Community-Based Bayesian Aggregation Models for Crowdsourcing

Matteo Venanzi
University of Southampton
mv1g10@ecs.soton.ac.uk

John Guiver
Microsoft Research
joguiver@microsoft.com

Gabriella Kazai
Microsoft
a-gabkaz@microsoft.com

Pushmeet Kohli
Microsoft Research
pkohli@microsoft.com

Milad Shokouhi
Microsoft
milads@microsoft.com

ABSTRACT

This paper addresses the problem of extracting accurate labels from crowdsourced datasets, a key challenge in crowdsourcing. Prior work has focused on modeling the reliability of individual workers, for instance, by way of confusion matrices, and using these latent traits to estimate the *true labels* more accurately. However, this strategy becomes ineffective when there are too few labels per worker to reliably estimate their quality. To mitigate this issue, we propose a novel community-based Bayesian label aggregation model, *CommunityBCC*, which assumes that crowd workers conform to a few different types, where each type represents a group of workers with similar confusion matrices. We assume that each worker belongs to a certain community, where the worker's confusion matrix is similar to (a perturbation of) the community's confusion matrix. Our model can then learn a set of key latent features: (i) the confusion matrix of each community, (ii) the community membership of each user, and (iii) the aggregated label of each item. We compare the performance of our model against established aggregation methods on a number of large-scale, real-world crowdsourcing datasets. Our experimental results show that our CommunityBCC model consistently outperforms state-of-the-art label aggregation methods, gaining, on average, 8% more accuracy with the same amount of labels.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence–Distributed Artificial Intelligence.

General Terms

Models, Machine learning.

Keywords

Crowdsourcing, Community detection, Bayesian inference.

WWW 2014, April 7-11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2567989

1. INTRODUCTION

Crowdsourcing has become one of the cornerstones of research in the development of human computation based intelligent systems. There are now a number of popular crowdsourcing platforms, like Amazon Mechanical Turk[1], CrowdFlower[2] or Clickworker [3], that are increasingly used for collecting labeled training data for image or document classification and ranking [7, 17, 19]. However, there are a number of challenges associated with gathering labeled data from the crowd. These challenges are related to the uncertainty in the trustworthiness of individual workers and the quality of the crowdsourced labels overall. Specifically, workers might be unreliable and may provide incorrect labels depending on their skills, expertise and motivations. In addition, they may be unintentionally biased towards particular labels. For instance, in tasks where a rating is required, certain workers may be overly conservative and always give medium scores, while others may be overly opinionated and always give extreme scores. These problems make the task of aggregating labels and obtaining a consistent answer challenging, particularly when the dataset includes only a few labels per worker.

Simple solutions to the aggregation of crowd labels typically use heuristic methods such as majority voting (MV) [16]. However, these approaches fail to take into account the reliabilities of different workers. To overcome this problem, probabilistic models have been proposed that do take worker reliability into account when aggregating crowd labels [1, 6, 14, 19, 20, 18]. While these methods can model the accuracy of the workers, they still fail to consider other potential biases, e.g., the tendency for a worker to consistently over or underrate items. Such biased labels can be captured in the form of a worker's *confusion matrix*. That is a matrix that for a given worker, expresses the probability of each possible labeling outcome for each possible true label of the item. Indeed, a number of models have been proposed in the literature to incorporate confusion matrices into the label aggregation process and can thus represent such labeling biases [2, 8, 22]. These models have been shown to produce more accurate aggregations compared to heuristic alternatives [5]. A common feature of these models is the simultaneous estimation of the latent confusion matrices associated with the workers, thus providing the task requester with the

[1] www.amt.com
[2] www.crowdflower.com
[3] www.clickworker.com

individual worker's reliability profile. However, these models typically need to observe a large number of labels per worker to reliably estimate their profiles (as encoded by their confusion matrices). Such a requirement can be particularly difficult to meet in a crowdsourcing setup, where often the distribution of workers to tasks follows a powerlaw curve due to most workers only contributing very few labels, while a few workers contributing a lot of labels [4].

In this paper, we propose a probabilistic model, which is able to merge sparse crowdsourced datasets more efficiently by modeling the correlations between the behaviors of different workers. The key feature that we introduce in our model is the concept of latent worker communities, which allows us to capture characteristic classes of worker behaviors. This is motivated by crowdsourcing studies, where empirical analysis of worker quality has shown that workers tend to exhibit similar behavior traits in the way they perform tasks. In particular, groups of workers can share similar confusion matrices, i.e., with similar reliabilities and biases [15]. We refer to such groups of workers as *communities*. For example, one community may be the class of reliable workers, while other communities can represent spammers, random voters, or, more finely, groups of overly opinionated workers, who tend to give very high or very low scores, versus more conservative workers who have a tendency to pick the middle ratings. Our model encodes this information by modelling worker communities in the generative precess of inferring the crowd labels. Each community is associated with a confusion matrix, which represents the average confusion matrix of its members. Each worker belongs to a certain community where the worker 's confusion matrix is similar to (a perturbation of) the community's confusion matrix. Thus, by embedding the community model inside the generative process, our model can learn a set of key latent features: (i) the community membership of each user, (ii) the confusion matrix of each community, and (iii) the true label of each item. Furthermore, since the number of communities for a specific pool of workers is unknown a priori, our model also learns this information through marginal likelihood based model selection.

The advantage of applying our community based aggregation approach to crowdsourced data is twofold: First, the model can achieve faster convergence to accurate label aggregations with less input data by exploiting the detected worker community structures. Second, the model can accurately learn a worker's confusion matrix even when faced with sparse data, i.e., when only few labels per worker are available, by transferring learning of community profiles to the individual workers. Specifically, our model can be regarded as an extension of the Bayesian classifier combination (BCC) model [8], introduced by Kim and Ghahramani for merging categorical labels, where we incorporate latent workers community structures.

We apply our model to a number of large scale crowdsourced classification datasets which, all together, include 1,367,889 labels collected for 194,052 human intelligence tasks (HITs) of different types, such as adult query suggestion identification, document classification and relevance judgments for search results. In total, 3,948 workers contributed labels to these datasets. Experimental results demonstrate that our model produces more accurate aggregation results compared to commonly used baselines, such as MV and state-of-the-art probabilistic models, e.g., [2, 8]. We also show that our model correctly learns the community structures and the community assignment of each worker, which we verify by comparing the model's output to the results of kmeans clustering over the workersŠ profiles. Finally, we show that our method is able to provide accurate aggregated labels with less training data. In summary, our main contributions are as follows:

- We define a probabilistic Bayesian model that jointly learns latent community profiles of crowd workers, together with the individual workers' and communities' reliability profiles and the items' true labels.
- We provide a scalable implementation of our model using model decomposition techniques which allows us to process hundreds of thousands of crowd labels.
- We empirically show that our model outperforms existing non-community based aggregation methods with an exhaustive comparison against three benchmark methods on four real-world datasets for different crowdsourcing tasks.

