PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

A FULL PROBABILISTIC MODEL FOR YES/NO TYPE CROWDSOURCING IN MULTI-CLASS CLASSIFICATION

BELÉN CAROLINA SALDÍAS FUENTES

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
KARIM PICHARA

Santiago de Chile, August 2017

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

A FULL PROBABILISTIC MODEL FOR YES/NO TYPE CROWDSOURCING IN MULTI-CLASS CLASSIFICATION

BELÉN CAROLINA SALDÍAS FUENTES

Members of the Committee:
KARIM PICHARA
DENIS PARRA
PAVLOS PROTOPAPAS
ALEJANDRO JARA
NESTOR ESCALONA

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, August 2017

To my mother, siblings and friends

ACKNOWLEDGEMENTS

I would like to thank my advisor Karim Pichara for supporting me through all my learning progress, for always trusting me and encourage me to do my best under any scenario. Also, for giving me many opportunities to get new skills, for sharing his constant interest for science and let me see beyond what I thought possible.

I would also like to thank Pavlos Protopapas for his strong commitment to my Master research. For letting me learn from scratch. Also, for being such a great person with all his students, what shows me the friendly side of science. Also, for extending my wings and make me believe everything is as possible as we focus on that.

Special thanks to my mother and siblings, for always giving me opportunities to develop my curiosity for computer sciences. Also, for never pulling me down when I came with a new idea. Thanks to my mother for teaching me how to work hard to achieve all our dreams, especially when it comes about Shakira (who inspired me).

I would also thank all my friends, especially Teresita Irarrázaval, Francisca Chadwick and María Juanita del Río, for always supporting me and showing me how fortunate I am to achieve this Master degree. Thanks to Christian Pieringer and Adrián Soto for their continuous support and wise advices. Also thanks to my work partners Nebil Kawas, Dalal Chahuán and Tamara Covacevich for their advices and for creating a productive place where to work.

Finally, I want to thank all the PUC Computer Science Department staff and professors, who make me feel at home. I would also thank the Harvard-Chile Data Science School for its support through all my research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Crowdsourcing has become widely used in supervised scenarios where training sets are scarce and hard to obtain. Most crowdsourcing models in the literature assume labelers can provide answers for full questions. In classification contexts, full questions mean that a labeler is asked to discern among all the possible classes. Unfortunately, that discernment is not always easy in realistic scenarios. Labelers may not be experts in differentiating all the classes. In this work, we provide a full probabilistic model for a shorter type of queries. Our shorter queries just required a "yes" or "no" response. Our model estimates a joint posterior distribution of matrices related to the labelers confusions and the posterior probability of the class of every object. We develop an approximate inference approach using Monte Carlo Sampling and Black Box Variational Inference, where we provide the derivation of the necessary gradients. We build two realistic crowdsourcing scenarios to test our model. The first scenario queries for irregular astronomical time-series. The second scenario relies on animals image classification. Results show that we can achieve comparable results with full query crowdsourcing. Furthermore, we prove that empirical bias plays an important role to model the labelers failures. Finally, we provide the community with two real data sets obtained from our crowdsourcing experiments. All our code is publicly available[1].

Keywords: Machine learning, supervised learning, crowdsourcing, training sets, variational inference, classification, query type, graphical models.

[1]Available at: `https://github.com/bcsaldias/yes-no-crowdsourcing`

RESUMEN

Crowdsourcing se ha convertido en una técnica ampliamente adoptada en escenarios donde los conjuntos de entrenamiento para modelos supervisados son escasos y difíciles de obtener. La mayoría de los modelos de *crowdsourcing* en la literatura asumen que los anotadores pueden proporcionar respuestas para preguntas completas, estas se refieren a preguntarle a un anotador que discierna entre todas las clases posibles para un objeto. Desafortunadamente, ese discernimiento no siempre es fácil en escenarios realistas, pueden haber muchas clases donde se desconoce cómo diferenciarlas. En este trabajo, se propone un modelo probabilístico para un tipo más corto y fácil de preguntas. Estas preguntas más simples sólo requieren una respuesta del tipo "sí" o "no". Este modelo estima una distribución posterior conjunta de matrices relacionadas con las confusiones y errores de los anotadores, además de la probabilidad posterior de la clase de cada objeto. La solución se lleva a cabo mediante inferencia aproximada, se usa en primer lugar muestreo de Monte Carlo y en segundo lugar el método de Inferencia Variacional como Caja Negra (BBVI). Para este último enfoque se provee la derivación de los gradientes necesarios para la aproximación del modelo. Se construyeron dos escenarios web reales de *crowdsourcing*, donde anotadores fueron invitados a participar. En el primer escenario se muestran series de tiempo astronómicas a ingenieros y astrónomos. El segundo escenario se basa en clasificación de animales mediante la observación de imágenes. Los resultados muestran que es posible lograr resultados comparables con la pregunta completa para clasificación en *crowdsourcing*. Además, se prueba que tomar muestras de cómo los anotadores se equivocan al responder preguntas es importante para la convergencia del modelo. Finalmente, se deja disponible para la comunidad los dos conjuntos de datos obtenidos desde los experimentos reales generados. Todo el código está públicamente disponible[2].

[2]Disponible en: `https://github.com/bcsaldias/yes-no-crowdsourcing`

Palabras Claves:Aprendizaje automático, aprendizaje supervizado, *crowdsourcing*, conjuntos de entrenamiento, inferencia variacional, clasificación, tipos de preguntas, modelos gráficos probabilísticos.

1. INTRODUCTION

Labeled data is the very first requirement for training machines. In classification tasks, models need to be taught about which objects belong to which classes. The issue is that unlabeled data increases much faster than labeled data. Moreover, the availability of data has stimulated great breakthroughs in AI, more than the latest models have done. For example, convolutional neural networks (CNNs) were first proposed by LeCun et al. (1989), but only when ImageNet (Deng et al., 2009) achieved a corpus of 1.5 million labeled images and 1,000 object categories allowed Google's GoogLeNet (Krizhevsky et al., 2012; Russakovsky et al., 2014) to perform object classification almost as good as humans by using CNNs. This encourages us to focus on creating new mechanisms for producing labels. Nevertheless, labeling means getting ground truths, which is in most cases hard, expensive, or impossible to get.

In order to increase the amount of labeled data, we can use crowdsourcing (Dawid and Skene, 1979; Raykar et al., 2010; Simpson et al., 2013; Vondrick et al., 2013). In the absence of a perfect oracle, we can take advantage of crowd's knowledge to create training sets and classify, even though they may give imperfect opinions. Working with crowds of labelers requires less effort from each labeler since each one does not need to tag all the data. A big challenge is to combine unreliable crowd information: not entirely accurate, but cheaper (Yan et al., 2012). A typical case dealing with crowds is to trust each labeler equally. For example, taking the simple majority of votes for each object. For this to work we must assume everyone knows the same about the topic. This is in many cases a wrong assumption. For instance, in an observatory with two senior astronomers and four new interns, the astronomers propose their best telescopes setting, but the interns reject to that configuration because they may not understand all the implications. We cannot just trust the majority.

From a different strategy, creating training sets has been approached with active learning (AL) solutions (Settles, 2010; Zhang and Chaudhuri, 2015; Yan et al., 2011). AL is a semi-supervised scenario in which a learning algorithm, iteratively selects the best instances to ask

a user for labels (for example the objects that confuse most the model). AL assumes a user as perfect labeler, whose responses are used to retrain the model.

Several different solutions for crowdsourcing with active learning scenarios have been proposed (Simpson et al., 2013; Zhang and Chaudhuri, 2015; Vondrick et al., 2013; Liu and Wang, 2012; Yan et al., 2012; Raykar et al., 2009) (see chapter 3). Most of them approach the solution by selecting the best instances as candidates to be tagged, considering the labelers' expertise and the way to combine the crowd's answers. Nevertheless, the main challenge remains on that manual labeling is extremely difficult when unlabeled data is abundant.

In this paper, instead of selecting the best instances as candidate labels for training the model, which as described above it is the traditional approach to crowdsourcing and active learning, we propose a novel approach based on the query type. Commonly in a four classes scenario, a labeler is asked the class of an object, with possible responses "A" or "B" or "C" or "D". Under this settings, we refer to the full question as ABCD question. The proposed model generates low-cost queries where each response produces partial information. This method iteratively selects, per labeler, a random object along with a random class' label, and then asks if that object belongs "yes" or "no" to that class (proposed YN question). See figure 1.1 for an example.

To represent the labelers' errors per response in a full ABCD question scenario, traditional models use different confusion matrices (Liu and Wang, 2012; Simpson et al., 2013). In our context, we need to measure the probability per labeler of giving a right answer when the class asked is $\mathcal{M}_{k'}$ and the true class is $\mathcal{M}_k$. We represent those probabilities in what we call a *credibility matrix* per labeler. Furthermore, we introduce the idea and proof the importance of having a supervised stage to learn the YN model. In a first stage, we ask for few known object labels with the aim of estimating prior credibility matrices. As we show in chapter 7, using that information we can ensure convergence, other way makes the inference model hard or impossible to converge to the true labels. The second stage is the labeling of unknown objects.

Complete **ABCD** query type:	
What class of animal is this?	***Bird*** or ***Reptile*** or ***Mammal*** or ***Amphibian***

Proposed **YN** query type:	
Is this animal a Bird?	***Yes*** *or* ***No***
Is this animal a Reptile?	***Yes*** *or* ***No***
Is this animal a Mammal?	***Yes*** *or* ***No***
Is this animal an Amphibian?	***Yes*** *or* ***No***

(a) Query types

Questions for ***Labeler 1***:	
Is this animal a Mammal?	***Yes*** *or* ***No***

Questions for ***Labeler 2***:	
Is this animal a Bird?	***Yes*** *or* ***No***
Is this animal a Mammal?	***Yes*** *or* ***No***
Is this animal a Reptile?	***Yes*** *or* ***No***

Questions for ***Labeler 3***:	
Is this animal a Bird?	***Yes*** *or* ***No***
Is this animal a Mammal?	***Yes*** *or* ***No***
Is this animal an Amphibian?	***Yes*** *or* ***No***
Is this animal a Reptile?	***Yes*** *or* ***No***

Questions for ***Labeler 4***:	
Is this animal a Bird?	***Yes*** *or* ***No***
Is this animal a Reptile?	***Yes*** *or* ***No***

(b) Questions for a crowd

Figure 1.1. Different query scenarios. Note that for the YN question the labelers do not need to know what is the ground truth to give accurate partial information. The figure 1.1(a) shows the spotted salamander, an amphibian. Figure 1.1(b) shows a possible scenario with four labelers, four classes, and "yes" or "no" questions for the showed animal.

The proposed method has many advantages over traditional approaches. First, the YN model focuses on the importance of learning a prior estimation of how the labelers fail. The YN strategy probabilistically learns this estimation as a prior parameters estimation for the labeling stage. Second, it provides partial information with less error. Third, the labelers do not need to know about all the classes. Finally, the method is independent of the kind of data, given that we only need to include labelers' votes, without worry about any representation of the objects to be classified.

This work makes the following summarized contributions:

(i) *Crowdsourcing query type*: We propose a new crowdsourcing framework to obtain labeled data focused on the query type. This method of getting data reduces the cost of other models because it can reconstruct the ground truth labels dealing only with

partial information. We show that the aggregation of partial information allows the YN model to ask fewer questions than others, for getting convergence.

(ii) *The training importance*: We propose a new first supervised stage for crowdsourcing scenarios. This first stage estimates the labelers' error per question per class (their credibility matrices) by asking for known objects. Then, in a second stage, the YN model uses that estimation as prior information. We find that learning how the labelers fail helps the model to reach better results.

(iii) *Implementations*: We compare two different implementations: The first approach is solving the problem with Monte Carlo Markov Chain, and the second is by using Black Box Variational Inference (Ranganath et al., 2014). We present the equations needed to solve the model which can be easily extended to any model with similar variable types.

(iv) *New data released*: We develop two real-world experiments with human crowds and publish the data.

The rest of this work is organized as follow: In chapter 2 we describe the main background theory of this work. In chapter 3 we present a brief description of the related work in crowdsourcing with imperfect labelers and active learning approaches. Chapter 4 explains the proposed model specification. Then, in chapter 5 we describe our implementations of the model. Chapter 6 describes the different dataset used for the experimental results exposed in chapter 7. Then, chapter 7 presents extensive experiments and analysis. Finally, in chapter 8 we discuss and conclude the main results of our work.

2. BACKGROUND THEORY

This chapter describes the main theory behind this work: how to represent probabilistic joint distributions by using graphs, following by how to approximate distributions with Markov chain Monte Carlo (MCMC) inference and variational inference (VI). We explore deeply VI by presenting the evidence lower bound (ELBO), mean field inference, stochastic variational inference (SVI), and black box variational inference (BBVI). We base our discussion strongly on Blei et al. (2017) and Murphy (2012).

2.1. Probabilistic Graphical Models

We represent the joint distribution of the proposed method with a probabilistic graphical model (PGM) (Koller and Friedman, 2009; Wainwright et al., 2008) (see section 4.2 for our model). A PGM is a graph-based representation for compactly encoding a complex distribution over a high-dimensional space. For example, figure 2.1 illustrates the elemental DawidSkene (Dawid and Skene, 1979) distribution for a crowdsourcing classification scenario. Where the circles represent random variables, observed variables are gray circles, and the points represent hyperparameters. When a set of variables share the same probability distribution, we can use the "plate" notation, which stacks identical objects in a rectangle representation. In that case, the plates dimensions are written in capital letters within the rectangles.

