A Full Probabilistic Model for Yes/No Type Crowdsourcing in Multi-Class Classification - **Supplementary Material**

Belen Saldias-Fuentes*† Pavlos Protopapas‡ Karim Pichara B.*‡

1 Background Theory

This section describes the main theory behind this work. We based this discussion mainly on [36] and [40].

1.1 Probabilistic Graphical Models We represented the joint distribution of the proposed method with a probabilistic graphical model (PGM) [14, 29]. A PGM is a graph-based representation for compactly encoding a complex distribution over a high-dimensional space. For example, figure 1 illustrates the elemental DawidSkene [4] distribution for a crowdsourcing classification scenario. The circles represent random variables, observed variables are gray circles, and the points represent hyperparameters. When a set of variables shares the same probability distribution, we can use the "plate" notation, which stacks identical objects in a rectangle. In that case, the plates' dimensions are written in capital letters within the rectangles.

In the PGM shown in figure 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 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 a hyperparameter, $\mathbf{Z}$ and Θ will be conditionally dependent given all the labelers' votes $\mathbf{R}$ due to the graph structure. Following the notation from [17], in that model definition the variable distributions are:

$$z_i \sim \text{Multinomial}(\rho) \tag{1.1}$$

$$r_i^j \sim \text{Multinomial}(\Theta^j(z_i, :)) \tag{1.2}$$

This structure allows inferring a compact representation of the explicit joint distribution. To get the posterior distribution, we can either use sampling-based methods or variational inference. In this work, we address the proposed probabilistic model solution with approximate inference. In the following subsections, we explain two approaches to infer the posterior target distribution by approximating a distribution: Markov chain Monte Carlo (MCMC) and Variational Inference (VI).

DawidSkene

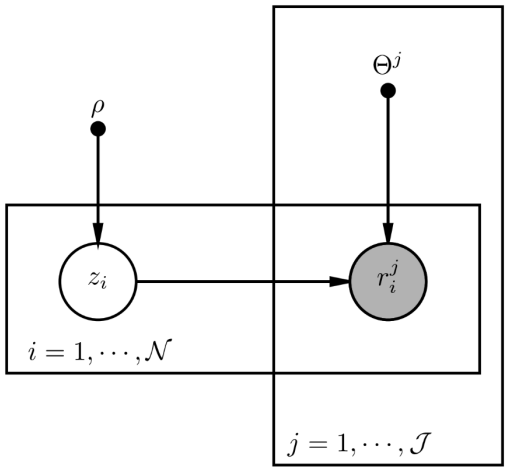

Figure 1: The DawidSkene model represented as a PGM in plate notation.

1.2 Markov Chain Monte Carlo MCMC [10, 7] 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 $\text{p}(z|x)$, 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 approximating $\text{p}(z|x)$.

MCMC approaches Bayesian inference with developments as the Gibbs sampler [9]. 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 rest of the

*Computer Science Department, Pontificia Universidad Católica de Chile, Santiago, Chile.

†MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA (present affiliation, belen@mit.edu).

‡Institute for Applied Computational Science, Harvard University, Cambridge, MA, USA.

distributions in the model. Suppose we want to infer $p(z_1, z_2)$. In each iteration, we would turn the samples iteratively: $z_1^{s+1} \sim p(z_1|z_2^s)$ and $z_2^{s+1} \sim p(z_2|z_1^s)$.

No-U-Turn Hamiltonian Monte Carlo (NUTS) To avoid the random walk and converge the sampling more quickly than with simple MCMC, we used NUTS [9], an MCMC algorithm based on a Hamiltonian Monte Carlo sampler (HMC). As an advantage, NUTS uses an informed walk and avoids the random walk by using a recursive algorithm to obtain a set of candidate points widely spread over the target distribution. Furthermore, NUTS stops when the recursion starts to go back to trace the dropped steps again. Nevertheless, HMC requires computing the gradient of the log-posterior to inform the walk, which can be difficult.

