AAAI Technical Report FS-12-06
Machine Aggregation of Human Judgment

Selective Sampling of Labelers for Approximating the Crowd

Şeyda Ertekin
MIT Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142
seyda@mit.edu

Haym Hirsh
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
hirsh@cs.rutgers.edu

Cynthia Rudin
MIT Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142
rudin@mit.edu

Abstract

In this paper, we present CrowdSense, an algorithm for estimating the crowd's majority opinion by querying only a subset of it. CrowdSense works in an online fashion where examples come one at a time and it dynamically samples subsets of labelers based on an exploration/exploitation criterion. The algorithm produces a weighted combination of a subset of the labelers' votes that approximates the crowd's opinion. We also present two probabilistic variants of CrowdSense that are based on different assumptions on the joint probability distribution between the labelers' votes and the majority vote. Our experiments demonstrate that we can reliably approximate the entire crowd's vote by collecting opinions from a representative subset of the crowd.

Introduction

The problem of "approximating the crowd" is that of estimating the crowd's majority opinion by querying only a subset of it. Our goal is to determine the majority opinion of a crowd, on a series of questions ("examples"), with a limited budget. Because of the open nature of crowdsourcing systems, it is not necessarily easy to approximate the majority vote of a crowd on a budget by sampling a representative subset of the voters. For example, the crowd may be comprised of labelers with a range of capabilities, motives, knowledge, views, personalities, etc. Without any prior information about the characteristics of the labelers, a small sample of votes is not guaranteed to align with the true majority opinion. In order to effectively approximate the crowd, we need to determine who are the most representative members of the crowd, in that they can best represent the interests of the crowd majority. This is even more difficult to accomplish when items arrive over time, and it requires our budget to be used both for 1) estimating the majority vote, even before we understand the various qualities of each labeler, and 2) exploring the various labelers until we can estimate their qualities well. Estimating the majority vote in particular can be expensive before the labelers' qualities are known, and we do not want to pay for additional votes that are not likely to impact the decision.

In order to make economical use of our budget, we could determine when just enough votes have been gathered to confidently align our decision with the crowd majority. The budget limitation necessitates a compromise: if we pay for a lot of votes per decision, our estimates will closely align with the crowd majority, but we will only make a smaller number of decisions, whereas if we pay for less votes per decision, the accuracy may suffer, but more decisions are made. There is clearly an exploration/exploitation tradeoff: before we can exploit by using mainly the best labelers, we need to explore to determine who these labelers are, based on their agreement with the crowd.

The main contributions of this work can be broken down into two main parts: In the first part, we propose a modular algorithm, *CrowdSense*, that approximates the wisdom of the crowd. In an online fashion, CrowdSense dynamically samples a subset of labelers, determines whether it has enough votes to make a decision, and requests more if the decision is sufficiently uncertain. CrowdSense keeps a balance between exploration and exploitation in its online iterations – exploitation in terms of seeking labels from the highest-rated labelers, and exploration so that enough data are obtained about each labeler to ensure that we learn each labeler's accuracy sufficiently well.

The second part presents probabilistic variations of CrowdSense. One of the main challenges to approximating the crowd has to do with the fact that the majority vote is taken as the ground truth (the truth we aim to predict). This means that there is a complicated relationship (a joint probability distribution) between the labelers' accuracies and the ground truth. Based on this observation, we introduce two variations of CrowdSense, called CrowdSense.Ind and CrowdSense.Bin, that make various distributional assumptions to handle distinct crowd characteristics. In particular, the first algorithm makes a statistical independence assumption of the probabilities for large crowds, whereas the second algorithm finds a lower bound on how often the current sub-crowd agrees with the crowd majority vote, using the binomial distribution. For both CrowdSense.Ind and CrowdSense.Bin, even though probabilistic assumptions were made, the accuracy is comparable to (or lower than) CrowdSense itself.

Throughout the paper, a "majority vote" refers to the simple, every day sense of voting wherein every vote is equal,

with no differential weighting of the votes. This is in contrast to a weighted majority vote, as we use in CrowdSense, wherein each labeler's vote is multiplied by the labeler's quality estimate. This weighting scheme ensures that the algorithm places a higher emphasis on the votes of higher quality labelers.

Related Work

The low cost of crowdsourcing labor has increasingly led to the use of resources such as Amazon Mechanical Turk[1] (AMT) to label data for machine learning purposes, where collecting multiple labels from non-expert annotators can yield results that rival those of experts. This cost-effective way of generating labeled collections using AMT has also been used in several studies (Snow et al. 2008; Kaisser and Lowe 2008; Bernstein et al. 2010).

