This document contains derivations and other supplemental information for the NIPS 2012 submission, “Variational Inference for Crowdsourcing”.

A Derivation of the Belief Propagation Algorithm

A.1 Sum-product Belief Propagation

We derive the belief propagation algorithm (15) in Theorem 3.1.

Theorem 3.1.

Let $\hat{x}_i = \log \frac{b_i(+1)}{b_i(-1)}, \quad x_{i\to j} = \log \frac{m_{i\to j}(+1)}{m_{i\to j}(-1)}, \quad$ *and* $\quad y_{j\to i} = L_{ij} \log \frac{m_{j\to i}(+1)}{m_{i\to j}(-1)}.$

Then, sum-product BP (5)-(7) can be expressed as

$$x_{i\to j}^{t+1} = \sum_{j'\in\mathcal{M}_{i\setminus j}} L_{ij} y_{j'\to i}^{t}, \qquad\qquad y_{j\to i}^{t+1} = \log \frac{\sum_{k=0}^{\gamma_j-1} \psi(k+1,\gamma_j) e_k^{t+1}}{\sum_{k=0}^{\gamma_j-1} \psi(k,\gamma_j) e_k^{t+1}}, \tag{1}$$

and $\hat{x}_i^{t+1} = \sum_{j\in\mathcal{M}_i} L_{ij} y_{i\to j}^{t+1}$*, where the terms* e_k *for* $k = 0,\dots,N_j - 1$*, are the elementary symmetric polynomials in variables* $\{\exp(L_{i'j} x_{i'\to j})\}_{i'\in\mathcal{N}_{j\setminus i}}$*, that is,* $e_k = \sum_{s:\,|s|=k} \prod_{i'\in s} \exp(L_{i'j} x_{i'\to j})$*. In the end, the true labels are decoded as* $\hat{z}_i^t = \mathrm{sign}[\hat{x}_i^t]$*.*

Proof. First, by update (5), we have

$$x_{i\to j}^{t+1} = \log \frac{m_{j\to i}^{t}(+1)}{m_{j\to i}^{t}(-1)} = \log \frac{\prod_{j'\to\mathcal{M}_{i\setminus j}} m_{j\to i}^{t}(+1)}{\prod_{j'\to\mathcal{M}_{i\setminus j}} m_{j\to i}^{t}(-1)} = \sum_{j'\in\mathcal{M}_{i\setminus j}} L_{ij} y_{i\to j'}^{t}.$$

Similar derivation applies to update (7). We just need to consider the update (6) in the following.

For a given z, we define $\mathcal{N}_{j\setminus i}^{+}[z] = \{i' \in \mathcal{N}_{j\setminus i} \colon z_{i'} = L_{i'j}\}$. Let $\mathcal{A}_k := \{z_{\mathcal{N}_{j\setminus i}} \colon |\mathcal{N}_{j\setminus i}^{+}[z]| = k\}$. By update (6) we have,

$$m_{j\to i}^{t+1}(+L_{ij}) = \sum_{z_{\mathcal{N}_{j\setminus i}}} \psi_j(z_{\mathcal{N}_j}) \prod_{i'\in\mathcal{N}_{j\setminus i}} m_{i'\to j}^{t+1}(z_i') \tag{2}$$

$$= \sum_{k=0}^{\gamma_j-1} \psi_j(k+1,\gamma_j) \sum_{z\in\mathcal{A}_k} \prod_{i'\in\mathcal{N}_{j\setminus i}} m_{i'\to j}^{t+1}(z_i') \tag{3}$$

$$= C \sum_{k=0}^{\gamma_j-1} \psi_j(k+1,\gamma_j) \sum_{z\in\mathcal{A}_k} \prod_{i'\in\mathcal{N}_{j\setminus i}} \frac{m_{i'\to j}^{t+1}(z_i')}{m_{i'\to j}^{t+1}(-L_{i'j})} \tag{4}$$

$$= C \sum_{k=0}^{\gamma_j-1} \psi_j(k+1,\gamma_j) \sum_{z\in\mathcal{A}_k} \prod_{i'\in\mathcal{N}_{j\setminus i}^{+}[z]} \frac{m_{i'\to j}^{t+1}(L_{i'j})}{m_{i'\to j}^{t+1}(-L_{i'j})} \tag{5}$$