The paper is structured as follows. Section 2 reviews related work. Section 3 introduces the preliminary notation and the BCC model, building the core of our framework. Section 4 details our community model and its probabilistic inference. Section 5 details our experimental setting and Section 6 presents the empirical evaluation of our method on real-world datasets. Section 7 concludes the paper and presents directions for future work.

2. RELATED WORK

In the context of compiling patient records, errors in measuring and interpreting different symptoms are not uncommon. In 1979, Dawid and Skene [2] proposed an Expectation-Maximisation (EM) algorithm for modelling these error rates, even when the true values for patient responds were not available. This is analogues to crowdsourcing scenarios in which the true label of an object or task (symptom/facet) may be subject to noise and error, as shown in [5]. Inspired by this classic statistical model, Kim and Ghahramani [8] introduced a general framework for Bayesian Classifier Combination (BCC) in which the relationship between the – unobserved – true label and the model's output are modeled. We provide more details about BCC in the following section as it is highly related to our proposed model, and it is also one our experimental baselines.

In CrowdSynth [6], a set of Bayesian models are trained for (a) predicting the correct labels and (b) modeling the workers and predicting their votes. These models allow the system to maintain a cost-accuracy trade-off under budget constraints by deciding whether to hire a new worker or not. Similarly, Welinder *et al.* [19] proposed an online model for reducing the cost of crowdsourcing by modeling label uncertainty and worker ability. Their method actively selects the next item for rating depending on label uncertainty and the desired level of confidence. Furthermore, the posterior distribution over the quality of workers are constantly updated as new labels are collected.

In the difficulty-ability-response estimation model (DARE) [1], the correct answers (to IQ questions), the task difficulty, and the ability of workers were jointly modeled using a probabilistic graphical model. The authors presented an active

learning testing scheme for selecting the next question to be answered based on previous responses and showed that their model achieves the same level of accuracy with less judgments. Raykar *et al.* [14] take a related approach where inference is iterative. In each iteration, a new *golden set* of true labels is established based on high-confidence labels, and the worker quality estimations are adjusted accordingly. In a similar fashion, the GLAD model [20] also simultaneously updates its beliefs about the true label, worker quality and task difficulty in a probabilistic framework.

Zhou *et al.* [22] considered a separate probabilistic distribution for each worker-item pair and proposed a minimax entropy principle to jointly infer the worker quality and the true labels. They argued that labels are generated by a probability distribution over workers and by maximising the entropy of this distribution the workers' quality can be naturally inferred. They compared the performance of their method against majority voting and the method proposed by Dawid & Skene [2] and showed that the labels inferred by their minmax entropy model are closer to the ground-truth.

Our work builds upon these empirical findings to model and simultaneously estimate worker qualities and the items' true labels, while also modelling workers' quality correlations within a probabilistic inference framework.

In the field of *community detection*, there are a number of graph based approaches that deal with crowdsourcing data categorizations [3] or with the identification of (potentially overlapping) community structures in worker networks [13]. These networks represent community members as nodes and encode distances between them by weighted (or unweighted) links. In this vein, Simpson *et al.* [15] applied a community detection model [13], which groups workers based on the Hellinger distance between their confusion matrices. The authors illustrate that several natural clusters are formed in which workers with similar behavior are grouped together. We take this line of work to the next level by applying community detection in the generative model as opposed to previous work that applies it after inference. We show that this joint inference of community models leads to more efficient and effective label aggregations on sparse datasets.

3. PRELIMINARIES

Notation. Let us start by introducing our notation. Consider that there are K workers classifying N objects into C possible classes. Let t_i be the latent true class[4] of object i. Let $\pi_{c,j}^{(k)}$, with $c, j \in C$, be the probability that worker k will classify an object of class c as j. Finally, $c_i^{(k)}$ is the label submitted by worker k for the object i and $\boldsymbol{C}$ be the set of all the observed labels. For simplicity of notation, in what follows we will assume a dense set of labels in which all the workers label all the objects. However, our model implementation allows us to also support sparsity in the label set.

Bayesian Classifier Combination (BCC) Model. Our approach is an extension of the BCC model [8] that has been successfully applied to crowdsourcing problems in previous work [15]. The factor graph for the BCC model is shown in

[4]In what follows, we will use the terms 'label' and 'class' of objects interchangeably.

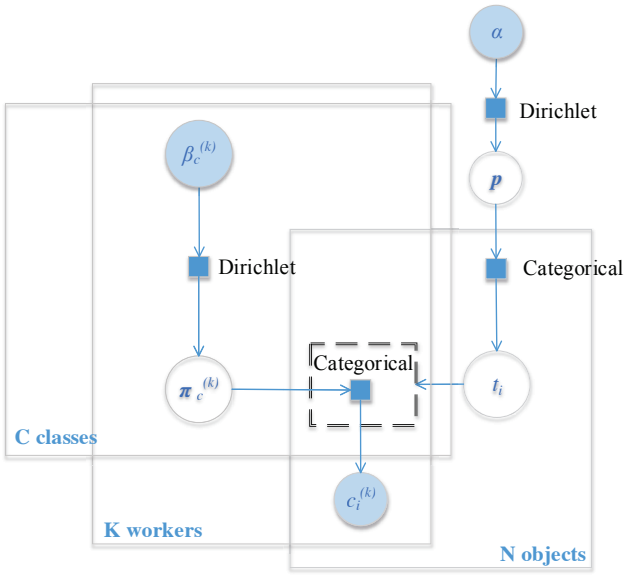


Figure 1: The factor graph of BCC. The users plate includes the user's confusion matrix $\boldsymbol{\pi}^{(k)}$ while the objects plate includes the true object class variable t_i. The observed (shaded) user's vote $c_i^{(k)}$ lies at their intersection

.

Figure 1. The BCC model assumes that the true class, t_i, of an object i is generated from a categorical distribution with parameters $\boldsymbol{p}$:

$$t_i|\boldsymbol{p} \sim \text{Cat}(t_i|\boldsymbol{p})$$

where $\boldsymbol{p}$ denotes the class proportions for all the objects. The label, $c_i^{(k)}$, submitted by worker k for object i is generated from a categorical distribution with parameters $\boldsymbol{\pi}_c^{(k)}$:

$$c_i^{(k)}|\boldsymbol{\pi}^{(k)}, t_i \sim \text{Cat}\big(c_i^{(k)}|\boldsymbol{\pi}_{t_i}^{(k)}\big) \tag{1}$$

where $\boldsymbol{\pi}^{(k)} = \{\boldsymbol{\pi}_1^{(k)}, \dots \boldsymbol{\pi}_C^{(k)}\}$ is the set of row probability vectors representing the confusion matrix of worker k. In other words, t_i is a random variable that acts as selector of the categorical distribution, the rows of $\boldsymbol{\pi}^{(k)}$, from which the label by worker k is generated. Notice that this probabilistic relationship is visually described in the factor graph using the *gate* notation (dashed box) – introduced in [11] – using t_i as gating variable. Further, if we assume that the labels are independent and identically distributed, then the likelihood for a given set of labels $\boldsymbol{C}$ factorises over the data points and can be expressed as follows,

$$p(\boldsymbol{C}|\boldsymbol{\pi}, \boldsymbol{t}, \boldsymbol{p}) = \prod_{i=1}^{N} \text{Cat}(t_i|\boldsymbol{p}) \prod_{k=1}^{K} \text{Cat}\big(c_i^{(k)}|\boldsymbol{\pi}_{t_i}^{(k)}\big)$$