In the PGM showed in figure 2.1, $\mathcal{N}$ is the number of instances to be labeled, and $\mathcal{J}$ is the number of labelers, where $i \in \{1, ..\mathcal{N}\}$ and $j \in \{1, ..\mathcal{J}\}$. In the DawidSkene model, ρ is the initial parameter for the distribution over the hidden labels $\mathbf{Z}$, where z_i is the predicted label for the object $\mathcal{X}_i$. In that scenario, r_i^j represents the class given by labeler L_j to object $\mathcal{X}_i$, whose confusion matrix is Θ^j. In this case, if each Θ^j is a random variable instead of an hyperparameter, $\mathbf{Z}$ and Θ will be conditionally dependent given all the labelers votes $\mathbf{R}$ due to the graph structure. Following the Liu and Wang (2012) notation, in that model definition

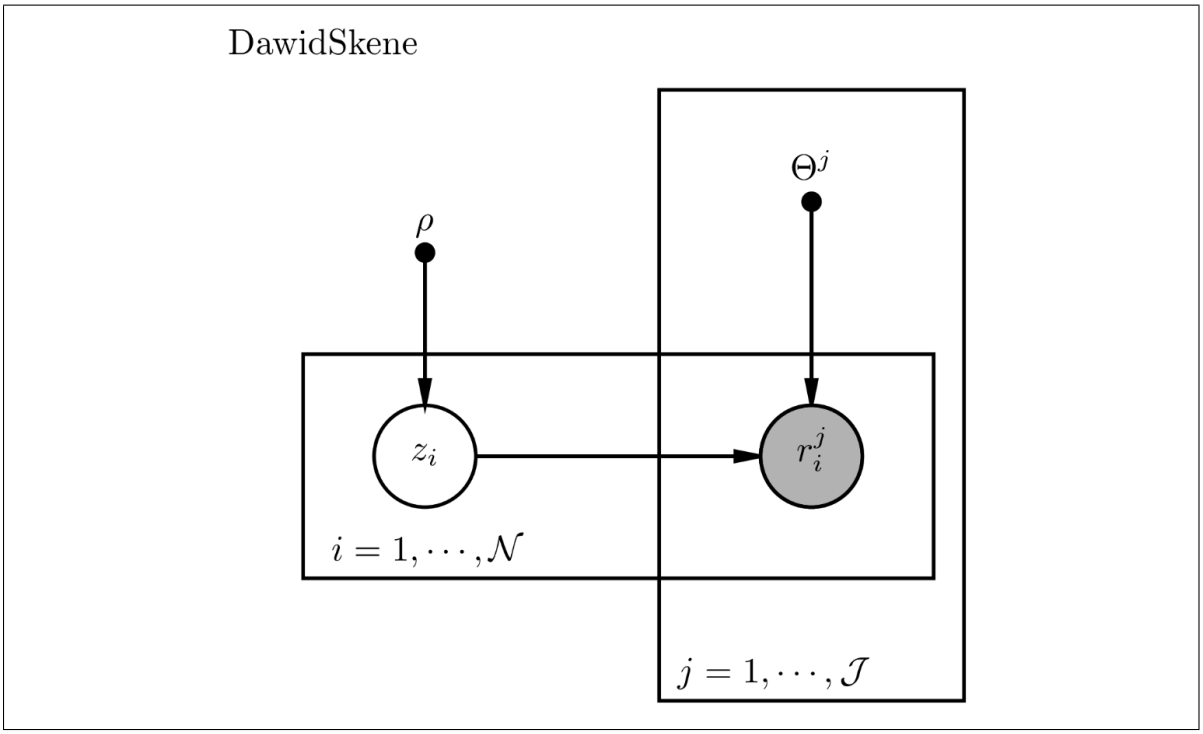

Figure 2.1. The DawidSkene model represented as a PGM in plate notation.

the variable distributions are:

$$z_i \sim \text{Multimonial}(\rho) \tag{2.1}$$

$$r_i^j \sim \text{Multimonial}(\Theta^j(z_i, :)) \tag{2.2}$$

This structure allows us to infer a compact representation of the explicit joint distribution. To get the posterior distribution, we can either use Bayesian Inference or Variational Inference. In this work, we address the proposed probabilistic model (see section 4.2) solution with approximate inference. In the following sections, we explain two approaches for inferring the posterior target distribution by approximating a distribution: Markov chain Monte Carlo and Variational Inference.

2.2. Markov Chain Monte Carlo

MCMC (Hastings, 1970; Gelfand and Smith, 1990) is the most popular method for sampling when simple Monte Carlo methods do not work well in high-dimensional spaces. The key idea is to build a Markov chain on the state space $\mathcal{Z}$ where the stationary distribution is the target, for instance a posterior distribution $\mathrm{p}(z|x)$, where x is observed data. MCMC performs a random sampling walk on the $\mathcal{Z}$ space, where the time spent in each state z is proportional to the target distribution. The samples allow us to approximate $\mathrm{p}(z|x)$.

MCMC approaches Bayesian inference with developments as the Gibbs sampler (Geman and Geman, 1984). The key idea behind Gibbs sampling is to turn the sampling among the variables. In each turn the sampler conditions a new variable sample s on the recent values of the rests of the distributions in the model. Suppose we want to infer $\mathrm{p}(z_1, z_2)$. In each iteration, we turn the samples iteratively: $z_1^{s+1} \sim \mathrm{p}(z_1|z_2^s)$ and $z_2^{s+1} \sim \mathrm{p}(z_2|z_1^s)$.

2.3. No-U-Turn Hamiltonian Monte Carlo (NUTS)

In this work, we use mainly NUTS (Hoffman and Gelman, 2014), an MCMC algorithm based on a Hamiltonian Monte Carlo sampler (HMC). As an advantage, NUTS avoids the random walk by using a recursive algorithm to obtain a set of candidate points widely spread over the target distribution. This informed walk makes the sampling converge faster. Furthermore, NUTS stops when the recursion starts to back and trace the dropped steps again. Nevertheless, HMC requires computing the gradient of the log-posterior to inform the walk, which can be hard.

Using NUTS does not oblige to establish the step size and the number of steps to converge, compared to what a simple MCMC or HMC sampler does. Setting those parameter would require preliminary runs and some expertise. This sampling stops when drawing more samples does no longer increase the distance between the proposal $\tilde{z}$ and the initial values of z.

Even though MCMC algorithms can be very slow when working with large datasets or very complex models, they asymptotically drawn exact samples from the target density (Robert, 2004). Under these heavy computational settings, we can use variational inference (VI) as an approximation to the target distribution. VI does not guarantee to find the density distribution, it only finds a close distribution, but usually it is faster than MCMC.

2.4. Variational Inference

Variational inference (VI) (Jordan et al., 1999; Wainwright et al., 2008) proposes a solution to the problem of posterior inference. VI selects an approximation $\mathrm{q}(z)$ from some tractable family and then it tries to make this $\mathrm{q}(z)$ as close as possible to the true posterior $\mathrm{p}^*(z) \triangleq \mathrm{p}(z|x)$. The VI approach reduces this approximation to an optimization problem, the minimization of the KL divergence (Kullback and Leibler, 1951) from q to p^*.

The KL divergence is a measure of dissimilarity of two probability distributions, p^* and q. Given that the forward KL divergence $\mathbb{KL}(\mathrm{p}^*||\mathrm{q})$ includes taking expectation over the intractable $\mathrm{p}^*(z)$, a natural alternative is the reverse KL divergence $\mathbb{KL}(\mathrm{q}||\mathrm{p}^*)$, defined in (2.3).

$$\mathbb{KL}(\mathrm{q}||\mathrm{p}^*) = -\int \mathrm{q}(z) \log \frac{\mathrm{q}(z)}{\mathrm{p}^*(z)} dz \tag{2.3}$$

2.5. The Evidence Lower Bound

Variational inference minimizes the KL divergence from q to p^*. It can be shown to be equivalent to maximize the lower bound (ELBO) on the log-evidence $\log \mathrm{p}(x)$. The ELBO is equivalent to the negative KL divergence plus a constant, as we show in the following definitions.

Lets say x are the observations, z the latent variables, and λ the free parameters of $\mathrm{q}(z|\lambda)$. We want to approximate $\mathrm{p}(z|x)$ by setting λ such as the KL divergence is minimum. In this

case we can rewrite (2.3), and expand the conditional in (2.4).

$$\mathbb{KL}(\mathrm{q}||\mathrm{p}^*) = \mathbb{E}_{\mathrm{q}}[\log \mathrm{q}(z|\lambda)] - \mathbb{E}_{\mathrm{q}}[\log \mathrm{p}(z,x)] - \log \mathrm{p}(x) \tag{2.4}$$

Therefore, the minimization of the KL in (2.5) is equivalent to maximizing the ELBO:

$$\mathcal{L}(\mathrm{q}) = \mathbb{E}_{\mathrm{q}}[\log \mathrm{p}(z,x) - \log \mathrm{q}(z|\lambda)] \tag{2.5}$$

2.6. Mean Field Inference

The optimization over a given family of distributions is determined by the complexity of the family. This optimization can be as difficult to optimize as complex is the family used. To keep the variational inference approach simple, Opper and Saad (2001) proposes to use the mean field approximation. This approach assumes that the posterior can be approximated by a fully factorized q, where each factor is an independent mean field variational distribution, as it is defined in (2.6).

$$\mathrm{q}(z) = \prod_{i=1}^{m} \mathrm{q}_i(z_i) \tag{2.6}$$

The goals is to solve the optimization in (2.7) over the parameters of each marginal distribution q.

$$\min_{\lambda_1,\ldots,\lambda_m} \mathbb{KL}(\mathrm{q}||\mathrm{p}^*) \tag{2.7}$$

2.7. Stochastic Variational Inference

Common posterior inference algorithms do not easily scale to work with high amounts of data. Furthermore, several algorithms are very computationally expensive because they require passing through the full dataset in each iteration. Under these settings, stochastic variational inference (SVI) (Hoffman et al., 2013) approximates the posterior distribution by computing and following its gradient in each iteration over subsamples of data. SVI iteratively takes samples from the full data, computes its optimal local parameters, and finally, it updates the global parameters.

SVI solves the ELBO optimization by using the natural gradient (Amari, 1998) in a stochastic optimization algorithm. This optimization consists in to estimate a noisy but cheap to compute gradient to reach the target distribution.

2.8. Black Box Variational Inference

The BBVI (Ranganath et al., 2014) avoids any model-specific derivations. Black Box VI proposes to stochastically maximize the ELBO using noisy estimates of its gradient. The estimator of this gradient is computed using samples from the variational posterior. Then, we need to write the gradient of the ELBO (2.5) in (2.8).

$$\nabla_\lambda \mathcal{L} = \mathbb{E}_{\mathrm{q}}[\nabla_\lambda \log \mathrm{q}(z|\lambda)(\log \mathrm{p}(z,x) - \log \mathrm{q}(z|\lambda))] \tag{2.8}$$

Using this equation, we can compute the noisy unbiased gradient of the ELBO sampling the variational distribution with Monte Carlo, as it is showed in equation (2.9), where S is the number of samples we take from each distribution to be estimated.

$$\nabla_\lambda \mathcal{L} \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log \mathrm{q}(z_s|\lambda)(\log \mathrm{p}(z_s,x) - \log \mathrm{q}(z_s|\lambda)) \tag{2.9}$$

where,

$$z_s \sim \mathrm{q}(z|\lambda) \tag{2.10}$$

For estimating the approximating q distribution, in BBVI the variational distributions $\mathrm{q}(z_i)$ are mean field factors with free variational parameters λ_i, for each index i (see (2.6)). In appendix A we show how to apply this method to the proposed model.

3. RELATED WORK

We discuss three relevant areas of related work in this chapter. First, the importance of training sets. Second, crowdsourcing with active learning for classification scenarios. Third, approximate variational inference approaches to crowdsourcing models.

3.1. Creating Training Sets

Creating training sets has become a central topic of supervised learning and generative models. Scientific works show that a model is as good as its training set[1]. For creating labels, we can manually label as many objects as we can. Furthermore, we can distribute the work among a crowd of labelers who give labels. An independent approach is to use active learning (AL) (Settles, 2010). AL assumes the presence of a perfect oracle, which gives labels only for the objects that improve most the model performance. We explore deeply crowdsourcing and active learning in the following section.

From another point of view, it is possible to use a programmatic paradigm for creating training sets called *data programming* (Ratner et al., 2016). In this paradigm, labelers give no labels but heuristics or strategies as labeling functions that return the asked labels. Furthermore, this approach is extended to crowdsourcing scenarios. There is a big difference between the proposed YN model and *data programming*, starting with the query type. *Data programming* asks for a function while we ask only for "yes" or "no" responses. An additional mechanism for labeling data is *co-training* (Blum and Mitchell, 1998), where the unlabeled data is classified under two conditional independent views. Each view is trained with a small amount of labeled data. In comparison to *co-training*, we can handle as many views as we want. Finally, close to our approach is the semi-supervised *boosting* (Schapire and Freund, 2012) procedure, which combines the output of several "weak" classifiers to create a "strong" classifier. In this case, we approach the weaknesses by modeling the labelers errors to infer the true labels probabilistically.