$$= C \sum_{k=0}^{\gamma_j-1} \psi_j(k+1,\gamma_j) \sum_{z\in\mathcal{A}_k} \prod_{i'\in\mathcal{N}_{j\setminus i}^{+}[z]} \exp(x_{i'\to j}^{t+1}) \tag{6}$$

$$= C \sum_{k=0}^{\gamma_j-1} \psi_j(k+1,\gamma_j) e_k, \tag{7}$$

where C is a constant, $C = (\prod_{i'\in\mathcal{N}_{j\setminus i}} m_{i'\to j}^{t+1}(-L_{i'j}))$. Similarly, one can show that

$$m_{j\to i}^{t+1}(-L_{ij}) = C \sum_{k=0}^{\gamma_j-1} \psi_j(k,\gamma_j) e_k. \tag{8}$$

Therefore, we have

$$y_{j\to i}^{t+1} = \log \frac{m_{j\to i}^{t+1}(+L_{ij})}{m_{j\to i}^{t+1}(-L_{ij})} = \log \frac{\sum_{k=0}^{\gamma_j-1} \psi_j(k+1,\gamma_j)e_k}{\sum_{k=0}^{\gamma_j-1} \psi_j(k,\gamma_j)e_k}.$$

The proof is completed. □

It remains a problem to calculate the elementary symmetric polynomials e_k. Here we present a divide and conquer algorithm with a running time of $O(\gamma_j(\log\gamma_j)^2)$. Note that e_k is the k-th coefficient of polynomial $\prod_{i=0}^{\gamma_j-1}(x+e^{x_i})$, where $e^{x_i} = \exp(x_{i\to j})$. We divide the polynomial into a product of two polynomials,

$$\prod_{i=0}^{\gamma_j-1}(x+e^{x_i}) = \Big\{\prod_{i=0}^{\lceil\frac{\gamma_j}{2}\rceil}(x+e^{x_i})\Big\}\cdot\Big\{\prod_{i=\lceil\frac{\gamma_j}{2}\rceil}^{\gamma_j-1}(x+e^{x_i})\Big\}.$$

Since the merging step requires a polynomial multiplication, which is solved by fast Fourier transformation with $O(\gamma_j\log\gamma_j)$, by the master theorem, we get a total cost of $O(\gamma_j(\log\gamma_j)^2)$.

A more straightforward algorithm can be derived via dynamic programming, but with a higher cost of $O(\gamma_j^2)$. Let $e(k,n)$ be the k-th symmetric polynomial of the first n numbers of $\{x_i^e\}_{i'\in\mathcal{N}_{j\setminus i}}$; one can calculate e_k through the recursive formula $e(k,n) = e(k,n-1)e^{x_n} + e(k,n)$.

A.2 Max-product Belief Propagation

Similarly to the sum-product BP that we focus on in the main text, one can derive an efficient max-product belief propagation to find the joint maximum *a posterior* configuration, $\hat{z} = \arg\max_z p(z|L,\theta)$, which minimizes the block-wise error rate $\mathrm{prob}[\exists i : z_i \neq \hat{z}_i]$ instead of the bit-wise error rate $\frac{1}{N}\sum_{i\in[N]}\mathrm{prob}[z_i\neq\hat{z}_i]$. The max-product belief propagation update, in its general form, is

From tasks to workers:
$$m_{i\to j}^{t+1}(z_i) \propto \prod_{j'\in\mathcal{M}_{i/j}} m_{j'\to i}^{t}(z_i), \tag{9}$$

From workers to taskers:
$$m_{j\to i}^{t+1}(z_i) \propto \max_{z_{\mathcal{N}_{j/i}}}\Big\{\psi_j(z_{\mathcal{N}_j})\prod_{i'\in\mathcal{N}_j} m_{i'\to j}^{t+1}(z_{i'})\Big\}, \tag{10}$$

Calculating the beliefs:
$$b_i^{t+1}(z_i) \propto \prod_{j\in\mathcal{M}_i} m_{j\to i}^{t+1}(z_i). \tag{11}$$

Similarly to Theorem 3.1, max-product BP can be performed efficiently by exploiting the special structure of the high order potential $\psi(z_{\mathcal{N}_j})$.