Consequently, the posterior distribution over model parameters, given the data, can be written as:

$$p(\boldsymbol{\pi}, \boldsymbol{t}, \boldsymbol{p}|\boldsymbol{C}) \propto p(\boldsymbol{C}|\boldsymbol{\pi}, \boldsymbol{t}, \boldsymbol{p}) p(\boldsymbol{t}|\boldsymbol{p}) p(\boldsymbol{p}) p(\boldsymbol{\pi}) \tag{2}$$

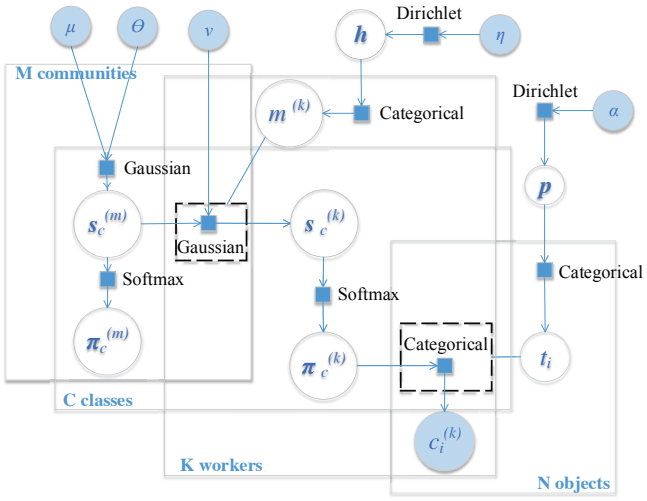


Figure 2: Factor graph of CommunityBCC. The extension to the BCC model (Figure 1) is depicted as the community plate which includes the community score $s_c^{(m)}$. This is connected to the user score $s_c^{(k)}$ which generates the user confusion matrix $\pi_c^{(k)}$.

Conjugate Dirichlet prior distributions are used for the parameters $\boldsymbol{p}$ and $\boldsymbol{\pi}$:

$$\boldsymbol{p} \sim \text{Dir}(\boldsymbol{p}|\boldsymbol{\alpha})$$
$$\boldsymbol{\pi}_c^{(k)} \sim \text{Dir}(\boldsymbol{\pi}_c^{(k)}|\boldsymbol{\beta}_c^{(k)}).$$

This simplifies Equation 2 to:

$$p(\boldsymbol{\pi}, \boldsymbol{t}, \boldsymbol{p}|\boldsymbol{C}) \propto \text{Dir}(\boldsymbol{p}|\boldsymbol{\alpha}) \prod_{i=1}^{N} \Big\{ \text{Cat}(t_i|\boldsymbol{p}) \prod_{k=1}^{K} \text{Cat}\big(c_i^{(k)}|\boldsymbol{\pi}_{t_i}^{(k)}\big) \text{Dir}(\boldsymbol{\pi}_c^{(k)}|\boldsymbol{\beta}) \Big\}$$

The marginal posterior distribution of individual parameters can be obtained by integrating out all the remaining joint parameters. Since this integration is intractable analytically, it needs to be computed using numerical methods. In particular, our implementation of BCC uses the Expectation-Propagation (EP) message passing algorithm [10] provided by the Infer.NET probabilistic programming framework [12].

A key feature of BCC is the assumption that workers are independent. On the one hand, this assumption reduces the complexity of the model and simplifies the inference. On the other hand, the model is inefficient as the relationships between the confusion matrices of workers are not modelled. Encoding these relationships, as we will show in Section 5, is key to achieving better aggregation performance with sparse sets of labels. In the next section, we describe our community-based BCC model, which encodes the feature that workers conform to a few different types, where each type represents a group of workers with similar confusion matrices.

4. COMMUNITY BCC MODEL

Our proposed Community BCC model – referred to hereafter as CommunityBCC – is an extension of the BCC model, enhanced by introducing the concept of worker communities. The factor graph of CommunityBCC can be seen in Figure 2. Specifically, we assume that there are M communities of workers within the crowd. Each community m is associated with a confusion matrix $\boldsymbol{\pi}^{(m)}$. Let $m^{(k)}$ be the community membership of worker k. Then, we assume that $m^{(k)}$ is generated from a categorical distribution with parameters $\boldsymbol{h}$:

$$m^{(k)}|\boldsymbol{h} \sim \text{Cat}(m^{(k)}|\boldsymbol{h})$$

where $\boldsymbol{h}$ represents the proportions of community memberships across all the workers. Further, each community has a probability score $\boldsymbol{s}_c^{(m)}$ representing the log probability vector of the c^{th} row of the confusion matrix $\boldsymbol{\pi}^{(m)}$ of community m. We apply a softmax operator to the community score $\boldsymbol{s}_c^{(m)}$ for all the classes c to derive a normalised exponentiated version of $\boldsymbol{s}_c^{(m)}$, which we refer to as the community confusion matrix $\boldsymbol{\pi}_c^{(m)}$:

$$p(\boldsymbol{\pi}_c^{(m)}|\boldsymbol{s}_c^{(m)}) = \delta(\boldsymbol{\pi}_c^{(m)} - \text{softmax}(\boldsymbol{s}_c^{(m)}))$$

here δ is the Dirac delta function imposing the equality constraint between $\boldsymbol{\pi}_c^{(m)}$ and the softmax of $\boldsymbol{s}_c^{(m)}$.

In contrast to the standard BCC model where the confusion matrices of workers are independent, the CommunityBCC model encodes the relationship between the confusion matrix of the community and the workers that belong to it. That is, each worker k has an individual score vector $\boldsymbol{s}_c^{(k)}$, one for each class c, generated from the score vector $\boldsymbol{s}_c^{(m^{(k)})}$ of the community $m^{(k)}$ (i.e., the worker k's community score), through a multivariate Gaussian draw:

$$\boldsymbol{s}_c^{(k)}|\boldsymbol{s}_c^{(m^{(k)})} \sim \mathcal{N}(\boldsymbol{s}_c^{(k)}|\boldsymbol{s}_c^{(m^{(k)})}, \nu^{-1}\boldsymbol{I}) \quad (3)$$

where ν is the hyperparameter representing the isotropic inverse variance of the community confusion matrices. From this, $\boldsymbol{\pi}_c^{(k)}$ is derived as the normalised exponentiated version of $\boldsymbol{s}_c^{(k)}$, i.e., the softmax of $\boldsymbol{s}_c^{(k)}$. Formally:

$$p(\boldsymbol{\pi}_c^{(k)}|\boldsymbol{s}_c^{(k)}) = \delta(\boldsymbol{\pi}_c^{(k)} - \text{softmax}(\boldsymbol{s}_c^{(k)}))$$

Finally, we follow the same assumption as BCC that c_i^k is generated by a categorical distribution conditioned on $\boldsymbol{\pi}^k$ and $\boldsymbol{t}_i$ (see Equation 1).