(Dawid and Skene 1979) presented a methodology to estimate the error rates based on the results from multiple diagnostic tests without a gold standard using latent variable models. Smyth et al. (1994a; 1994b) used a similar approach to investigate the benefits of repeatedly labeling same data points via a probabilistic framework that models a learning scheme from uncertain labels. Although a range of approaches are being developed to manage the varying reliability of crowdsourced labor (Callison-Burch and Dredze 2010; Wallace et al. 2011; Law and von Ahn 2011), the most common method for labeling data via the crowd is to obtain multiple labels for each item from different labelers and treat the *majority label* as an item's true label. (Sheng, Provost, and Ipeirotis 2008), for example, demonstrated that repeated labeling can be preferable to single labeling in the presence of label noise, especially when the cost of data preprocessing is non-negligible. (Dekel and Shamir 2009a) proposed a methodology to identify low quality annotators for the purpose of limiting their impact on the final attribution of labels to examples. To that effect, their model identifies each labeler as being either good or bad, where good annotators assign labels based on the marginal distribution of the true label conditioned on the instance and bad annotators provide malicious answers. (Dekel and Shamir 2009b) proposed an algorithm for pruning the labels of less reliable labelers in order to improve the accuracy of the majority vote of labelers. First collecting labels from labelers and then discarding the lower quality ones presents a different viewpoint than our work, where we achieve the same "pruning effect" by estimating the qualities of the labelers and not asking the low quality ones to vote in the first place. A number of researchers have explored approaches for learning how much to trust different labelers, typically by comparing each labeler's predictions to the majority-vote prediction of the full set. These approaches often use methods to learn both labeler quality characteristics and latent variables representing the *ground-truth* labels of items that are available (Kasneci et al. 2011; Dekel, Gentile, and Sridharan 2010; Warfield, Zou, and Wells 2004) sometimes in tandem with learning values for other latent variables such as task difficulty (Whitehill et al. 2009; Welinder et al. 2010), classifier parameters (Yan et al. 2010a; 2010b; Raykar et al. 2010), or domain-specific information about the labeling task (Welinder et al. 2010).

[1] http://www.mturk.com

CrowdSense

Let us first model the labelers' quality estimates as a measure of their agreement with the crowd majority. These quality estimates indicate whether a labeler is "representative" of the crowd. Let $L = \{l_1, l_2, \ldots, l_M\}$, $l_k : \mathcal{X} \to \{-1, 1\}$ denote the set of labelers and $\{x_1, x_2, \ldots, x_t, \ldots, x_N\}$, $x_t \in \mathcal{X}$ denote the sequence of examples, which could arrive one at a time. We define $V_{it} := l_i(x_t)$ as l_i's vote on x_t and $S_t \subset \{1, \ldots, M\}$ as the set of labelers selected to label x_t. For each labeler l_i, we then define c_{it} as the number of times we have observed a label from l_i so far and a_{it} as how many of those labels were consistent with the other labelers:

$$c_{it} := \sum_{\tilde{t}=1}^{t} \mathbb{1}_{[i \in S_{\tilde{t}}]} \qquad a_{it} := \sum_{\tilde{t}=1}^{t} \mathbb{1}_{\left[i \in S_{\tilde{t}}, V_{i\tilde{t}} = V_{S_{\tilde{t}}\tilde{t}}\right]}$$

where $V_{S_t t} = \text{sign}\left(\sum_{i \in S_t} V_{it} Q_{it}\right)$ is the weighted majority vote of the labelers in S_t. Labeler l_i's quality estimate is then defined as

$$Q_{it} := \frac{a_{it} + K}{c_{it} + 2K} \tag{1}$$

where t is the number of examples that we have collected labels for and K is a smoothing parameter. Q_{it} is a Bayesian shrinkage estimate of the probability that labeler i will agree with the crowd, pulling values down toward $1/2$ when there are not enough data to get a more accurate estimate. This ensures that labelers who have seen fewer examples are not considered more valuable than labelers who have seen more examples and whose performance is more certain. CrowdSense uses a pessimistic estimate of the mean rather than an upper (e.g. 95%) confidence interval that IEThresh (Donmez et al., 2009) (and UCB algorithms) use.

At the beginning of an online iteration to label a new example, the labeler pool is initialized with three labelers; we select two "exploitation" labelers that have the highest quality estimates Q_{it} and select another one uniformly at random for "exploration". This initial pool of seed labelers enables the algorithm to maintain a balance between exploitation of quality estimates and exploration of the quality of the entire set of labelers. We ask each of these 3 labelers to vote on the example. Their votes for this example are then used to generate a *confidence score*, given as

$$\text{Score}(S_t) = \sum_{i \in S_t} V_{it} Q_{it}$$

which represents the weighted majority vote of the labelers. Next, we determine whether we are certain that the sign of $\text{Score}(S_t)$ reflects the crowd's majority vote, and if we are not sure, we repeatedly ask another labeler to vote on this example until we are sufficiently certain about the label. To measure how certain we are, we greedily see whether adding an additional labeler might make us uncertain about the decision. Specifically, we select the labeler with the highest quality estimate Q_{it}, who is not already in S_t, as a candidate

Algorithm 1 Pseudocode for CrowdSense.

