Crowdsourcing with Contextual Uncertainty

Viet-An Nguyen
Meta

Peibei Shi
Meta

Jagdish Ramakrishnan
Meta

Narjes Torabi
Meta

Nimar S. Arora
Meta

Udi Weinsberg
Meta

Michael Tingley
Meta

ABSTRACT

We study a crowdsourcing setting where we need to infer the latent truth about a task given observed labels together with context in the form of a classifier score. We present Theodon, a hierarchical nonparametric Bayesian model, developed and deployed at Meta, that captures both the prevalence of label categories and the accuracy of labelers as functions of the classifier score. Theodon uses Gaussian processes to model the non-uniformity of mistakes over the range of classifier scores. For our experiments, we used data generated from integrity applications at Meta as well as public datasets. We showed that Theodon (1) obtains 1–4% improvement in AUC-PR predictions on items' true labels compared to state-of-the-art baselines for public datasets, (2) is effective as a calibration method, and (3) provides detailed insights on labelers' performances.

CCS CONCEPTS

• **Information systems → Crowdsourcing**; • **Computing methodologies → Latent variable models**.

KEYWORDS

Bayesian methods; crowdsourcing; probabilistic models

ACM Reference Format:
Viet-An Nguyen, Peibei Shi, Jagdish Ramakrishnan, Narjes Torabi, Nimar S. Arora, Udi Weinsberg, and Michael Tingley. 2022. Crowdsourcing with Contextual Uncertainty. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539184

1 INTRODUCTION

Crowdsourcing is a well studied domain in which tasks are assigned to humans for a variety of applications, including training machine learning models, removing abusive content from social platforms, and measuring the prevalence of labels in a population. A key challenge is that humans are noisy decision makers [15], thus the obtained labels may be incorrect. A common mitigation strategy is to collect labels from several labelers, and aggregate them while taking into account the accuracy of the individual labelers.

Early work characterized labeler accuracy using a confusion matrix for each labeler [7], with numerous extensions, e.g. [13, 14, 35]. An intuitive extension is to model labeler accuracy within some context. This context improves the ability to aggregate labels and provide a higher resolution into how each labeler performs. Prior work by Yan et al. [48, 49] captures this dependency between labeler accuracy and tasks' features using logistic regression. However, Wenger et al. [44] later showed that Gaussian processes (GPs) are more effective than logistic regression, as GPs allows non-linear relationship that cannot be modeled by logistic regression.

Our work is focused on crowdsourcing for prevalence estimation, specifically of content that violates community standard of online platforms, such as Meta[1] [29] and Google[2] [21]. We propose Theodon, a Bayesian non-parametric model that learns the *prevalence of label categories* and the *accuracy of labelers* as functions of a given context. Our model leverages GPs as flexible priors to model the prevalence, sensitivity, and specificity functions.

Figure 1 illustrates this setup. We start with the entire population (content, accounts or other entities on the platform) over which we want to measure the prevalence of violations. Due to the large volume, it is impossible to label the entire population; and due to the low prevalence of violations (often below 0.1%), sampling uniformly from the population would result in few labeled violations. Thus, we upsample likely violations, which are sent to one or more labelers. This upsampling process is done using a classifier that predicts the likelihood of a violation for each entity in the population. We note that this is different from enforcement classifiers that remove violating content with high certainty.

The upsampling classifier, which is trained using content features, provides a very natural context for learning both prevalence and labelers' performance. The labels together with the classifier score is passed to Theodon, which infers labelers' performance, and aggregated labels. These, in turn, enable downstream applications, specifically prevalence measurement, classifier calibration, and labelers' accuracy measurement.

Our focus in this work is on binary labels (i.e., violating or not violating), and a single feature for context – the output of a classifier capturing the (potentially uncalibrated) likelihood of having the "positive" label. However, GPs enable easy extensions to settings with multiple classes and a feature vector. In addition, the prevalence function estimates, which maps the raw classifier scores to the calibrated probabilities, allows Theodon to be an effective calibration method under the presence of labeling error.

KDD '22, August 14–18, 2022, Washington, DC, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539184

[1] https://transparency.fb.com/policies/community-standards/
[2] https://www.youtube.com/howyoutubeworks/policies/community-guidelines/

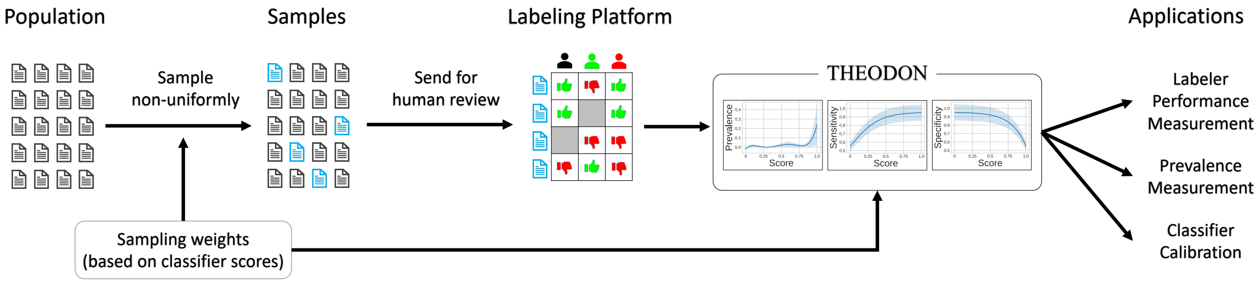


Figure 1: Overview of THEODON's crowdsourcing setup

Key Contributions

- We present THEODON, a novel Bayesian non-parametric model that is deployed in production at Meta to aggregate crowd-sourced labels using Gaussian processes that capture the dependencies of the label's prevalence and the labelers' accuracy on the task's context, represented by a classifier score.
- Using Meta's crowdsourcing data and publicly available datasets, we study the effectiveness of THEODON compared with state-of-the-art baselines on a range of applications.
- We show that compared to state-of-the-art baselines, THEODON (1) obtains 1–4% improvement in AUC-PR predictions on items' true labels, (2) is effective as a calibration method, and (3) learns detailed accuracy properties of the labelers, enabling better auditing and training of their labeling performances.

Related Work

One of the earliest works aggregating multiple labels was by Dawid and Skene (D&S) [7], which in its binary form, estimates the *prevalence* of labels, and the *sensitivity* and *specificity* of labelers. Specifically, prevalence is the rate at which items with true positive labels occur, and sensitivity and specificity capture the true positive and true negative rates of each labeler, respectively. This D&S model has inspired many modeling extensions [4, 30, 31, 51].

The most related works here are extensions to advance the way prevalence and/or sensitivity/specificity can be captured in various settings. Raykar et al. [36, 37] proposed a method to model the prevalence using a logistic regression where each item's feature vector is available. Rodrigues et al. [40] proposed a GP-based classifier to capture the prevalence function, with sensitivity and specificity as fixed parameters; while Ruiz et al. [41] extended the work by modeling both sensitivity and specificity as stochastic latent variables. Other work also focused on supervised learning, but did so for continuous labels using GP-based regression model [9]. The closest work to ours is the model introduced by Yan et al. [48, 49] which captures feature-dependent sensitivity/specificity using logistic regression.

Other works focus on using the classifier as an additional noisy label that can be leveraged to improve label aggregation. Snorkel [35] is a weak-supervised framework that enables users to provide "labeling functions", and a binarized classifier score here can also be used as a labeling function. Another work, CLARA [29], extends the D&S model to use the continuous score as another labeler. In addition, by capturing the likelihood of having a "positive" label, the score also represents the item difficulty, for which previous works [4, 43, 45] jointly inferred from the labels. In this work, we focus on settings where the number of labels per item is typically small and explicitly do not infer the item difficulty.