4.1 Probabilistic Inference

We now describe how inference is performed in the CommunityBCC model. Again, for simplicity of notation this description assumes a dense set of labels. Specifically, we assume that we have observed the class assignments of all objects from all workers (denoted by $\boldsymbol{C}$). Given $\boldsymbol{C}$, we wish to infer the posterior distributions of the set of the parameters: $\Theta = \{\, \boldsymbol{s}^{(m)}, \boldsymbol{\pi}^{(m)}, \boldsymbol{s}^{(k)}, \boldsymbol{\pi}^{(k)}, \boldsymbol{t}, \boldsymbol{p} \,\}$. To do so, we apply Bayes' theorem to compute the joint posterior distribution of Θ conditioned on $\boldsymbol{C}$:

$$p(\Theta|\boldsymbol{C}) \propto p(\boldsymbol{C}|\boldsymbol{\pi}^{(k)}, \boldsymbol{t}) p(\boldsymbol{t}|\boldsymbol{p}) p(\boldsymbol{p}) p(\boldsymbol{\pi}^{(k)}|\boldsymbol{s}^{(k)}) p(\boldsymbol{s}^{(k)}|\boldsymbol{s}^{(m^{(k)})}) p(m^{(k)}|\boldsymbol{h}) p(\boldsymbol{h}) p(\boldsymbol{s}^{(m^{(k)})}) \quad (4)$$

The priors for the parameters $\boldsymbol{h}$ and $\boldsymbol{s}_m$ are specified by the following conjugate distributions:

$$\boldsymbol{h}|\boldsymbol{\alpha} \sim \text{Dir}(\boldsymbol{h}|\boldsymbol{\alpha})$$
$$\boldsymbol{s}_c^{(k)}|\boldsymbol{\mu}, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{s}_c^{(k)}|\boldsymbol{\mu}, \boldsymbol{\theta}^{-1}\boldsymbol{I})$$

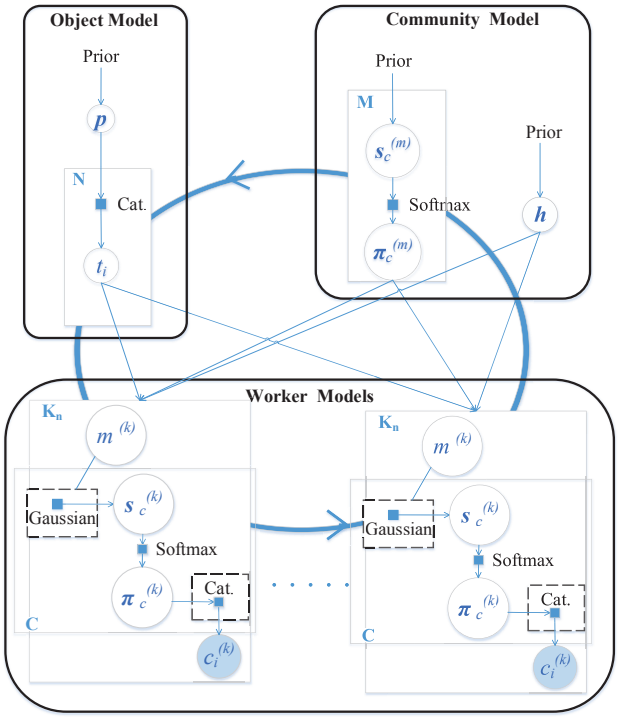


Figure 3: Scalable CommunityBCC implementation. The figure shows the three sub-models and their cyclic schedule for the inference computation.

That is, the prior for $\boldsymbol{h}$ is a Dirichlet distribution with parameters $\boldsymbol{\alpha}$, while the prior for $\boldsymbol{s}_c^{(k)}$ is a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and isotropic precision $\boldsymbol{\theta}$. If we assume that labels are generated independently, then factoring in Equation 5 in the model's distributions, we obtain the following expression for the joint posterior distribution:

$$p(\Theta|\boldsymbol{C}) \propto \mathrm{Dir}(\boldsymbol{p}|\boldsymbol{\alpha}) \prod_{i=1}^{N} \Big\{ \mathrm{Cat}(t_i|\boldsymbol{p}) \prod_{k=1}^{K} \Big\{ \mathrm{Cat}(c_i^{(k)}|\boldsymbol{\pi}_{t_i}^{(k)}) \\ \delta(\boldsymbol{\pi}_{t_i}^{(m^{(k)})} - \mathrm{softmax}(\boldsymbol{s}_{t_i}^{(m^{(k)})})) \mathrm{Dir}(\boldsymbol{h}|\boldsymbol{\alpha}) \\ \mathcal{N}(\boldsymbol{s}_{t_i}^{(k)}|\boldsymbol{s}_{t_i}^{(k)}, \boldsymbol{\nu}^{-1}) \mathcal{N}(\boldsymbol{s}_{t_i}^{(m^{(k)})}|\boldsymbol{\mu}, \boldsymbol{\theta}^{-1}) \mathrm{Cat}(m^{(k)}|\boldsymbol{h}) \Big\} \Big\}$$

To compute the approximate marginal distribution of each parameter, we use variational message passing algorithm [21], which is also provided by the Infer.NET inference engine[5]. Finally, to find the optimal number of communities M^*, we use the maximum marginal likelihood model selection criterion that is computed through marginalising out all the parameters in Θ. Formally:

$$M^* = \arg\max_{M} \int_{\Theta} p(C|\Theta, M) p(\Theta) d\Theta \qquad (5)$$

This marginal likelihood optimisation can be performed through a simple line search on the discrete set of the M values.

[5]The messages for the softmax factor (which is not conjugate) make use of the Taylor expansion in [9].

4.2 Scalable Implementation

To train our model on large-scale datasets (i.e., in the order of hundreds of thousands of labels), we provide a scalable implementation of CommunityBCC using a model decomposition implementation technique. The main idea is to split the model into three sub-models: object model, community model and a set of worker models. Then, we can run (cheaper) local inferences in each sub-model, and merge the post-inference results using standard message passing calculations. To define the user models, we split the set of workers $\boldsymbol{K}$ into a complete and disjoint partition: $\boldsymbol{K} = \{\boldsymbol{K}_1, \ldots, \boldsymbol{K}_l\}$. Each sub-set K_i is associated with a single worker model instance which infers the variables $\boldsymbol{\pi}^{(k)}$ and $\boldsymbol{s}^{(k)}$ for the workers in K_i. The object model infers the variables t_i and $\boldsymbol{p}$ of all the objects. Similarly, the community model infers the variables $\boldsymbol{h}$, $\boldsymbol{s}^{(k)}$ and $\boldsymbol{\pi}^{(k)}$ of all the workers. The three models are connected together through the probabilistic relationships between t_i and $\boldsymbol{\pi}^{(k)}$ and between $\boldsymbol{s}^m$ and $\boldsymbol{s}^k$, as described in Equations 1 and 3, respectively. To run inference in this model architecture, we iteratively run the inference message passing algorithm in each of the three sub-models in a scheduled order and repeat this process until all the merged posterior distributions reach convergence. In particular, we use the EP algorithm [10] to run inference in the object model and the VMP algorithm [21] to run inference in the community and the worker models.