[1] http://www.kdnuggets.com/2016/05/datasets-over-algorithms.html

3.2. Crowdourcing Scenarios

There are several approaches to crowdsourcing with active learning that use the most relevant instances for being labeled by unreliable or noisy labelers (Simpson et al., 2013; Zhang and Chaudhuri, 2015; Vondrick et al., 2013). They all try to make the labeling task easier. Some works propose probabilistic graphical models as well as we do (Liu and Wang, 2012; Yan et al., 2012). However, we differ from them mainly on that they require the labelers to give the class as a full ABCD answer while in our work we can handle partial information asking for YN questions. In the YN scenario, labelers do not need to discern among all the possible classes. Furthermore, their selection criterion focuses the analysis by either selecting the instances to be labeled or asking the best labeler for classification (Raykar et al., 2009). We address the creation of training sets and labeling to a new goal, a new query type that diminishes the effort required from the labelers. In addition, several efforts have been made on how to estimate the labelers' expertises (Yan et al., 2010, 2011; Liu and Wang, 2012; Simpson et al., 2013; Zhang et al., 2014). We propose to estimate those expertises as prior of the inference model. Results show that using this semi-supervised approach the proposed model can converge quickly by asking only for few training objects.

There are some works that propose new query types on active learning scenarios. For instance, a different approach to ours (Rashidi and Cook, 2011) reduces the number of questions by asking generic queries. These queries are generated by rule induction from multiple unlabeled instances. Another approach classifies images by asking a different type or queries (Huang et al., 2015). This approach relies on asking about what label (given a pair of labels) is more relevant for each instance, where each image can be classified into more than one category. The closest research to the YN query type (Qi et al., 2008) assumes each instance could belong to more than one class. This assumption allows the model to take into account the label correlation between the classes for the classification. The main difference between the YN model and all these works relies on the fact that they do not propose a crowdsourcing

context to improve the scenario. They keep mainly the existence of a perfect oracle assumption. A work that changes this perfect oracle and the query type is closer to the hierarchical crowdsourcing (Otani et al., 2015), but it still asks full questions.

Until now, no paper has been presented for the integration of the query type importance, partial information requested to the labelers, and the crowd's power. We propose a mechanism that outperforms and handles many difficulties, as we expose in chapter 1.

3.3. Variational Inference Approaches

Another relevant theory behind this work is approximate inference. The Bayesian model and techniques proposed by Blei et al. (2003) pushed up several works based on probabilistic graphical models and variational inference methods. In this area, works such as Raykar et al. (2010); Yan et al. (2012) solve the semi-supervised learning scenario from a probabilistic perspective using mainly EM and MAP algorithms. The works with techniques closest to the YN model use the Gibbs sampler to infer ground truth labels for objects in crowdsourcing scenarios (Liu and Wang, 2012; Carpenter, 2011). There is also a nonparametric approach to labels inference that uses Gibb sampling (Moreno et al., 2015).

Simpson et al. (2013) propose a full variational Bayesian treatment. In this last work, they provide all the mean field equations to approximate the target distribution and evaluate the lower bound. We prefer to avoid this full derivation and focus only on the MCMC implementation and the Black Box (Ranganath et al., 2014) variational inference method. Following the full variational approach, the crowdsourcing scenario with unreliable labelers can be transformed into a standard inference problem in graphical models (Liu et al., 2012). This approach proposes to use belief propagation and the mean field theory connected to EM. The performance of this method critically depends on the choice of a prior distribution on the labelers' confusions. We propose an automatic solution for this problem, to separate the method into two stages: the first estimates how reliable the labelers are by asking for known objects labels, and the second stage asks for unknown objects labels.

All works proposed in this section have always studied either the method for a two-coin model or a multi-class scenario where they ask for full questions about what is the class, never changing the question asked the labelers as we do.

4. PROPOSED MODEL

4.1. Model Specification

Consider a dataset with $\mathcal{N}$ objects, each object $\mathcal{X}_i$ having only one true class z_i, between $\mathcal{K}$ possible classes, where $i \in \{1, ..., \mathcal{N}\}$ and $\mathbf{Z} = \{z_1, ..., z_i, ..., z_{\mathcal{N}}\}$. Each labeler $\mathcal{L}_j$, is then presented with a series of binary "yes" or "no" (YN) questions, where $j \in \{1, ..., \mathcal{J}\}$.

Formally, we define a YN question k_i^j as the question asked to the labeler $\mathcal{L}_j$ about whether $\mathcal{X}_i$ belongs "yes" or "no" to the class M_k, $k \in \{1, ..., \mathcal{K}\}$. We define $\mathcal{K}_i^j$ as the set of k_i^j queries asked to the labeler $\mathcal{L}_j$ for the object $\mathcal{X}_i$. Let r_{ik}^j be the response assigned by $\mathcal{L}_j$ to the question k_i^j, and $\mathbf{R}$ the set of every response r_{ik}^j. Note that classes are not necessarily in equal proportions, every labeler is not necessarily asked for all objects, and a labeler is not asked twice for the same class for the same object.

4.1.1. Responses/Votes

For object $\mathcal{X}_i$, labeler $\mathcal{L}_j$ and question k_i^j, it is convenient to encode the response as a two dimensional vector: $r_{ik}^j : [0, 1]$ where $[0, 1] \leftarrow$ [YES, NO]. Figure 4.1 shows an example of votes for the object $\mathcal{X}_i$ given by the labeler $\mathcal{L}_j$. Note that $r_{ik}^j = [0, 0]$ means the question k_i^j was not asked.

Question for	*Yes*	*No*
Class $\mathcal{M}_1$	1	0
Class $\mathcal{M}_2$	0	0
$\vdots$	$\vdots$	$\vdots$
Class $\mathcal{M}_k$	0	1
$\vdots$	$\vdots$	$\vdots$
Class $\mathcal{M}_\mathcal{K}$	0	1

Figure 4.1. Responses/votes r_i^j.

4.2. The Model

We propose a probabilistic graphical model to infer the true labels $\mathbf{Z}$. Figure 4.3 shows the YN graphical model. This represents the joint distribution of the labels and the rest of the variables involved. These variable distributions are assumed as it is presented in this chapter.

4.2.1. Credibility Matrices Modeling

Common Bayesian approaches for crowdsourcing classification use the confusion matrix of each labeler to represent their errors, due to the nature of the full question. We represent the YN error per labeler as a *credibility matrix*. In our context, we need to measure the probability per labeler of giving the right answer when the class asked is $M_{k'}$, and the true class is M_k. Figure 4.2 presents the credibility matrix of a specific labeler, where $\theta^j_{kk'}$ represents the probability of the labeler $\mathcal{L}_j$ of saying "yes" to the question k'^j_i when $z_i = k$.

The main goal of our model is to decide the most likely class for each object, given the votes of different labelers and their *credibility matrices* Θ. A side goal is to estimate those credibility matrices. In particular, we consider conjugate priors. Given that each "yes" or "no" response can be modeled as a $\mathrm{Bernoulli}$ distribution, the prior for each $\theta^j_{kk'}$ distributes as (4.1), where $\hat{\alpha}^j_{kk'}$ and $\hat{\beta}^j_{kk'}$ are the estimated prior initial parameters (see subsection 4.2.2).

$$\theta^j_{kk'} \sim \mathrm{Beta}(\hat{\alpha}^j_{kk'}, \hat{\beta}^j_{kk'}) \tag{4.1}$$

Finally, the likelihood is:

$$r^j_{kk'} \sim \mathrm{Bernoulli}(\theta^j_{kk'})$$

Modeling the prior of $\theta^j_{kk'}$ as a Beta distribution that lives in a 0 to 1 space allows us to model the probability of a response. Also, is a proper distribution because it is very flexible and can model any expertise (either high or low).

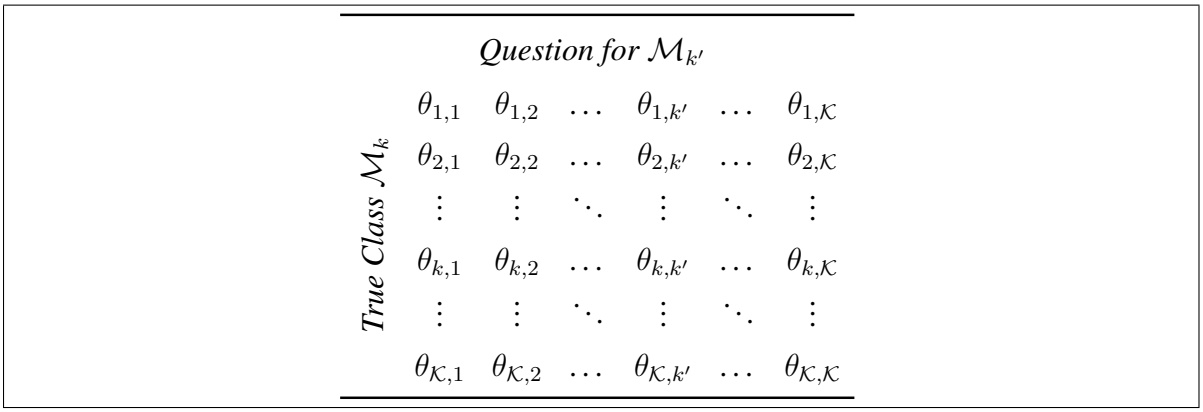

Figure 4.2. Credibility matrix. Note that the rows are not required to sum 1.

4.2.2. Probabilistic Model

In the crowdsourcing problem we address in this work, every instance in the dataset could be voted by no, one, or many unreliable labelers. In consequence, we need to be aware of the labeler's credibilities. The approached solution is to ask them for a training set to estimate their initial credibilities. In this estimation stage, the main issue is how much to ask each labeler.

Each YN vote r_{ik}^j depends on the real, but unknown label z_i. Furthermore, the vote also depends on the credibility $\theta_{z_i k}^j$ of the labeler $\mathcal{L}_j$. The conditioning to z_i allows the labeler to be more accurate in subsets of classes. The dependency on Θ^j allows us to model the labeler's biases and errors for all the classes. These dependencies are represented by the conditional distribution $\mathrm{P}(r_{ik}^j | z_i, \theta_{z_i k}^j)$ (Liu and Wang, 2012). Finally, we apply MAP to estimate the true labels $\mathbf{Z}$.

From prior information, we can estimate the initial classes proportions ρ, and we can define a global Dirichlet variable π in charge of this unknown distribution of the vector $\mathbf{Z}$. The distribution governing this random variable is:

$$\pi \sim \mathrm{Dirichlet}(\rho)$$

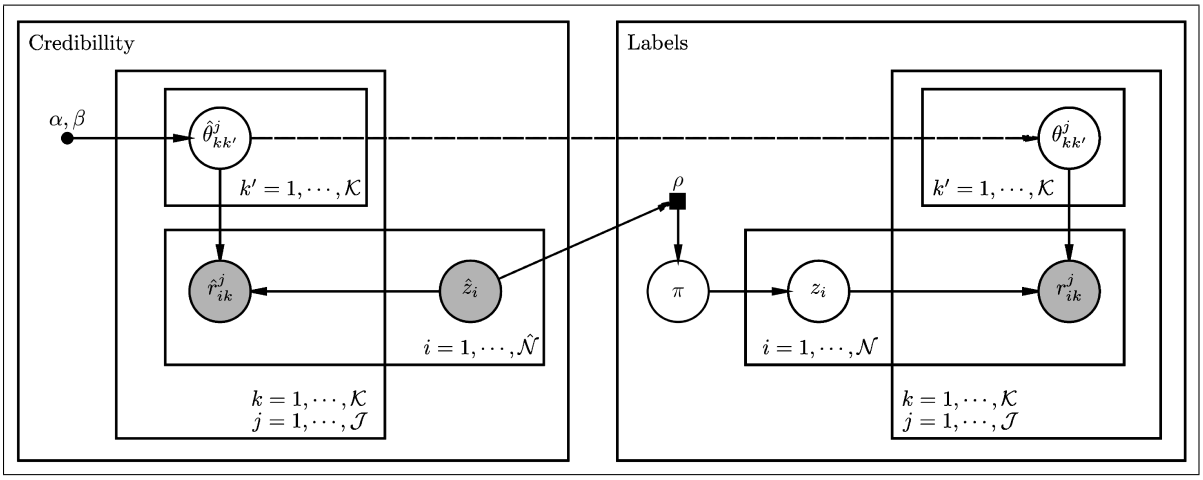

Figure 4.3. Proposed PGM. The proposed model can be implemented in two stages: first estimating the prior credibility $\hat{\Theta}$ and then the classification labeling model. In this plate notation, random variables are clear circles, observed variables are shaded in gray. Hyperparameters are represented by black points. The long dashed line represents prior parameters estimation, and the short dashed line points a parameter derived deterministically shown in a square.