Notation	Description
N	Number of items
L_i	Number of labels that item i receives
A	Number of unique reviewers/labelers
$a_{i,j}$	j^{th} reviewer of item i
$r_{i,j}$	j^{th} label of item i (by reviewer $a_{i,j}$)
s_i	classifier score associated with item i
y_i	True label category of item i
$\theta(s)$	Prevalence function
$\psi_a(s)$	Sensitivity function of labeler a
$\phi_a(s)$	Specificity function of labeler a
α and ρ	Hyperparameters defining the kernel of GP

Table 1: Main notations used in this paper

Learning the accuracy of labelers is an important component of crowdsourcing models [13, 14]. In addition to using confusion matrices to characterize labeling errors, other approaches have been proposed including using item response theory [45], signal detection theory [43] or minimax entropy principle [52]. Another line of work tries to learn the structure of labelers' errors by using a hierarchical prior to improve the estimate for individuals with few labels [4], using clustering approaches to group similar labelers [26, 42], modeling common confusions [6], and capturing the correlations among labelers [3, 19, 28]. Recent work use deep neural networks to capture annotators' labeling behaviors [10, 38].

2 BACKGROUND

2.1 Data Properties

Our crowdsourced data contains a set of items, each is associated with a continuous score and one or more labels provided by human labelers. For simplicity, we assume that the task requires a binary outcome, thus human labels are binary and there is a single score.

More formally, the input dataset consists of N items, each receives L_i binary labels denoted by $\{r_{i,j}\}$. There are A unique labelers (annotators) who assign labels to a subset of the items. The j^{th} label $r_{i,j}$ of item i is provided by labeler $a_{i,j} \in [1, A]$. In addition, each item i also has a continuous *score* $s_i \in [0, 1]$, which captures the likelihood of i being in the positive class. We assume that we do not have access to other features of each item, and if they do exist, they are fed to the classifier that outputs the scores s_i. In addition, in practice labeling capacity is very limited, therefore each item is typically labeled by only a small number of reviewers, e.g., $L_i \leq 3$.

2.2 Gaussian Processes

A Gaussian process (GP) [34] is a stochastic process which defines a probability distribution over the function $p(f \mid x) = \mathcal{GP}\ (m(x), K(x, x'))$ where $m(x)$ and $K(x, x')$ denote the mean and covariance function respectively. One commonly used kernel is the *exponentiated quadratic kernel* which is defined as

$$K_{\alpha,\rho}(x, x') = \alpha^2 \exp\left(-(x - x')^2/2\rho^2\right) + \sigma^2 I_N \tag{1}$$

where α^2 is the variance determining the distance of the function away from the mean, ρ is the length scale controlling the smoothness of the function, σ^2 is the variance modeling the observation noise, and I_N is the identity matrix. As a default, we set $\sigma = e^{-9}$ to represent very little observation noise and exclude it from the notation of $K_{\alpha,\rho}$ for brevity.

Leveraging GPs to improve crowdsourcing models has been an active area of research. One key direction is to treat the problem of learning from multiple annotators a special case of supervised learning and use GPs to build either a regressor [9] or a classifier [22, 40, 41] to map item features to a latent ground truth label. These works use either a Gaussian distribution or a confusion matrix to characterize labeling errors, which is independent of the input item features. Another line of research focuses on developing scalable and efficient learning methods [24], while others apply GP-based approaches to different domains including classifying remote sensing images [25] and detecting signal glitches [23].

2.3 Classifier Calibration

A classifier is considered *perfectly calibrated* if the confidence in its class prediction matches the probability of its prediction being correct. More formally, this means $\mathbb{E}[\mathbf{1}_{\hat{y}=y} \mid \hat{z}] = \hat{z}$ where y is the true class, $\hat{y}$ is the predicted class, and $\hat{z}$ is the confidence associated with the prediction. The goal of *calibrating a classifier* is to transform its raw scores to the true correctness probabilities.

Developing methods for classifier calibration is an active research area, especially that many modern deep neural networks are shown to be overconfident and miscalibrated [11, 12, 18]. Some popular calibration methods include Platt scaling [20, 32], isotonic regression [50], Beta calibration [17], and BBQ [27]. Recent work introduces GPcalib [44], a non-parametric calibration method using GP which is shown to be effective for various base classifiers. In this paper, we extend the idea to consider and incorporate labeling error using crowdsourced data for classifier calibration.

3 MODELING CROWDSOURCED DATA WITH GAUSSIAN PROCESSES

3.1 Generative Model

Following a rich body of research on crowdsourcing models [31], we take a Bayesian probabilistic approach to define different latent variables and the generative process of the observed data (Figure 2). Specifically, the observed data include N items, each has L_i binary labels $\{r_{i,j}\}$ given by labelers $\{a_{i,j}\}$ and a continuous score s_i.

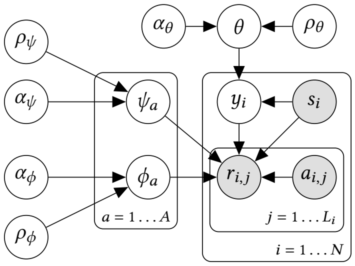


Figure 2: Graphical representation of Theodon. Nodes are random variables (shaded ones are observed), edges are probabilistic dependencies, and plates represent repetition.

Similar to the D&S model and many of its extensions, we define three main latent variables: *prevalence*, *sensitivity*, and *specificity*. The key modeling novelty here is that our proposed model Theodon assumes all three quantities are dependent on the score s, and are captured by the *prevalence function* $\theta(s)$, the *sensitivity function* $\psi(s)$, and the *specificity function* $\phi(s)$, respectively. We also assume that each item i has a true, but latent label, denoted by y_i.

In this work, we do not assume anything about the selection of the scores and the labelers. Hence, we are interested in the joint probability conditional on $\mathbf{a}$ and $\mathbf{s}$ which is given by

$$p(\mathbf{r}, \mathbf{y}, \theta, \psi, \phi \mid \mathbf{a}, \mathbf{s}) =$$
$$p(\mathbf{r} \mid \mathbf{y}, \psi, \phi, \mathbf{a}, \mathbf{s}) \cdot p(\mathbf{y} \mid \theta, \mathbf{s}) \cdot p(\theta \mid \mathbf{s}) \cdot p(\psi \mid \mathbf{s}) \cdot p(\phi \mid \mathbf{s})$$

$$\text{where} \quad p(\psi \mid \mathbf{s}) = \prod_{a=1}^{A} p(\psi_a \mid \mathbf{s}); \;\; p(\phi \mid \mathbf{s}) = \prod_{a=1}^{A} p(\phi_a \mid \mathbf{s});$$
$$p(\mathbf{y} \mid \theta, \mathbf{s}) = \prod_{i=1}^{N} p(y_i \mid \theta(s_i));$$
$$p(\mathbf{r} \mid \mathbf{y}, \psi, \phi, \mathbf{a}, \mathbf{s}) = \prod_{i=1}^{N} \prod_{j=1}^{L_i} p(r_{i,j} \mid y_i, \psi_{a_{i,j}}(s_i), \phi_{a_{i,j}}(s_i)).$$

3.1.1 *Prevalence.*

We assume the prevalence of the positive label depends on the input score s. This dependency is captured by the prevalence function $\theta(s)$, which is modeled by first drawing a function f_θ from a GP prior and then transforming it into a probability distribution using a link function

$$f_\theta \sim \mathcal{GP}\left(m_\theta, K_{\alpha_\theta,\rho_\theta}\right); \qquad \theta(s) = \Omega(f_\theta(s))$$

where $\Omega(x)$ denotes a link function (typically logit or probit) which transforms $x \in \mathcal{R}$ into the $[0, 1]$-range. This $\theta(s)$ function essentially maps the input scores to the true probabilities, which enables Theodon to calibrate the base classifier.

