Adaptive Crowdsourcing via EM with Prior

Peter Maginnis and Tanmay Gupta

May 4, 2015

In this work, we make two primary contributions: derivation of the EM update for the shifted and rescaled beta prior and the development of an algorithm to adaptively select new edges (task assignments) for more efficient use of an edge budget (i.e. reduced probability of task labeling error for a fixed number of edges). The algorithm exploits several observable features of a task (namely the entropy of its label probabilities, the number of workers already assigned to it, and a histogram of estimated worker abilities for its neighbors) to produce a ranking of tasks most likely to be currently mislabeled. An additional worker (sampled uniformly) is then assigned to each of the highest ranking tasks and the graph is updated. Rankings are produced by a random forest classifier that is given the current graph structure and prior information, and trained on ensembles of randomly generated data.

First, we derive the EM updates for the crowdsourcing problem both without and with knowledge of the transformed beta prior. Next we outline the adaptive algorithm used to assign edges in graph construction using the random forest classifier. Finally, we present numerical results demonstrating the effectiveness of these techniques over a range of edge budgets.

1 Expectation-Maximization for Crowdsourcing

EM without prior

We treat $\mathbf{A}$ as our observed variables, $\mathbf{t}$ as latent variables and $\mathbf{p}$ as parameters. Then the joint distribution will be given as:

$$\begin{aligned}\mathbb{P}(\mathbf{A},\mathbf{t}|\mathbf{p}) &= \mathbb{P}(\mathbf{A}|\mathbf{t},\mathbf{p})\mathbb{P}(\mathbf{t}|\mathbf{p})\\ &= \mathbb{P}(\mathbf{A}|\mathbf{t},\mathbf{p})\mathbb{P}(\mathbf{t}) && \text{(Since true labels are independent of worker abilities)}\\ &= \prod_{(i,j)\in E}\mathbb{P}(A_{ij}|t_i,p_j)\prod_{i}^{n}\mathbb{P}(t_i)\end{aligned}$$

where,

$$\mathbb{P}(A_{ij}|t_i,p_j) = p_j\mathbb{I}(A_{ij}=t_i) + (1-p_j)\mathbb{I}(A_{ij}=-t_i)$$
$$\mathbb{P}(t_i) = \frac{3}{4}\mathbb{I}(t_i=1) + \frac{1}{4}\mathbb{I}(t_i=-1)$$

E-Step:

In the **E** step we find the distribution of the latent variables as a function of the observed variables and parameters estimated in the previous iteration.

$$\mathbb{P}(\mathbf{t}|\mathbf{A},\mathbf{p}^{\text{old}}) = \prod_{i=1}^{n}\mathbb{P}(t_i|\mathbf{A},\mathbf{p}^{\text{old}})$$

where,

$$\begin{aligned}
\mathbb{P}(t_i|\mathbf{A},\mathbf{p}^{\text{old}}) &= \mathbb{P}(t_i|A_{i\partial i}, p^{\text{old}}_{\partial i}) \\
&= \frac{\mathbb{P}(A_{i\partial i}, t_i|p^{\text{old}}_{\partial i})}{\mathbb{P}(A_{i\partial i}|p^{\text{old}}_{\partial i})} \\
&= \frac{\mathbb{P}(t_i)\prod_{j\in\partial i}\mathbb{P}(A_{ij}|t_i, p^{\text{old}}_j)}{\sum_{t_i}\mathbb{P}(t_i)\prod_{j\in\partial i}\mathbb{P}(A_{ij}|t_i, p^{\text{old}}_j)} \\
&= \gamma_i(t_i)
\end{aligned}$$