The factor graph and the inference schedule of the scalable CommunityBCC implementation is depicted in Figure 3. This implementation of CommunityBCC has several advantages in terms of scalability. First, the space complexity is reduced by running local inference in simpler sub-model and merging the results using only a few messages exchanged between the connecting factors. Second, we can parallelise the computation of each sub-model with an evident saving of time complexity. Using just the first of these techniques, we were able to train our CommunityBCC model on our largest dataset of 722,970 data points in approximately 20 minutes on a standard machine with 3.60 GHz CPU and 16GB RAM.

5. EMPIRICAL EVALUATION

In order to evaluate the efficacy of our CommunityBCC model, we make use of four large-scale, real-world datasets and compare performance against three state-of-the-art baselines for crowdsourcing scenarios. We measure performance based on agreement with gold standard data. Each of these aspects are described below.

5.1 Datasets

Our four crowdsourcing datasets include a publicly available dataset, provided by CrowdFlower, and three datasets that were crowdsourced using Clickworker, a platform similar to Amazon's Mechanical Turk, where workers remain anonymous. The types of documents and the types of labels vary in the different datasets. The label types include categorical classes, with 3 or 5 classes of labels. In addition to the labels collected from workers on Clickworker, we also obtained a set of gold labels from highly trained editorial judges to be used as ground truth in our experiments. These gold labels were collected using the exact same interface as those shown to the workers. CrowdFlower's dataset includes gold labels with an undisclosed origin. In total,

the four label sets contain $1,367,889$ judgments for $194,052$ HITs from $3,948$ workers, with $38,876$ judgments by $1,488$ trained expert judges for $1,209$ gold HITs. Table 1 provides an overview of some basic statistics over the datasets.

Auto-completion Judgments (ACJ) This task collected preference judgments over lists of query suggestions (auto-completion queries) that were shown to judges side-by-side to each other. Workers were asked to indicate their preferences over the two lists of auto-completions on a 3 point scale $(-1, 0, 1)$, where 0 indicates no preference and 1 or -1 signal preference for one side or the other. This dataset includes 722,970 judgments for 72,142 HITs by 722 workers. Although this is the largest set in terms of volume, the number of workers is only the third largest across the four datasets. This set is thus somewhat unusual in terms of averaging 1,000 judgments per worker. Such a high per-worker throughput may, however, simply be an indication of the task's popularity with these workers. Among the 72k HITs in the set, only 155 are gold tests, which were judged only by 77 workers, resulting in 3,100 judgments with known answers. Note that the relatively low number of gold tests in this task is compensated for by the high redundancy of 10 judgments per HIT.

Search Relevance Judgments (SRJ) Similarly to the ACJ set, this dataset contains preference based relevance judgments, but, this time, the preferences are over two search engine result pages (SERPs), shown side-by-side, for a given query. Judges were asked to indicate their preferences over the two SERPs, which was then, again, mapped to a 3 point scale of $(-1, 0, 1)$. The set contains a total of 50,840 judgments for 22,623 HITs by 1,118 judges. Each HIT is a query and a pair of SERPs that may be judged multiple times by different judges. The set also contains 399 gold HITs with known labels, for which 9,352 judgments have been collected from 802 judges out of the total 1,118.

CrowdFlower (CF) This dataset is provided by CrowdFlower as part of the 2013 Crowdsourcing at Scale shared task challenge[6]. It consists of 569,375 sentiment analysis judgments for 98,980 HITs (questions) by 1,960 workers. The questions are tweets and the judgments reflect the sentiment of the tweets discussing weather. The labels can take values from four categories: negative (0), neutral (1), positive (2), tweet not related to weather (4) and cannot tell (5). The data also includes 300 gold HITs, which were judged by 461 of the 1,960 workers in the full dataset, resulting in a total of 1,720 judgments with known answers.

Adult Relevance Judgments (ARJ) Workers in this task were asked to rate whether a given set of query suggestions contained various types of adult content on a four point scale. For our experiments, we use the binary label whether adult content was detected (1) or not (0), ignoring the actual classification. This dataset contains 24,704 judgments for 307 HITs by 148 workers. We have gold labels for all the HITs in this set.

5.2 Baselines

We compare our CommunityBCC model against the following set of commonly used baselines:

[6]https://sites.google.com/site/crowdscale2013/shared-task

Table 1: Datasets: Avg-J (Avg-GJ) is the average number of judgments (on gold HITs) per judge, Avg-H (Avg-GH) is the average number of judgments per (gold) HIT. Gold judges are the judges who were tested on gold HITs.

Dataset:	ACJ	SRJ	CF	ARJ
HITs	72,142	22,623	98,980	307
Judgments	722,970	50,840	569,375	24,704
Judges	722	1,118	1,960	148
Avg-J	1,001.34	45.47	290.5	166.92
Avg-H	10.02	2.25	5.75	80.47
Gold HITs	155	399	300	307
Gold judgments	3,100	9,352	1,720	24,704
Gold judges	77	802	461	148
Avg-GJ	40.26	11.66	3.73	166.92
Avg-GH	20.00	23.44	5.73	80.47

Majority Voting (MV) The MV method, which is often used for the heuristic computation of crowd consensus [16], estimates the aggregated label as the one with the most votes, where each vote is considered with equal weight. As such, this method assumes that all workers are equally reliable. Importantly, in this paper we use a deterministic version of MV, which provides the most voted label only for unimodal vote distributions. Other stochastic interpretations of MV might use random draws for tie-breaking, e.g. weighted MV, which could marginally improve our deterministic MV in expectation.

Dawid&Skene The well-known crowd consensus model, presented by Dawid and Skene [2], allows the joint estimation of the items' true labels and the workers' confusion matrices. This model represent the *frequentist* version of BCC and is trained through the Expectation-Maximisation (EM) algorithm.

BCC The BCC model [8], which learns the true object labels and the workers' confusion matrices using the EP inference algorithm, as described in Section 3. In particular, we used uninformative priors for $\boldsymbol{p}$ and informative priors for the workers confusion matrices. Each row of $\boldsymbol{\pi}_c^{(k)}$ has a Dirichlet prior with pseudo counts 1 expect for the diagonal count set to $C - 1$. This means that workers are initially assumed to be better than random.

These three methods were tested against our Community-BCC with uninformative priors for $\boldsymbol{h}$ and $\boldsymbol{p}$, a noise precision prior for the worker score matrices $\nu = 100$. The softmax of the community score matrices is set to be approximately equivalent to the prior of the BCC's confusion matrices.