Using NUTS does not require establishing the step size and the number of steps to converge, compared to what a simple MCMC or HMC sampler does. Setting those parameters would require preliminary runs and some expertise. This sampling stops when drawing more samples no longer increases 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 draw exact samples from the target density [42]. Under these heavy computational settings, we can use variational inference (VI) as an approximation to the target distribution. VI does not guarantee finding the density distribution, it only finds a close distribution; however, it is usually faster than simple MCMC.

1.3 Variational Inference Variational inference (VI) [38] proposes a solution to the problem of posterior inference. VI selects an approximation $q(z)$ from some tractable family and then tries to make this $q(z)$ as close as possible to the true posterior $p^*(z) \triangleq p(z|x)$. The VI approach reduces this approximation to an optimization problem: the minimization of the KL divergence [39] from q to p^*.

The KL divergence is a measure of the dissimilarity of two probability distributions, p^* and q. Given that the forward KL divergence $\mathbb{KL}(p^*||q)$ includes taking expectations over the intractable $p^*(z)$, a natural alternative is the reverse KL divergence $\mathbb{KL}(q||p^*)$, defined in (1.3).

$$\mathbb{KL}(q||p^*) = -\int q(z) \log \frac{q(z)}{p^*(z)} dz \tag{1.3}$$

1.4 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 $\log p(x)$. The ELBO is equivalent to the negative KL divergence plus a constant, as we show in the following definitions.

Assume x is the observations, z the latent variables, and λ the free parameters of $q(z|\lambda)$. We want to approximate $p(z|x)$ by setting λ such that the KL divergence is minimum. In this case, we can rewrite (1.3) and expand the conditional in (1.4).

$$\mathbb{KL}(q||p^*) = \mathbb{E}_q[\log q(z|\lambda)] - \mathbb{E}_q[\log p(z, x)] - \log p(x) \tag{1.4}$$

Therefore, the minimization of the KL in (1.5) is equivalent to maximizing the ELBO:

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(z, x) - \log q(z|\lambda)] \tag{1.5}$$

1.5 Mean Field Inference Optimization over a given family of distributions is determined by the complexity of the family. This optimization can be difficult to optimize when a complex family is used. To keep the variational inference approach simple, [41] 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 is defined in (1.6).

$$q(z) = \prod_{i=1}^{m} q_i(z_i) \tag{1.6}$$

The goal is to solve the optimization in (1.7) over the parameters of each marginal distribution q.

$$\min_{\lambda_1,\ldots,\lambda_m} \mathbb{KL}(q||p^*) \tag{1.7}$$

1.6 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) [37] 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 updates the global parameters.

SVI solves the ELBO optimization by using the natural gradient [35] in a stochastic optimization algorithm. This optimization consists of estimating a noisy but cheap-to-compute gradient to reach the target distribution.

1.7 Black Box Variational Inference The BBVI [20] avoids any model-specific derivations. Black Box

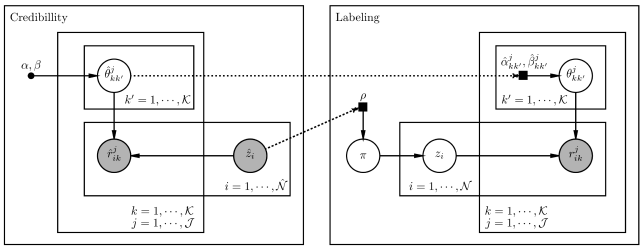

Figure 2: Proposed inference scheme for the YN model. Dashed lines represent prior parameters estimated in the `Credibility` stage. These parameters were then used as input to the `Labeling` stage.

VI proposes stochastically maximizing 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 (1.5) in (1.8).

(1.8)
$$\nabla_\lambda \mathcal{L} = \mathbb{E}_q[\nabla_\lambda \log\, q(z|\lambda)(\log\, p(z,x) - \log\, q(z|\lambda))]$$

Using this equation, we can compute the noisy unbiased gradient of the ELBO sampling the variational distribution with Monte Carlo, as shown in equation (1.9), where S is the number of samples we take from each distribution to be estimated.

(1.9)
$$\nabla_\lambda \mathcal{L} \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda \log\, q(z_s|\lambda)(\log\, p(z_s,x) - \log\, q(z_s|\lambda))$$

where,

(1.10)
$$z_s \sim q(z|\lambda)$$

For estimating the approximating q distribution, in BBVI the variational distributions $q(z_i)$ are mean field factors with free variational parameters λ_i, for each index i (see (1.6)). In appendix 4, we show how to apply this method to the proposed model.