3.1.2 *Sensitivity and specificity.*

We assume that the labeling mistakes each labeler makes depend on both the true label of each item and its associated input score. More specifically, each labeler a is characterized by a sensitivity function $\psi_a(s)$ and a specificity function $\phi_a(s)$, which respectively define the true positive rate and true negative rate of labeler a at each score s.

$$f_{\psi_a} \sim \mathcal{GP}\left(m_\psi, K_{\alpha_\psi,\rho_\psi}\right); \qquad \psi_a(s) = \Omega(f_\psi(s))$$
$$f_{\phi_a} \sim \mathcal{GP}\left(m_\phi, K_{\alpha_\phi,\rho_\phi}\right); \qquad \phi_a(s) = \Omega(f_\phi(s))$$

3.1.3 *Labels.*

Given θ, ψ, and ϕ, the generative processes for the true labels $\mathbf{y}$ and the observed labels $\mathbf{r}$ are similar to the those in the D&S model. For an item i, $y_i \sim \text{Bern}(\theta(s_i))$ and

$$r_{i,j} \sim \begin{cases} \text{Bern}(\psi_{a_{i,j}}(s_i)) & \text{if } y_i = 1, \\ \text{Bern}(1 - \phi_{a_{i,j}}(s_i)) & \text{if } y_i = 0. \end{cases}$$

For simplicity, we are focusing on binary labels with a single classifier score associated with each item. In principle, it is straightforward to extend Theodon for multi-class labels by using a *softmax* link function instead. It is also straightforward to extend to settings where there are more than one related classifier scores, or in general, there exists a feature vector for each item by using GPs with multi-dimensional kernel

$$K_{\alpha,P}(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T P^{-1} (\mathbf{x} - \mathbf{x}')\right)$$

where $P = \text{diag}(\rho_1^2, \rho_2^2, \cdots, \rho_d^2)$ comprises the length scales along d dimensions.

3.2 GP Hyperparameters

One approach for setting the hyperparameters of the GPs could be to maximize the marginal likelihood of the data. However, as pointed out in [1] such a maximization approach can lead to very unsmooth GP functions. An alternate approach presented in that article is to put priors on the hyperparameters, and either perform regularized maximization or simply include these as latent variables in the model. We take the latter approach in our work. The settings for these priors can be highly subjective and often rely on domain expertise. We use the following prior distributions, where the priors for α_* and σ_* are fairly generic while the prior for ρ_* is chosen to put most of the mass around a value of 0.1 which results in smooth GP functions. More specifically,

$$\alpha_* \sim \text{Normal}(0, 2) \quad \alpha_* > 0$$
$$\sigma_* \sim \text{Normal}(0, 1) \quad \sigma_* > 0$$
$$\rho_* \sim \text{Inverse-Gamma}(3, 0.3)$$

For the mean values, we assume a constant mean function. Since we subjectively expect that most labelers tend to be fairly accurate, the prior mean of sensitivity and specificity is set to 0.8, while for prevalence we use a uninformative prior.[3] Therefore we have

$$m_\theta = \Omega^{-1}(0.5); \quad m_\psi = \Omega^{-1}(0.8); \quad m_\phi = \Omega^{-1}(0.8)$$

where $\Omega(x)$ is the link function.

3.3 Posterior Inference

We perform posterior inference using Hamiltonian Monte Carlo (HMC) by implementing Theodon (and all other model-based baselines) in Stan [5]. However, for Gaussian processes, a direct implementation in Stan would lead to a runtime complexity of $O(N^3)$ and storage cost of $O(N^2)$, which can be intractable for most datasets. In the interest of scalability, we propose to use an inducing point sparse approximation (see for example, [33]) with M inducing points located uniformly on the grid, which is $[0, 1]$ in our setting. This approximation reduces the runtime complexity to $O(M^2 N + M^3)$ with $O(MN + M^2)$ storage. In fact, this runtime complexity is still too high in many applications. We further use the Structured Kernel Interpolation (SKI) method as presented in [46] with linear interpolation between $c = 2$ nearest inducing points. The SKI method reduces the runtime cost to $O(N + M^2)$ with $O(M^2)$ storage, which is quite reasonable. In higher dimensions, the SKI paper describes a method to exploit the Kronecker structure of the kernel, K, to keep the complexity w.r.t. M to $O(dM^{1+\frac{1}{d}})$.

4 EVALUATION

We begin our evaluation using data from real crowdsourcing applications at Meta. Among the numerous applications of crowdsourcing at Meta, in this section, we focus on the problem of prevalence measurement, which has a setup similar to the one depicted in Figure 1. Since we want to study the performance of Theodon under a range of different scenarios, we derive distributions from the logged data, and use them to explore a range of operating parameters, which enables us to have complete control over the different parameters, while using realistic crowdsourcing scenarios.

4.1 Deployment

Figure 3 illustrates an overview of how Theodon was deployed at scale in production at Meta. The content is sampled in a nonuniformly way from the population and sent to a centralized labeling platform for human review. The human labels and the sampling weights are stored in the distributed file system (based on Hadoop), which are then used to fit Theodon models. The posterior inference is done using Stan [5] on the FBLearner platform [8]. The output estimates from Theodon are stored back to the distributed file system and then used for monitoring and analytical purposes, including: (1) measuring and monitoring the labeler performances using the sensitivity and specificity functions, (2) measuring and monitoring the prevalence of the positive class by aggregating the prevalence function, and (3) calibrating the input classifier using the prevalence function.

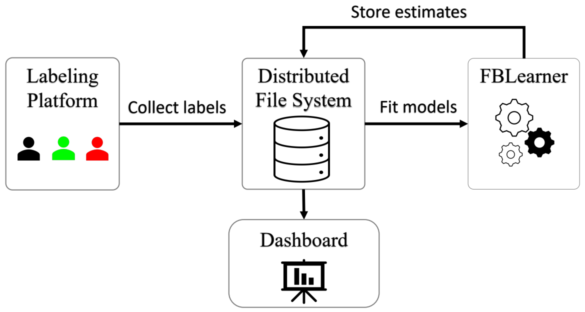
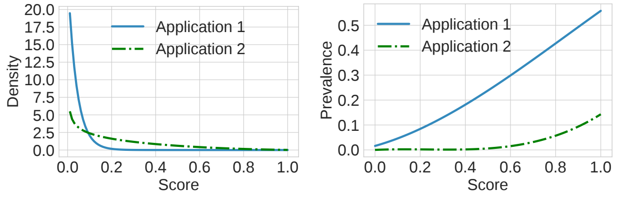


Figure 3: Overview of how Theodon was deployed at Meta

We use this deployment and logged data to generate the distributions, on which we perform the analysis detailed in this section.

[3] The same prior mean of 0.8 for sensitivity and specificity is also used for other baseline models in our experiments.



(a) Score distributions S^* **(b) Prevalence functions θ^***

Figure 4: Score distributions and prevalence functions obtained from two crowdsourcing applications at Meta

4.2 Data Generation Process

Given known (obtained from the logged data) score distribution S^*, prevalence function θ^*, global sensitivity function ψ^*, and global specificity function ϕ^*, Algorithm 1 describes the data generation process in details. First, we generate a score for each item from S^*. We then compute the mean function $\Omega^{-1}(\psi^*)$ (and $\Omega^{-1}(\phi^*)$) and covariance matrix $K_{\alpha,\rho}$ for the multivariate Gaussian distributions, and use them to sample the sensitivity (and specificity) function of each labeler. Finally, for each item, we simulate a true binary label, sample its labelers uniformly at random, and generate the crowdsourced labels based on its score, true label, and the labelers' sensitivity and specificity functions.

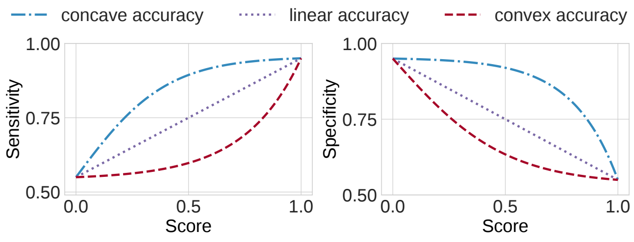


Figure 5: Sensitivity (ψ^*) and specificity (ϕ^*) functions

To generate data that closely reflects reality, we obtain the score distribution S^* and true prevalence function θ^* by collecting data from two crowdsourcing applications at Meta. For each application, there exists a classifier which provides a continuous score. We extract the empirical distribution of the classifier scores and fit a Beta distribution to generate S^* for each application (Figure 4a). Similarly, we obtain the true prevalence function θ^* by fitting a third degree polynomial function on the empirical prevalence function (Figure 4b).

To simulate different types of labeling errors, we generate three global sensitivity and specificity functions (see Figure 5): a *linear* function representing a simple correlation between scores and accuracy, and two non-linear functions, one *concave*, indicating high levels of accuracy and one *convex*, indicating lower levels of accuracy. We use $K_{0.5,0.5}$ as the covariance matrix to generate the per-labeler sensitivity and specificity functions, which are relatively smooth and not too far away from the respective global functions. The resulting per-labeler functions are detailed in the supplementary material.