In the sequel, allowing for a slight abuse of notation, we need only refer to

$$\begin{aligned}
\gamma_i := \gamma_i(1) &= \mathbb{P}(t_i = 1|\mathbf{A},\mathbf{p}^{\text{old}}) \\
&= \frac{\mathbb{P}(A_{i\partial i}|t_i = 1,\mathbf{p})\mathbb{P}(t_i = 1)}{\mathbb{P}(A_{i\partial i}|t_i = 1,\mathbf{p})\mathbb{P}(t_i = 1) + \mathbb{P}(A_{i\partial i}|t_i = -1,\mathbf{p})\mathbb{P}(t_i = -1)} \\
&= \frac{\frac{3}{4}\prod_{j\in\partial i} p_j^{\mathbb{I}(A_{ij}=1)}(1-p_j)^{\mathbb{I}(A_{ij}=-1)}}{\frac{3}{4}\prod_{j\in\partial i} p_j^{\mathbb{I}(A_{ij}=1)}(1-p_j)^{\mathbb{I}(A_{ij}=-1)} + \frac{1}{4}\prod_{j\in\partial i} p_j^{\mathbb{I}(A_{ij}=-1)}(1-p_j)^{\mathbb{I}(A_{ij}=1)}} \\
&= \frac{\frac{3}{4}\prod_{j\in\partial i}\left[\frac{1}{2} + \frac{2p_j-1}{2}A_{ij}\right]}{\frac{3}{4}\prod_{j\in\partial i}\left[\frac{1}{2} + \frac{2p_j-1}{2}A_{ij}\right] + \frac{1}{4}\prod_{j\in\partial i}\left[\frac{1}{2} - \frac{2p_j-1}{2}A_{ij}\right]}
\end{aligned}$$

M-Step:

In this we approximate the log likelihood of the observed data using the expected log likelihood of the observed and latent variables together where the expectation is with respect to the distribution of the latent variables computed in the **E** step

$$\begin{aligned}
Q(\mathbf{p}|\mathbf{p}^{\text{old}}) &= \mathbb{E}_\gamma\left[\log\mathbb{P}(\mathbf{A},\mathbf{t}|\mathbf{p})\right] \\
&= \mathbb{E}_{\mathbf{t}|\mathbf{A},\mathbf{p}^{\text{old}}}\left[\log\mathbb{P}(\mathbf{A},\mathbf{t}|\mathbf{p})\right] \\
&= \sum_{\mathbf{t}\in\{-1,1\}^n}\mathbb{P}(\mathbf{t}|\mathbf{A},\mathbf{p}^{\text{old}})\log\mathbb{P}(\mathbf{A},\mathbf{t}|\mathbf{p}) \\
&= \sum_{\mathbf{t}\in\{-1,1\}^n}\left[\left(\prod_{i=1}^n\mathbb{P}(t_i|\mathbf{A},\mathbf{p}^{\text{old}})\right)\sum_{i=1}^n\left(\log\mathbb{P}(t_i|\mathbf{p}) + \sum_{j\in\partial i}\log\mathbb{P}(A_{ij}|t_i,p_j)\right)\right] \\
&= \sum_{i=1}^n\sum_{t_i}\gamma_i(t_i)\left[\log\mathbb{P}(t_i|\mathbf{p}) + \sum_{j\in\partial i}\log\mathbb{P}(A_{ij}|t_i,p_j)\right] \\
&= \sum_{i=1}^n\Bigg\{\gamma_i\left[\log\mathbb{P}(t_i = 1) + \sum_{j\in\partial i}\log\mathbb{P}(A_{ij}|t_i = 1,p_j)\right] \\
&\qquad + (1-\gamma_i)\left[\log\mathbb{P}(t_i = -1) + \sum_{j\in\partial i}\log\mathbb{P}(A_{ij}|t_i = -1,p_j)\right]\Bigg\}
\end{aligned}$$

Now to maximize $Q(\mathbf{p}|\mathbf{p}^{\text{old}})$ with respect to $\mathbf{p}$ we set the derivatives to zero.