Theorem A.1.

Let $\hat{x}_i = \log\frac{b_i(+1)}{b_i(-1)}$, $x_{i\to j} = \log\frac{m_{i\to j}(+1)}{m_{i\to j}(-1)}$, *and* $y_{j\to i} = L_{ij}\log\frac{m_{j\to i}(+1)}{m_{i\to j}(-1)}$.

Then max-product BP (9)-(11) *can be rewritten as*

$$x_{i\to j}^{t+1} = \sum_{j'\in\mathcal{M}_{i\setminus j}} L_{ij}y_{i\to j'}^{t}, \qquad y_{j\to i}^{t+1} = \log\frac{\max_{0\le k\le\gamma_j-1}\{\psi_j(k+1,\gamma_j)v_k^{t+1}\}}{\max_{0\le k\le\gamma_j-1}\{\psi_j(k,\gamma_j)v_k^{t+1}\}}, \tag{12}$$

and $\hat{x}_i^{t+1} = \sum_{j\in\mathcal{M}_i} L_{ij}y_{i\to j}^{t+1}$, *where* $v_k = \exp(\sum_{n=0}^{k} x_{[n]})$ *and* $x_{[n]}$ *is the n-th largest number in* $\{L_{i'j}x_{i'\to j}\}_{i'\in\mathcal{N}_{j\setminus i}}$. *In the end, the true labels are decoded as* $\hat{z}_i^t = \mathrm{sign}[\hat{x}_i^t]$.

The main cost of (12) is for sorting $\{L_{i'j}x_{i'\to j}\}_{i'\in\mathcal{N}_{j\setminus i}}$, requiring a running time of $O(\gamma_j\log\gamma_j)$; this is slightly faster than sum-product BP, which requires $O(\gamma_j(\log\gamma_j)^2)$. See Tarlow et al. [2010] for a similar derivation and more general treatment of structured high-order potentials.

B Extensions of Belief Propagation

Compared to KOS, our BP algorithm is derived using a principled Bayesian paradigm, and hence can be easily extended to more general models and cases beyond the assumptions made in the paper. In this section, we show in detail how to extend the BP algorithm to work on the two-coin worker-error model, to estimate the hyperparameters of the algorithmic priors, and to incorporate additional task features and other side information.

B.1 Extending to the Two-Coin Model

In the paper, we initially assumed that the sensitivities equals the specificities, i.e., $q_j = s_j = t_j$. Here we extend the BP algorithm to the more general case when s_j and t_j are defined separately. Assume s_j and t_j independently follow two beta priors, i.e., $p(s_j|\theta) \propto s_j^{\alpha_s - 1}(1-s_j)^{\beta_s - 1}$ and $p(t_j|\theta) \propto t_j^{\alpha_t - 1}(1-t_j)^{\beta_t - 1}$. The $\psi_j(z_{\mathcal{N}_j})$ defined in (4) becomes

$$\psi_j(z_{\mathcal{N}_j}) = \int p(s_j|\theta)p(t_j|\theta)\prod_{i\in\mathcal{N}_j} p(L_{ij}|s_j,t_j)ds_j dt_j = B(c_{11}+\alpha_s, c_{21}+\beta_s)B(c_{22}+\alpha_t, c_{12}+\beta_t),$$

where $c_{11} = \sum_{i\in\mathcal{N}_j}\mathbb{I}[L_{ij}=1, z_i=1]$, $c_{21} = \sum_{i\in\mathcal{N}_j}\mathbb{I}[L_{ij}=-1, z_i=1]$, $c_{22} = \sum_{i\in\mathcal{N}_j}\mathbb{I}[L_{ij}=-1, z_i=-1]$, $c_{12} = \sum_{i\in\mathcal{N}_j}\mathbb{I}[L_{ij}=1, z_i=-1]$. In addition, let $\gamma_j^+ = \sum_{i\in\mathcal{N}_j}\mathbb{I}[L_{ij}=1]$ and $\gamma_j^- = \sum_{i\in\mathcal{N}_j}\mathbb{I}[L_{ij}=-1]$. Note that $c_{21} = \gamma_j^- - c_{11}$, $c_{12} = \gamma_j^+ - c_{22}$, we have