Algorithm 1: Data generation procedure

```
Input: score distribution $S^*$, prevalence function $\theta^*$, global
       sensitivity function $\psi^*$, global specificity function
       $\phi^*$, $N$, $A$, $L_i$, $\alpha$, $\rho$
foreach item $i$ in $[1, N]$ do
    Draw a score $s_i$ from $S^*$;
foreach labeler $a$ in $[1, A]$ do
    Draw $f_{\psi_a} \sim \mathcal{N}\left(\Omega^{-1}(\psi^*), K_{\alpha,\rho}\right)$;
    Draw $f_{\phi_a} \sim \mathcal{N}\left(\Omega^{-1}(\phi^*), K_{\alpha,\rho}\right)$;
    Define $\psi_a = \Omega(f_{\psi_a})$ and $\phi_a = \Omega(f_{\phi_a})$ where
      $\Omega(x) = 1/(1 + \exp(-x))$;
foreach item $i$ in $[1, N]$ do
    Draw a true label $y_i \sim \mathrm{Bern}(\theta^*(s_i))$;
    foreach observed label $j$ in $[1, L_i]$ do
        Choose a labeler $a_{i,j}$ uniformly at random;
        Draw $r_{i,j} \sim \begin{cases} \mathrm{Bern}(\psi_{a_{i,j}}(s_i)) & \text{if } y_i = 1, \\ \mathrm{Bern}(1 - \phi_{a_{i,j}}(s_i)) & \text{if } y_i = 0. \end{cases}$
```

For each combination of S^*, θ^*, ψ^*, and ϕ^*, we generate 50 datasets. Each dataset contains $N = 5000$ items, $A = 10$ unique labelers, and each item i has $L_i = 3$ observed labels.

4.3 Evaluation Metrics

We evaluate the ability of Theodon and the baselines (Table 2) to recover the true prevalence, sensitivity, and specificity functions from observed data. In particular, for prevalence estimate, our model produces (1) a mean function $\hat{\theta}$ and (2) the lower bound function $\hat{\theta}_L$ and upper bound function $\hat{\theta}_U$ to capture the 95% confidence intervals. To measure the performance of the estimate with respect to the true function θ^*, we compute the *absolute error* and the *coverage* at different scores in a pre-defined set and take the average. More specifically, let $\mathcal{S} = \{0, 0.01, 0.02, \cdots, 1.0\}$ denote the set of scores to be evaluated at, we compute the follow two metrics:

- **Mean absolute error** (MAE) of the mean $\hat{\theta}$:
$$\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left| \hat{\theta}(s) - \theta^*(s) \right|$$
- **Coverage rate** of the confidence interval (CI):
$$\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathbf{1}\left\{ \theta^*(s) \in [\hat{\theta}_L(s), \hat{\theta}_U(s)] \right\}$$

For sensitivity and specificity, we compute the same metrics for each labeler and report the mean values averaged over all labelers.

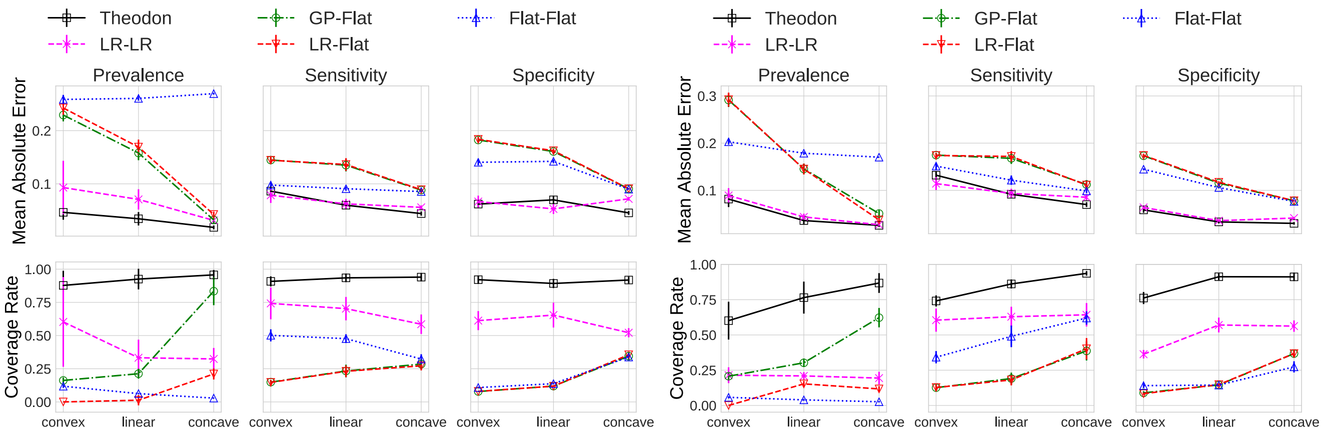
4.4 Results

For each configuration of our simulations, we average the evaluation metrics over 50 datasets and report the results in Figure 6. We compare our model against several baselines detailed in Table 2. Additional details about the generative process of each baseline can be found in the supplementary material.

Overall, Theodon and Lr-Lr, with the ability to capture the dependency between sensitivity and specificity on the scores, outperform the other methods for both datasets across all metrics. Both Theodon and Lr-Lr's performances are relatively consistent

Baseline	Prev.	Sens. & Spec.	Description
Fl-Fl [7]	Flat	Flat	Models prevalence, sensitivity, and specificity using scalar probabilities, independent of the input score, similar to the Bayesian version of D&S model and its extensions [7, 16, 30]
Lr-Fl [36, 37]	Lr	Flat	Models prevalence using a logistic regression, sensitivity and specificity using scalar probabilities, similar to Raykar et al.'s [36, 37], with the score as the only feature
Gp-Fl [40, 41]	Gp	Flat	Models prevalence using GP, sensitivity and specificity using scalar probabilities. Similar to Rodrigues et al. [40] that modeled sensitivity and specificity as point parameters, and Ruiz et al. [41] that used stochastic variables
Lr-Lr [48, 49]	Lr	Lr	Models prevalence, sensitivity and specificity using logistic regression. This is based on Yan et al.'s [48, 49] and is closest to our work

Table 2: Baseline methods we use in the paper, each is named X-Y, X is the model for prevalence and Y is the model for specificity and sensitivity. Theodon can be described Gp-Gp, as it uses GP to model the prevalence, sensitivity and specificity.



(a) Application 1

(b) Application 2

Figure 6: Simulation results of Theodon and the baselines on two datasets generated using real distributions

across the three different types of labeling errors, showing their robustness in capturing sensitivity and specificity functions with different shapes. For the mean function estimates, the MAEs of the two methods are similar, except for Application 1's prevalence function which changes much more rapidly w.r.t. the score compared to that of Application 2. For the coverage rate, however, Theodon outperforms Lr-Lr significantly, which shows the effectiveness of our method in capturing the uncertainty in its estimates for prevalence, sensitivity, and specificity functions.

5 EVALUATION USING PUBLIC DATA

In this section, we compare our approach to baselines on publicly available data. We study the performance of approaches when varying the performance of classifiers, representing context quality.

5.1 Evaluation Setup

Datasets. We use two public datasets (collected and released by Rodrigues et al. [39]). The first dataset, *Sentiment*, contains 4,999 sentences from movie reviews, each of which has a ground truth *positive* or *negative* label. Each sentence is sent to Amazon Mechanical Turk (AMT) to collect crowdsourced binary labels from multiple annotators. The second dataset, *Music*, contains 30s length samples from 700 songs, each of which is categorized into one of 10 different genres: classical, country, disco, hiphop, jazz, rock, blues, reggae, pop, and metal. Each song is also sent to AMT to collect crowdsourced labels and has a 124-dim feature vector. We transform this dataset into 10 separate binary datasets, one for each music genre. More details on the datasets are in the supplementary materials.

Generating classifier scores. We generate the scores by training different classifiers using scikit-learn [2]. For the *Sentiment* dataset, we extract TF-IDF features for each sentence; while for the *Music* dataset, we use the 124-dim feature vector provided. Since the *Music* dataset contains 10-class labels, convert it into 10 binary datasets and train a classifier for each binary label.