1. **Input:** Examples $\{x_1, x_2, \ldots, x_N\}$, Labelers $\{l_1, l_2, \ldots, l_M\}$, confidence threshold ε, smoothing parameter K.
2. **Define:** $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates.
3. **Initialize:** $a_{i1} \leftarrow 0,\ c_{i1} \leftarrow 0$ for $i = 1, \ldots, M$. Initialize $Q_{it} \leftarrow 0$ for $i = 1 \ldots M, t = 1 \ldots N$
4. **For** $t = 1, ..., N$
   (a) Compute quality estimates $Q_{it} = \frac{a_{it}+K}{c_{it}+2K},\ i = 1, \ldots, M$. Update L_Q.
   (b) $S_t = \{l^{(1)}, l^{(2)}, l^{(k)}\}$, where k is chosen uniformly at random from the set $\{3, \ldots M\}$.
   (c) **For** $j = 3 \ldots M,\ j \neq k$
      i. $\text{Score}(S_t) = \sum_{i \in S_t} V_{it} Q_{it},\ \ l_{\text{candidate}} = l^{(j)}$.
      ii. If $\frac{|\text{Score}(S_t)| - Q_{l_{\text{candidate}},t}}{|S_t|+1} < \varepsilon$, then $S_t \leftarrow S_t \cup l_{\text{candidate}}$. Otherwise exit loop to stop adding new labelers to S_t.
   (d) Get the weighted majority vote of the labelers $V_{S_t t} = \text{sign}\left(\sum_{i \in S_t} V_{it} Q_{it}\right)$
   (e) $\forall i \in S_t$ where $V_{it} = V_{S_t t},\ \ a_{it} \leftarrow a_{it} + 1$
   (f) $\forall i \in S_t,\ \ c_{it} \leftarrow c_{it} + 1$
5. **End**

to label this example. We then check whether this labeler could potentially either change the weighted majority vote if his vote were included, or if his vote could bring us into the *regime of uncertainty* where the $\text{Score}(S_t)$ is close to zero, and the vote is approximately a tie. We check this before we pay for this labeler's vote, by temporarily assigning him a vote that is opposite to the current subcrowd's majority. The criteria for adding the candidate labeler to S_t is defined as:

$$\frac{|\text{Score}(S_t)| - Q_{l_{\text{candidate}},t}}{|S_t| + 1} < \varepsilon \qquad (2)$$

where ε controls the level of uncertainty we are willing to permit, $0 < \varepsilon \leq 1$. If (2) is true, the candidate labeler is added to S_t and we get (pay for) this labeler's vote for x_t. We then recompute $\text{Score}(S_t)$ and follow the same steps for the next-highest-quality candidate from the pool of unselected labelers. If the candidate labeler is not added to S_t, we are done adding labelers for example t, and assign the weighted majority vote as the predicted label of this example. We then proceed to label the next example in the collection. Pseudocode for CrowdSense is given in Algorithm 1.

Datasets and Baselines

We conducted experiments on four datasets that model a crowd with different characteristics. In particular, the first dataset is a real world collection of movie ratings from humans, the second dataset is the decisions of software agents on prediction tasks, and the remaining two datasets are the classification outputs of supervised machine learning algorithms.

