Exploration of Questioning Strategies for Unsupervised Labeling

Neil Chainani, Christian Junge, Abhishek Malali

AM207, Harvard University

Abstract

In this project, we contrast two different questioning schemes to assess efficacy in recovering true class labels from noisy data provided by crowdsourced non-experts, when there are more than two classes to choose from. We generate data with a confusion matrix for each worker, and an underlying class distribution, and attempt to recover both parameters and the true labels. We use Expectation Maximization (EM), Simulated Annealing, and PyMC to compare efficiency and verify results.

1. Introduction

Having reliable labeled data is crucial for any supervised learning task. But given large datasets, manually determining these labels may be intractable. Services like Mechanical Turk can be particularly useful to crowdsource labeling, but it is difficult to guarantee that the workers will reputably provide correct labels, so often the study will aggregate labeling on a particular item from multiple workers. In addition, each worker will have their own subjectivity when issuing labels, and it is important to capture this bias somehow in the model.

The task becomes even more difficult with more classes from which to choose. In the case of labeling flowers, for example, an amateur labeler may not be well versed enough to differentiate between families of orchids, when faced with over a hundred options. There also may be ambiguity between classes. Take the classic example of lions, tigers, and ligers. To the layperson, an image of a liger may fall closer to a tiger on the lion-tiger spectrum, and thus be hard to classify.

To address these concerns, we compare two strategies for structuring a labeling question. J workers are given N items to label, each of which belongs to one of K classes. The first strategy, the multiclass paradigm (MC), is to ask each worker a single question for each item, where the worker has to select just one option from the K classes provided. This is a commonly employed technique in Mechanical Turk and other crowdsourcing platforms. The second strategy, which we refer to as the yes-no paradigm (YN), is to pose K questions to each worker for each item, where the worker has to answer "Is this item of class K?", and he or she must answer yes or no. The worker has the ability to answer yes to more than one question, or no to all K classes. We hypothesized that YN would perform better than MC, because it allows the worker to focus on a single class at a time.

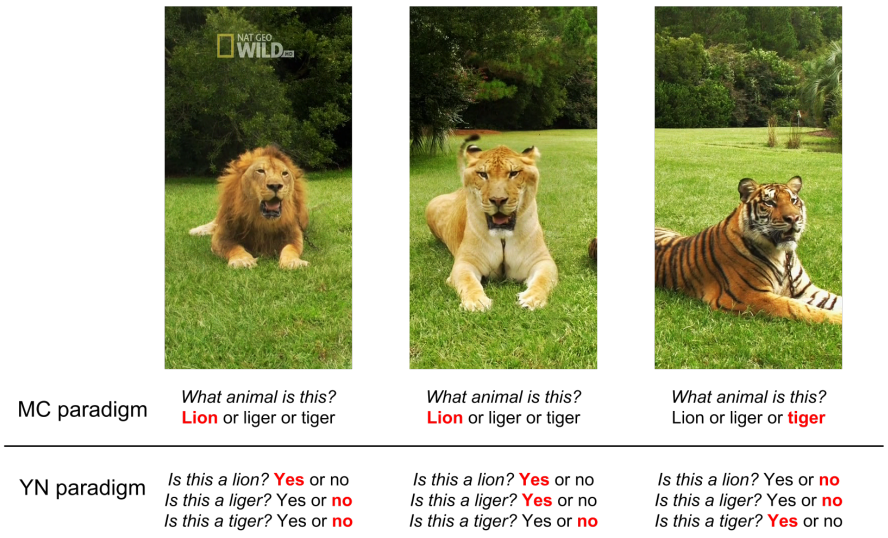
NAT GEO WILD
MC paradigm
What animal is this? Lion or liger or tiger
What animal is this? Lion or liger or tiger
What animal is this? Lion or liger or tiger
YN paradigm
Is this a lion? Yes or no
Is this a liger? Yes or no
Is this a tiger? Yes or no
Is this a lion? Yes or no
Is this a liger? Yes or no
Is this a tiger? Yes or no
Is this a lion? Yes or no
Is this a liger? Yes or no
Is this a tiger? Yes or no

Figure 1: Imagine a worker is faced with these three images. They may classify the images of tigers and lions correctly, regardless of the questioning strategy. However, we postulate that their answer with the YN paradigm will allow us to more accurately recover the true class of the liger.

