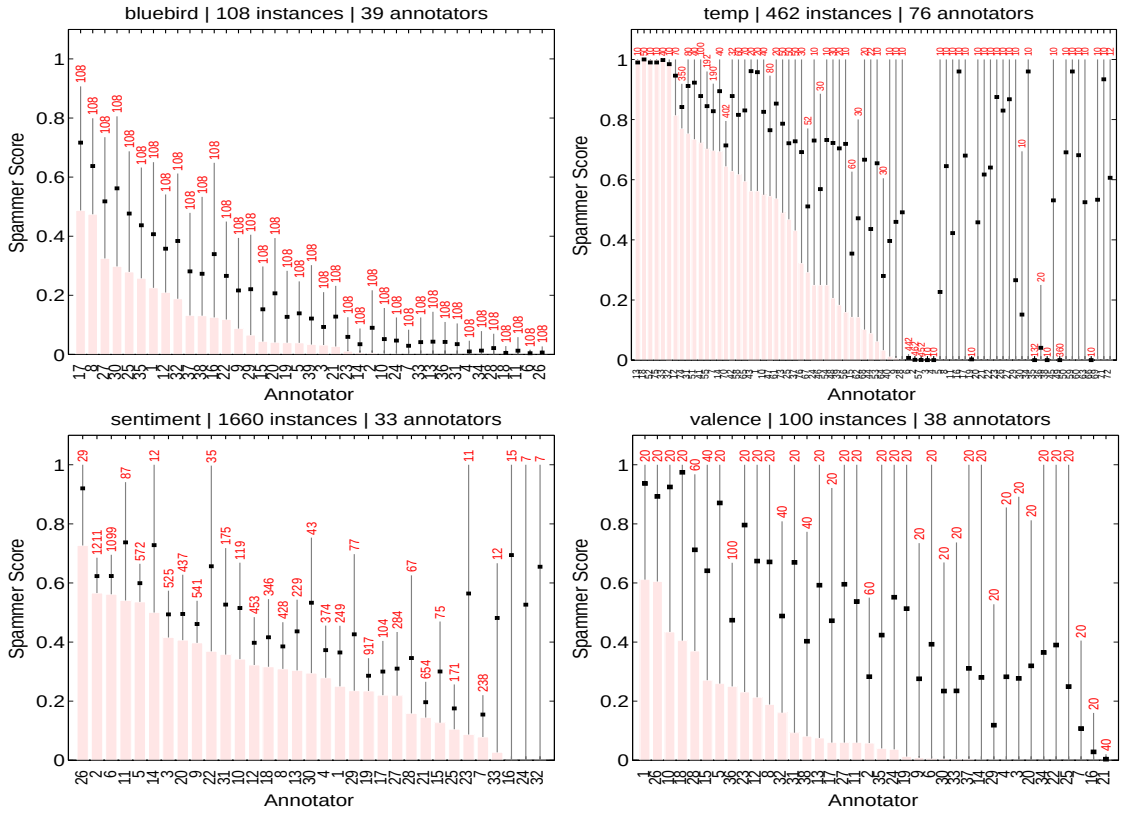


Figure 11: *Annotator Rankings* The rankings obtained for the data sets in Table 1. The spammer score ranges from 0 to 1, the lower the score, the more spammy the annotator. The mean spammer score and the 95% confidence intervals (CI) are shown, obtained from 100 bootstrap replications. The annotators are ranked based on the lower limit of the 95% CI. The number at the top of the CI bar shows the number of instances annotated by that annotator. Note that the CIs are wider when the annotator labels only a few instances.

term can be further simplified as follows,

$$
\begin{aligned}
N(\lambda^j) &= \frac{\sqrt{2\pi}}{\lambda^j}\left(\sqrt{\lambda^j}(2\Phi(\sqrt{\lambda^j})-1)+2\phi(\sqrt{\lambda^j})-2\phi(0)\right)\\
&= \frac{\sqrt{2\pi}}{\lambda^j}\left(\sqrt{\lambda^j}\mathrm{erf}(\sqrt{\lambda^j/2})+2\phi(\sqrt{\lambda^j})-2\phi(0)\right)\\
&= \frac{1}{\lambda^j}\left(\sqrt{2\pi\lambda^j}\mathrm{erf}(\sqrt{\lambda^j/2})+2\exp(-\lambda^j/2)-2\right)\\
&= \sqrt{\frac{2\pi}{\lambda^j}}\left(\frac{2}{\sqrt{\lambda^j}}\int_0^{\sqrt{\lambda^j}}\Phi(t)dt-1\right).
\end{aligned}
$$