Number of labels per item. To study the performance of different approaches using different number of labels per item, for each item i, we sample $\min(l, L_i)$ labels without replacement from the original L_i labels, with $l \in \{3, 5, 7, 9\}$.

Evaluation metrics. For each configuration, we randomly split the data into training and test sets with an 80/20 ratio and repeat the process 5 times. For each split, we fit each model using the training

	Metric	l	Base	MV	FV	Snorkel	Fl-Fl	Lr-Fl	Gp-Fl	Lr-Lr	Theodon
Sentiment	ECE	3	0.2304	0.1360	0.0754	0.0989	0.0952	0.0855	0.0814	0.0729	**0.0661**
		9	0.2394	0.1116	0.0887	0.0765	0.0842	0.0725	0.0726	0.0704*	**0.0656**
	AUC-PR	3	0.9210	0.8931	0.9169	0.9440	0.9283	0.9578	0.9604*	0.9641*	**0.9649**
		9	0.9345	0.9151	0.9498	0.9694	0.9536	0.9701	0.9716	0.9718	**0.9771**
Music	ECE	3	0.1816	0.0580	0.0514	0.0649	0.0487	0.0424*	**0.0403**	0.0470	0.0413*
		7	0.1835	0.0570	0.0635	0.0582	0.0448	0.0420*	**0.0413**	0.0443	0.0423*
	AUC-PR	3	0.7245	0.7139	0.7094	0.7808	0.7719	0.8276	0.8513*	0.8276	**0.8619**
		7	0.7660	0.7261	0.7399	0.8132	0.7906	0.8378*	0.8474*	0.8329*	**0.8515**

Table 3: Results averaged over 5 training/test splits on the *Sentiment* and *Music* datasets when varying the number of labels per item l. Bold denotes the best results and * denotes the results within one standard deviation of the best value per row.

data (using only the score and crowdsourced labels), and apply them on the test set. We compute the AUCs of the P/R (AUC-PR) curves by comparing predictions against the *ground truth* labels.

In addition, to evaluate the effectiveness of Theodon as a calibration method, we compute the **expected calibration error** (ECE) [11, 27, 44]. More specifically, with B fixed uniform bins between 0 and 1, the ECE is defined as $\sum_{b=1}^{B}(N_b \cdot |o_b - e_b|)/N$ where o_b is the true fraction of positive items in bin b, e_b is the mean of the calibrated probabilities for the items in bin b, and N_b is the number of items that fall into bin b. Following Wenger et al. [44], we use 100 equally spaced bins.

For the *Sentiment* dataset, we report the averaged values for all metrics over the 5 splits of training and test sets. For the *Music* dataset, we first average over the 10 binary datasets before reporting the averaged values over the 5 data splits.

***Baselines*.** We consider the same set of baselines described in Table 2, with the addition of the following:

- Majority Vote (MV): uses the majority voted label as the aggregated label. This is often used in practice as it is a simple and intuitive baseline, and requires no “training” phase.
- Fraction Vote (FV): uses the fraction of positive labels as the score for each item. Similar to MV, no training is needed.
- Snorkel [35]: uses matrix-completion to estimate the conditional probabilities, which are then used to re-weight and aggregate the labels. We assign Snorkel’s “labeling functions” to each labeler, and an additional labeling function that represents the binarized classifier score.

5.2 Varying Number of Labels per Item

First, we study the performance of Theodon compared with different baselines when the number of labels per item varies. Here, we use logistic regression to generate the classifier scores. Table 3 shows the AUC-PR and ECE, averaged over 5 splits of training and test sets for both the *Sentiment* and *Music* datasets.

Overall, the results clearly highlight the need to model labeling errors (and is consistent with prior work [31]) – the methods that incorporate labeling errors and classifier scores consistently outperform MV, FV and Snorkel. We also see that for all approaches, increasing the number of labels per item generally improves the performance for the *Sentiment* dataset. This trend is, however, less clear for the *Music* dataset, which is partly due to its small size and class imbalance.

As expected, MV is generally the worst performing method, and FV is significantly better but still underperforms the more complex approaches. Consistently, Snorkel outperforms both MV and FV in predicting the items labels, but performs comparably or worst as a calibration method.

Considering the top performing models, the results shows that modeling the dependency between the input score and either (a) only the prevalence (Lr-Fl, Gp-Fl) or (b) both the prevalence and sensitivity/specificity (Lr-Lr, Theodon), clearly outperforms the Fl-Fl model (that assumes no score dependency).

Gp-Fl outperforms Lr-Fl in all metrics on both datasets. The main difference between the two methods is that, as their names suggested, Gp-Fl uses a GP to capture the prevalence function while Lr-Fl uses a logistic regression. It is worth noting that, as calibration methods, Lr-Fl and Gp-Fl are equivalent to Platt scaling [32] and GPcalib [44] respectively, but with labeling error being incorporated and captured by a flat function. The results here are consistent with prior work [44] which shows the effectiveness of using GP in calibrating classifier scores compared with logistic regression.

Our proposed method Theodon consistently provides the best results in all metrics for the *Sentiment* dataset. For the *Music* dataset, Theodon outperforms other baselines in AUC-PR, but does not perform better than Gp-Fl in ECE. The results confirm that improving both accuracy and calibration can be conflicting objectives [44] and suggest that for small datasets like *Music*, it might be more beneficial to use a simple scalar to capture the sensitivity/specificity of labelers when the method is used for calibration.

5.3 Varying the Base Classifier

We now study how varying the base classifier affects the performance of Theodon and other baselines. We consider five types of base classifiers: 1-layer neural network (NN), logistic regression (LR), naive Bayes (NB), random forest (RF), and AdaBoost (AB) provided by scikit-learn [2]. In this experiment, we use all labels available for each item. Table 4 shows the results on ECE and AUC-PR, again averaged over 5 data splits for the two *Sentiment* and *Music* datasets. Details on the base classifiers’ implementations and additional results can be found in the supplementary material.

All the base classifiers perform reasonably well in predicting the items' ground truth labels. However, their performances as calibration methods vary significantly. For the *Sentiment* dataset, NN provides the best ECE (which is consistent with prior work [44]), followed by RF and AB; while LR and NB perform the worst. However, for the *Music* dataset, AB is significantly worse in ECE.

Consistent with the results in Section 5.2, Snorkel improves upon the base classifier, MV, and FV in AUC scores but generally performs worse as a calibration method. We also see similar improvements in all metrics for both datasets when capturing the dependencies of prevalence/sensitivity/specificity w.r.t. to the classifier compared with Fl-Fl.

For all metrics across the two datasets, Theodon's results are consistently the best or within one standard deviation of the best, which shows the effectiveness of Theodon in both predicting per-item labels and calibrating the base classifiers.

5.4 Sensitivity and Specificity Estimates

One main advantage of Theodon is to provide additional insights into how the accuracy of each individual labeler changes when labeling items with different scores. Since the sensitivity and specificity functions are unknown in the real-world datasets, we focus this section on a qualitative analysis.

An intuitive method to compute the sensitivity and specificity of labelers when the score and ground truth labels are available is to bucketize the scores and compute the per-bucket sensitivity and specificity. Figure 7a shows the counts for a selected labeler $\mathscr{A}$ (`WorkerId=A207OR9LV0PAPY`). The top and bottom plot shows the labels provides by $\mathscr{A}$ when the ground truth is 1 and 0, respectively. We selected this specific labeler since they have the most labels in the *Sentiment* dataset, which should enable us to obtain a good estimate of their performance.

The figure depicts a clear challenge in this process – even for the labeler with the most labels in the dataset, there are score buckets that have almost no labels, which makes computing accuracy scores in these regions impossible, or highly inaccurate. To illustrate, we used the true and observed labels to compute, for each score bucket, the sensitivity and specificity and their Wilson score 95% CIs [47]. The resulting sensitivity and specificity are labeled by GT in Figures 7b and 7c, respectively. As shown, due to the lack of labels, it is impossible to compute the sensitivity of $\mathscr{A}$ in the lowest score bucket, while the specificity's CI in the highest score bucket is so wide making it practically unusable. Much narrower CIs are obtained by the estimates from Gp-Fl, which are independent of the score. As the figures show, these estimates capture mostly the true sensitivity and specificity for the buckets with the most labels. While these methods make it possible to draw broad conclusions about the accuracy of the labeler, they fail to leverage the contextual information about each task.