The distribution for each z_i is:

$$z_i \sim \text{Categorical}(\pi)$$

4.2.2.1. Likelihood

We start from a single labeler, one object, and one question. For labeler $\mathcal{L}_j$ and question k^j_j the likelihood is in (4.2).

$$\mathrm{P}(r^j_{ik}|\theta^j_{z_ik}, z_i) = \overbrace{(\theta^j_{z_ik})^{r^j_{ik}[0]}}^{\overbrace{r^j_{ik}[0]=1, r^j_{ik}[1]=0}^{YES}} \times \overbrace{(1-\theta^j_{z_ik})^{r^j_{ik}[1]}}^{\overbrace{r^j_{ik}[0]=0, r^j_{ik}[1]=1}^{NO}} \tag{4.2}$$

For all responses $\mathbf{R}$, all labelers $\mathcal{L}$, and all data $\mathcal{N}$ the likelihood is in (4.3).

$$\mathrm{P}(\mathbf{R}|\Theta, \mathbf{Z}) \propto \prod_{i=1}^{\mathcal{N}} \prod_{j=1}^{\mathcal{J}} \prod_{k \in \mathcal{K}^j_i} \left\{ (\theta^j_{z_ik})^{r^j_{ik}[0]} \left(1-\theta^j_{z_ik}\right)^{r^j_{ik}[1]} \right\}. \tag{4.3}$$

4.2.2.2. Credibility Estimation

For training a graphical model, we need a supervised stage. In the YN scenario, we show examples to the model to obtain an initial parameters estimation. The training part is crucial to converge the model; otherwise, the model cannot know if it is doing well. Then, the training stage of the YN model is the prior Credibility Estimation. Following this idea, in this stage the users are presented with a set of $\hat{\mathcal{N}}$ objects $\hat{\mathcal{X}}_i$ that we know the true labels $\hat{\mathbf{Z}}$ (we refer to this set as Training Set). In this scenario, the model can learn beforehand an estimation $\hat{\Theta}$ and converge faster, as we show in chapter 7. In this scenario, $\hat{\mathbf{Z}}$ are observed values. Therefore the likelihood for all responses $\hat{\mathbf{R}}$, all labelers $\mathcal{L}$, and all data $\hat{\mathcal{N}}$ is in 4.4.

$$\mathrm{P}(\hat{\mathbf{R}}, \hat{\mathbf{Z}} \,|\hat{\Theta}) \propto \prod_{i=1}^{\hat{\mathcal{N}}} \prod_{j=1}^{\mathcal{J}} \prod_{k \in \hat{\mathcal{K}}_i^j} \left\{ (\hat{\theta}^j_{\hat{z}_i k})^{\hat{r}^j_{ik}[0]} \, (1 - \hat{\theta}^j_{\hat{z}_i k})^{\hat{r}^j_{ik}[1]} \right\} \tag{4.4}$$

The prior distribution of each $\hat{\theta}$ is chosen to be uninformative, but flexible enough to represent either a labeler with high or low expertise. We select $\mathrm{Beta}(\alpha, \beta)$ distributions with expected value equivalent to 0.5. We test using $\alpha = \beta = 0.5$ and $\alpha = \beta = 2$ (see chapter 7). Therefore,

$$\hat{\theta}^j_{kk'} \sim \mathrm{Beta}(\alpha, \beta)$$

Finally, this model works in two different stages: first estimating the prior credibility $\hat{\Theta}$ and then running the classification labeling model to infer $\mathbf{Z}$ and Θ. Then, $\hat{\alpha}^j_{kk'}$ and $\hat{\beta}^j_{kk'}$ from equation (4.1) are the estimated parameters of $\hat{\theta}^j_{kk'}$ from $\hat{\Theta}$, the Credibility Estimation stage.

5. IMPLEMENTATION

Our implementation is in Python 3.5 (van Rossum and , eds), using mainly MCMC sampling. Besides, we compare performance between solving the model with MCMC sampling and the Black Box Variational Inference (Ranganath et al., 2014) implementation of our model. The plots were generated using Matplotlib (Hunter, 2007) and Skit-learn tools (Pedregosa et al., 2011). All the data preprocessing, and analysis were done with Pandas (McKinney et al., 2010), Numpy (S. van der Walt and Varoquaux, 2011) and Scipy (Jones et al., 2014).

5.1. Monte Carlo Sampling - MCMC

We used MCMC sampling from PyMC3 (Salvatier et al., 2016). PyMC3 uses NUTS. This implementation includes mainly a DensityDist PyMC3 variable to evaluate the log-likelihood. All the presented results were run with this implementation, except for the experiment that compares this with the BBVI implementations due to the classification performance results.

5.2. Variational Inference - BBVI

Due to the convergence time of the PyMC3 implementation, we decided to compare with BBVI (Ranganath et al., 2014). This approach tries to find a simple probability distribution that is closest (in KL divergence) to the true posterior distribution. As we did before, we separate the custom developed implementation into two stages as shown in figure 4.3. First, the latent variables to estimate are $\hat{\Theta}$. Second, for the labeling part, the latent variables are Θ, $\mathbf{Z}$, and π. We provide the derivation of the necessary gradients in appendix A.

6. DATA

We use simulated votes and real-world datasets with human crowds. The goal is to compare the performance of our proposed model to other methods and baselines. Since multiple binary YN votes were not available for the first experiments, we needed to simulate several labelers with different expertise. The three sources of labels we used are described in the following sections.

6.1. Synthetic Votes for Synthetic Data

To simulate the labelers and their votes we proceeded as follows: First, we created labels ($\hat{\mathbf{Z}}$ and $\mathbf{Z}$). Then, for each labeler, we created a credibility matrix. The modeled labelers have high expertise in at most half of the classes. Therefore each row is simulated using a $\mathrm{Beta}(0.5, 0.5)$ distribution except where the labeler is more accurate; those are with $\mathrm{Beta}(20, 1)$ distributions (because its expected value is close to 1). Finally, we simulate the votes using the labelers and true labels. When the labeler $\mathcal{L}_j$ is presented with the object $\mathcal{X}_i$ of class z_i, and it is asked k_i^j, we go to its credibility matrix to obtain the response for k_i^j. We take r_{ik}^j by flipping a coin with the probability given by $\theta_{z_i k}^j$.

6.2. Synthetic Votes for Real-World Data

The real dataset is a subset of the astronomical MACHO data (Cook et al., 1995) (we used 250 objects of 65 features). We train six different classifiers as labelers (2 Random Forest classifiers, 2 Logistic Regressions, and 2 Support Vector Machines) with equal training set sizes). We proceed as follow: First, we split the data into three different sets; one to train classifiers, another to infer $\hat{\Theta}$ and the last to test the model. Each labeler is composed of a pool of $\mathcal{K}$ one-vs-all classifiers, one per class. When a labeler is asked for k_i^j, we go to its one-vs-all binary classifier for the class $\mathcal{M}_k$ to get the probability of the object to belong to the class M_k. Then we flip a coin with that probability to obtain $\hat{\mathbf{R}}$ and $\mathbf{R}$.

MACHO data: The Massive Compact Halo Objects (MACHO) project provides a time series (light curves) dataset. The main purpose this project is the detection of microlensing events in the Magellanic Clouds and Galactic bulge. Table 6.1 shows the distribution of the subset used in our experiments. The four classes selected are: Eclipsing Binary stars (EB), Be stars (BE), Long period variable stars (LPV), and Cepheid stars (CEP). Random sampling was used to divide the selected dataset into three different sets: i) for training classifiers, ii) for training the credibility stage, and iii) for testing the final classification. The main challenges dealing with this data relies on the fact that those time series are not uniformly sampled. In addition, they do not have a defined shape or structure to be easily handled by every model. Several works have put their efforts on classifying astronomical irregular time series (Pichara and Soto, 2011; Pichara and Protopapas, 2013; Pichara et al., 2016). Proposing a low-cost model to classify this type of data extends to the astronomers a strategy for executing their classification tasks faster.

Table 6.1. Real datasets classes distributions

Macho Data		*The Catalina Surveys*		*Animals Data*	
EB	104	CEP	119	Mammal	232
BE	57	RRLYR	99	Bird	73
LPB	49	EB	80	Amphibian	31
CEP	40	LPV	60	Reptile	23

6.3. Real Votes for Real-World Data

Two websites[1] were set up to acquire data from human crowds. Each of them presented a contest to people related to the dataset domain. The domains were:

(i) Astronomical irregular time series: The relevance of irregular time series encourages us to handle their challenges in classification. From the human experiments,

[1]Available at: the link will be revealed at soon as the paper gets published.

we can show proofs on how our model can assist the astronomers' work. The first contest was to classify images of irregular time series of the Catalina Surveys (Drake et al., 2009). This surveys is composed first by the Catalina Sky Survey (CSS), which involves searches for rapidly moving Near Earth Objects (NEOs), and second by the Catalina Real-Time Transient Survey (CRTS), which includes searches for stationary optical transients (OTs). The labelers were astronomers and engineers related to the field. The experiments were done with a total of 8 labelers. Table 6.1 shows the selected subset of the Catalina Surveys.

(ii) Animals classes: Each labeler was presented with several images of each object. The objective of classifying animals was to compare the model in different fields. In chapter 7, we show that even though the presence of unreliable labelers, the YN model can reach high accuracy score. Table 6.1 shows the characteristics of this dataset[2]. The labelers selected were 11 university students.

Each dataset contains 4 classes and 318 unknown objects, for about 15 people. Each user was presented with 1 to 4 random YN questions per instance. Also, the sets have (i) 40 and (ii) 41 known objects. For those known objects and 80 of the 318 unknown objects, the users were asked the full ABCD questions as well, which ask for what is the class. The following results are based only on those labelers who finished at least 70% of the questions.

[2]The full dataset is available at: https://a-z-animals.com/animals/pictures/. We filter the data mainly in the number of mammals to make the contests commensurable. The class fish was removed to work with only four classes, and to increase the difficulties as well.

7. RESULTS

In this chapter, we develop several experiments to evaluate the YN model performance. We start checking for convergence to be able to apply the model to different scenarios, not only to the synthetic ones. Then we set up a full synthetic scenario where we have many noisy labelers; we want to simulate a real context before going for human votes. In this experiment, we want to ensure the YN method can differentiate whom to trust more and whom to ignore.

After that, we start with the classifiers' simulated votes. We need to decide how much to ask the labelers as a trade-off between the number of questions for instances within the training set and the testing set. Related to that relation of sizes, we check for how fast the proposed method can learn the labelers' errors, and how this learning rate affects the final classification accuracy score. Furthermore, using the classifiers' votes, we approximate the number of questions needed to recover as much information as the full ABCD question does. We call to that result ABCD equivalent questions, which are deeply explored in the experiments with human crowds.

Finally, with the real-world votes, we compare the two implementations that we develop. We evaluate classification performance, implementation technique, and memory and time spent to converge. Following that, we want to measure how well our model performs working with a crowd in two different real-world scenarios. Therefore, we compare the YN full model results against what we get by running the model with only one labeler (in which case we have much less available votes). To conclude, we define different ABCD equivalent questions, from where we can take conclusions on how many YN queries we need to achieve comparable results with the full ABCD query type.

Our experiments were divided into ten parts: First, two full experiments with synthetic data. Then, four aspects using classifiers as labelers, and finally, we set up the websites to get real crowd's results, which we present in four experiments. All the results are with the MCMC implementation, except the benchmark against BBVI. We applied 10 sampling chains per experiment to validate our results.

7.1. Convergence Simulations

Given that the goal of the YN model is to perform inference for the unknown variables $\hat{\Theta}$, π, $\mathbf{Z}$ and Θ, in this section, we check for convergence of all of these variables and accuracy score as well.

To check for convergence, we generate votes as we explained before. For synthetic and classifiers' votes, we work with six labelers and four classes. We ask each labeler a random number of questions for about 250 objects in each case. We ask from 1 to 4 (Random(1,4)) YN queries per instance to each labeler. Between 25 and 40 instances were used to approximate each credibility matrix $\hat{\Theta^j}$, the rest was used for testing.

7.1.1. Accuracy score convergence

For all the experiments we perform, the samples became completely stable after 3000 iterations. For every analysis done we *burn* the first 1500 samples, *thinning* the chains each 10 samples. Similar results for convergence are obtained from both classifiers' scenarios, and the two set up contests with real-world data.

In addition, to check for convergence of the full model, we analyze each variable convergence. The convergence diagnostics for our random variables is based on Gelman-Rubin statistic (Gelman and Rubin, 1992). To try this diagnostic, we need multiple chains to compare the similarity between them. Our experiments are based on 10 chains each. When the Gelman-Rubin ratio (potential scale reduction factor) is less than 1.1, it is possible to conclude that the estimation has converged. Figure 7.1 presents the potential scale reduction factors for all the estimated variables.

7.1.2. Labeling convergence

The main variables we want to converge are the labels $\mathbf{Z}$. According to figure 7.1 there is not disagreement on that each z_i converges.