2 Inference Schema

As stated in the paper, the proposed inference scheme works in two stages. The PGM in 2 shows both stages.

3 Complementary Results

3.1 Convergence Simulations - Synthetic Data.

To check for convergence of the full model, we analyzed each variable convergence. The convergence diagnostics for our random variables was based on the Gelman-Rubin statistic [7]. To try this diagnostic, we needed multiple chains to compare the similarity between them. Our experiments were 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 3 presents the potential scale reduction factors for all the estimated variables. According to this figure, there is no disagreement on whether each z_i converges.

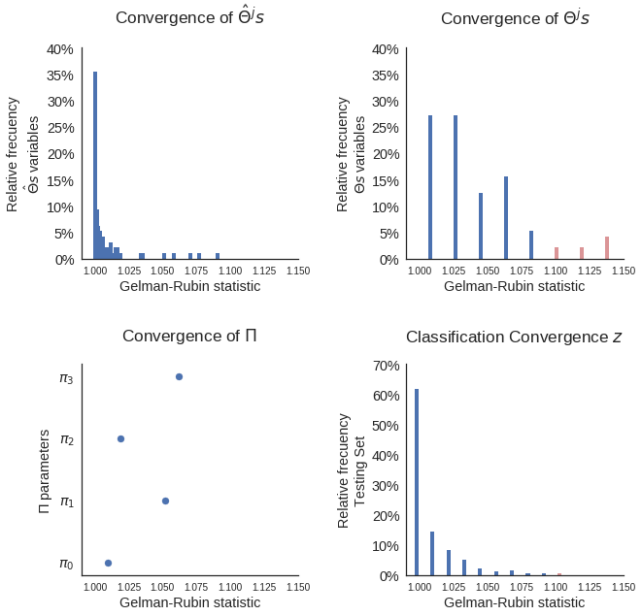

Figure 3: Accuracy score convergence. It is possible to see that Θ has more variance than any other variable, and some of the θs have not completely converged. We cannot conclude that the model has not converged, we can only say that one of the chains has not converged. In practice, that minimum percentage does not condition the full model.

3.2 Performance Simulations - MACHO Data.

Figure 4 shows the results for the experiment in subsection **7.6** in a five-class scenario. We can see that when all classes were asked per object per labeler, the YN model outperformed the ABCD strategy. However, three labelers are not enough for this scenario because the only way they reached the ABCDE performance (five classes implies ABCDE) was when we asked them about all five classes. In this five-class scenario, six labelers outperformed the ABCDE model when giving responses for only four classes. This means that the labelers were not required to discern among the five classes to reach a high accuracy score.

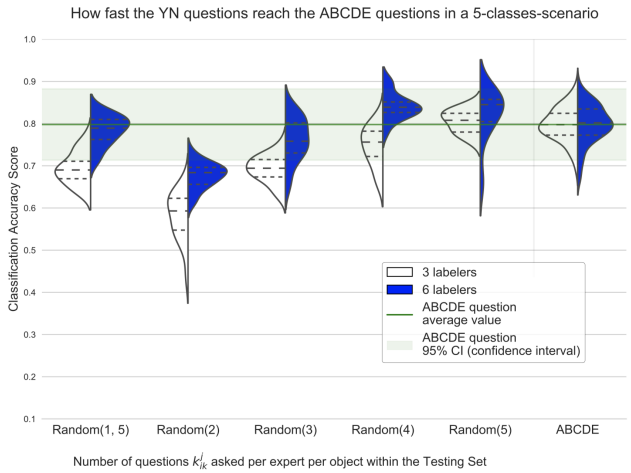

Figure 4: Classifiers voting for MACHO data. Random(w) means we asked each labeler for w different classes M_k a question k^j_{ik}, $w \leq \mathcal{K}$. The violin shape represents the cross validation results distribution.