5.3 Metrics

The performance of each method is evaluated by two measures: (1) accuracy and (2) negative log probability density (NLPD). Accuracy is defined as the absolute estimation error given by the number of correct labels assigned, divided by the total number of assigned labels N:

$$Accuracy = \frac{num.\ correct\ labels}{num.\ gold\ labels}$$

The NLPD provides a more comprehensive error measure,

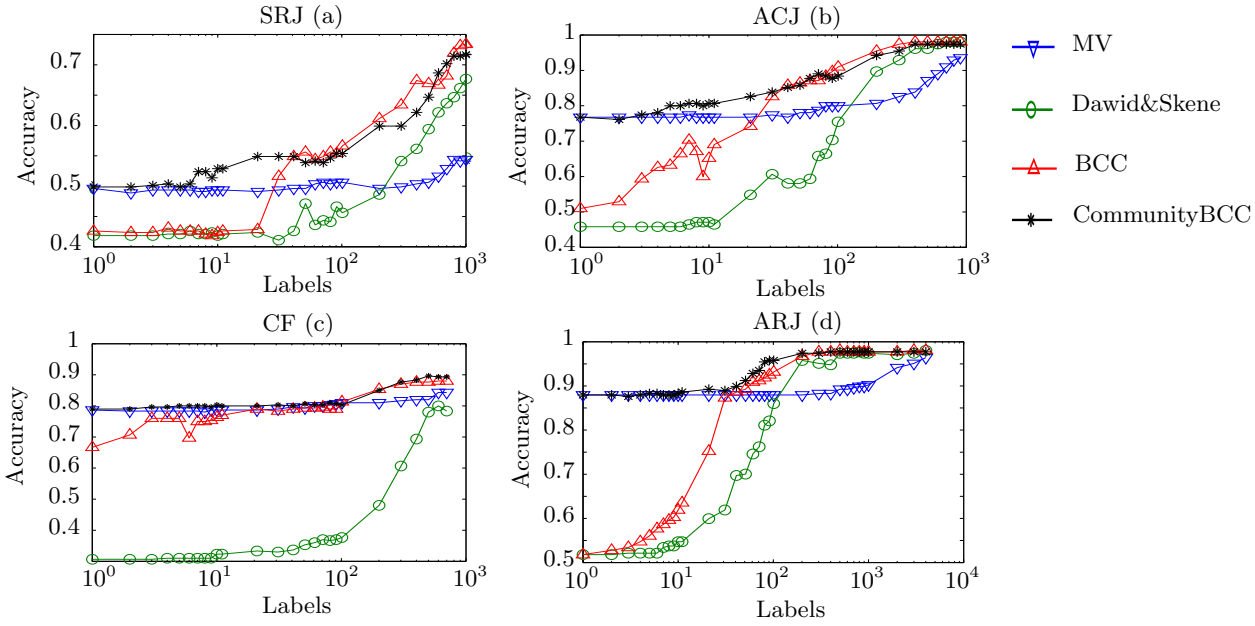


Figure 4: Accuracy of the four methods measured with increasing proportions of labels for each dataset.

which also takes into account the uncertainty in the predictive label distribution, and is calculated as follows. Let $\{q_{i,1}, \ldots, q_{i,j}\}$ be the parameters of the predictive categorical distribution for item i. Then, the NLPD score is:

$$\text{NLPD} = \frac{1}{N} \sum_{i=1}^{N} -\log\left(q_{i,t_i^*}\right)$$

where t_i^* is the true label and $\hat{t}_i = \arg\max_c t_{i,c}$ is the estimated label obtained from the predictive distribution $\boldsymbol{t}_i = \{t_{i,c} | c = 1 \ldots C\}$ for object i computed by the algorithm[7].

Overall, we seek the best method with highest accuracy and lowest uncertainty, i.e., lowest NLPD.

6. EVALUATION

In this section, we present the results of the empirical evaluation of three key features of our CommunityBCC model applied to crowdsourcing scenarios.

Robustness to Sparsity. We start by investigating the effect of sparsity on the accuracy of the different methods for estimating the true label for each object. We present the results from our accuracy evaluation against various sparsity levels within each dataset. For each of the four datasets, we simulate an adaptive learning setting in which labels are progressively selected. In detail, at each iteration, each method performs an intelligent selection of the set of n items for which to get an extra label in the next iteration by using the standard information theoretic criterion of entropy minimisation. That is, after an initial exploration phase in which one label for each item is observed, the methods select the next labels by taking the n items with the highest uncertainty in their predictive class distribution. Specifically, we used $n = 3$ for ACJ, $n = 5$ for SRJ, $n = 2$ for CF and $n = 5$ for ARJ. Figure 4 shows the accuracy measured for the four methods for each dataset. In particular, it shows that the accuracy of all methods improves as they get more data. The accuracy of CommunityBCC increases rapidly with the amount of data and is consistently higher compared to other methods when the size of label set is small. Specifically, looking at the points of the maximum gap between CommunityBCC and the other methods, CommunityBCC's accuracy is 8.4% higher than the second best method on SRJ, 7.5% higher on ACJ, 2.7% higher on CF and 3.9% higher on ARJ. Generally speaking, these results show that CommunityBCC is able to make more effective use of small sets of labels by exploiting community patterns between workers and transferring learning of community knowledge across community members. In large sets, CommunityBCC performs comparably or slightly better than the other methods. Overall, CommunityBCC offers the best trade-off between effectiveness and efficiency compared to alternative state-of-the-art baselines across all experimental testbeds.

Community Detection. Figure 5 shows the communities that our method inferred for each dataset. It is interesting to note that CommunityBCC finds four communities for ACJ (Figure 5a) and SRJ (Figure 5b) although with different proportions. The first community is the group of workers who provide reliable answers and the community confusion matrix has consistently high value on the diagonal. We call this group *Accurate* workers. This community represents 28% and 4% of the workers in the ACJ and the SRJ dataset, respectively. The second community outlines the profile of workers who tend to provide correct answers although with lower probability (the diagonal value) compared to the *Accurate* community. We refer to this group of workers as *Mostly accurate*. The other two communities were found to correspond to biased workers. The first community of biased workers is the group of workers who mostly vote -1 or 1, who are present in 29% and 47% of the workers in the ACJ and the SRJ datasets, respectively. The second community of biased workers is the group of workers

[7]To avoid infinite errors, we threshold the NLPD with a with minimum probability of 0.001.