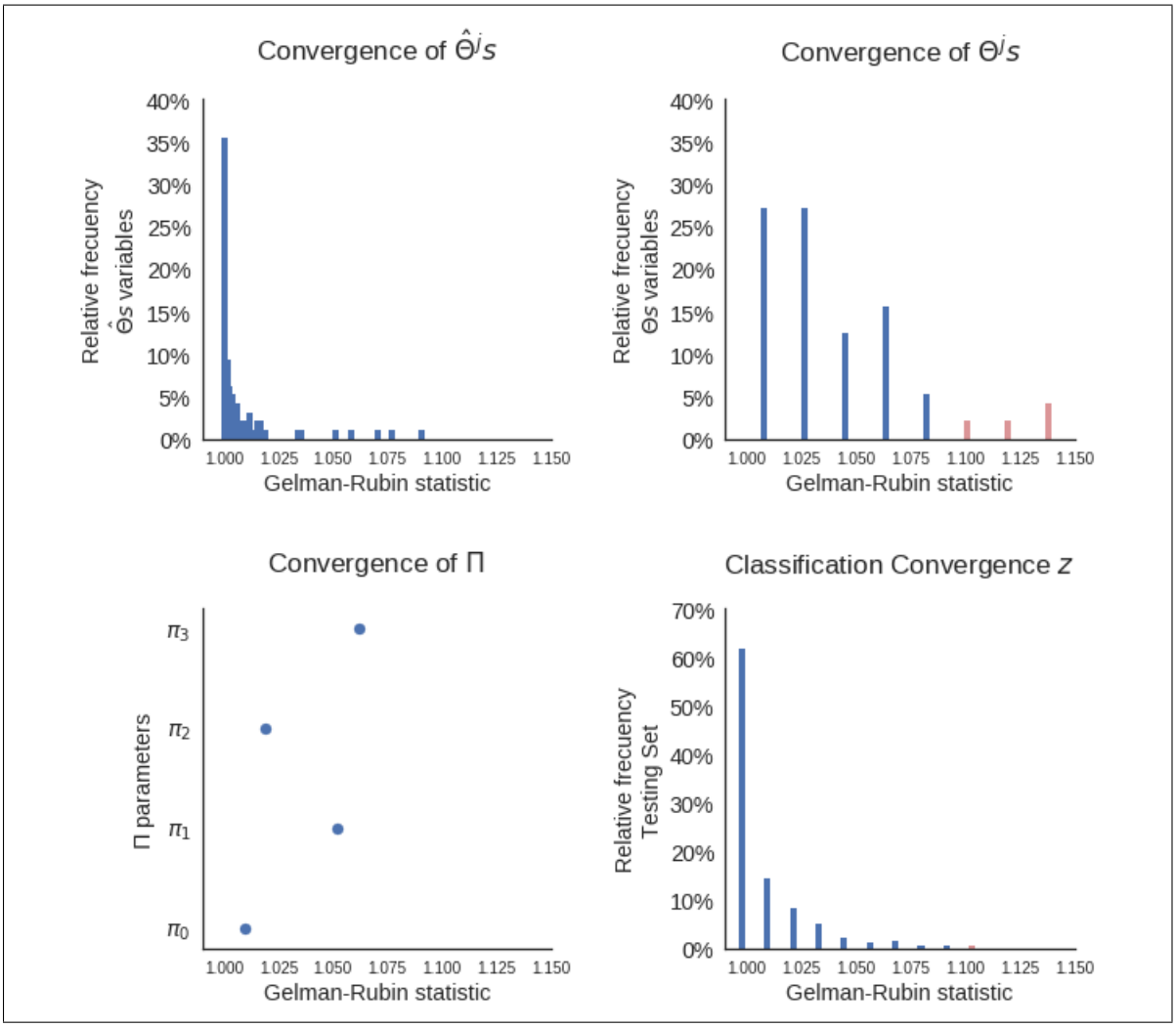

Figure 7.1. Accuracy score convergence. It is possible to see that Θ has more variance than any other variable, then some of the θs have not completely converged. We cannot conclude that the model has not converged, we only can say one of the chains has not converged. In practice that minimum percentage is not conditioning the full model.

7.2. Modeling the Crowd Expertise on Synthetic Data

To prove that our model can effectively differentiate between master and rookie labelers, we propose the comparison baselines as the ones used in Yan et al. (2010). In this scenario, we work with 7 synthetic labelers with higher expertise for at most two of four classes (as explained in chapter 6).

- *Proposed YN query*: As prediction we select the MAP of $\mathrm{P}(\mathbf{Z}|\mathbf{R})$.
- *Each labeler's ABCD simulated responses*: We ask one YN question k_i^j per object to each labeler, where $k = z_i$. It means we ask if $\mathcal{X}_i$ belongs "yes" or "no" (YN questions), to what we know is the true label z_i. Per labeler, we consider these answers as the full ABCD votes. We get the classification accuracy score as the proportion of right answers.
- *Majority*: As prediction we take the majority of the labelers' votes given the ABCD simulated responses.
- *Concatenate*: We select as the prediction the average of concatenating all the votes per each object using the ABCD simulated responses.

Figure 7.2 shows the performance of each method after convergence. We can see how our method outperforms all the baselines when the labelers do not have equal knowledge about all the classes.

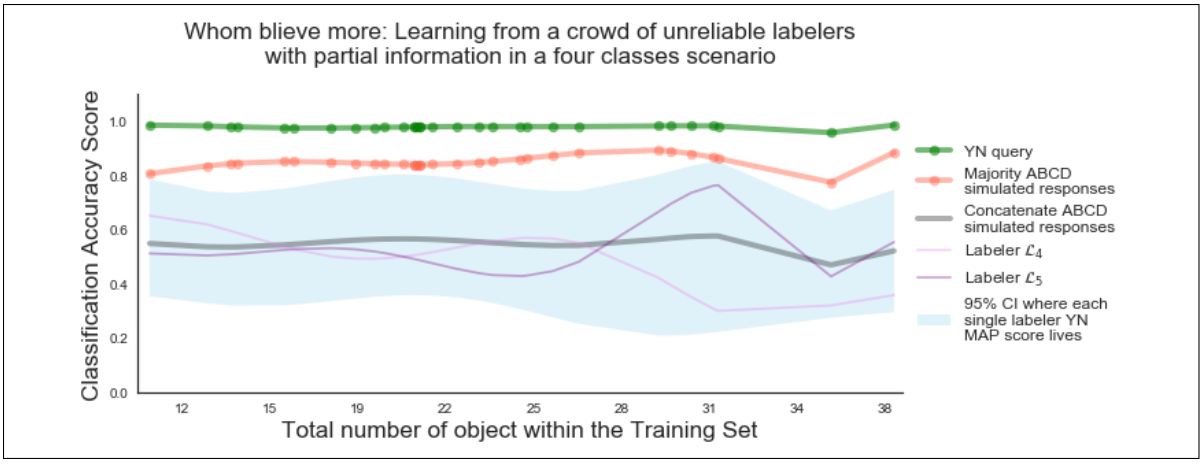

Figure 7.2. Modeling the crowd expertise on synthetic data. Note that each labeler score lives in a range of lower accuracy classification score than the YN and majority methods.

7.3. Performance Depending on the Training Set Size

We want to check how sensitive is our model to the hyperparameters α and β. In addition, we want to know how many objects we need, to converge the $\hat{\Theta}$ parameters estimation quickly. Figure 7.3 shows that the YN model can achieve equal results independent on the initial values of the hyperparameters. Furthermore, we can see that the learning rate growths logarithmically. It means that by asking just for few known objects $\hat{\mathcal{X}}_i$, the model can quickly converge to a good estimation of $\hat{\Theta}$ and $\mathrm{P}(\mathbf{Z}|\mathbf{R})$, independent on the testing set size $\mathcal{N}$.

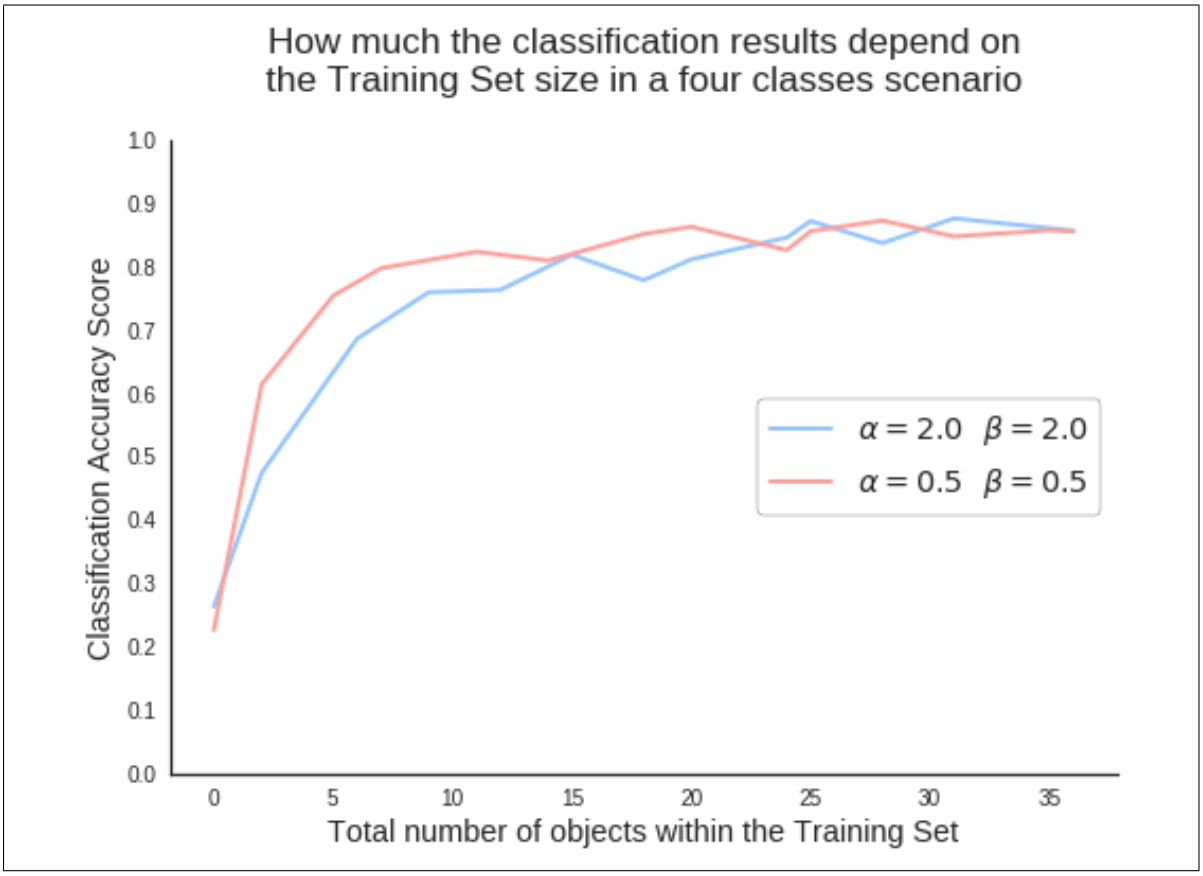

Figure 7.3. Classifiers voting for MACHO data. Note that increasing the training set size in about 10 instances produces that the model gets an increment of 50% (from 20% to 80%) in the classification accuracy score. This figure shows that after a small number of instances the accuracy remains more or less stable, independent to the training set size.

7.4. Recovery of Credibility Matrices $\hat{\Theta}$

The relation between the accuracy classification score and the training set size are closely related, as we show in figure 7.3. In addition, figure 7.4 shows the convergence speed of $\hat{\Theta}$ versus the training set size. From figure 7.3 we have that the accuracy score depends on the training set size. Figure 7.4 shows that the convergence of $\hat{\Theta}$ also depends on that size. Therefore, we have that if we can estimate a good prior $\hat{\Theta}$ of Θ, we can converge to a higher classification accuracy score. In consequence, the classification accuracy score depends on the convergence of $\hat{\Theta}$.

7.5. Performance Simulations Depending on Θ Convergence

Figure 7.5 shows that the better the model estimates the labelers' credibilities Θ, better is the classification accuracy score.

7.6. Performance Simulations on MACHO Data

As is shown in figure 7.6, in a four classes scenario our method reach the performance of the ABCD question, when we ask a $\mathrm{Random}(1, 4)$ number of YN questions per object to each labeler. The implemented baseline is the Bayesian ABCD model. This baseline is the Hybrid Confusion Matrix (Liu and Wang, 2012) based on the DawidSkene model (Dawid and Skene, 1979) plus the prior estimation stage of credibility matrices. Furthermore, the proposed model outperformed the ABCD method when we asked a YN question for every possible class $\mathcal{M}_k$ for every object $\mathcal{X}_i$.