2. Data Generation

We began with a confusion matrix $\Theta^{(j)}$ per worker, where $j \in \{1, 2, ..., J\}$. $\Theta^{(j)}$ is a $K \times K$ matrix, where $\Theta^{(j)}[k_1, k_2]$ is the probability of predicting class k_2, when the true class is k_1. To generate each $\Theta^{(j)}$, we took an identity matrix of size K and added a bias term over the entire matrix, and then normalized over the rows. This puts the most weight along the diagonal, which is intuitive because we would expect that a worker predicts the correct class most often. We also generated true classes, using a categorical distribution. We determined the true label y_n on item n by pulling class k with probability λ_k.

2.1. *MC paradigm data generation*

From the confusion matrices and true labels, we generated the data as a $N \times J \times K$ size matrix. $r_{njk} = 1$ when a worker j predicts class k on item n, and $r_{njk} = 0$ otherwise for that same worker on the same item.

2.2. *YN paradigm data generation*

In the Yes-No case, our data matrix size becomes $N \times J \times K \times 2$. Now, for each worker j on item n, $r_{njk0} = 1, r_{njk1} = 0$ if they answered no on the question regarding class k, and $r_{njk0} = 0, r_{njk1} = 1$ if they answered yes. We have two binary values for each question rather than a single one (where 0 is no and 1 is yes) because in the event that a worker does not answer a question on an item, it will not affect the likelihood, because both r_{njk0} and r_{njk1} will be 0.

3. Approach

For both strategies, we attempt EM and SA to converge on our true labels, confusion matrices, and class distributions. We then benchmark against PyMC. Regardless of the method, we were required to calculate the complete data log likelihood.

3.1. Likelihood

3.1.1. MC Paradigm

For the likelihood, we will need a prior π to specify the class distribution. The r_{nj} is the label itself (rather than r_{njk} being a binary variable). For our a single item belonging to a class k, the likelihood is simply:

$$p(\{r_{njk}\}_{j=1}^{J} | \{\Theta^{(j)}\}_{j=1}^{J}, \pi_k) = \pi_k \prod_{j=1}^{J} \Theta^{(j)}[k, r_{nj}]$$

Then, to scale this up for the entire data-set,

$$\prod_{n=1}^{N} \prod_{k=1}^{K} p(\{r_{njk}\}_{j=1}^{J} | \{\Theta^{(j)}\}_{j=1}^{J}, \pi_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k \prod_{j=1}^{J} \Theta^{(j)}[k, r_{nj}]$$

We take the log of our complete data likelihood to get the log-likelihood:

$$L(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \Big(\log \pi_k + \sum_{j=1}^{J} \log \Theta^{(j)}[k, r_{nj}] \Big)$$

3.1.2. YN Paradigm

The YN likelihood is a little bit trickier. For a single item, we need to consider each question for an item as being drawn from a Bernoulli distribution. The probability of a single class on a particular item will require a product over all of the questions:

$$p(\{r_{njk0}, r_{njk1}\}_{j=1}^{J} | \{\Theta^{(j)}\}_{j=1}^{J}, \pi_k) = \pi_k \prod_{j=1}^{J} \prod_{k'=1}^{K} \Theta^{(j)}[k, k']^{r_{njk'0}} (1 - \Theta^{(j)}[k, k'])^{r_{njk'1}}$$

Again, we scale this up for the whole data set:

$$\prod_{n=1}^{N} \prod_{k=1}^{K} p(\{r_{njk0}, r_{njk1}\}_{j=1}^{J} | \{\Theta^{(j)}\}_{j=1}^{J}, \pi_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k \prod_{j=1}^{J} \prod_{k'=1}^{K} \Theta^{(j)}[k, k']^{r_{njk'1}} (1 - \Theta^{(j)}[k, k'])^{r_{njk'0}}$$

The log likelihood of the above term becomes:

$$L(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \Bigg(\log \pi_k + \sum_{j=1}^{J} \sum_{k'=1}^{K} \Big(r_{njk'1} \log \Theta^{(j)}[k, k'] + r_{njk'0} \log(1 - \Theta^{(j)}[k, k']) \Big) \Bigg)$$