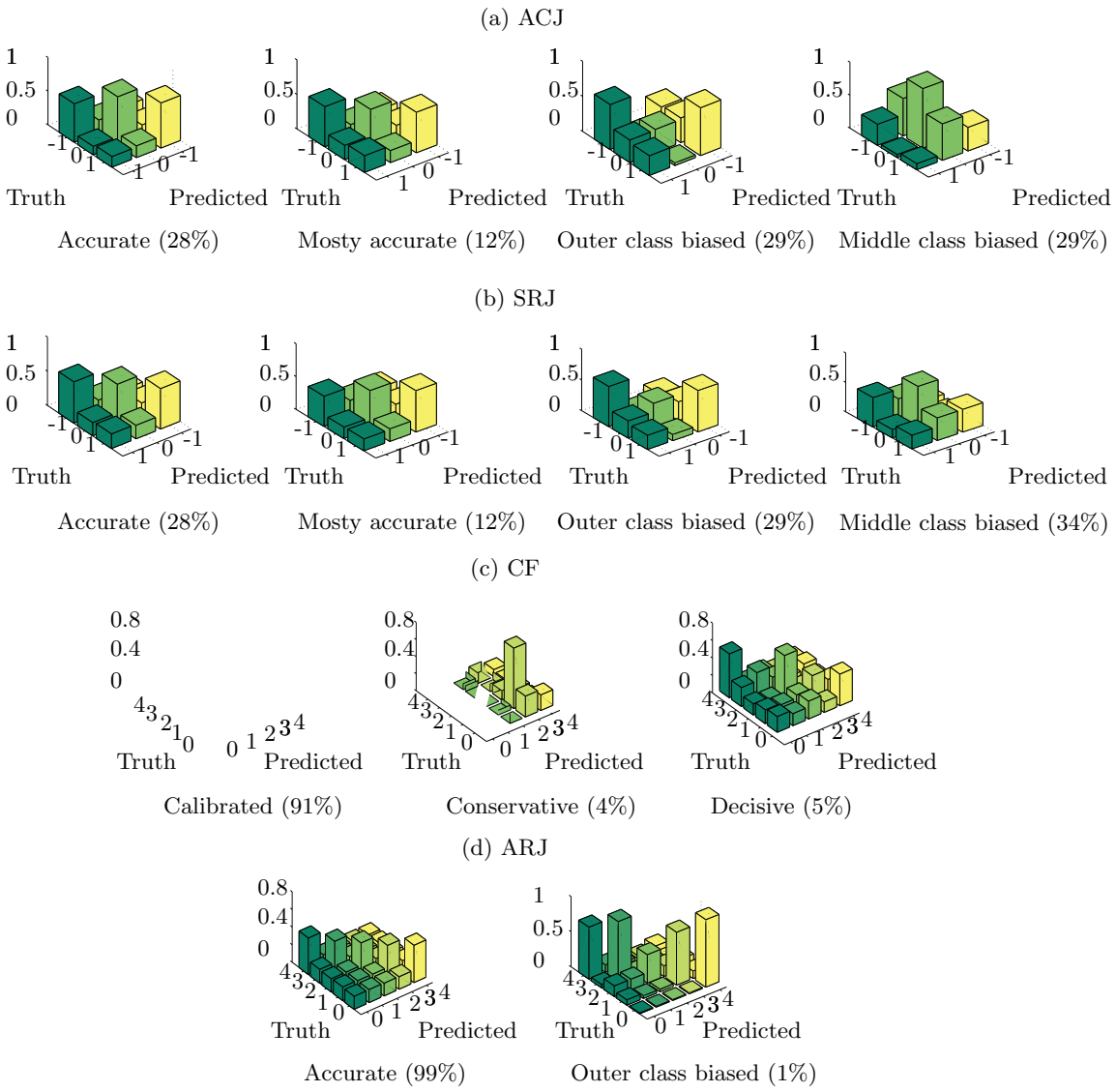


Figure 5: Communities of workers and members proportions (in brackets) inferred from CommunityBCC in each dataset. The bar plots are the communities confusion matrices inferred for each dataset. The values on the x, y axis are the object classes and values on the z axis are the probabilities.

who almost always vote 0, i.e., they tend not to pick a side in their judgment. This community is present with similar proportions within the two datasets: 31% for ACJ and 34% for SRJ. We respectively refer to these two communities as *Outer class biased* and *Middle class biased*.

On the CF testbed (Figure 5c), there is a large community of calibrated workers (91%), who almost never use the "cannot tell" option (i.e., vote 4), and frequently select the correct judgment (i.e., using votes 1, 2 or 3), for all the tweets. The second (small) community is the one of conservative workers who mostly opt for the neutral (1) or the "not related to weather" (3) judgments. By contrast, the third community includes workers who often confidently pick negative (0) or positive (2) judgments. We refer to these communities as *Calibrated*, *Conservative*, and *Decisive* respectively.

On the ARJ testbed (Figure 5e), there is a large majority of workers that are *Accurate* (99%), and there is only 1% workers biased towards outer classes (referred to as the *Outer class biased* community). We will later provide more insights about how the reported community percentages relate to the quality of the predicted label.

To evaluate the community detection accuracy of CommunityBCC, we run a kmeans clustering algorithm (with k=4, that is the number of communities found by CommunityBCC) on the confusion matrices estimated by BCC for SRJ[8]. Then we compare the centroid of each cluster to our community matrices. Similar to [15], we first calculate the cumulative pairwise distance (CPD) for the confusion matrix of each worker defined as:

$$\text{CPD}(k) = \sum_{k' \neq k} HD(\boldsymbol{\pi}^{(k)}, \boldsymbol{\pi}^{(k')})$$

where $HD(\boldsymbol{\pi}^{(k)}, \boldsymbol{\pi}^{(k')})$ is the Hellinger distance between the

[8]for brevity we report only the results of SRJ, however the conclusions for the other datasets are similar.

Table 2: The accuracy scores of the four methods. The numbers for each dataset (rows) are the absolute estimation error of each method (column). The best performing run in each dataset is highlighted in bold.

	MV	Dawid&Skene	BCC	CommunityBCC
ACJ	0.974	0.972	**0.987**	0.974
SRJ	0.692	0.670	**0.720**	0.711
CF	0.773	0.830	0.860	**0.886**
ARJ	**0.980**	0.960	0.964	0.976

Table 3: The NLPD scores (the lower the better) of the three probabilistic methods: Dawid&Skene, BCC and CommunityBCC computed from the predictive object class distribution. The best performing run in each dataset is highlighted in bold.

	Dawid&Skene	BCC	CommunityBCC
ACJ	0.146	**0.076**	0.137
SRJ	1.826	1.226	1.414
CF	1.045	**0.437**	0.526
ARJ	**0.135**	0.157	0.157

two confusion matrices. Then, kmeans is run on the set of CPD values. From results showed in Figure 6, it can be seen that the four communities of CommunityBCC clearly match the profiles of the kmeans clusters. In particular, Figure 6b shows the *actual* confusion matrix of the worker corresponding to the centroid of each cluster. The centroid worker's confusion matrices are similar to the estimated community matrices. Furthermore, the proportions of workers in each communities reported in brackets are equivalent to the ones of computed by kmeans as showed in Figure 6b. This means that our method correctly learns both the community profiles and the worker's community memberships.

Accuracy. Table 2 reports the accuracy scores of all methods evaluated on the subset of gold items of each of the four datasets. Importantly, we use only gold items as training set in order to evaluate the user's accuracy profile only based on labels for items with gold labels. The scores show that the three confusion matrix-based methods (Dawid&Skene, BCC and CommunityBCC) are generally more accurate than MV. In particular, CommunityBCC is the best method which outperforms the best benchmark (BCC) by up to 3% on the CF dataset. On average, its performance is slightly higher than BCC. From the NLPD scores reported in Table 3, it can also be seen that, except for SRJ, CmmunityBCC is close to the best NLPD scoring method which means that its predictions also have low uncertainty, hence it is able to provide informative predictions. These accuracy scores reflect also the difficulty of the label aggregation task for each dataset. For instance, all the methods are consistently more accurate on ACJ (avg. 0.975) and less accurate on SRJ (avg. 0.689). Notice that the scores also relate to the proportions of accurate users available in the crowd, as discussed earlier in the community detection results. Indeed, SRJ has only 19% of accurate or mostly accurate workers while this percentage increases up to 40% in ACJ (Figure 5). Consequently, all the methods are able to produce higher quality results for sets where more accurate labels are available. In summary, we conclude that our method performs comparably or slightly higher than the tested baselines in predicting the true labels, while it is much more efficient and less prone to error when only sparse data is available per worker.