7.7. Performance Real-World Votes MCMC vs BBVI

We developed all the previous simulations using the PyMC3 implementation mainly for two reasons. First, because even though when we used the AdaGrad (Ranganath et al., 2014) algorithm for setting the learning rate, this setting presents more parameters tunning than

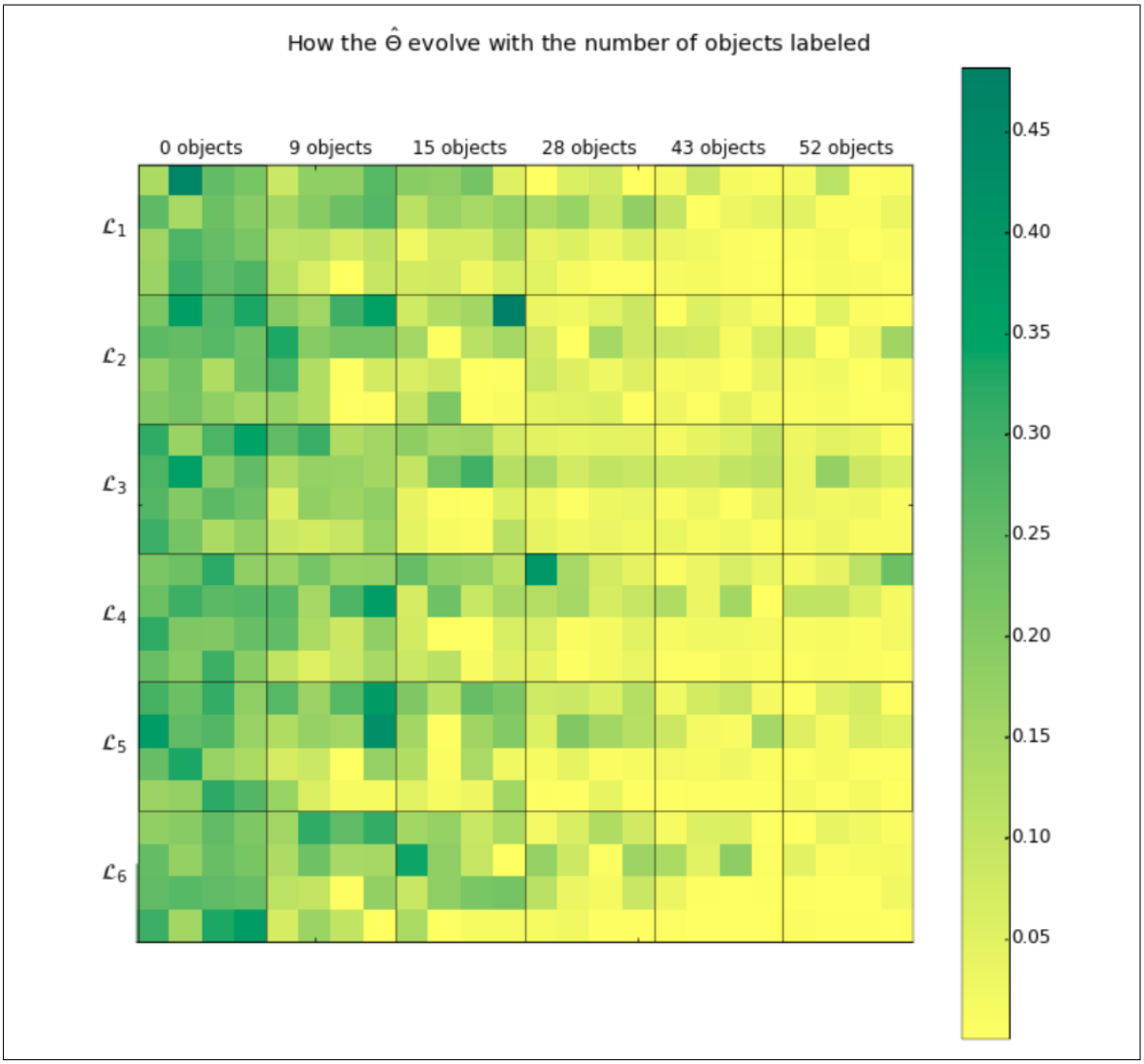

Figure 7.4. Classifiers voting for MACHO data MSE. MSE between each original Credibility Matrix row and its recovered $\hat{\theta}^j_{kk'}$ estimation with our method. Note that both figures 7.3 and 7.4 converge by the 35 objects. If we have 36 objects and 4 classes, each labeler votes for about 9 objects per class. $\mathrm{E}\{\mathrm{Random}(1,4)\} = 2.5$ questions per object implies 22.5 questions per class, it means about 5.6 votes for estimating each $\hat{\theta}^j_{kk'}$. We can see that the convergence of $\hat{\Theta}$ depends directly on that set size.

the MCMC parametrization. Second, because the results where most of the time slightly outperformed by the PyMC3 implementation.

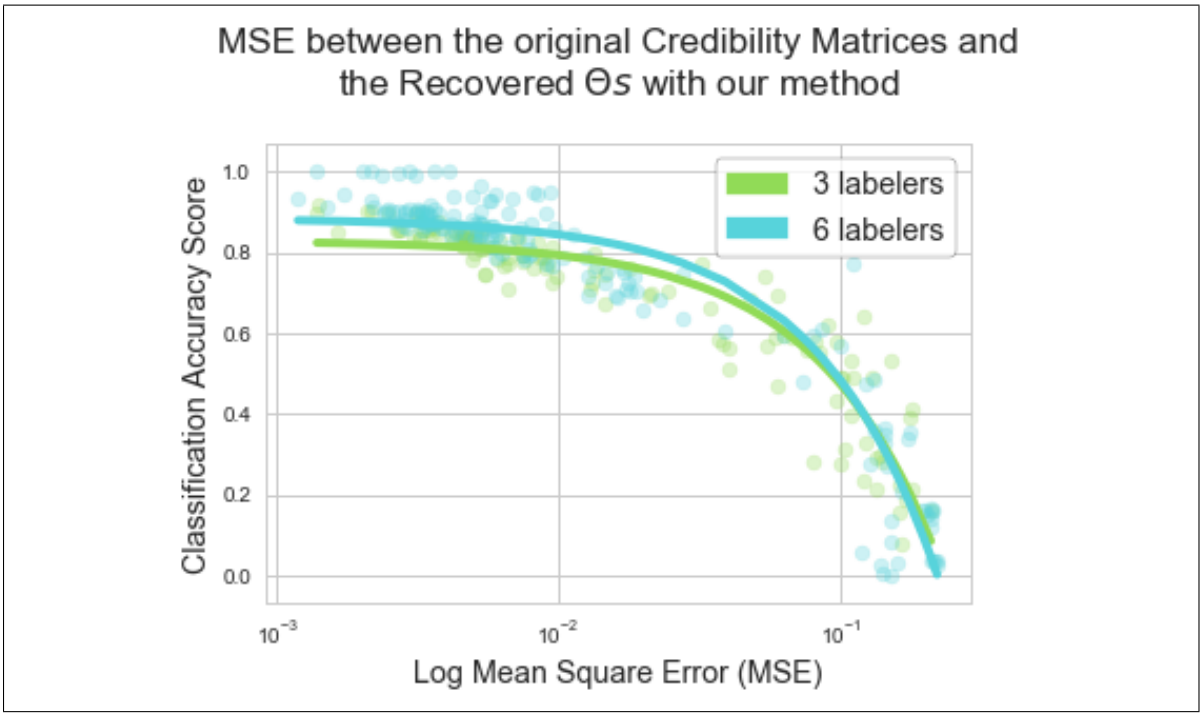

Figure 7.5. MSE between original Credibility Matrices and the Recovered ones. Classifiers voting for MACHO data. The error has two possible sources: i) an insufficient size of the training set, and ii) a lack of convergence in the model. In conclusion, how accurate is the estimation of $\mathrm{P}(\mathbf{Z}|\mathbf{R})$ depends on the quality of the estimation of Θ.

7.7.1. Iterations Until Converge

As we said before, PyMC3 needs about 3000 iterations until converge when running one chain. BBVI needs only 4 iterations, but each iteration implies to estimate the gradient of each latent variable, that means to take samples from the variational approximation distribution of every variable. This estimation converges at 3072 total samples.

7.7.2. Time and Memory Complexity

The model has $\mathcal{J} \times \mathcal{K} \times \mathcal{K} \times 2$ and $\mathcal{J} \times \mathcal{K} \times \mathcal{K} \times 2 + \mathcal{N} \times \mathcal{K} + \mathcal{K}$ parameters to estimate, respectively in each stage. If we assume that always $\mathcal{J} \times \mathcal{K} < \mathcal{N}$, this model is $\Omega(\mathcal{N} \times \mathcal{K})$. For both implementation we need samples. The memory complexity for the PyMC3 model is $\mathcal{O}(\mathcal{N} \times \mathcal{K} \times NumOfSamplesMCMC)$ and for the BBVI is $\mathcal{O}(\mathcal{N} \times$

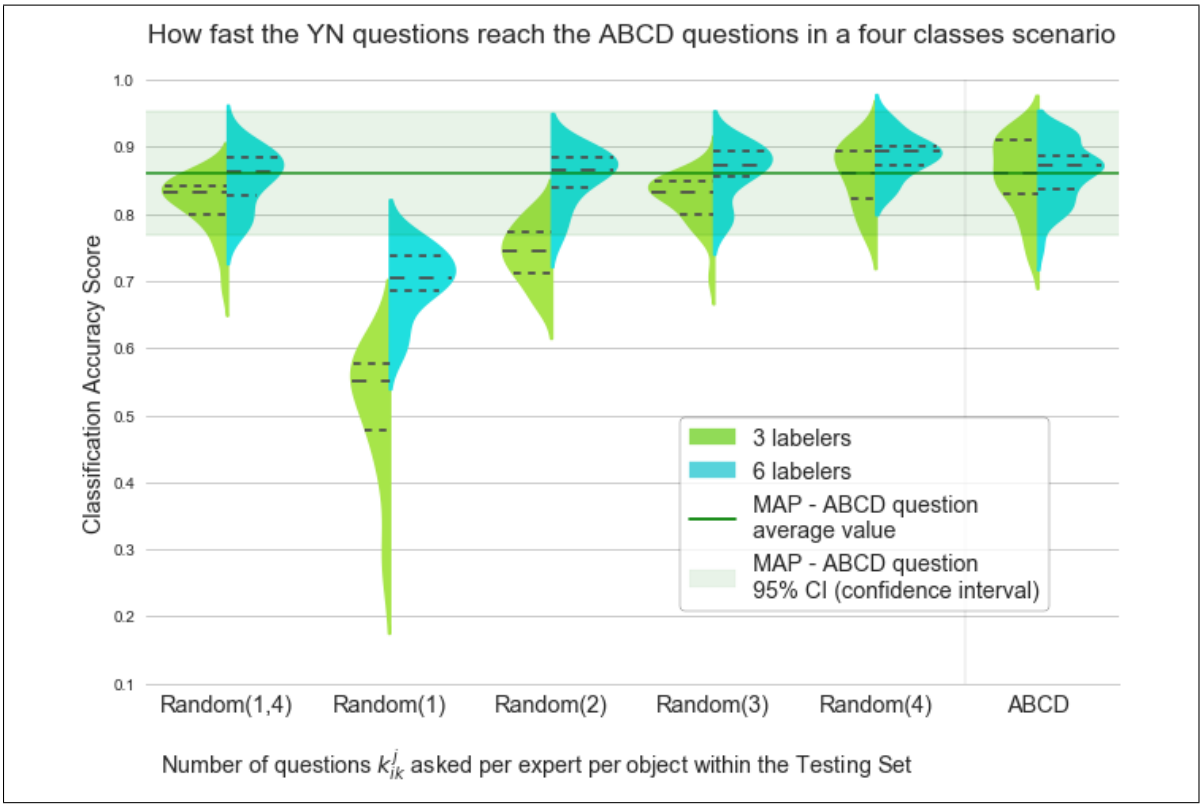

Figure 7.6. Scenarios with different amounts of questions. Classifiers voting for MACHO data. Random(w) means we asked each labeler for w different classes M_k a question k_{ik}^{j}, $w \leq \mathcal{K}$. The violin shape represents the cross validation results distribution.

$\mathcal{K} \times NumOfSamplesBBVI$), in that case both number of samples are constant, then the complexity is $\mathcal{O}(\mathcal{N} \times \mathcal{K})$. Both time complexities are equivalent.

7.7.3. Time Until Complete Convergence

The experiments performed a time of 10 minutes (PyMC3) versus 5 minutes (BBVI) for The Catalina Survey full model running 1 chain. Moreover, for the Animals Dataset 14 minutes (PyMC3) versus 7 minutes (BBVI) respectively. Taking into account the sizes were equal, those times depend only on the number of labelers, 8 and 11 respectively for each dataset. The time spent is linear on the number of chains for both models.

The results for The Catalina Surveys are shown in figure 7.7. It is possible to observe that for this data the MCMC model outperforms the BBVI implementation. For the Animals Data, both implementations get 100% of accuracy score. The BBVI implementations are both parametrized equally. We found that the BBVI approach can get higher accuracy if we fine-tune each learning rate of the latent variables.