$$\begin{aligned}
&\frac{\partial Q}{\partial p_j} = 0 \\
\implies &\sum_{i\in\partial j}\left[\gamma_i\left(\frac{\mathbb{I}(A_{ij}=1)}{p_j} - \frac{\mathbb{I}(A_{ij}=-1)}{1-p_j}\right) + (1-\gamma_i)\left(\frac{\mathbb{I}(A_{ij}=-1)}{p_j} - \frac{\mathbb{I}(A_{ij}=1)}{1-p_j}\right)\right] = 0 \\
\implies &\sum_{i\in\partial j}\left[(\gamma_i - p_j)\,\mathbb{I}(A_{ij}=1) + (1-\gamma_i-p_j)\,\mathbb{I}(A_{ij}=-1)\right] = 0 \\
\implies &p_j = \frac{1}{|\partial j|}\sum_{i\in\partial j}\left[\gamma_i\mathbb{I}(A_{ij}=1) + (1-\gamma_i)\,\mathbb{I}(A_{ij}=-1)\right]
\end{aligned}$$

EM with Beta Prior

Instead, we seek to maximize the objective

$$\begin{aligned}
Q(\mathbf{p}|\mathbf{p}^{\text{old}}) &= \mathbb{E}_\gamma\left[\ln\mathbb{P}(\mathbf{A}|\mathbf{p}) + \ln\mathbb{P}(\mathbf{p})\right] \\
&= \sum_{i=1}^n\left[\gamma_i\ln\frac{3}{4}a_i + (1-\gamma_i)\ln\frac{1}{4}b_i\right] + \sum_{j=1}^m \ln f\left(\frac{p_j-0.1}{0.9}\right),
\end{aligned}$$

where

$$\begin{aligned}
a_i &:= \mathbb{P}(A_{i\partial i}|t_i=1,\mathbf{p}) = \prod_{j\in\partial i} p_j^{\mathbb{I}(A_{ij}=1)}(1-p_j)^{\mathbb{I}(A_{ij}=-1)} \\
b_i &:= \mathbb{P}(A_{i\partial i}|t_i=-1,\mathbf{p}) = \prod_{j\in\partial i} p_j^{\mathbb{I}(A_{ij}=-1)}(1-p_j)^{\mathbb{I}(A_{ij}=1)},
\end{aligned}$$

and

$$f(z) = \frac{1}{B(\alpha,\beta)}(z)^{\alpha-1}(1-z)^{\beta-1}$$

We take the gradient

$$\begin{aligned}
\frac{\partial Q}{\partial p_j} &= \sum_{i\in\partial j}\left\{\gamma_i\left[\frac{\mathbb{I}(A_{ij}=1)}{p_j} - \frac{\mathbb{I}(A_{ij}=-1)}{1-p_j}\right] + (1-\gamma_i)\left[\frac{\mathbb{I}(A_{ij}=-1)}{p_j} - \frac{\mathbb{I}(A_{ij}=1)}{1-p_j}\right]\right\} + \frac{\alpha-1}{p_j-0.1} - \frac{\beta-1}{1-p_j} \\
&= \sum_{i\in\partial j}\left\{\frac{\gamma_i\mathbb{I}(A_{ij}=1) + (1-\gamma_i)(\mathbb{I}(A_{ij}=-1))}{p_j} - \frac{(1-\gamma_i)\mathbb{I}(A_{ij}=1) + \gamma_i(\mathbb{I}(A_{ij}=-1))}{1-p_j}\right\} + \frac{\alpha-1}{p_j-0.1} - \frac{\beta-1}{1-p_j} \\
&= \sum_{i\in\partial j}\left\{\frac{\gamma_iA_{ij} - A_{ij} + 1}{p_j} - \frac{A_{ij} - \gamma_iA_{ij} + 1}{1-p_j}\right\} + \frac{\alpha-1}{p_j-0.1} - \frac{\beta-1}{1-p_j}.
\end{aligned}$$

We then solve for the critical point

$$\frac{\partial Q}{\partial p_j} = 0$$

which implies

$$0 = (p_j - 0.1)\sum_{i\in\partial j}\left\{\gamma_i\mathbb{I}(A_{ij}=1) + (1-\gamma_i)\mathbb{I}(A_{ij}=-1) - p_j\right\} + (\alpha-1)p_j(1-p_j) - (\beta-1)p_j(p_j-0.1).$$