MovieLens is a movie recommendation dataset of user ratings on a collection of movies, and the goal is to find the majority vote of these reviewers. This is a real world dataset with humans as labelers. The dataset is originally very sparse, meaning that only a small subset of users have rated each movie. We compiled a smaller subset of this dataset where each movie is rated by each user in the subset, to enable comparative experiments. We mapped the original rating scale of [0-5] to votes of $\{$-1,1$\}$ by using 2.5 as the decision boundary. ChemIR is a dataset of chemical patent documents from the 2009 TREC Chemistry Track. This track defines a "Prior Art Search" task, where the competition is to develop algorithms that, for a given set of patents, retrieve other patents that they consider relevant to those patents. The labelers for this dataset can be considered as software agents specifically designed for this particular task. The evaluation criteria is based on whether there is an overlap of the original citations of patents and the patents retrieved by the algorithm. The ChemIR dataset that we compiled is the complete list of citations of several chemistry patents, and the $\{$+1,-1$\}$ votes indicate whether or not an algorithm has successfully retrieved a true citation of a patent. Both MovieLens and ChemIR datasets have 11 labelers in total. Reuters is a popular dataset of articles that appeared on the Reuters newswire in 1987. We selected documents from the *money-fx* category. The Reuters data is divided into a "training set" and a "test set," which is not the format we need to test our algorithms. We used the first half of the training set (3,885 examples) to develop our labelers. Specifically, we trained several machine learning algorithms on these data: AdaBoost, Naïve Bayes, SVM, Decision Trees, and Logistic Regression. Each algorithm with its specific parameter setting was used to generate one labeler, and there were 10 labelers generated this way. Additionally, we selected 3 features of the dataset as labelers, for a total of 13 labelers. We combined the other half of the training set with the test set, which provided 6,904 total examples over which we used to measure the performance of CrowdSense and the baselines. The same simulation of a crowd that we conducted for the Reuters dataset was also used for the Adult dataset from the UCI Machine Learning Repository. Both of these datasets use well known machine learning algorithms as their labelers.

We compared CrowdSense with several baselines: (a) the accuracy of the *average* labeler, represented as the mean accuracy of the individual labelers, (b) the accuracy of the overall best labeler in hindsight, and (c) the algorithm that selects just over half the labelers (*i.e.* $\lceil 11/2 \rceil = 6$ for ChemIR and MovieLens, $\lceil 13/2 \rceil = 7$ for Reuters and Adult) uniformly at random, which combines the votes of labelers with no quality assessment using a majority vote. We also compare CrowdSense against IEThresh (Donmez et al., 2009). IEThresh builds upon Interval Estimation (IE) learning, and estimates an upper confidence interval UI for the mean reward for an action, which is a technique used in reinforcement learning. In IEThresh, an action refers to asking a labeler to vote on an item, and a reward represents the labeler's agreement with the majority vote. IEThresh updates the UI scores of the labelers after observing new votes,

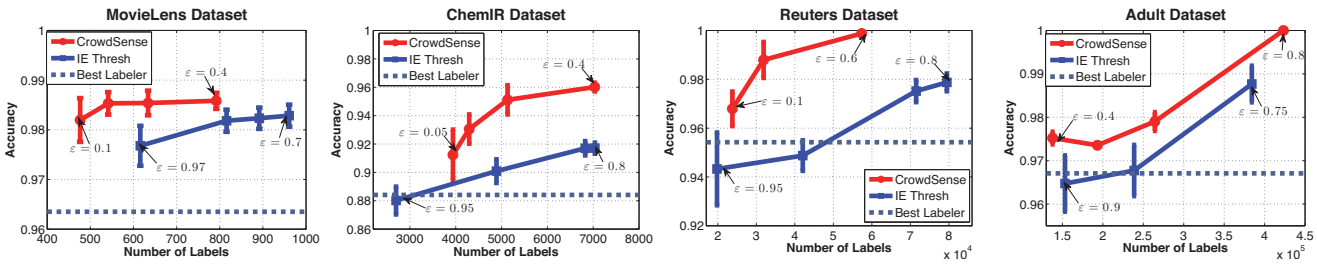

Figure 1: Tradeoff curves, averaged over 100 runs. The x-axis is the total number of votes (the total cost) used by the algorithm to label the entire dataset. The y-axis indicates the accuracy on the full dataset.

and given $\{UI_1, UI_2, \ldots, UI_k\}$ for the labelers, IEThresh selects all labelers with $UI_{\tilde{j}} > \varepsilon \times \max_{\tilde{j}}(UI_{\tilde{j}})$. The ε parameter in both CrowdSense and IEThresh algorithms tunes the size of the subset of labelers selected for voting, so we report results for a range of ε values. Note that tuning ε exhibits opposite behavior in CrowdSense and IEThresh; increasing ε relaxes CrowdSense's selection criteria to ask for votes from more labelers, whereas larger ε causes IEThresh to have a more strict selection policy.

It is worth noting that we do not assess the performance of the algorithms on separate test splits of the datasets. Rather, we make a single pass over the *entire* dataset and select labelers for each example based on the quality estimates available at that particular time. This is different than how IEThresh was evaluated in (Donmez, Carbonell, and Schneider 2009), where the labelers' qualities were first learned on a training set, and then the single best labeler with the highest accuracy was selected to label all the examples in a separate test set. In order to have a valid comparative assessment of the iterative nature of quality estimation, the majority vote for each example in IEThresh is computed based on the majority vote of the selected sub-crowd, similar to CrowdSense.