Theodon on the other hand provides a continuous function, while being able to capture the trends of GT's estimates well, with the sensitivity increasing while specificity decreasing as the score increases. In addition, lack or low number of labels is captured well using Theodon's CIs. We note that similar results are observed (not shown due to space constraint) when using Lr-Lr as it also models the specificity and sensitivity as functions of the scores.

6 CONCLUSION

In this paper, we introduce Theodon, a Bayesian non-parametric model to aggregate labels in crowdsourced data by capturing the dependencies of label's prevalence and labelers' sensitivity and specificity on the input classifier score using Gaussian processes. We conduct extensive empirical studies on simulated generated based on real applications at Meta as well as real-world crowdsourced data to show the effectiveness of Theodon compared with various baselines on a range of tasks. In addition to obtaining good per-item aggregated labels, Theodon is effective in calibrating the base classifier under the presence of labeling error and provides useful insights into how the prevalence of labels and the mistakes that labelers might change with respect to the input classifier score. Understanding and leveraging these insights for labeling quality management is a fruitful direction for future work.

REFERENCES

[1] Michael Betancourt. 2020. Robust Gaussian Process Modeling. https://betanalpha.github.io/assets/case_studies/gaussian_processes.html

[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[3] Peng Cao, Yilun Xu, Yuqing Kong, and Yizhou Wang. 2019. Max-MIG: an Information Theoretic Approach for Joint Learning from Crowds. In *ICLR*.

[4] Bob Carpenter. 2008. Multilevel Bayesian models of categorical data annotation. *Unpublished manuscript* 17, 122 (2008), 45–50.

[5] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: A probabilistic programming language. *Journal of Statistical Software*.

[6] Zhendong Chu, Jing Ma, and Hongning Wang. 2021. Learning from Crowds by Modeling Common Confusions. *AAAI* 35, 7 (2021), 5832–5840.

[7] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics* (1979).

[8] Facebook. 2016. *Introducing FBLearner Flow: Facebook's AI backbone*. https://code.fb.com/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/

[9] Perry Groot, Adriana Birlutiu, and Tom Heskes. 2011. Learning from Multiple Annotators with Gaussian Processes. In *ICANN*. 159–164.

[10] Melody Y. Guan, Varun Gulshan, Andrew M. Dai, and Geoffrey E. Hinton. 2017. Who Said What: Modeling Individual Labelers Improves Classification. In *AAAI*.

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML*. 1321–1330.

[12] M. Hein, M. Andriushchenko, and J. Bitterwolf. 2019. Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem. In *CVPR*. 41–50.

[13] Panagiotis G Ipeirotis, Foster Provost, Victor S Sheng, and Jing Wang. 2014. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery* 28, 2 (2014), 402–441.

[14] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on Amazon Mechanical Turk. In *ACM SIGKDD Workshop on Human Computation*.

[15] Daniel Kahneman, Andrew M. Rosenfield, Linnea Gandhi, and Tom Blaser. 2016. *Noise: How to Overcome the High, Hidden Cost of Inconsistent Decision Making*. https://hbr.org/2016/10/noise

[16] Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *AISTATS*. 619–627.

[17] Meelis Kull, Telmo Silva Filho, and Peter Flach. 2017. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *AISTATS*. 623–631.

[18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *NeurIPS*.

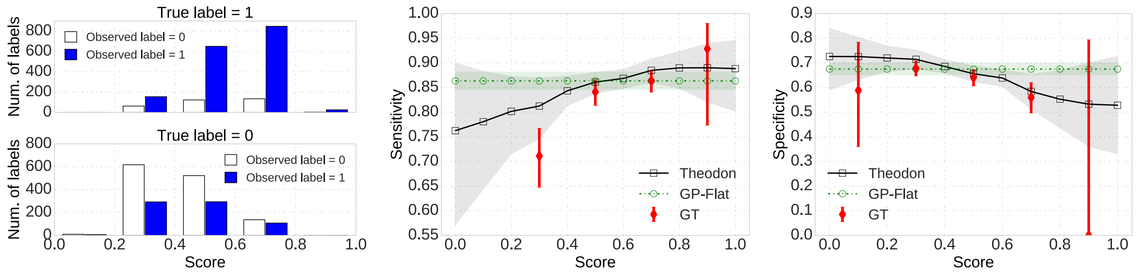
[19] Yuan Li, Benjamin Rubinstein, and Trevor Cohn. 2019. Exploiting Worker Correlation for Label Aggregation in Crowdsourcing. In *ICML*. 3886–3895.

[20] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. 2007. A Note on Platt's Probabilistic Outputs for Support Vector Machines. *MLJ* 68, 3 (Oct. 2007), 267–276.

[21] Yu Liu. 2019. *Estimating the prevalence of rare events — theory and practice*. https://www.unofficialgoogledatascience.com/2019/08/estimating-prevalence-of-rare-events.html

	Metric	Classifier Type	Base	MV	FV	Snorkel	Fl-Fl	Lr-Fl	Gp-Fl	Lr-Lr	Theodon
Sentiment	ECE	1-layer NN	0.0926	0.1128	0.0841	0.0769	0.0844	0.0690	0.0694	0.0677	**0.0582**
		Logistic Regression	0.2293	0.1068	0.0878	0.0839	0.0881	0.0789	0.0772	0.0737	**0.0654**
	AUC-PR	1-layer NN	0.9371	0.9169	0.9508	0.9681	0.9618	0.9776*	0.9777*	0.9761	**0.9815**
		Logistic Regression	0.9361	0.9177	0.9471	0.9638	0.9538	0.9698	0.9715	0.9714	**0.9761**
Music	ECE	1-layer NN	0.1442	0.0583	0.0661	0.0573	0.0430	0.0393	0.0374*	0.0404	**0.0367**
		Logistic Regression	0.1853	0.0573	0.0608	0.0618	0.0433	0.0390*	**0.0382**	0.0420	0.0384*
	AUC-PR	1-layer NN	0.8253	0.7242	0.7279	0.8310	0.8036	0.8716*	0.8784*	0.8697*	**0.8844**
		Logistic Regression	0.7398	0.7203	0.7375	0.7791	0.7812	0.8269	0.8442*	0.8306*	**0.8507**

Table 4: Results averaged over 5 training/test splits on the *Sentiment* and *Music* datasets when varying the base classifier. Bold denotes the best results and * denotes the results within one standard deviation of the best value per row.



(a) Confusion counts **(b) Sensitivity estimate** **(c) Specificity estimate**

Figure 7: Sensitivity and specificity functions for the selected labeler in the *Sentiment* dataset

[22] C. Long, G. Hua, and A. Kapoor. 2013. Active Visual Recognition with Expertise Estimation in Crowdsourcing. In *ICCV*. 3000–3007.

[23] P. Morales-Alvarez, P. Ruiz, S. Coughlin, R. Molina Soriano, and A. Katsaggelos. 2020. Scalable Variational Gaussian Processes for Crowdsourcing: Glitch Detection in LIGO. *TPAMI* (2020).

[24] Pablo Morales-Álvarez, Pablo Ruiz, Raúl Santos-Rodríguez, Rafael Molina, and Aggelos K Katsaggelos. 2019. Scalable and efficient learning from crowds with Gaussian processes. *Information Fusion* 52 (2019), 110–127.

[25] P. Morales-Álvarez, A. Pérez-Suay, R. Molina, and G. Camps-Valls. 2018. Remote Sensing Image Classification With Large-Scale Gaussian Processes. *IEEE Transactions on Geoscience and Remote Sensing* 56, 2 (2018), 1103–1114.

[26] Pablo G. Moreno, Antonio Artés-Rodríguez, Yee Whye Teh, and Fernando Perez-Cruz. 2015. Bayesian Nonparametric Crowdsourcing. *JMLR* (2015), 1607–1627.

[27] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining Well Calibrated Probabilities Using Bayesian Binning. In *AAAI*. 2901–2907.