Appendix B. Derivatives of the Log-marginal—Binary Case

The derivative of the approximation to the log-marginal likelihood can be written as

$$\begin{aligned}\frac{\partial}{\partial\lambda^j}\log\Pr[\mathcal{D}|\boldsymbol{\lambda}] &\approx \frac{\partial}{\partial\lambda^j}\log\Pr[\widehat{\boldsymbol{\theta}}_{\text{MAP}}|\boldsymbol{\lambda}] - \frac{1}{2}\text{Tr}\left[\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})\frac{\partial}{\partial\lambda^j}\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})\right] \\ &= \frac{\partial}{\partial\lambda^j}\log\Pr[\widehat{\alpha}^j,\widehat{\beta}^j|\lambda^j] - \frac{1}{2}\sigma(\lambda^j)\end{aligned}$$

where we have defined $\sigma(\lambda) = \text{Tr}\left[\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})\frac{\partial}{\partial\lambda}\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})\right]$. From the ASD prior we can show that

$$\frac{\partial}{\partial\lambda^j}\log\Pr[\widehat{\alpha}^j,\widehat{\beta}^j|\lambda^j] = -\frac{1}{2}(\widehat{\alpha}^j+\widehat{\beta}^j-1)^2 + \frac{1}{2\lambda^j}\delta(\lambda^j),$$

where we have defined

$$\delta(\lambda) = \left[2 - \frac{\sqrt{2\pi\lambda}\,\text{erf}(\sqrt{\lambda/2})}{\sqrt{2\pi\lambda}\,\text{erf}(\sqrt{\lambda/2}) + 2\exp(-\lambda/2) - 2}\right].$$

To compute $\sigma(\lambda^j)$, let us compute the Hessian matrix first. Since $\log\Pr[\mathcal{D}|\boldsymbol{\theta}]$ is again not tractable, we use the following lower bound (as used by the EM algorithm earlier) to compute the likelihood term:

$$\log\Pr[\mathcal{D}|\boldsymbol{\theta}] \geq \sum_{i=1}^{N}\left[\mu_i\log pa_i + (1-\mu_i)\log(1-p)b_i\right],$$

where $\mu_i = \Pr[\hat{y}_i|\boldsymbol{\theta}]$ is the expected class label for item i (calculated in the E-step). Then we have

$$\begin{aligned}\Psi(\boldsymbol{\theta}) &= \log\Pr[\mathcal{D}|\boldsymbol{\theta}] + \log\Pr[\boldsymbol{\theta}|\boldsymbol{\lambda}] \\ &\approx \sum_{i=1}^{N}\left[\mu_i\log pa_i + (1-\mu_i)\log(1-p)b_i\right] - \sum_{j=1}^{M}\frac{\lambda^j}{2}(\alpha^j+\beta^j-1)^2 - \sum_{j=1}^{M}\log C(\lambda^j).\end{aligned}$$

Note that this is equal to $\mathcal{L}_{\boldsymbol{\theta}}$ as defined in (12). The first-order derivatives with respect to α^j and β^j are:

$$\begin{aligned}\frac{\partial\Psi(\boldsymbol{\theta})}{\partial\alpha^j} &= \frac{\sum_{i=1}^{N}\mu_i y_i^j - \alpha^j\sum_{i=1}^{N}\mu_i}{\alpha^j(1-\alpha^j)} - \lambda^j(\alpha^j+\beta^j-1), \\ \frac{\partial\Psi(\boldsymbol{\theta})}{\partial\beta^j} &= \frac{\sum_{i=1}^{N}(1-\mu_i)(1-y_i^j) - \beta^j\sum_{i=1}^{N}(1-\mu_i)}{\beta^j(1-\beta^j)} - \lambda^j(\alpha^j+\beta^j-1).\end{aligned}$$