$$\psi_j(z_{\mathcal{N}_j}) = B(c_{11}+\alpha_s, \gamma_j^- - c_{22}+\beta_s)B(c_{22}+\alpha_t, \gamma_j^+ - c_{11}+\beta_t) \overset{def}{=} \psi_j(c_{11}, c_{22}),$$

where we rewrite $\psi_j(z_{\mathcal{N}_j})$ as $\psi(c_{11}, c_{22})$ (with a slight abuse of notation), because $\psi_j(z_{\mathcal{N}_j})$ depends on $z_{\mathcal{N}_j}$ only through c_{11} and c_{22}, the true positive and true negative counts. Similar to Theorem 3.1, one can show that belief propagation (6) can be reduced to

$$y_{j\to i} = \begin{cases} \log[\sum_{k_1=0}^{\gamma_{j\backslash i}^+}\sum_{k_2=0}^{\gamma_{j\backslash i}^-}\psi(k_1+1,k_2)e_{k_1}^+e_{k_2}^-] - \log[\sum_{k_1=0}^{\gamma_{j/i}^+}\sum_{k_2=0}^{\gamma_{j/i}^-}\psi(k_1,k_2)e_{k_1}^+e_{k_2}^-], & \text{if } L_{ij}=+1 \\ \log[\sum_{k_1=0}^{\gamma_{j\backslash i}^+}\sum_{k_2=0}^{\gamma_{j\backslash i}^-}\psi(k_1,k_2+1)e_{k_1}^+e_{k_2}^-] - \log[\sum_{k_1=0}^{\gamma_{j/i}^+}\sum_{k_2=0}^{\gamma_{j/i}^-}\psi(k_1,k_2)e_{k_1}^+e_{k_2}^-], & \text{if } L_{ij}=-1 \end{cases}$$

where $\gamma_{j/i}^+ = \sum_{i'\in\mathcal{N}_{j\backslash i}}\mathbb{I}[L_{i'j}=1]$ and $\gamma_{j/i}^- = \sum_{i'\in\mathcal{N}_{j\backslash i}}\mathbb{I}[L_{i'j}=-1]$; and $\{e_k^+\}$ and $\{e_k^-\}$ are the symmetric polynomials of the $\{\exp(L_{i'j}x_{i'\to j})\colon i'\in\mathcal{N}_{j\backslash i}, L_{i'j}=1\}$ and $\{\exp(L_{i'j}x_{i'\to j})\colon i'\in\mathcal{N}_{j\backslash i}, L_{i'j}=-1\}$, respectively. Each step of the above update requires a running time of $O(\gamma_j^-\gamma_j^+ + \gamma_j^+(\log\gamma^+)^2 + \gamma_j^-(\log\gamma^-)^2)$. The update for $x_{i\to j}$ and the decoding step remain the same as in Theroem 3.1.

B.2 Learning the hyper-parameters via EM

The optimal choice of the algorithmic prior $p(q_j|\theta)$ in BP (15) should match the true data prior. One can adopt an empirical Bayesian approach to estimate the hyper-parameters θ from the data. Here we present an EM algorithm for estimating the hyper-parameters θ, that alternates between performing the belief propagation (15) (E-step) and adjusting the parameters via maximizing the expected marginal likelihood (M-step).

The EM algorithm, in its general form, is

$$\text{E-step: } Q(\theta|\theta^{old}) = \mathbb{E}_z[\log p(z|L,\theta)|\theta^{old}], \qquad \text{M-step: } \theta^{new} = \arg\max_\theta Q(\theta|\theta^{old}).$$

The E-step in our case is performed by running the belief propagation algorithm. First, we approximate the posterior distribution $p(z_{\mathcal{N}_j}|L,\theta^{old})$ with belief $b_j^{old}(z_{\mathcal{N}_j})$ on the factor nodes, defined by

$$b_j^{old}(z_{\mathcal{N}_j}) \propto \psi(z_{\mathcal{N}_j})\prod_{i\in\mathcal{N}_j} m_{i\to j}^{old}(z_i),$$

where $m^{old}_{i\to j}$ are the messages of belief propagation when $\theta = \theta^{old}$. The E-step becomes

$$Q(\theta|\theta^t) = \mathbb{E}_z[\log p(z|L,\theta)|\theta^t] = \sum_j \sum_{z_{\mathcal{N}_j}} b_j^{old}(z_{\mathcal{N}_j}) \log \psi_j(z_{\mathcal{N}_j}|L,\theta). \tag{13}$$