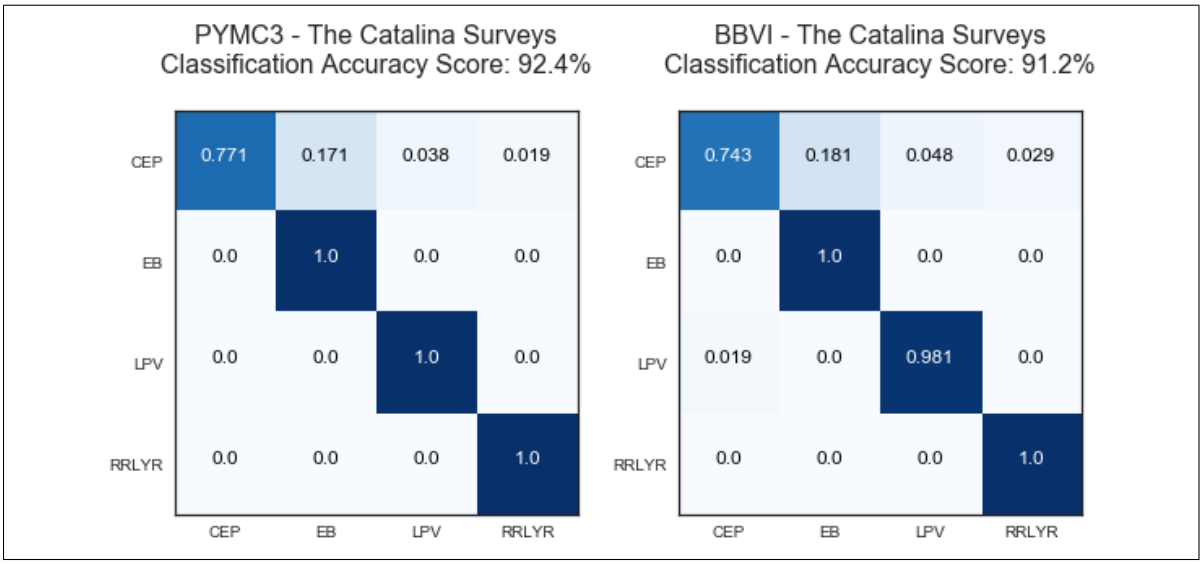

Figure 7.7. PyMC3 vs. BBVI. Confusion matrices for the learned models from Real-World Data.

7.8. Performance Crowd Versus Each Labeler

To evaluate the individual performance of each labeler versus the mixture of them, we take the MAP of the YN model for each labeler. Figure 7.8 shows the individual performance of each labeler selected from the contest with the Catalina Surveys. According to that, it is possible to see that our model can effectively model and integrate the unreliable crowd knowledge. The labelers' behavior for the Animals datasets is quite similar; many of them are unreliable, but the full model is more accurate than all the labelers.

We do not assume that during the credibility estimation labelers give responses to estimate every $\hat{\theta}^j_{kk'}$. This produces that the performance of the YN model with only one labeler can increase if we ask for more training set.

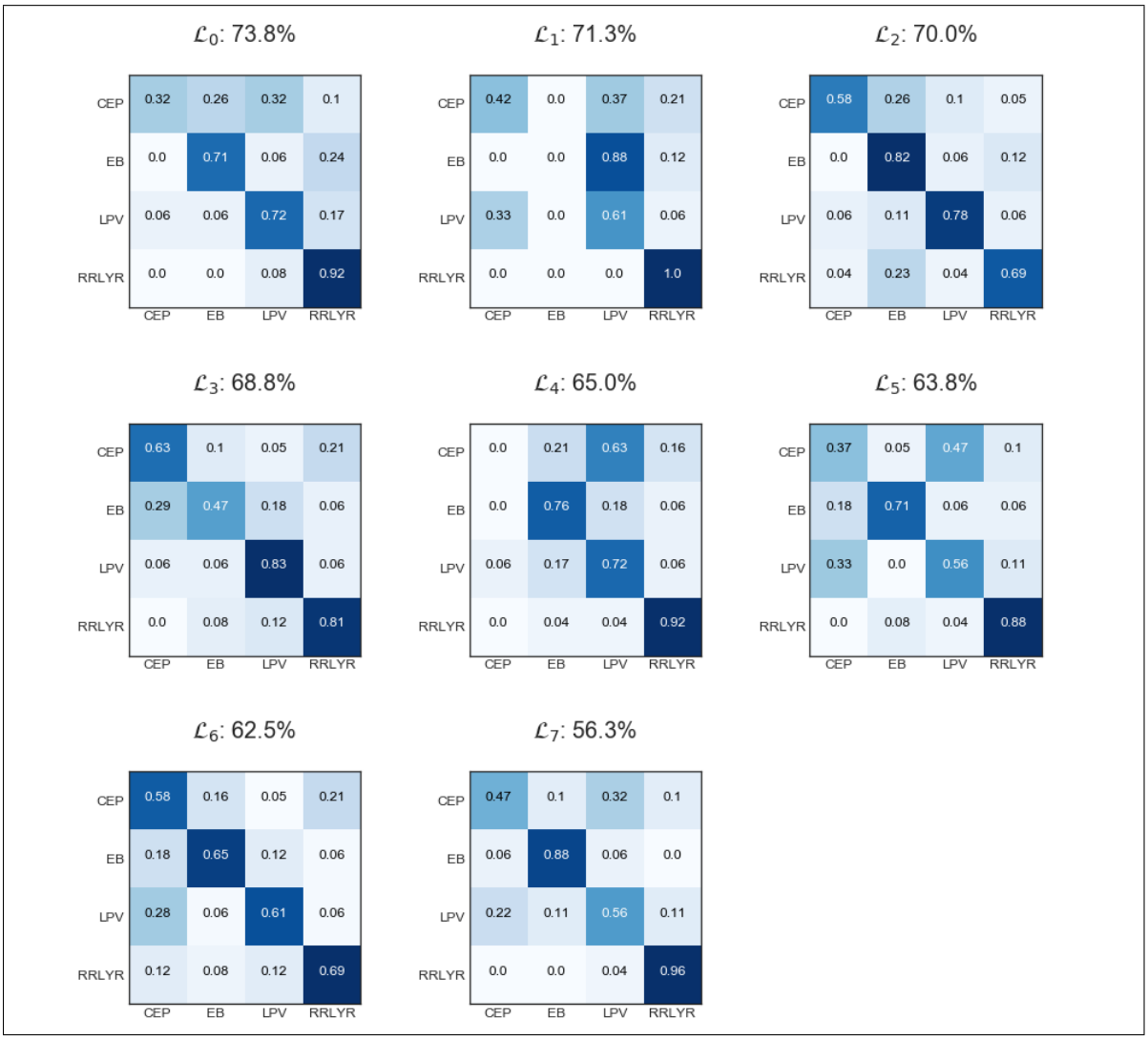

Figure 7.8. The Catalina Surveys contest's participants. Even though the labelers are confused, the YN model can learn how the labelers fail. We can observe in figures 7.7 that our model outperforms each labeler.

7.9. Performance Real-World Votes YN vs ABCD

As we explained in chapter 6, each labeler was presented to a series of ABCD questions for 80 objects of all data, for which the labelers were asked for $\mathrm{Random}(1,4)$ YN queries as well. Table 7.1 shows those results.

Table 7.1. Accuracy scores - Objects with responses YN and ABCD

Contest	*YN query type*	*ABCD query type*
The Catalina Survey	91.2%	90.0%
Animals Dataset	100%	100%

7.10. Performance Analysis YN Question vs ABC Question

Finally, we analyzed the cost and performance of the number of YN queries versus the number of ABCD queries needed for convergence of the classification accuracy score. Although, the YN query requires less expertise than the full ABCD question, the time for giving a response is not proportional to the number of possible classes $\mathcal{K}$. It is shown in the websites time records, where each ABCD question needs at most two times the time that a YN question needs for being answered. To measure the cost, we compare how many YN queries versus how many ABCD queries are needed for the model to converge. We assume that each ABCD query is equivalent to give $\mathcal{K}$ YN votes (Welinder and Perona, 2010), because the ABCD response requires the labeler to recognize the YN response for all the $\mathcal{K}$ possible classes. Despite our assumption, that 4 YN queries are equivalent to 1 ABCD query, in figure 7.9 we present an analysis based on different ABCD equivalents questions.

Each experiment at the left side in figure 7.9 has a completely converged $\hat{\Theta}$. The analysis corresponds to how fast the accuracy classification score converges. We compare the accuracy score versus the number of ABCD equivalent question asked for each object within the testing set. The Training Set has an equal size for all experiments; then the comparison is based only on the number of extra question needed for labeling the Testing Set.

The experiments showed at the right side in figure 7.9 have a completely converged Θ, $\mathbf{Z}$, and π. This analysis is related to how many ABCD equivalent queries the model needs for the Training Set to reach the best performance possible. These experiments have the same testing sets for all the experiments; we only change the Testing Set size. This right side is equivalent to the analysis presented in figure 7.3.

Independent of our equivalent questions assumption, it is possible to see in figure 7.9 that the YN strategy outperforms the ABCD strategy on the learning rate of $\hat{\Theta}$ at any equivalence. That higher learning rate implies lower human cost, because we need to bother the labelers with fewer questions for converging the model.

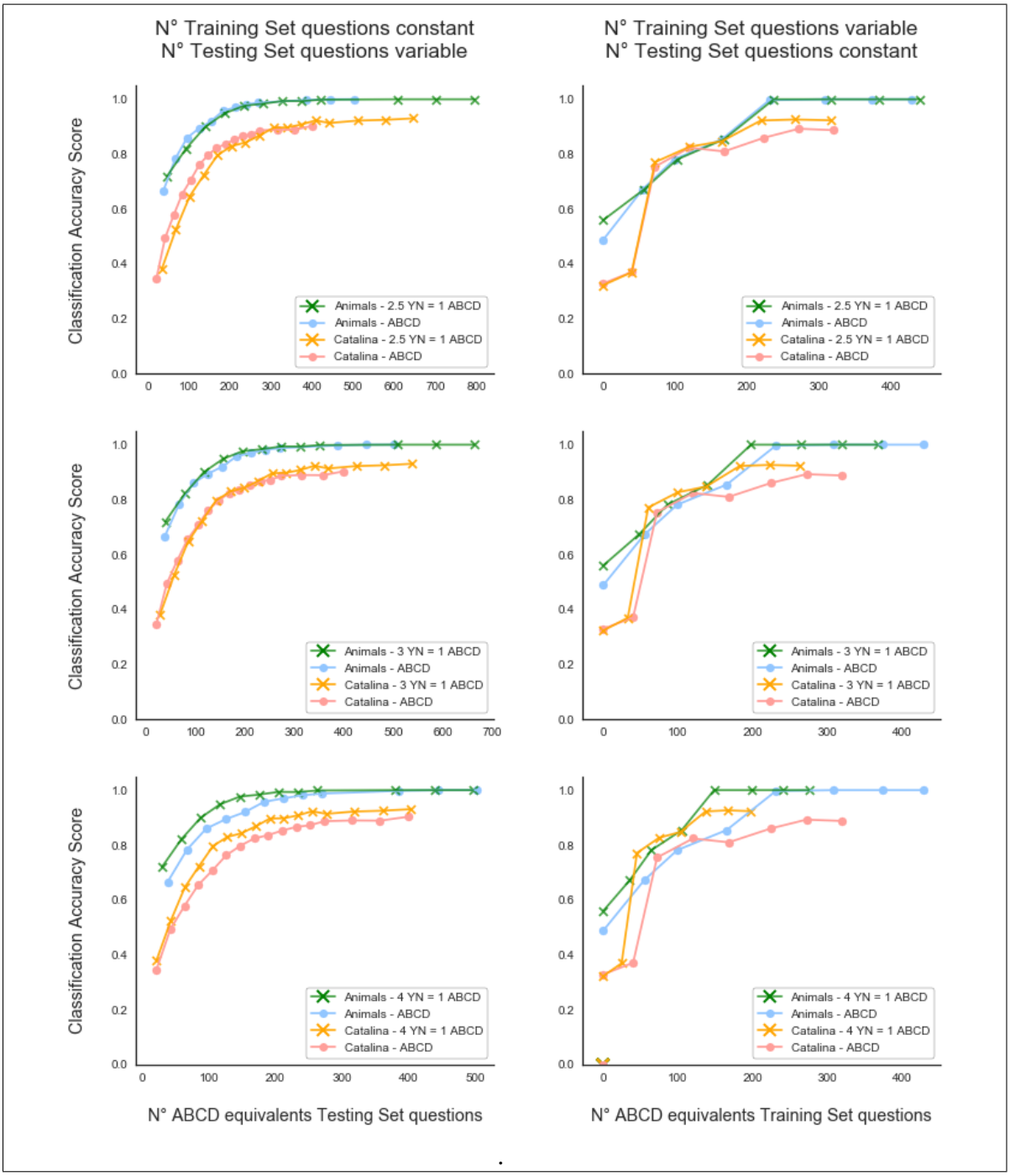

Figure 7.9. Results from two web contests. Real-World votes on two different scenarios.

8. CONCLUSIONS

We developed a new model for crowdsourcing with "yes" or "no" types of queries. Results show that our model obtains comparable results with models that ask full questions to the labelers. The proposed model can be applied to any crowdsourcing context, reducing the amount of effort spent by the labelers. From the results, we could see that the effort reduction comes from the ability of our model to realize quickly the main mistakes made by the labelers. This ability is reflected as a more rapid convergence in the posterior probability of the credibility matrices parameters. Moreover, that fast convergence does not come with a sacrifice in accuracy, because our model can quickly detect most of the classes where the labelers have low confidence. That observation is key to the proposed approach, because contrarily to what we may think apriori, detecting where the labelers are weak is as informative as detecting where the labelers are strong. From the mixture-of-labelers perspective, we could see from the results that in cases where most of the labelers were unreliable in many classes, the full probabilistic model was able to capture the right posterior of the classes by taking advantage of crowds. That occurs mainly because the model learns to map crowds' responses between the credibility estimation stage and the labeling stage.