Define

$$\lambda_j := \frac{1}{|\partial j|} \sum_{i \in \partial j} \left[\gamma_i \mathbb{I}(A_{ij} = 1) + (1 - \gamma_i) \mathbb{I}(A_{ij} = -1) \right].$$

Then, we may more compactly write the quadratic equation

$$0 = (p_j - 0.1)|\partial j|(\lambda_j - p_j) + (\alpha - 1)p_j(1 - p_j) - (\beta - 1)p_j(p_j - 0.1).$$

Numerical results indicate that the smaller root lies outside the support, $[0.1, 1]$, of the worker ability distribution, and the larger root is always taken for the EM update.

Note that if, instead, we model p_j as $\text{Beta}(\alpha, \beta)$, then the above analysis recovers the update found in [1,2]. Specifically, the critical equation becomes

$$\begin{aligned} 0 &= |\partial j|(\lambda_j - p_j) + (\alpha - 1)(1 - p_j) - (\beta - 1)p_j \\ \implies \lambda_j|\partial j| + \alpha - 1 &= (|\partial j| + \alpha + \beta - 2)p_j \\ \implies p_j &= \frac{\sum_{i \in \partial j} \left[\gamma_i \mathbb{I}(A_{ij} = 1) + (1 - \gamma_i) \mathbb{I}(A_{ij} = -1) \right] + \alpha - 1}{|\partial j| + \alpha + \beta - 2} \end{aligned}$$

2 Adaptive Graph Construction

We adapt to the current state of the EM estimate and graph by training a random forest classifier to estimate which label assignments are most likely incorrect. Given the current graph, the algorithm for adaptive edge assignment proceeds as follows. At iteration ℓ, we are given a current graph $G^{(\ell)}$ and an EM estimate $(\gamma^{(\ell)}, \hat{\mathbf{p}}^{(\ell)})$. The current graph $G^{(\ell)}$ is passed to a random forest generation algorithm that has access to the $\mathbf{t}$ and $\mathbf{p}$ prior distributions. The random forest generator then runs EM on ensembles of i.i.d. randomly generated instances of $(\mathbf{A}^{(\ell)}, \mathbf{t}^{(\ell)}, \mathbf{p}^{(\ell)})_e$ to produce task label probabilies $\hat{\gamma}_e^{(\ell)}$ and worker ability maximum likelihood estimates $\hat{\mathbf{p}}_e^{(\ell)}$ and trains a random forest classifier using entropy of label probability estimates $h(\hat{\gamma}_e^{(\ell)})$, number of workers assigned to a task $|\partial i^{(\ell)}|$, and a histogram of $\{\hat{p}_{\partial i}^{(\ell)}\}$ as features and incorrectness of task assignments as the true labels. The resulting random forest classifier is then returned to the main algorithm, which uses the random forest and EM estimates of the current graph to rank the actual tasks by likelihood of mislabeling. The K highest risk tasks $\{i_k\}_{k=1}^K$ are then assigned one new worker chosen uniformly from the population of workers who are not already assigned to that task. These edges are then added to $G^{(\ell)}$ to produce $G^{(\ell+1)}$.

2.1 Random Forest

Random forest is an ensemble machine learning algorithm that has wide applicability in classification and regression tasks. The basic unit of a random forest is a decision tree. Each node in the tree represents a binary decision on a feature and every path from the root to a leaf represents a logical conjunction of the binary decisions encountered along the path. Learning a decision tree from data $\mathcal{D} = \{(\mathbf{x}_i, l_i) | i = 1, \cdots, M\}$ comprising of M <features,label> pairs involves choosing a feature and a threshold that forms the splitting criterion for each node and assigning each $\mathbf{x}_i$ to one of the leaves, L_i in the tree where $L_i \in \{1, \cdots, N\}$ if there are N leaves in the tree. During inference a new feature vector $\mathbf{v}$ follows a path from the root to some leaf L in the tree and gets a class posterior distribution given by

$$P(l|\mathbf{v}) = \frac{\sum_{i=1}^M \mathbb{I}(L_i = L)\mathbb{I}(l_i = l)}{\sum_{i=1}^M \mathbb{I}(L_i = L)}$$