3.3 Performance Real-World Votes MCMC vs. BBVI on Websites Results We developed all the previous simulations using the PyMC3 implementation mainly for two reasons. First, even though we used the AdaGrad [20] algorithm to set the learning rate, this setting presents more parameters tuning than the MCMC parametrization. Second, the results were usually slightly outperformed by NUTS.

Iterations Until Convergence As we said before, PyMC3 needs about 3000 iterations until convergence when running one chain. BBVI needs only 4 iterations, but each iteration implies estimating the gradient of each latent variable, which means taking samples from the variational approximation distribution of every variable. This estimation converges at 3072 total samples.

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 $\Omega(\mathcal{N} \times \mathcal{K})$. Both implementation require samples. The memory complexity for the PyMC3 model is $\mathcal{O}(\mathcal{N} \times \mathcal{K} \times NumOfSamplesMCMC)$ and for the BBVI is $\mathcal{O}(\mathcal{N} \times \mathcal{K} \times NumOfSamplesBBVI)$. When the number of samples remains constant, as in this work, the complexity is $\mathcal{O}(\mathcal{N} \times \mathcal{K})$. Both time complexities are equivalent.

4 Derivation Black Box Inference Equations.

The BBVI minimizes the KL divergence from an approximating distribution q to the true p posterior. Lets say x is 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. This optimization is equivalent to maximizing the ELBO in (1.5):

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

BBVI proposes stochastically maximizing the ELBO using noisy estimates of its gradient. The estimator of this gradient is computed using samples from the variational posterior. This require writing the gradient of the ELBO as in (1.8):

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

Using (1.8), we can compute the noisy unbiased gradient of the ELBO sampling the variational distribution with Monte Carlo, as shown in (1.9), where S is the number of samples taken from each distribution to be estimated:

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

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

Then λ is set at each iteration t as:

$$\lambda_t = \lambda_{t-1} + \rho \nabla_\lambda \mathcal{L}$$

Where the learning rate ρ can be fine-tuned as a global rate for all λs or as a unique rate per λ_s.

To estimate the approximating q distribution, BBVI uses the mean field theory. Then we define the approximating distribution q as in (1.6):

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

The variational mean field distributions q from (1.6) in the Credibility Estimation (first stage) of the YN model are found in (4.11). Their free variational parameters to estimate are in (4.12).

$$\mathrm{q}(\hat{\theta}^j_{kk'}) \sim \mathrm{Beta}(\hat{\alpha}^j_{kk'}, \hat{\beta}^j_{kk'}) \ \forall\ jkk' \tag{4.11}$$

$$\hat{\Theta}:\ (\hat{\alpha}^j_{kk'}, \hat{\beta}^j_{kk'}) \ \forall\ \hat{\theta}^j_{kk'} \tag{4.12}$$

For the Labeling part (second stage) of the proposed model, the mean field distributions q from (1.6) are defined in (4.13), (4.14), and (4.15).

$$\mathrm{q}(\theta^j_{kk'}) \sim \mathrm{Beta}(\alpha^j_{kk'}, \beta^j_{kk'}) \ \forall\ jkk' \tag{4.13}$$

$$\mathrm{q}(z_i) \sim \text{Categorical}(\mathbf{p}_i) \; \forall \; i \tag{4.14}$$

$$\mathrm{q}(\pi) \sim \text{Dirichlet}(\mathbf{d}) \tag{4.15}$$

Their free variational parameters to estimate are defined in (4.16), (4.17), and (4.18) respectively.

$$\Theta: \; (\alpha^j_{kk'}, \beta^j_{kk'}) \; \forall \; \theta^j_{kk'} \tag{4.16}$$

$$\mathbf{z}: \; (p_{ik} \; \forall k \in K) \; \forall \; z_i \tag{4.17}$$

$$\pi: \; (d_k \; \forall k \in K) \tag{4.18}$$

As shown in (1.8), to maximize the ELBO, we need the expectations under q. Given that we prefer to avoid the derivation for the YN model joint distribution, we used the black box method by approximating the gradient of the ELBO as defined in (1.9).

To apply this method to our model, we needed to write the needed functions for both the Credibility stage and the Labeling stage. 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).