3.2. Expectation Maximization

3.2.1. MC Paradigm

We need to find the expectation of the log likelihood:

$$E[L(\theta)] = \sum_{n=1}^{N}\sum_{k=1}^{K} Z_{nk}\Big(\log \pi_k + \sum_{j=1}^{J} \log \Theta^{(j)}[k, r_{nj}]\Big)$$

The E-step (from the Dawid-Skene model) is:

$$Z_{nk} = \frac{\pi_k \prod_{j=1}^{J} \Theta^{(j)}[k, r_{nj}]}{\sum_{k'=1}^{K} \pi_{k'} \prod_{j=1}^{J} \Theta^{(j)}[k', r_{nj}]}$$

For the M-step, we calculate the estimates for each element in the confusion matrix, as well as π:

$$\hat{\Theta}^{(j)}[k, k'] = \frac{\sum_{n=1}^{N} Z_{nk}\mathbf{1}(r_{nj} = k')}{\sum_{k''=1}^{K}\sum_{n=1}^{N} Z_{nk}\mathbf{1}(r_{nj} = k'')}$$

$$\hat{\pi}_k = \frac{\sum_{n=1}^{N} Z_{nk}}{\sum_{k=1}^{K}\sum_{n=1}^{N} Z_{nk}}$$

The indicator function for the $\Theta^{(j)}$ update selects only the items which belong to class k'. We iterate between E and M until convergence.

3.2.2. YN Paradigm

Again, we begin with the ECDLL (expected complete-data log likelihood):

$$E[L(\theta)] = \sum_{n=1}^{N}\sum_{k=1}^{K} Z_{nk}\Big(\log \pi_k + \sum_{j=1}^{J}\sum_{k'=1}^{K}\Big(r_{njk'1}\log \Theta^{(j)}[k, k'] + r_{njk'0}\log(1 - \Theta^{(j)}[k, k'])\Big)\Big)$$

For the E-step, we can similarly find the posterior to be:

$$Z_{nk} = \frac{\pi_k \prod_{j=1}^{J}\prod_{k'=1}^{K} \Theta^{(j)}[k, k']^{r_{njk'0}}(1 - \Theta^{(j)}[k, k'])^{r_{njk'1}}}{\sum_{k''=1}^{K} \pi_{k''} \prod_{j=1}^{J}\prod_{k'=1}^{K} \Theta^{(j)}[k'', k']^{r_{njk'0}}(1 - \Theta^{(j)}[k'', k'])^{r_{njk'1}}}$$

In the M-step, it is easy to see that the predicted class distribution $\hat{\pi}_k$ will be identical to that from the MC case; however, the estimated confusion matrix is not as straight-forward:

$$\frac{\partial E[L(\theta)]}{\partial \Theta^{(j)}[k, k']}\sum_{n=1}^{N}\sum_{k=1}^{K} Z_{nk}\Big(\log \pi_k + \sum_{j=1}^{J}\sum_{k'=1}^{K}\Big(r_{njk'1}\log \Theta^{(j)}[k, k'] + r_{njk'0}\log(1 - \Theta^{(j)}[k, k'])\Big)\Big) = 0$$

Each term in the confusion matrix fixes the j, k, and k', but not n. We can thus simplify this expression:

$$\frac{\partial E[L(\theta)]}{\partial \Theta^{(j)}[k, k']}\sum_{n=1}^{N} Z_{nk}\Big(r_{njk'1}\log \Theta^{(j)}[k, k'] + r_{njk'0}\log(1 - \Theta^{(j)}[k, k'])\Big) = 0$$

Then, computing the derivative:

$$\sum_{n=1}^{N} Z_{nk}\Big(\frac{r_{njk'1}}{\Theta^{(j)}[k,k']} - \frac{r_{njk'0}}{1-\Theta^{(j)}[k,k']}\Big) = 0$$

$$\sum_{n=1}^{N} Z_{nk}\Big(\frac{r_{njk'1} - r_{njk'1}\Theta^{(j)}[k,k'] - r_{njk'0}\Theta^{(j)}[k,k']}{\Theta^{(j)}[k,k'](1-\Theta^{(j)}[k,k'])}\Big) = 0$$

$$\sum_{n=1}^{N} Z_{nk}(r_{njk'1}) = \Theta^{(j)}[k,k'] \sum_{n=1}^{N} Z_{nk}(r_{njk'1} + r_{njk'0})$$

$$\hat{\Theta}^{(j)}[k,k'] = \frac{\sum_{n=1}^{N} Z_{nk}(r_{njk'1})}{\sum_{n=1}^{N} Z_{nk}(r_{njk'1} + r_{njk'0})}$$