The splitting criterion is chosen using some notion of information gain such as decrease in entropy after splitting. Let $\text{left}(i)$ and $\text{right}(i)$ denote the left and right children of node i and $p_i(l)$ be the fraction of

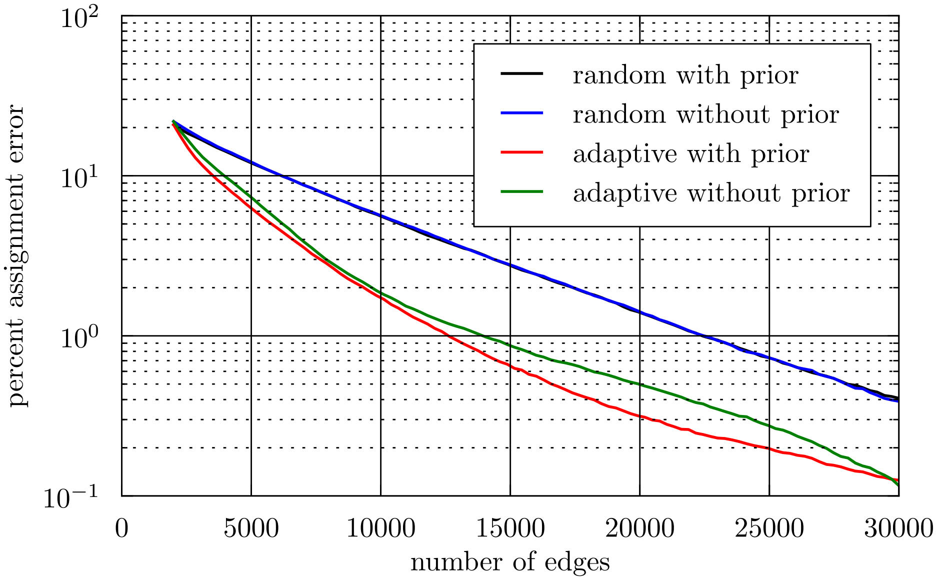

Figure 1: Mean percent classification error for the 4 modes of estimation: random graph construction both with and without transformed beta prior and adaptive graph construction using EM with and without the transformed beta prior. Note that in the random graph generation cases, the results of EM both with and without prior are almost identical. However, when graph construction is done adaptively, and thus depends on the output of EM, we see a marked improvement by including worker ability prior data.

training data points that reached node i which had label l. Then the information gain for a particular splitting criterion is evaluated as follows:

$$\Delta = \sum_{j=1}^{C} p_i(l_j) \log p_i(l_j) - \left[\sum_{j=1}^{C} p_{\text{left}(i)}(l_j) \log p_{\text{left}(i)}(l_j) - \sum_{j=1}^{C} p_{\text{right}(i)}(l_j) \log p_{\text{right}(i)}(l_j) \right]$$

The feature and threshold used for splitting at a node is the one that maximizes Δ.

3 Numerical Results

To examine the performance of our two contributions over a range of edge budget values, we consider a bipartite graph initialization $G^{(0)}$ that assigns one worker to every task and at least one task to each of $m < n$ workers, using the fewest edges possible (2000). As a baseline for comparison, we consider a technique where at each step ℓ, we add edges by sampling uniformly from the tasks $i \in [n]$ and then uniformly selecting a worker not already assigned to that task, $j \in [m] \setminus \partial i$. We then proceed by adding edges (either randomly or adaptively) and computing the EM estimates periodically.

References

[1] Q. Liu, J. Peng, and A. T. Ihler. “Variational Inference for Crowdsourcing”. *Advances in Neural Information Processing Systems*, 2012.

[2] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. “Learning from crowds”. *The Journal of Machine Learning Research*, 11:1297-1322, 2010.

[3] L. Breiman. “Random Forests”. *Machine Learning*, vol. 45(1):5-32, 2001.