Overall Performance

We compare CrowdSense with the baselines to demonstrate its ability to accurately approximate the crowd's vote. Accuracy is calculated as the proportion of examples where the algorithm agreed with the majority vote of the entire crowd. Figure 1 indicates that uniformly across different values of ε, CrowdSense consistently achieved the highest accuracy against the baselines, indicating that CrowdSense uses any fixed budget more effectively than the baselines, including IEThresh. Generally, the quality estimates of CrowdSense better reflect the true accuracy of the members of the crowd and therefore, it can identify and pick a more representative subset of the crowd. The other baselines that we considered did not achieve the same level of performance as CrowdSense and IEThresh. On the subplots in Figure 1, the accuracy of the best labeler in hindsight (baseline (b)) is indicated as a straight line. Note that the accuracy of the best labeler is computed separately as a constant. Baselines (a) and (c), which are the average labeler and the unweighted random labelers, achieved performance beneath that of the best labeler. For the MovieLens dataset, the values for these baselines are 74.05% and 83.69% respectively; for ChemIR these values are 68.71% and 73.13%, for Reuters, the values are 84.84% and 95.25%, and for Adult they are 83.94% and 95.03%. The results demonstrate that asking for labels from labelers at random may yield poor performance for representing the majority vote, highlighting the importance of making informed decisions for selecting the representative members of the crowd. Asking the best and average labeler were also not effective approximators of the majority vote.

Effect of the parameters

As we discussed earlier, the ε parameter is a trade-off between the total cost that we are willing to spend and the accuracy that we would like to achieve. We compared the running accuracy of CrowdSense at various ε values, and in all datasets, the results consistently demonstrated that increasing epsilon leads to increased accuracy, at a higher cost.

The K parameter helps with both exploration at early stages and exploitation at later stages. In terms of exploration, K ensures that labelers who have seen fewer examples (smaller c_{it}) are not considered more valuable than labelers who have seen many examples. Additionally, since the quality estimates are all approximately equal in early stages due to the effect of K, the weighted majority vote becomes almost a simple majority vote. This prevents CrowdSense from trusting any labeler too much early on.

Increasing K also makes the quality estimates more stable, which helps to permit exploration. The quality estimates in the first few iterations that the labeler is involved in are not often accurate, and not very stable, and as a result, the algorithm's accuracy could be sensitive to what happens in the early iterations. Consider two equally accurate labelers. After each has labeled many items, it would have been clear that they are equal. If one labeler coincidentally makes more mistakes in the earlier iterations than the other, then the mistakes at these early iterations will constitute a larger fraction of the votes from that labeler (a_{it}/c_{it} will be small, and c_{it} will be small). Because of this, it is possible that this labeler will not be chosen as often as the other one, and it may not be possible to recover from the bias caused by the mistakes in the early stages. One of the purposes of K is to help reduce this bias – both labelers will be assigned almost equal quality estimates in early stages.

Since the quality estimates are all approximately equal in early stages, the weighted majority vote becomes almost a

simple majority vote. This prevents CrowdSense from trusting any labeler too much early on. Having the Q_{it}'s be almost equal also increases the chance to put CrowdSense into the "regime of uncertainty" where it requests more votes per example, allowing it to explore the labelers more. The impact of K on the quality estimates gradually reduces as more votes are observed, and exploitation-based selection of labelers will then start favoring labelers that are indeed higher quality than others, and trusting them more in the weighted majority vote.

Specific Choices for Exploration and Exploitation in CrowdSense

The algorithm template underlying CrowdSense has three components that can be instantiated in different ways: (1) the composition of the initial seed set of labelers (step 4(b) in the pseudocode), (2) how subsequent labelers are added to the set (step 4(c)), and (3) the weighting scheme, which affects the selection of the initial labeler set, the way the additional labelers are incorporated, as well as the strategy for combining the votes of the labelers (steps 4(b)(c)(d)).

We tested the effect of the first component by running separate experiments that initialized the labeler set with three (3Q), one (1Q) and no (0Q) labelers that had the highest quality estimates, where for the latter two, additional labelers were selected at random to complete the set of three initial labelers. 3Q removes the exploration capability of the initial set whereas the latter two make limited use of the quality estimates. Figure 2 shows an example for a specific choice of ε, where all three variants have lower predictive performance than CrowdSense.

We experimented with the second component by adding labelers randomly rather than in order of their qualities. In this case, exploitation is limited, and the algorithm again tends not to perform as well (as shown in Figure 2 for the curve marked "Component 2").

To test the effect of the weighting scheme in the third component, we removed the use of weights from the algorithm. This corresponds to selecting the initial seed of labelers and the additional labelers at random without using their quality estimates. In addition, when combining the votes of the individual labelers, we use majority voting, rather than a weighted majority vote. This approach performed dramatically worse than the rest of the variants in Figure 2, demonstrating the significance of using quality estimates for labeler selection and the calculation of weighted vote.