7. CONCLUSIONS

In this paper, we presented a novel Bayesian model for the efficient aggregation of crowdsourced labels for classification tasks based on modelling worker's profiles as latent communities. The key innovation of our method is to model the worker communities within the label aggregation process. More specifically, we model workers' community structures within the generative model of crowd labels and we apply Bayesian inference to learn (i) the accuracy profile, i.e., the confusion matrix of each community, (ii) the community membership of each worker, and the true label of each object. We also discover the optimal number of communities within the crowd using marginal likelihood optimisation. Thus, the key capabilities of our model are to (i) abstract inferred information about workers' reliability from a single individual to the general community and (ii) transfer knowledge about the community profile to workers. We also provided a scalable implementation of CommunityBCC using model decomposition techniques that allow us to train the model on hundred of thousands of labels within minutes on a standard desktop machine. This makes our new method particularly suitable for large-scale crowdsourcing applications. We ran an extensive experimental evaluation with more than one million real-word crowdsourced labels for different tasks contributed by more than four thousand workers. We compared the performance of our method against three state-of-the-art methods as baselines. We empirically showed that CommunityBCC is more accurate than the baseline methods, and is more effective on sparse datasets where, on average, it gains 8% more accuracy with the same amount of sparse labels compared to the baseline methods.

While this work is a step forward towards relaxing the assumption of independence between workers in crowdsourcing by incorporating the notion of communities, there are several other aspects related to dynamic and task-specific worker behaviours that have not yet been addressed and will be considered as future work. In future work, we will investigate the application of our community-based aggregation approach to other crowdsourcing problems, such as ranking or to continuous labels.

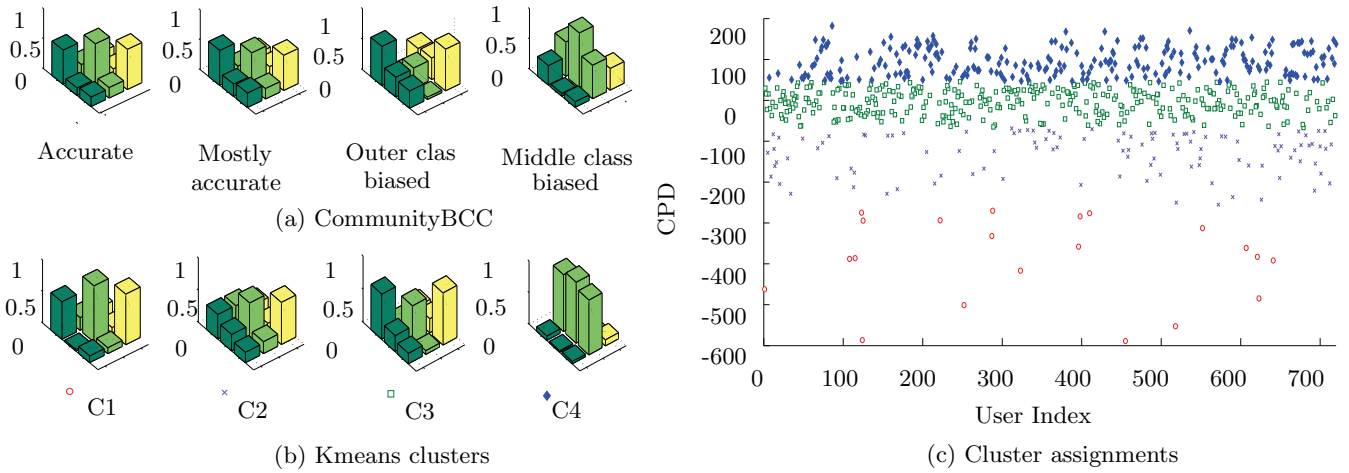
8. ACKNOWLEDGMENTS

The authors gratefully thank Tom Minka for the support and discussion about the model. Matteo Venanzi would also like to thank Oliver Parson for early discussions about this work.

9. REFERENCES

[1] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver. How to grade a test without knowing the answers. a bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *Proc. of the 29th Int. Conf. on Machine Learning*, pages 1183–1190, 2012.

[2] A. P. Dawid and A. M. Skene. Maximum likelihood



(a) CommunityBCC

(b) Kmeans clusters

(c) Cluster assignments

Figure 6: Comparison between CommunityBCC and Kmeans clustering for community detection on the SRJ dataset. On the left, there is the comparison between the CommunityBCC communities (first row) and the kmeans clusters (second row). On the right, there is the plot of the cumulative pairwise distances and kmeans cluster assignments of each user.

estimation of observer error-rates using the EM algorithm. *Applied Statistics*, pages 20–28, 1979.

[3] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *Advances in Neural Information Processing Systems*, 2011.

[4] P. G. Ipeirotis. Analyzing the Amazon Mechanical Turk marketplace. *XRDS*, 17:16–21, December 2010.

[5] 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, New York, NY, USA, 2010. ACM.

[6] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proc. of the 11th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 467–474, 2012.

[7] G. Kazai. In search of quality in crowdsourcing for search engine evaluation. In *Advances in information retrieval*, pages 165–176. Springer, 2011.

[8] H.-C. Kim and Z. Ghahramani. Bayesian classifier combination. *Int. Conf. on Artificial Intelligence and Statistics*, pages 619–627, 2012.

[9] J. D. Lafferty and D. M. Blei. Correlated topic models. In *Advances in Neural Information Processing Systems*, pages 147–154, 2005.

[10] T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[11] T. Minka and J. Winn. Gates. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2008.

[12] T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer .net 2.5. *Microsoft Research Cambridge, Cambridge, UK, available at http://research. microsoft. com/infernet*, 115, 2010.

[13] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011.

[14] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *The Journal of Machine Learning Research*, 99:1297–1322, 2010.

[15] E. Simpson, S. Roberts, I. Psorakis, and A. Smith. Dynamic bayesian combination of multiple imperfect classifiers. In *Decision Making and Imperfection*, pages 1–35. Springer, 2013.

[16] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proc. of the 2013 Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 901–908, 2013.

[17] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *Proc. of the 21st international Conf. on World Wide Web*, pages 989–998. ACM, 2012.

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

[19] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conf. on*, pages 25–32. IEEE, 2010.

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

[21] J. M. Winn and C. M. Bishop. Variational message passing. In *Journal of Machine Learning Research*, pages 661–694, 2005.

[22] D. Zhou, J. Platt, S. Basu, and Y. Mao. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Springer, 2012.