4.1 Labeling Parameters Estimation The joint distribution to be inferred is:

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

$$\log \mathrm{p}(\theta^j_{kk'}|\alpha_0, \beta_0) = \log \text{Beta}(\theta^j_{kk'}|\alpha_0, \beta_0) \tag{4.19}$$

$$\log \mathrm{p}(z_i|\theta^j_{kk'}, \pi) = \log \text{Categorical}(z_i|\pi) \tag{4.20}$$

$$\log \mathrm{p}(\pi|\rho) = \log \text{Dirichlet}(\pi|\rho) \tag{4.21}$$

$$\log \mathrm{p}(r^j_{ik}|z_i, \theta^j_{z_ik}) = \log \left\{ (\theta^j_{z_ik})^{r^j_{ik}[0]} \times (1 - \theta^j_{z_ik})^{r^j_{ik}[1]} \right\} \tag{4.22}$$

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

$$\log \mathrm{q}(\theta^j_{kk'}|\alpha^j_{kk'}, \beta^j_{kk'}) = \log \text{Beta}(\theta^j_{kk'}|\alpha^j_{kk'}, \beta^j_{kk'}) = \frac{\Gamma(\alpha^j_{kk'} + \beta^j_{kk'})}{\Gamma(\alpha^j_{kk'})\Gamma(\beta^j_{kk'})} \times (\theta^j_{kk'})^{\alpha^j_{kk'}-1} \times (1 - \theta^j_{kk'})^{\beta^j_{kk'}-1} \tag{4.23}$$

$$\log \mathrm{q}(z_i|\mathbf{p}_i) = \log \text{Categorical}(z_i|\mathbf{p}_i) = \sum_{k=1}^{\mathcal{K}} \{z_i k \times \log p_{ik}\} \tag{4.24}$$

$$\log \mathrm{q}(\pi|\mathbf{d}) = \log \text{Dirichlet}(\pi|\mathbf{d}) = \sum_{k=1}^{\mathcal{K}} \{(d_k - 1) \times \log \pi_k\} \tag{4.25}$$

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

$$\nabla_{\alpha^j_{kk'}} \log \mathrm{q}(\theta^j_{kk'}|\alpha^j_{kk'}, \beta^j_{kk'}) = \log \theta^j_{kk'} + \Psi(\alpha^j_{kk'} + \beta^j_{kk'}) - \Psi(\alpha^j_{kk'}) \tag{4.26}$$

$$\nabla_{\beta^j_{kk'}} \log \mathrm{q}(\theta^j_{kk'}|\alpha^j_{kk'}, \beta^j_{kk'}) = \log (1 - \theta^j_{kk'}) + \Psi(\alpha^j_{kk'} + \beta^j_{kk'}) - \Psi(\beta^j_{kk'}) \tag{4.27}$$

$$\nabla_{p_{ik}} \log \mathrm{q}(z_i|\mathbf{p}_i) = \frac{z_{ik}}{p_{ik}} \tag{4.28}$$

$$\nabla_{d_k} \log \mathrm{q}(\pi|\mathbf{d}) = \log d_k - \Psi(d_k) - \Psi(\sum_{k=1}^{\mathcal{K}} d_k) \tag{4.29}$$

4.2 Constrained 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 used the soft-plus function and a normalized soft-plus function to deal with these constraints.

References

[35] Amari SI (1998) Natural gradient works efficiently in learning. Neural computation 10(2):251–276

[36] Blei DM, Kucukelbir A, McAuliffe JD (2017) Variational inference: A review for statisticians. Journal of the American Statistical Association (just-accepted)

[37] Hoffman MD, Blei DM, Wang C, Paisley J (2013) Stochastic variational inference. The Journal of Machine Learning Research 14(1):1303–1347

[38] Jordan MI, Ghahramani Z, Jaakkola TS, Saul LK (1999) An introduction to variational methods for graphical models. Machine learning 37(2):183–233

[39] Kullback S, Leibler RA (1951) On information and sufficiency. The annals of mathematical statistics 22(1):79–86

[40] Murphy KP (2012) Machine learning: a probabilistic perspective. MIT press

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

[42] Robert CP (2004) Monte carlo methods. Wiley Online Library