Similar to Theorem 3.1, one can calculate (13) in terms of the elementary symmetric polynomials e_k; one can show that

$$Q(\theta|\theta^t) = \sum_j \sum_{k=0}^{\gamma_j} \tilde{y}_k^{old} \log \psi_j(k,\gamma_j|L,\theta), \tag{14}$$

where $\tilde{y}_k^{old} = \psi_j(k,\gamma_j|L,\theta^{old}) e_k^{old}$, where e_k^{old} are the elementary symmetric polynomials of $\{\exp(L_{ij} x^{old}_{i\to j}) \colon i \in \mathcal{N}_j\}$.

The M-step in our case can be efficiently solved using standard numerical methods. For example, when the algorithmic priors of q_j is $\mathrm{Beta}(\alpha,\beta)$ where $\theta = [\alpha,\beta]$, one can show that $Q(\theta|\theta^{old})$ equals (up to a constant) the log-likelihood of Beta-binomial distribution, and can be efficiently maximized using standard numerical methods.

The E-step above takes a *soft* combination of the posterior evidence. An alternative is to use *hard*-EM, which replaces the E-step with

$$\text{E-step (hard-EM): } Q(\theta|\theta^{old}) = \log p(\hat{z}^{old}|L,\theta),$$

where $\hat{z}^{old}$ are the estimated labels via belief propagation on $\theta = \theta^{old}$. The hard-EM form is very intuitive; it iteratively estimates the labels with belief propagation, and fits the hyper-parameters imputed with the labels found by the last estimation.

Note that this form of EM (in the outer loop) for estimating the hyper-parameters is different from EM (in the inner loop) of Dawid and Skene [1979], Smyth et al. [1995], Raykar et al. [2010], which maximizes q_j with fixed hyper-parameter θ; it is closer to the SpEM of Raykar and Yu [2012], which also estimates a hyper-parameter with q_j marginalized, but uses a different EM and Laplacian approximation in the inner loop.

B.3 Incorporating Task Features

In some cases the tasks are associated with known features that provide additional information about the true labels, and the problem is formulated as a supervised learning task with crowdsourced (redundant but noisy) labels [Raykar et al., 2010]. Our method can be easily extended to these cases by representing the task features as singleton potentials on the variables nodes in the factor graph, that is, the posterior distribution (4) is modified to

$$p(z|F,L,\theta,\omega) = \prod_i p(z_i|f_i;\omega) \prod_j \psi(z_{\mathcal{N}_j}),$$

where $F = \{f_j \colon j \in [N]\}$ are the features of the tasks, and ω are the regression coefficients. Our belief propagation works here with only minor modification. The regression coefficient ω, together with the hyper-parameter θ, can be estimated using the EM algorithm we discussed above.

B.4 Incorporating Partially Known Ground Truth

In case the ground truth labels of some tasks are known, these labels can help the prediction of the other tasks via a "wave effect", propagating information about the reliabilities of their associated workers. Our algorithm can also be easily extended to this case.

Specifically, assume the ground truth labels of a subset of tasks $G \in [N]$ are known, e.g., $z_G = z_G^0$. Let $\hat{\alpha}_j$ be the number of tasks in G that worker j labels correctly. To predict the remaining labels, one can simply modify the BP algorithm (15) into

$$x^{t+1}_{i\to j} = \sum_{j' \in \mathcal{M}_{i \setminus j}} L_{ij} y^t_{i\to j'}, \qquad y_{j\to i} = \log \frac{\sum_{k=0}^{\gamma_j - 1} \psi_j(k + \hat{\alpha}_j + 1, \gamma_j) e_k}{\sum_{k=0}^{\gamma_j - 1} \psi_j(k + \hat{\alpha}_j, \gamma_j) e_k}, \tag{15}$$

where the messages are passed only between the workers and the tasks with unknown labels. Intuitively, the known ground truth provides scores (in term of $\hat{\alpha}_j$) of the workers who have labeled them, which are used as "prior" information for predicting the remaining labels.