3.3. Simulated Annealing

For SA, our states are represented by our estimate of the labels. We initially randomly assign labels to each item without worrying about the class distribution. Using these random assignments and the data, we compute the initial confusion matrix. In a Simulated Annealing step, we assign a random new label to a random item, and then use this new set of assignments to compute the confusion matrix. This provides us with all the variables to compute the new likelihood. Using the standard SA procedure, we can now optimize parameters like the re-annealing schedule and temperature to converge to the right solution. Since we are effectively optimizing on the assignments as well as the confusion matrices, simulated annealing has a tough time finding the global optimum. To solve this problem, the initial assignments to the SA can be inferred by taking the mode class over all workers.

4. Results

We used two data sets: an easy data set and a hard data set. Both had 5 workers and 4 classes, but were generated with different confusion matrices. The easy data were created with confusion matrices close to an identity matrix, representing workers that are very accurate. In contrast, the confusion matrices used to generate the hard data set were much more muddled and varied vastly between workers, representing workers with varying accuracy. We compared overall classification accuracy (the percentage of predicted labels that match the true labels) for 40, 80, 100, 200, and 500 data points, on both the easy and hard data. We then ran the data through SA and EM for both MC and YN.

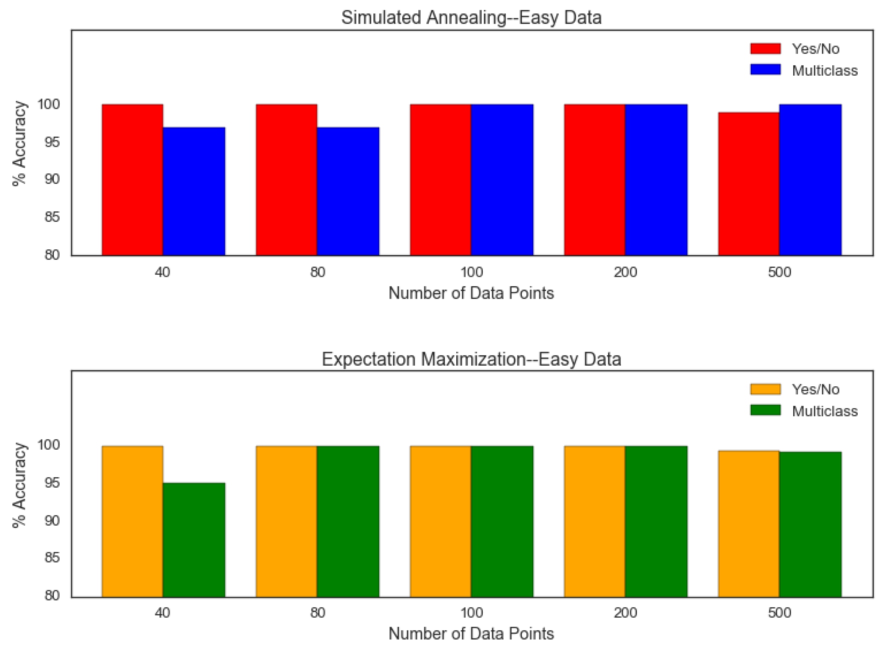

Figure 2: Comparing accuracy in Label Recovery for Yes/No vs Multiclass questioning on an easy data set

As we can observe in the plot, both paradigms are able to recover the true labels well, but YN has an edge over MC in almost every instance.