The second-order derivatives are:

$$\frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\alpha^j\partial\alpha^j} = \frac{\sum_i\mu_i y_i^j\cdot(2\alpha^j-1) - (\alpha^j)^2\sum_i\mu_i}{[\alpha^j(1-\alpha^j)]^2} - \lambda^j \tag{19}$$

$$\begin{aligned}\frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\alpha^j\partial\beta^j} &= \frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\beta^j\partial\alpha^j} = -\lambda^j \\ \frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\beta^j\partial\beta^j} &= \frac{\sum_i(1-\mu_i)(1-y_i^j)\cdot(2\beta^j-1) - (\beta^j)^2\sum_i(1-\mu_i)}{[\beta^j(1-\beta^j)]^2} - \lambda^j \qquad (20) \\ \frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\alpha^j\partial\alpha^k} &= \frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\alpha^j\partial\beta^k} = \frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\beta^j\partial\alpha^k} = \frac{\partial^2\Psi(\boldsymbol{\theta})}{\partial\beta^j\partial\beta^k} = 0,\ \forall j\neq k.\end{aligned}$$

If we arrange all the parameters column-wise as a vector $\{\alpha^1,\beta^1,\ldots,\alpha^M,\beta^M\}$, then the Hessian matrix can be written in a block diagonal form $\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda}) = \mathbf{A}(\boldsymbol{\alpha},\boldsymbol{\beta}) - \mathbf{B}(\boldsymbol{\lambda})$, where matrix $\mathbf{A}$ (a diagonal matrix with entries equal to the first terms in (19) and (20)) depends on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ only, and matrix $\mathbf{B}$ is a block diagonal matrix of the form

$$\mathbf{B}(\boldsymbol{\lambda}) = \begin{pmatrix} \lambda^1 & \lambda^1 & 0 & 0 & \cdots & 0 & 0 \\ \lambda^1 & \lambda^1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \lambda^2 & \lambda^2 & \cdots & 0 & 0 \\ 0 & 0 & \lambda^2 & \lambda^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \lambda^M & \lambda^M \\ 0 & 0 & 0 & 0 & \cdots & \lambda^M & \lambda^M \end{pmatrix}.$$

It is now easy to take the derivative of $\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})$ with respect to λ^j and compute $\sigma(\lambda^j)$. Let $\boldsymbol{\Sigma} = \mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})$, a block diagonal matrix, then we have we have

$$\sigma(\lambda^j) = \text{Tr}\left[\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})\frac{\partial}{\partial\lambda^j}\mathbf{H}(\widehat{\boldsymbol{\theta}}_{\text{MAP}},\boldsymbol{\lambda})\right] = -\boldsymbol{\Sigma}_{2j-1,2j-1} - \boldsymbol{\Sigma}_{2j-1,2j} - \boldsymbol{\Sigma}_{2j,2j-1} - \boldsymbol{\Sigma}_{2j,2j}.$$

That is, $\sigma(\lambda^j)$ is computed by taking the negative of the element-wise sum of the sub-matrix $\boldsymbol{\Sigma}(2j-1:2j,2j-1:2j)$.

Appendix C. Derivatives of the Log Marginal—Categorical Labels

Similarly the second-order derivatives for the categorical case can be written as

$$\begin{aligned} \frac{\partial^2\mathcal{L}}{\partial\alpha^j_{ck}\partial\alpha^j_{ck}} &= -\frac{\sum_i \mu_{ic}\delta(y^j_i,k)}{[\alpha^j_{ck}]^2} - \frac{(C-1)\lambda^j}{C}, \\ \frac{\partial^2\mathcal{L}}{\partial\alpha^j_{ck}\partial\alpha^j_{c'k}} &= \frac{\lambda^j}{C}, \\ \frac{\partial^2\mathcal{L}}{\partial\alpha^j_{ck}\partial\alpha^j_{ck'}} &= \frac{\partial^2\mathcal{L}}{\partial\alpha^j_{ck}\partial\alpha^j_{c'k'}} = 0. \end{aligned} \tag{21}$$

If we rearrange all the parameters in the multinomial term $\boldsymbol{\alpha}^j$ column-wise as a vector of the form $\{\alpha^j_{11},\alpha^j_{21},\ldots,\alpha^j_{C1},\alpha^j_{12},\ldots,\alpha^j_{C2},\ldots,\alpha^j_{CC}\}$, then Hessian matrix for the parameters $\theta = \{\boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^M\}$ can be written in a block diagonal form as $\mathbf{H} = \text{diag}(\mathbf{H}^1,\ldots,\mathbf{H}^M)$, with $\mathbf{H}^j = \text{diag}(\mathbf{D}^j_1,\ldots,\mathbf{D}^j_C)$, where each matrix $\mathbf{D}^j_c$ is a $C\times C$ matrix of the form $\mathbf{D}^j_c = \mathbf{A}_c(\boldsymbol{\alpha}^j) + \mathbf{B}(\lambda^j)$, where $\mathbf{A}_c(\boldsymbol{\alpha}^j)$ is a diagonal matrix with entries equal to the first term in (21) and $\mathbf{B}(\lambda^j) = \lambda^j\left((1/C)\mathbf{e}\mathbf{e}^\top - \mathbf{I}_C\right)$.