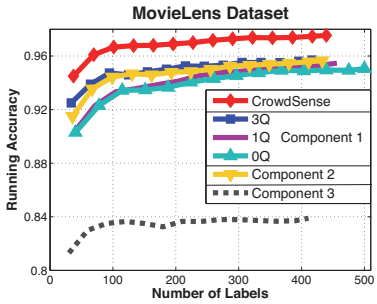

Figure 2: Performance of CrowdSense and several of its variants on the Movielens dataset, averaged over 100 runs. The x-axis is the total number of votes (the total cost), and the y-axis is the running accuracy on those examples.

Probabilistic Algorithms for Approximating the Crowd

CrowdSense has four important properties: 1) It takes labelers' votes into account even if they are right only half of the time. This property is especially beneficial for approximating small crowds, 2) It trusts "better" labelers more, and places higher emphasis on their votes, 3) Its decision rule is derived to be similar to boosting, and 4) Bayesian shrinkage of the quality estimates. These estimates provide the means for exploration of labelers in the earlier stages, and exploitation in later stages.

The fact that the majority vote determines the ground truth indicates a complicated relationship – a joint probability distribution – between the labelers' accuracies and the majority vote. CrowdSense makes an implicit assumption on the joint probability distribution through its boosting-style update. We believe it is possible to further improve its accuracy by making an explicit assumption on the joint probability distribution. To that effect, we propose two variations of CrowdSense that directly incorporate the joint probability distributions of the labelers under different assumptions. The first variation (CrowdSense.Ind) makes a statistical independence assumption of the labelers. That is, we will assume that the majority vote is independent of the votes that we have seen so far. The second variation (CrowdSense.Bin) makes a different probabilistic assumption, which is a lower bound on how often the current subcrowd agrees with the majority vote. This leads to a different voting scheme which is based on the binomial distribution of the votes. This also replaces the boosting-style decision rule in property 3. CrowdSense.Bin does not include the current subcrowd's weights in the vote, but the labeler selection criteria still favors labelers with high accuracy. Consequently, this approach covers the second property to some extent, but not as strong as the original CrowdSense.

CrowdSense.Ind

CrowdSense.Ind makes an assumption, namely that the quality of the individual labelers gives much more information than the count of votes received so far. In other words, that a labeler who is right 90% of the time and votes +1 should be counted more than a handful of mediocre labelers who vote -1. This assumption is approximately true when there a large number of labelers, but is not true for a small number of labelers. For instance, consider a case where we have received three votes so far: [+1,-1,-1], from labelers with qualities .8, .5, and .5 respectively. Let's say that there are a total of 500 labelers. The evidence suggests that the majority vote will be +1, in accordance with the high quality voter, and discounting the low-quality labelers. This is the type of assumption CrowdSense.Ind makes. Instead, if there were only 5 labelers total, evidence suggests that the

majority vote might be -1, since the low quality labelers still contribute to the majority vote, and only one more -1 vote is now needed to get a majority.

Let V_i denote the vote of labeler i on a particular example, and P_i denote the probability that the labeler will agree with the crowd's majority vote. Consider that two labelers have voted 1 and -1, and we want to evaluate the probability that the majority vote is 1. Then, from Bayes Rule, we have

$$P(y=1|V_1=1,V_2=-1)=\frac{P(V_1=1,V_2=-1,y=1)}{P(V_1=1,V_2=-1)}$$

and after some derivations, we obtain

$$P(y=1|V_1=1,V_2=-1) = \frac{P_1(1-P_2)P(y=1)}{P(y=1)P_1(1-P_2)+P(y=-1)(1-P_1)P_2}$$

where $P(y=1)$ and $P(y=-1)$ are the ratios of the crowd's approximated votes of 1 and -1 on the examples voted so far, respectively.

To expand this more generally to arbitrary size subsets of crowds, we define the following notation: Let V_i^{bin} denote the mapping of labeler i's vote from $\{-1,1\}$ to $\{0,1\}$, i.e. $V_i^{\text{bin}}=(V_i+1)/2$. Using the following notation for the joint probabilities of agreement and disagreement with the crowd:

$$\psi_i = P_i^{V_i^{\text{bin}}}(1-P_i)^{1-V_i^{\text{bin}}} \qquad \text{P(agreement}|V^{\text{bin}})$$

$$\theta_i = (1-P_i)^{V_i^{\text{bin}}}P_i^{1-V_i^{\text{bin}}} \qquad \text{P(disagreement}|V^{\text{bin}})$$