[28] An T. Nguyen, Byron C. Wallace, and Matthew Lease. 2016. A Correlated Worker Model for Grouped, Imbalanced and Multitask Data. In *UAI*. 537–546.

[29] Viet-An Nguyen, Peibei Shi, Jagdish Ramakrishnan, Udi Weinsberg, Henry C Lin, Steve Metz, Neil Chandra, Jane Jing, and Dimitris Kalimeris. 2020. CLARA: Confidence of Labels and Raters. In *SIGKDD*. 2542–2552.

[30] Rebecca J Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *TACL* 2 (2014), 311–326.

[31] Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian Models of Annotation. *TACL* (2018).

[32] John C. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*. 61–74.

[33] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. 2005. A unifying view of sparse approximate Gaussian process regression. *JMLR* (2005), 1939–1959.

[34] Carl Edward Rasmussen. 2004. *Gaussian Processes in Machine Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg, 63–71.

[35] Alexander J. Ratner, Stephen H. Bach, Henry E. Ehrenberg, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment* 11, 3 (2017), 269–282.

[36] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Anna Jerebko, Charles Florin, Gerardo Hermosillo Valadez, Luca Bogoni, and Linda Moy. 2009. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*. 889–896.

[37] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning From Crowds. *JMLR* 11 (Aug. 2010), 1297–1322.

[38] Filipe Rodrigues and Francisco Pereira. 2018. Deep learning from crowds. In *AAAI*, Vol. 32.

[39] Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2013. Learning from multiple annotators: distinguishing good from random labelers. *Pattern Recognition Letters* 34, 12 (2013), 1428–1436.

[40] Filipe Rodrigues, Francisco C. Pereira, and Bernardete Ribeiro. 2014. Gaussian Process Classification and Active Learning with Multiple Annotators. In *ICML*.

[41] Pablo Ruiz, Pablo Morales-Álvarez, Rafael Molina, and Aggelos K Katsaggelos. 2019. Learning from crowds with variational Gaussian processes. *Pattern Recognition* 88 (2019), 298–311.

[42] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. 2014. Community-based Bayesian Aggregation Models for Crowdsourcing. In *WWW*. 155–164.

[43] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. 2010. The multidimensional wisdom of crowds. In *NeurIPS*. 2424–2432.

[44] Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. 2020. Non-parametric calibration for classification. In *AISTATS*. PMLR, 178–190.

[45] Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *NIPS*. 2035–2043.

[46] Andrew Wilson and Hannes Nickisch. 2015. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *ICML*. 1775–1784.

[47] Edwin B Wilson. 1927. Probable inference, the law of succession, and statistical inference. *JASA* 22, 158 (1927), 209–212.

[48] Yan Yan, Rómer Rosales, Glenn Fung, Mark Schmidt, Gerardo Hermosillo, Luca Bogoni, Linda Moy, and Jennifer Dy. 2010. Modeling annotator expertise: Learning when everybody knows a bit of something. In *AISTATS*. 932–939.

[49] Yan Yan, Rómer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. 2014. Learning from Multiple Annotators with Varying Expertise. *Mach. Learn.* 95, 3 (June 2014), 291–327.

[50] Bianca Zadrozny and Charles Elkan. 2001. Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. In *ICML*. 609–616.

[51] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth Inference in Crowdsourcing: Is the Problem Solved? *VLDB* (2017), 541–552.

[52] Dengyong Zhou, Sumit Basu, Yi Mao, and John C. Platt. 2012. Learning from the Wisdom of Crowds by Minimax Entropy. In *NeurIPS*. 2195–2203.

7 SUPPLEMENTARY INFORMATION

7.1 Simulated Sensitivity/Specificity Functions

We include here the simulated per-labeler sensitivity and specificity functions for the three types of functions (convex, linear, and concave) for Application I (Figure 8) and Application II (Figure 9).

7.2 Additional Results on Public Data

Table 5 summarizes that statistics of the public *Sentiment* and *Music* datasets. Tables 6 and 7 provides additional results on the two public datasets, complementing Tables 3 and 4, respectively.

Dataset	N	A	$\sum_i L_i$	L_i		
				min	avg	max
Sentiment	4999	203	27746	4	5.6	10
Music	700	44	2945	1	4.2	7

Table 5: Statistics of the two datasets: N items, A reviewers, $\sum_i L_i$ total labels, and the minimum, average, and maximum number of labels per item L_i.

7.3 Details on Base Classifiers

Here are details on the base classifiers that were trained using `scikit-learn`.

- 1-layer Neural Network: `MLPClassifier` with the following parameters `hidden_layer_sizes` = (100,), `activation` = relu, and `solver` = adam.
- Logistic Regression: `LogisticRegression` with default parameters.
- Naive Bayes: `MultinomialNB` with default parameters.
- Random Forest: `RandomForestClassifier` with `max_depth`=2.
- AdaBoost: `AdaBoostClassifier` with `n_estimators` = 100.

7.4 Details on Baseline Models

In this section, we provide additional details about the various baseline models used in our experiments.

7.4.1 Fl-Fl. uses scalar probabilities to capture the prevalence, sensitivity, and specificity. In our experiment, we use the Bayesian version of the Fl-Fl model [7] with the following generative process:

- Draw prevalence $\theta \sim \text{Beta}(\alpha)$
- For each labeler $a \in [1, A]$, draw sensitivity $\psi_a \sim \text{Beta}(\beta_\psi)$ and specificity $\phi_a \sim \text{Beta}(\beta_\phi)$
- For each item $i \in [1, N]$
  - Draw a latent true binary label $y_i \sim \text{Bern}(\theta)$
  - For each observed binary label $r_{i,j}$ by labeler $a_{i,j}$
    - If $y_i = 1$, draw $r_{i,j} \sim \text{Bern}(\psi_{a_{i,j}})$
    - Otherwise, draw $r_{i,j} \sim \text{Bern}(1 - \phi_{a_{i,j}})$

We use the same weak priors $\beta_\psi = \beta_\phi = [0.8, 0.2]$ for both sensitivity and specificity, and a uniform prior $\alpha = [1, 1]$ for prevalence.

7.4.2 Lr-Fl. uses a logistic regression to capture the prevalence function and models sensitivity and specificity using scalar probabilities. This baseline is equivalent to Raykar et al.'s model [36, 37] in which the input score is the only feature used in the logistic regression. Here is the detailed generative process:

- Draw logistic regression weights $w_1 \sim \text{Normal}(0, 1)$ and $w_0 \sim \text{Normal}(0, 1)$
- For each labeler $a \in [1, A]$, draw sensitivity $\psi_a \sim \text{Beta}(\beta_\psi)$ and specificity $\phi_a \sim \text{Beta}(\beta_\phi)$
- For each item $i \in [1, N]$ with score s_i
  - Draw a latent true binary label $y_i \sim \text{Bern}(\Omega(w_1 s_i + w_0))$ where $\Omega(x) = 1/(1 + \exp(-x))$
  - For each observed binary label $r_{i,j}$ by labeler $a_{i,j}$
    - If $y_i = 1$, draw $r_{i,j} \sim \text{Bern}(\psi_{a_{i,j}})$
    - Otherwise, draw $r_{i,j} \sim \text{Bern}(1 - \phi_{a_{i,j}})$

Similarly, we use the same weak priors $\beta_\psi = \beta_\phi = [0.8, 0.2]$ for both sensitivity and specificity, and a prior of $\text{Normal}(0, 1)$ on the weights of the logistic regression.