$$\mathbf{B}^j_c(\lambda^j) = \frac{1}{C}\begin{pmatrix} -(C-1)\lambda^j & \lambda^j & \lambda^j & \cdots & \lambda^j \\ \lambda^j & -(C-1)\lambda^j & \lambda^j & \cdots & \lambda^j \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \lambda^j & \cdots & \lambda^j & -(C-1)\lambda^j & \lambda^j \\ \lambda^j & \cdots & \lambda^j & \lambda^j & -(C-1)\lambda^j \end{pmatrix}$$

Therefore, $\frac{\partial}{\partial \lambda^j}\mathbf{H}^j$ is a $C^2 \times C^2$ block diagonal matrix with $(1/C)\mathbf{e}\mathbf{e}^\top - \mathbf{I}_C$ on every diagonal. This greatly simplifies the computation of $\sigma(\lambda^j)$.

Since computing the normalization constant $N(\lambda^j)$ is analytically hard we numerically calculate $\delta(\lambda^j)$ by observing that

$$\begin{aligned}\delta(\lambda^j) &= -\frac{2\lambda^j}{N(\lambda^j)}\frac{\partial}{\partial \lambda^j}N(\lambda^j) \\ &= \lambda^j \cdot \frac{\int_S \exp\left(-\frac{\lambda^j}{2}\left\|\left(\mathbf{I}-\frac{1}{C}\mathbf{e}\mathbf{e}^\top\right)\mathbf{A}^j\right\|_F^2\right)\cdot\left\|\left(\mathbf{I}-\frac{1}{C}\mathbf{e}\mathbf{e}^\top\right)\mathbf{A}^j\right\|_F^2 d\mathbf{A}^j}{\int_S \exp\left(-\frac{\lambda^j}{2}\left\|\left(\mathbf{I}-\frac{1}{C}\mathbf{e}\mathbf{e}^\top\right)\mathbf{A}^j\right\|_F^2\right) d\mathbf{A}^j},\end{aligned}$$

where $S = \{\mathbf{A}^j = [\alpha_{ck}^j] \in \mathbb{R}^{C\times C} | \alpha_{ck} \in [0,1], \mathbf{A}^j\mathbf{e} = \mathbf{e}\}$. We compute the integration numerically via sampling.

References

A. Brew, D. Greene, and P. Cunningham. Using crowdsourcing and active learning to track sentiment in online media. In *Proceedings of the 6th Conference on Prestigious Applications of Intelligent Systems (PAIS'10)*, 2010.

B. Carpenter. Multilevel Bayesian models of categorical data annotation. Technical Report available at http://lingpipe-blog.com/lingpipe-white-papers/, 2008.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977.

P. Donmez, J. G. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD 2009: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2009.

P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP'10)*, pages 64–67, 2010.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

G. Parent and M. Eskenazi. Clustering dictionary definitions using amazon mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 21–29. Association for Computational Linguistics, 2010.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

V. C. Raykar and S Yu. Ranking annotators for crowdsourced labeling tasks. In *Advances in Neural Information Processing Systems 24*, pages 1809–1817. 2011.

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 International Conference on Machine Learning (ICML 2009)*, pages 889–896, 2009.

V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, April 2010.

V. S. 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*, pages 614–622, 2008.

P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of venus images. In *Advances in Neural Information Processing Systems 7*, pages 1085–1092. 1995.

R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and Fast—but is it good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 254–263, 2008.

A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Proceedings of the First IEEE Workshop on Internet Vision at CVPR 08*, pages 1–8, 2008.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23*, pages 2424–2432. 2010.

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. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043. 2009.

Y. Yan, R. Rosales, G. Fung, M. Schmidt, G. Hermosillo, L. Bogoni, L. Moy, and J. Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 932–939, 2010.