the likelihood of the majority vote y being 1 is estimated by the following conditional probability:

$$\begin{aligned} f(x_t|\text{votes}) &= P(y=1|\underbrace{V_1,V_2,\ldots,V_i}_{\text{votes of labelers in } S}) \\ &= \frac{\left(\prod_{l\in S}\psi_l\right)P_+}{\left(\prod_{l\in S}\psi_l\right)P_+ + \left(\prod_{l\in S}\theta_l\right)(1-P_+)} \\ &= \frac{1}{1+\frac{\left(\prod_{l\in S}\theta_l\right)(1-P_+)}{\left(\prod_{l\in S}\psi_l\right)P_+}} \end{aligned} \tag{3}$$

where probabilities higher than 0.5 indicate a majority vote estimate of 1, and -1 otherwise. Further, the more the probability in (3) diverges from 0.5, the more we are confident of the current approximation of the crowd based on the votes seen so far. As in CrowdSense, the decision of whether or not to get a vote from an additional labeler depends on whether his vote could bring us to the regime of uncertainty. This corresponds to hypothetically getting a -1 vote when $P(y=1|\text{votes})>0.5$, and getting a 1 vote otherwise. Defining

$$\psi_{\text{candidate}} = \begin{cases} 1-P_{\text{candidate}} & f(x_t|\text{votes})>0.5 \\ P_{\text{candidate}} & \text{otherwise} \end{cases}$$

$$\theta_{\text{candidate}} = \begin{cases} P_{\text{candidate}} & f(x_t|\text{votes})>0.5 \\ 1-P_{\text{candidate}} & \text{otherwise} \end{cases}$$

and expanding (3) to include the new hypothetical vote from the candidate labeler, we get

$$f(x_t|\text{votes},V_{\text{candidate}}) = \frac{1}{1+\frac{\left(\prod_{l\in S}\theta_l\right)\theta_{\text{candidate}}(1-P_+)}{\left(\prod_{l\in S}\psi_l\right)\psi_{\text{candidate}}P_+}}. \tag{4}$$

The decision as to whether get a new vote depends on the values of Eq (3) and (4). If we are sufficiently confident of the current majority vote to the point where the next best labeler we have could not change our view, or is not in the regime of uncertainty, then we simply make a decision and do not call any more votes.

CrowdSense.Bin

The statistical independence assumption that we made for CrowdSense.Ind is almost valid for sufficiently large crowds, but it does not hold when the crowd is small. For CrowdSense.Bin, we modify our treatment of the joint probability distribution by estimating a lower bound on how often the sub-crowd S_t agrees with the crowd's majority vote.

Consider again the scenario where there are 5 labelers in the crowd, and we already have observed [+1, -1, -1] votes from three labelers. The simple majority of the crowd would be determined by 3 votes in agreement, and we already have two votes of -1. So the problem becomes estimating the likelihood of getting one more vote of -1 from the two labelers that have not voted yet, which is sufficient to determine the simple majority vote. If the voters are independent, this can be determined from a computation on the binomial distribution. Let N_{needed} denote the number of votes needed for a simple majority and let N_{unvoted} denote the number of labelers that have not yet voted. (In the example above with 5 labelers, $N_{\text{needed}}=1$ and $N_{\text{unvoted}}=2$). In order to find a lower bound, we consider what happens if the remaining labelers have the worst possible accuracy, P=0.5. We then define the score using the probability of getting enough votes from the remaining crowd that agree with the current majority vote:

$$\begin{aligned} \text{Score} &= P_{\text{Bin}(N_{\text{unvoted}},0.5)}(X\geq N_{\text{needed}})-0.5 \\ &= \left[\sum_{X=N_{\text{needed}}}^{N_{\text{unvoted}}}\text{Bin}(X,N_{\text{unvoted}},0.5)\right]-0.5, \end{aligned} \tag{5}$$

where $\text{Bin}(X,N_{\text{unvoted}},0.5)$ is the X^{th} entry in the Binomial distribution with parameters N_{unvoted} and probability of success of 0.5. In other words, this score represents a lower bound on how confident we are that the breakdown of votes that we have observed so far is consistent with the majority vote. The score (5) is always nonnegative; therefore, the decision to ask for a new vote can then be tied to our level of confidence, and if it drops below a certain threshold ε, we can ask for the vote of the highest quality labeler that has not voted yet. The algorithm stops asking for additional votes once we are sufficiently confident that the subset of labels that we have observed is a good approximation of the crowd's vote.