As a future work, the model should include the possibility to increase the labelers expertise with time, something that makes much sense in real scenarios. That could be achieved by introducing an extra variable in the distributions governing the credibility matrices parameters. As iterations increase, the posterior update should take into account the labeler gained expertise. Another interesting idea we could exploit in future work is to consider that the order in which we present the questions may create a bias in the labelers. For example, if we ask if a cereal bar is healthy after asking for lettuce is not the same as asking if a cereal bar is healthy after showing a burger. Objects shown in previous iterations may modify the way that the labelers perceive our questions. There is also space for a further contribution in the way we select the labeler at each iteration. So far we just randomly select an object along with a class to ask each labeler available for an answer. This selection can be optimized by choosing the following labeler according to an expected information gain criterion.

It can also improve the selection criterion the inclusion of some features of the labelers in the model. Unfortunately, most of the approaches to estimate expectations make the model extremely slow, because on each iteration we would have to calculate integrals over all the possible objects and answers. Another future work, from a statistical perspective, could be going deep into the identifiability analysis of the proposed model.

REFERENCES

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted).

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Carpenter, B. (2011). A hierarchical bayesian model of crowdsourced relevance coding. In *TREC*.

Cook, K. H., Alcock, C., Allsman, R., Axelrod, T., Freeman, K., Peterson, B., Quinn, P., Rodgers, A., Bennett, D., Reimann, J., et al. (1995). Variable stars in the macho collaboration 1 database. In *International Astronomical Union Colloquium*, volume 155, pages 221–231. Cambridge University Press.

Dawid, A. P. and Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Drake, A., Djorgovski, S., Mahabal, A., Beshore, E., Larson, S., Graham, M., Williams, R., Christensen, E., Catelan, M., Boattini, A., et al. (2009). First results from the catalina real-time transient survey. *The Astrophysical Journal*, 696(1):870.

Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, pages 457–472.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

Huang, S.-J., Chen, S., and Zhou, Z.-H. (2015). Multi-label active learning: Query type matters. In *IJCAI*, pages 946–952.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Jones, E., Oliphant, T., and Peterson, P. (2014). {SciPy}: open source scientific tools for {Python}.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Liu, C. and Wang, Y.-M. (2012). Truelabel+ confusions: A spectrum of probabilistic models in analyzing multiple ratings. *arXiv preprint arXiv:1206.4606*.

Liu, Q., Peng, J., and Ihler, A. T. (2012). Variational inference for crowdsourcing. In *Advances in neural information processing systems*, pages 692–700.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. SciPy Austin, TX.

Moreno, P. G., Artés-Rodríguez, A., Teh, Y. W., and Perez-Cruz, F. (2015). Bayesian nonparametric crowdsourcing. *Journal of Machine Learning Research*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Opper, M. and Saad, D. (2001). *Advanced mean field methods: Theory and practice*. MIT press.

Otani, N., Baba, Y., and Kashima, H. (2015). Quality control for crowdsourced hierarchical classification. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 937–942. IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pichara, K. and Protopapas, P. (2013). Automatic classification of variable stars in catalogs with missing data. *The Astrophysical Journal*, 777:83.

Pichara, K., Protopapas, P., and Leon, D. (2016). Meta-classification for variable stars. *The Astrophysical Journal*, 819(1).

Pichara, K. and Soto, A. (2011). Active learning and subspace clustering for anomaly detection. *Intelligent Data Analisys*, 15(2):151–171.

Qi, G.-J., Hua, X.-S., Rui, Y., Tang, J., and Zhang, H.-J. (2008). Two-dimensional active learning for image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Ranganath, R., Gerrish, S., and Blei, D. M. (2014). Black box variational inference. In *AISTATS*, pages 814–822.

Rashidi, P. and Cook, D. J. (2011). Ask me better questions: active learning queries based on rule induction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 904–912. ACM.

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. (2016). Data programming: Creating large training sets, quickly. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3567–3575. Curran Associates, Inc.

Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., and Moy, L. (2009). Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual international conference on machine learning*, pages 889–896. ACM.

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

Robert, C. P. (2004). *Monte carlo methods*. Wiley Online Library.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2014). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*.

S. van der Walt, S. C. C. and Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55.

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Simpson, E., Roberts, S., Psorakis, I., and Smith, A. (2013). Dynamic bayesian combination of multiple imperfect classifiers. In *Decision making and imperfection*, pages 1–35. Springer.

van Rossum, G. and (eds), F. D. (2001). Python reference manual.

Vondrick, C., Patterson, D., and Ramanan, D. (2013). Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204.

Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

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

Yan, Y., Fung, G. M., Rosales, R., and Dy, J. G. (2011). Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1161–1168.

Yan, Y., Rosales, R., Fung, G., and Dy, J. (2012). Modeling multiple annotator expertise in the semi-supervised learning scenario. *arXiv preprint arXiv:1203.3529*.

Yan, Y., Rosales, R., Fung, G., Schmidt, M. W., Valadez, G. H., Bogoni, L., Moy, L., and Dy, J. G. (2010). Modeling annotator expertise: Learning when everybody knows a bit of something. In *International conference on artificial intelligence and statistics*, pages 932–939.

Zhang, C. and Chaudhuri, K. (2015). Active learning from weak and strong labelers. In *Advances in Neural Information Processing Systems*, pages 703–711.

Zhang, Y., Chen, X., Zhou, D., and Jordan, M. I. (2014). Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pages 1260–1268.

APPENDIX

A. DERIVATION BLACK BOX INFERENCE EQUATIONS

The BBVI (Ranganath et al., 2014) minimizes the KL divergence from an approximating distribution q to the true p posterior. Lets say x are the observations, z latent variables, and λ the free parameters of $\mathrm{q}(z|\lambda)$. And we want to approximate $\mathrm{p}(z|x)$ by setting λ. This optimization is equivalent to maximizing the ELBO in (2.5):

$$\mathcal{L}(\lambda) = \mathbb{E}_{\mathrm{q}}[\log \mathrm{p}(z, x) - \log \mathrm{q}(z|\lambda)]$$

BBVI proposes stochastically maximize the ELBO using noisy estimates of its gradient. The estimator of this gradient is computed using samples from the variational posterior. Then, we need to write the gradient of the ELBO in (2.8):

$$\nabla_\lambda \mathcal{L} = \mathbb{E}_{\mathrm{q}}[\nabla_\lambda \log \mathrm{q}(z|\lambda)(\log \mathrm{p}(z, x) - \log \mathrm{q}(z|\lambda))]$$

Using (2.8), we can compute the noisy unbiased gradient of the ELBO sampling the variational distribution with Monte Carlo, as it is showed in (2.9), where S is the number of samples we take from each distribution to be estimated:

$$\nabla_\lambda \mathcal{L} \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log \mathrm{q}(z_s|\lambda)(\log \mathrm{p}(z_s, x) - \log \mathrm{q}(z_s|\lambda))$$

$$\text{Where, } z_s \sim \mathrm{q}(z|\lambda)$$

For estimating the approximating q distribution, BBVI uses the mean field theory. Then we define the approximating distribution q as in (2.6):

$$\mathrm{q}(z) = \prod_{i}^{m} \mathrm{q}_i(z_i)$$

The variational mean field distributions q from (2.6) in the Credibility Estimation (first stage) of the YN model are in (A.1). Whose free variational parameters to estimate are in

(A.2).

$$\mathrm{q}(\hat{\theta}^{j}_{kk'}) \sim \mathrm{Beta}(\hat{\alpha}^{j}_{kk'}, \hat{\beta}^{j}_{kk'}) \; \forall \; jkk' \tag{A.1}$$

$$\hat{\Theta} : \; (\hat{\alpha}^{j}_{kk'}, \hat{\beta}^{j}_{kk'}) \; \forall \; \hat{\theta}^{j}_{kk'} \tag{A.2}$$

For the Labeling part (second stage) of the proposed model, the mean field distributions q from (2.6) are defined in (A.3), (A.4) and (A.5).

$$\mathrm{q}(\theta^{j}_{kk'}) \sim \mathrm{Beta}(\alpha^{j}_{kk'}, \beta^{j}_{kk'}) \; \forall \; jkk' \tag{A.3}$$

$$\mathrm{q}(z_i) \sim \mathrm{Categorical}(\mathbf{p}_i) \; \forall \; i \tag{A.4}$$

$$\mathrm{q}(\pi) \sim \mathrm{Dirichlet}(\mathbf{d}) \tag{A.5}$$

Whose free variational parameters to estimate are in (A.6), (A.7), and (A.8) respectively.

$$\Theta : \; (\alpha^{j}_{kk'}, \beta^{j}_{kk'}) \; \forall \; \theta^{j}_{kk'} \tag{A.6}$$

$$\mathbf{z} : \; (p_{ik} \; \forall k \in K) \; \forall \; z_i \tag{A.7}$$

$$\pi : \; (d_k \; \forall k \in K) \tag{A.8}$$

As it is shown in (2.8), to maximize the ELBO, we need the expectations under q. Given that we prefer to avoid that derivation for the YN model joint distribution, we use the black box method by approximating the gradient of the ELBO as defined in (2.9).

For applying this method to our model, we need to write the needed functions for the Credibility stage and the Labeling stage as well. In this appendix, we show only the derivation for that second stage (the gradients for the training part are a simplification of the presented derivations).

A.1. Labeling Parameters Estimation

The joint distribution to be inferred is:

$$\log \mathrm{p}(r_{ik}^j, z_i, \theta_{z_i k}^j, \pi) = \log \mathrm{p}(\theta_{z_i k}^j | \alpha_0, \beta_0) + \sum_{i=1}^{\mathcal{N}} \left\{ \log \mathrm{p}(z_i | \theta_{z_i k}^j, \pi) + \log \mathrm{p}(r_{ik}^j | z_i, \theta_{z_i k}^j) \right\} \quad \text{(A.9)}$$

First, for each variable, we define the log probability of all distributions containing the free parameters in order to obtain the mean field q. The priors are:

$$\log \mathrm{p}(\theta_{kk'}^j | \alpha_0, \beta_0) = \log \mathrm{Beta}(\theta_{kk'}^j | \alpha_0, \beta_0) \quad \text{(A.10)}$$

$$\log \mathrm{p}(z_i | \theta_{kk'}^j, \pi) = \log \mathrm{Categorical}(z_i | \pi) \quad \text{(A.11)}$$

$$\log \mathrm{p}(\pi | \rho) = \log \mathrm{Dirichlet}(\pi | \rho) \quad \text{(A.12)}$$

$$\log \mathrm{p}(r_{ik}^j | z_i, \theta_{z_i k}^j) = \log \left\{ (\theta_{z_i k}^j)^{r_{ik}^j[0]} \times (1 - \theta_{z_i k}^j)^{r_{ik}^j[1]} \right\} \quad \text{(A.13)}$$

Then, we can write those log probabilities to estimate the gradient with respect to the variational parameters:

$$\log \mathrm{q}(\theta_{kk'}^j | \alpha_{kk'}^j, \beta_{kk'}^j) = \log \mathrm{Beta}(\theta_{kk'}^j | \alpha_{kk'}^j, \beta_{kk'}^j) = \frac{\Gamma(\alpha_{kk'}^j + \beta_{kk'}^j)}{\Gamma(\alpha_{kk'}^j)\Gamma(\beta_{kk'}^j)} \times (\theta_{kk'}^j)^{\alpha_{kk'}^j - 1} \times (1 - \theta_{kk'}^j)^{\beta_{kk'}^j - 1} \quad \text{(A.14)}$$

$$\log \mathrm{q}(z_i | \mathbf{p}_i) = \log \mathrm{Categorical}(z_i | \mathbf{p}_i) = \sum_{k=1}^{\mathcal{K}} \{ z_i k \times \log p_{ik} \} \quad \text{(A.15)}$$

$$\log \mathrm{q}(\pi | \mathbf{d}) = \log \mathrm{Dirichlet}(\pi | \mathbf{d}) = \sum_{k=1}^{\mathcal{K}} \{ (d_k - 1) \times \log \pi_k \} \quad \text{(A.16)}$$

Finally, we can write the gradients for each parameter to be estimated, where $\Psi(x) = \frac{d\Gamma(x)}{dx}$:

$$\nabla_{\alpha_{kk'}^j} \log \mathrm{q}(\theta_{kk'}^j | \alpha_{kk'}^j, \beta_{kk'}^j) = \log \theta_{kk'}^j + \Psi(\alpha_{kk'}^j + \beta_{kk'}^j) - \Psi(\alpha_{kk'}^j) \quad \text{(A.17)}$$

$$\nabla_{\beta^j_{kk'}} \log \mathrm{q}(\theta^j_{kk'}|\alpha^j_{kk'}, \beta^j_{kk'}) = \log\,(1-\theta^j_{kk'}) + \Psi(\alpha^j_{kk'}+\beta^j_{kk'}) - \Psi(\beta^j_{kk'}) \tag{A.18}$$

$$\nabla_{p_{ik}} \log \mathrm{q}(z_i|\mathbf{p}_i) = \frac{z_{ik}}{p_{ik}} \tag{A.19}$$

$$\nabla_{d_k} \log \mathrm{q}(\pi|\mathbf{d}) = \log d_k - \Psi(d_k) - \Psi(\sum_{k=1}^{\mathcal{K}} d_k) \tag{A.20}$$

A.2. Constrain Parameters

All the estimated parameters must be positive to remain in their distribution domain. In fact, each vector $\mathbf{p}_i$ and the vector $\mathbf{d}$ must sum one. We use the soft-plus function, and a normalized soft-plus function to deal with these constrains.