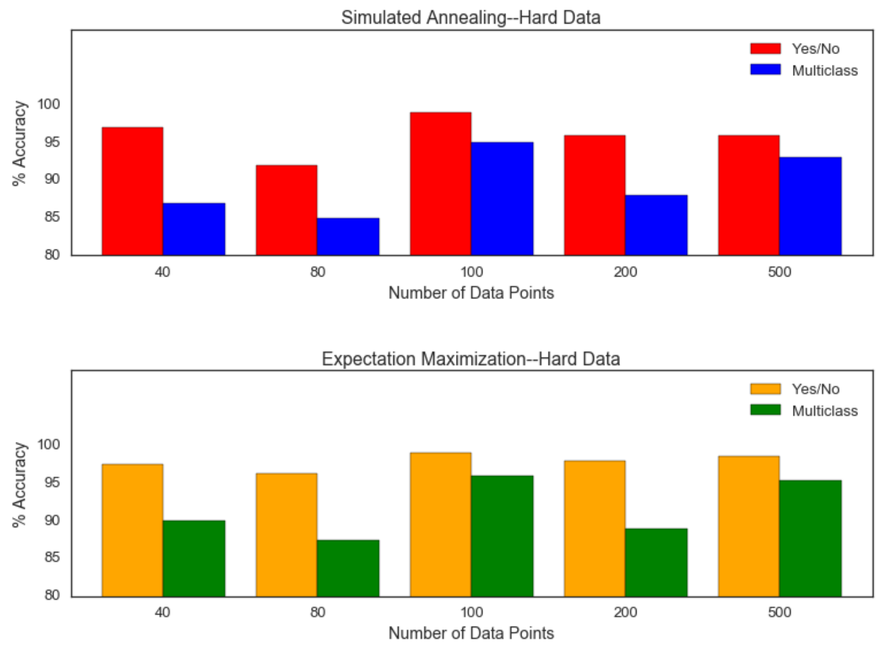

Figure 3: Comparing accuracy in Label Recovery using Expectation Maximization for easy and hard datasets.

The differences in accuracy between YN and MC are much more evident on the hard data, where our workers are less precise. Not only can YN consistently recover the true labels with almost 100% accuracy, but with EM it can also capture the confusion matrices for each worker. The implication of this is that we can pick out individual workers who are worse than others. As for differences in methods, EM is much faster than SA; it is able to converge in two iterations (a tenth of a second on average), versus the ten minutes it takes for SA to reach a solution. SA also requires tweaking of parameters, whereas EM is able to reach the optimal solution with just the formulas.

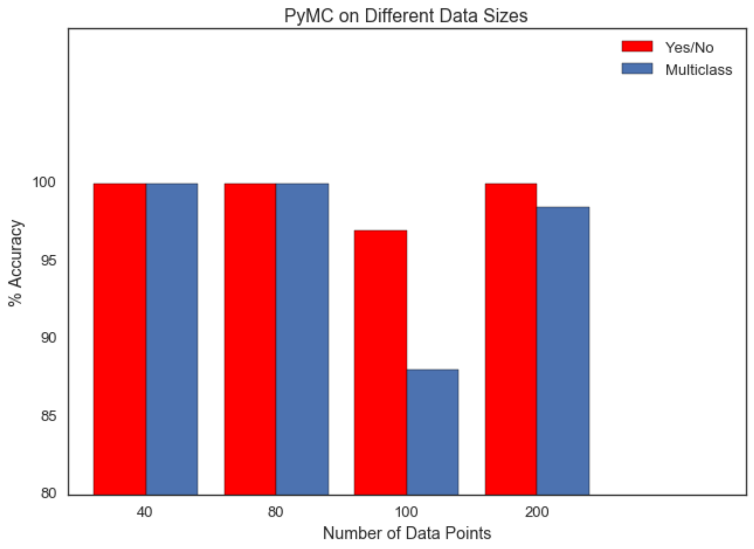

Figure 4: Comparing accuracy in Label Recovery using PyMC for both questioning paradigms.

In addition to the above methods, we validate the results by running the same problem in PyMC. PyMC corroborates our theory that YN is a better questioning strategy, and shows that provided the cost of asking questions can be kept low, the YN method proves effective in recovering the class labels correctly.

Online Resources

Github repo: http://github.com/abhishekmalali/questioning-strategy-classification
Video: https://www.youtube.com/watch?v=XEcamnmy0z8

Citations

[1] C.Liu and Y.M.Wang, TrueLabel + Confusions: A Spectrum of Probabilistic Models in Analyzing Multiple Ratings, Proceedings of the 29th International Conference on Machine Learning(ICML-12)

[2] A.P.Dawid and A.M.Skene, Maximum Likelihood Estimation of Observer Error-Rates using EM Algorithm, Journal of the Royal Statistical Society: Series C (Applied Statistics), Vol. 28, No. 1(1979), pp. 20-28