When we compared CrowdSense.Ind and CrowdSense.Bin with CrowdSense, CrowdSense achieved better

performance than its probabilistic variants on most datasets. However, on a larger MovieLens dataset that we compiled (with 51 labelers), CrowdSense.Bin was the best performer. In another dataset that we synthetically generated with 101 labelers, CrowdSense.Ind yielded accuracy that is marginally better than CrowdSense.

Conclusions

Our goal is to "approximate the crowd," that is, to estimate the crowd's majority vote by asking only certain members of it to vote. We discussed exploration/exploitation in this context, specifically, exploration for estimating the qualities of the labelers, and exploitation (that is, repeatedly using the best labelers) for obtaining a good estimate of the crowd's majority vote. We presented a modular outline that CrowdSense follows, which is that a small pool of labelers vote initially, then labelers are added incrementally until the estimate of the majority is sufficiently certain, then a weighted majority vote is taken as the overall prediction. We discussed specific choices within this modular outline, the most important one being the choice of ε, which determines the overall budget for labeling the entire dataset. We compared our results to several baselines, indicating that CrowdSense, and the overall exploration/exploitation ideas behind it, can be useful for approximating the crowd.

We then presented two variations of CrowdSense, namely CrowdSense.Ind, which is based on an independence assumption, and CrowdSense.Bin, where calculations are based on the binomial distribution. These two algorithms make probabilistic assumptions or bounds on the joint distribution of the votes and the labels.

References

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2010. Soylent: a word processor with a crowd inside. In *Proc. of the 23rd annual ACM symposium on User interface software and technology*, 313–322.

Callison-Burch, C., and Dredze, M. 2010. Creating speech and language data with amazon's mechanical turk. In *Proc. of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 1–12.

Dawid, A. P., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics* 28(1):20–28.

Dekel, O., and Shamir, O. 2009a. Good learners for evil teachers. In *Proc. of the 26th Annual International Conference on Machine Learning (ICML)*.

Dekel, O., and Shamir, O. 2009b. Vox populi: Collecting high-quality labels from a crowd. In *Proc. of the 22nd Annual Conference on Learning Theory*.

Dekel, O.; Gentile, C.; and Sridharan, K. 2010. Robust selective sampling from single and multiple teachers. In *23rd Conference on Learning Theory (COLT)*, 346–358.

Donmez, P.; Carbonell, J. G.; and Schneider, J. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*, 259–268.

Kaisser, M., and Lowe, J. 2008. Creating a research collection of question answer sentence pairs with amazons mechanical turk. In *Proc. of the 6th Intl. Language Resources and Evaluation*.

Kasneci, G.; Gael, J. V.; Stern, D.; and Graepel, T. 2011. Cobayes: bayesian knowledge corroboration with assessors of unknown areas of expertise. In *Proc. of the 4th ACM International Conference on Web Search and Data Mining (WSDM)*, 465–474.

Law, E., and von Ahn, L. 2011. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

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* 11:1297–1322.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proc. of the 14th International Conference on Knowledge Discovery and Data Mining (KDD)*, 614–622.

Smyth, P.; Burl, M. C.; Fayyad, U. M.; and Perona, P. 1994a. Knowledge discovery in large image databases: Dealing with uncertainties in ground truth. In *KDD Workshop*, 109–120.

Smyth, P.; Fayyad, U. M.; Burl, M. C.; Perona, P.; and Baldi, P. 1994b. Inferring ground truth from subjective labelling of venus images. In *Advances in Neural Information Processing Systems (NIPS)*, 1085–1092.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. Y. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proc. of the Conf. on Empirical Methods in Nat. Lang. Processing*, 254–263.

Wallace, B. C.; Small, K.; Brodley, C. E.; and Trikalinos., T. A. 2011. Who should label what? instance allocation in multiple expert active learning. In *Proc. of the SIAM International Conference on Data Mining (SDM)*.

Warfield, S. K.; Zou, K. H.; and Wells, W. M. 2004. Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation. *IEEE Transactions on Medical Imaging (TMI)* 23(7):903–21.

Welinder, P.; Branson, S.; Belongie, S.; and Perona, P. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems (NIPS)*.

Whitehill, J.; Ruvolo, P.; fan Wu, T.; Bergsma, J.; and Movellan, J. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2035–2043.

Yan, Y.; Rosales, R.; Fung, G.; and Dy, J. 2010a. Modeling multiple annotator expertise in the semi-supervised learning scenario. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-10)*, 674–682.

Yan, Y.; Rosales, R.; Fung, G.; Schmidt, M. W.; Valadez, G. H.; Bogoni, L.; Moy, L.; and Dy, J. G. 2010b. Modeling annotator expertise: Learning when everybody knows a bit of something. *Journal of Machine Learning Research - Proceedings Track (JMLR)* 9:932–939.