7.4.3 Gp-Fl. uses a GP classifier to capture the prevalence function and scalar probabilities to capture the sensitivity and specificity. A similar model was proposed by Rodrigues et al. [40] with the sensitivity and specificity as point parameters, which was later extended by Ruiz et al. [41] to use stochastic variables. Here is the detailed generative process used:

- Draw $f \sim \mathcal{GP}\left(m_f, K(\alpha, \rho)\right)$ and define prevalence function $\theta(s) = \Omega(f(s))$ where $\Omega(x) = 1/(1 + \exp(-x))$
- For each labeler $a \in [1, A]$, draw sensitivity $\psi_a \sim \text{Beta}(\beta_\psi)$ and specificity $\phi_a \sim \text{Beta}(\beta_\phi)$
- For each item $i \in [1, N]$ with score s_i
  - Draw a latent true binary label $y_i \sim \text{Bern}(\theta(s_i))$
  - For each observed binary label $r_{i,j}$ by labeler $a_{i,j}$
    - If $y_i = 1$, draw $r_{i,j} \sim \text{Bern}(\psi_{a_{i,j}})$
    - Otherwise, draw $r_{i,j} \sim \text{Bern}(1 - \phi_{a_{i,j}})$

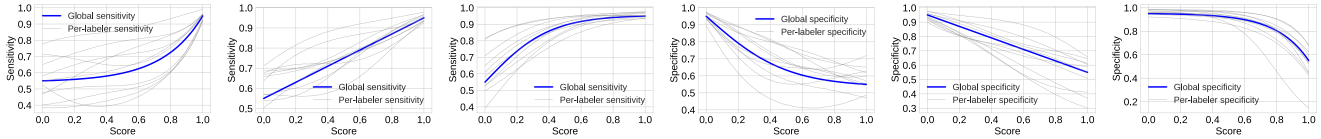
Similarly, we use the same weak priors $\beta_\psi = \beta_\phi = [0.8, 0.2]$ for both sensitivity and specificity. For prevalence, we set $m_f = \Omega^{-1}(0.5)$, and use $\alpha = 1.0$ and $\rho = 0.1$ to define the kernel.

7.4.4 Lr-Lr. uses logistic regression classifiers to capture both prevalence and sensitivity/specificity. This is equivalent to Yan et al.'s model [48, 49] and is closest to our work in capturing varying performance of reviewers depending on the input items. Here is the detailed generative process:

- Draw logistic regression weights $w_1 \sim \text{Normal}(0, 1)$ and $w_0 \sim \text{Normal}(0, 1)$. Define prevalence function $\theta(s) = \Omega(w_1 s + w_0)$.
- For each labeler $a \in [1, A]$
  - Draw $w^\psi_{a,1} \sim \text{Normal}(0, 1)$ and $w^\psi_{a,0} \sim \text{Normal}(0, 1)$.
  - Define sensitivity function $\psi_a(s) = \Omega(w^\psi_{a,1} s + w^\psi_{a,0})$.
  - Draw $w^\phi_{a,1} \sim \text{Normal}(0, 1)$ and $w^\phi_{a,0} \sim \text{Normal}(0, 1)$.
  - Define specificity function $\phi_a(s) = \Omega(w^\phi_{a,1} s + w^\phi_{a,0})$.
- For each item $i \in [1, N]$ with score s_i
  - Draw a latent true binary label $y_i \sim \text{Bern}(\theta(s))$
  - For each observed binary label $r_{i,j}$ by labeler $a_{i,j}$
    - If $y_i = 1$, draw $r_{i,j} \sim \text{Bern}(\psi_{a_{i,j}}(s_i))$
    - Otherwise, draw $r_{i,j} \sim \text{Bern}(1 - \phi_{a_{i,j}}(s_i))$

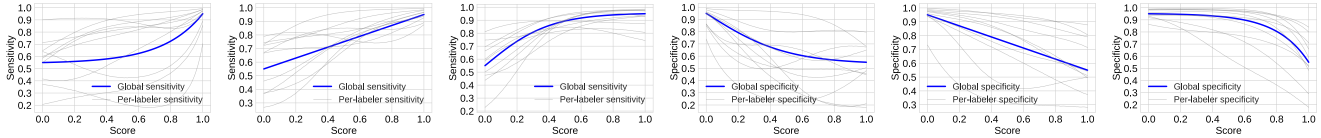
7.4.5 Snorkel. We used the `snorkel` [35] python package for our implementation[4]. We used the following parameters for `LabelModel`, as they provided the best results for our datasets: `lr_scheduler` = exponential, `lr` = 0.01, `optimizer` = adam, and `n_epochs` = 1000

[4] https://snorkel.readthedocs.io/en/v0.9.6/



(a) convex **(b) linear** **(c) concave** **(d) convex** **(e) linear** **(f) concave**

Figure 8: Generated per-labeler sensitivity and specificity functions for Application I



(a) convex **(b) linear** **(c) concave** **(d) convex** **(e) linear** **(f) concave**

Figure 9: Generated per-labeler sensitivity and specificity functions for Application II

	Metric	l	Base	MV	FV	Snorkel	Fl-Fl	Lr-Fl	Gp-Fl	Lr-Lr	Theodon
Sentiment	ECE	5	0.2431	0.1164	0.0764	0.0899	0.0830	0.0704	0.0681*	**0.0644**	0.0651*
		7	0.2417	0.1092	0.0854	0.0881	0.0789	0.0705	0.0672*	0.0666*	**0.0651**
	AUC-PR	5	0.9403	0.9091	0.9487	0.9633	0.9567	0.9727	0.9743	0.9746	**0.9769**
		7	0.9358	0.9157	0.9498	0.9637	0.9584	0.9733	0.9759*	0.9746*	**0.9776**
Music	ECE	5	0.1875	0.0526	0.0594	0.0577	0.0402	0.0370*	**0.0351**	0.0406	0.0362*
	AUC-PR	5	0.7068	0.7338	0.7508	0.7955	0.8146	0.8583*	0.8715*	0.8492*	**0.8730**

Table 6: Results averaged over 5 training/test splits on the *Sentiment* and *Music* datasets when varying the number of labels per item l. Bold denotes the best results and * denotes the results within one standard deviation of the best value per row.

	Metric	Classifier Type	Base	MV	FV	Snorkel	Fl-Fl	Lr-Fl	Gp-Fl	Lr-Lr	Theodon
Sentiment	ECE	Naive Bayes	0.2245	0.1104	0.0813	0.0806	0.0847	0.0738	0.0733	0.0695	**0.0651**
		Random Forest	0.1750	0.1070	0.0873	0.1551	0.0804*	0.0804*	0.0794*	**0.0766**	0.0770*
		AdaBoost	0.1765	0.1142	0.0843*	0.0974	0.0857*	0.0858*	0.0851*	**0.0814**	0.0825*
	AUC-PR	Naive Bayes	0.9475	0.9169	0.9490	0.9663	0.9574	0.9728	0.9741*	0.9749*	**0.9783**
		Random Forest	0.7380	0.9155	0.9555*	0.9491	0.9615*	0.9613*	0.9627*	**0.9627**	0.9623*
		AdaBoost	0.7714	0.9105	0.9467	0.9458	0.9513	0.9514	0.9539	0.9537	**0.9559**
Music	ECE	Naive Bayes	0.2583	0.0586	0.0661	0.0688	0.0450*	0.0431*	**0.0419**	0.0492	0.0465
		Random Forest	0.1533	0.0549	0.0626	0.0590	0.0427	0.0394*	**0.0386**	0.0427	0.0400*
		AdaBoost	0.6381	0.0571	0.0626	0.0604	0.0436*	0.0433*	**0.0394**	0.0510	0.0406*
	AUC-PR	Naive Bayes	0.4100	0.7270	0.7358	0.7592	0.7983	0.8232*	**0.8296**	0.8043	0.8218*
		Random Forest	0.7207	0.7318	0.7524	0.7928	0.8070	0.8536*	**0.8616**	0.8454*	0.8569*
		AdaBoost	0.7483	0.7243	0.7625	0.8007	0.8030	0.8131	0.8608*	0.8040	**0.8663**

Table 7: Results averaged over 5 training/test splits on the *Sentiment* and *Music* datasets when varying the base classifier. Bold denotes the best results and * denotes the results within one standard deviation of the best value per row.

Additionally, we chose the `l2` parameter for regularization by choosing from the set $\{0.0, 0.05, 0.1, 0.15, 0.2\}$ based on a 20% held out set from the training data.

We also provided as input the `class_balance` parameter by estimating class balance with majority vote. Additionally, for using the score in a labeling function, we simply binarized it with a threshold. We set the threshold to the $(1-p)$th percentile of the score distribution, where p is the percent of positive labels estimated from majority vote; this worked much better than simply setting the score threshold to 0.5, especially for class imbalanced datasets.