An Epistemic Approach in Multiple Social Issues

CS 238
Spring 2022

Max Guo, Rachel Li, Jacqueline Wei

1 Abstract

Previous work in epistemic democracy often relies on the assumption that voter competence levels are fixed, known quantities. However, in practice, these competence levels are often inaccessible. To address this challenge, we propose the formulation of an epistemic model where voter competence levels are heterogeneous and unknown. We consider the setting of voting on multiple social issues and develop an algorithm to determine the ground-truth given a set of voter responses. By estimating a difficulty ranking among issues and exploring various update rules, the algorithm iteratively updates voter competence estimates and uses these to inform voting on subsequent issues. In this way, we show that the algorithm is able to derive a ground truth that may differ substantially from the simple majority outcome. We use both simulations and empirical data to demonstrate the viability of our proposed algorithm. In our theoretical work, we 1) characterize a setting in which we are able to identify competent individuals and 2) analyze the convergence of our algorithm in this setting to show that it obtains a full set of correct answers with probability 1 as the number of questions tends towards infinity.

2 Introduction

In most epistemic democracy models, such as Condorcet's famous jury theorem, voter competencies are assumed to be fixed, known quantities. However, in practice, these are often inaccessible (and hard to measure). We consider the setting of voting on multiple social choice issues and seek to answer the question:

Given a set of voter responses, how do we estimate voter competencies and deduce the ground-truth answers?

This project serves as a natural extension of information aggregation techniques to the social choice setting. The problem of information aggregation has been well-explored in previous literature because of its extensive applications to machine learning and crowdsourcing. For example, labels for machine learning datasets are commonly sourced from workers on Amazon Mechanical Turk, and responses must be aggregated to estimate the true labels. In the "wisdom of crowds" setting, participant predictions must be aggregated to obtain an estimate of the ground truth for a particular forecasting question. In both of these applications, the participants have heterogenous areas of expertise, sources of information, and levels of effort that affect their response accuracy. In this case, it is natural to model each participant with an associated *competency level* that summarizes these differences in accuracy.

In our project, we consider the expanding the current work on information aggregation to the social choice setting where participants vote on a series of social issues. We assume that there is a socially optimal ground-truth answer for each issue and can interpret voter competencies as a metric for the extent to which voters are informed on the set of issues at hand. In particular, since we are aggregating voter responses, we seek to propose an algorithm that is intuitive and can be easily explained to the participants.

3 Related Work

Due to their prevalence in crowdsourcing and machine learning, information aggregation and competence estimate models have been extensively studied. Bachrach et al. propose a Bayesian adaptive learning model using an expectation propagation algorithm, which assumes that competence levels and question difficulties follow Gaussian priors [1]. Their algorithm follows from some earlier theoretical work by Herbrich et al., who use this expectation propagation approach to model player skill ratings for Xbox matchmaking purposes, an algorithm known as TrueSkill [2]. Venanzi et al. propose an assumption that voters conform into "types," such that individuals of each type vote similarly according to a confusion matrix [3]. They propose an extension of the Bayesian Classifier Combination model to learn confusion matrices and latent worker communities in order to produce aggregated labels. Furthermore, weighted majority algorithms often appear in online learning settings, where individual competencies are updated iteratively, but the correct answers are known in this setting. Many of these algorithms are theoretically complex; we seek to implement a simpler, more intuitive algorithm below.

4 Model

We model the following setting:

- A set N of voters ($|N| = n$).
- A set M of questions ($|M| = m$).
- Each question $j \in M$ has alternatives $a_{j1}, \ldots, a_{jk}$, where k is fixed. Assume that there exists a unique ground-truth correct answer to each question.
- Each voter $i \in N$ has a vector $\vec{v_i} = (v_{i1}, \ldots, v_{im})$ of responses, where v_{ij} is the response of person i to question j.
- Each voter $i \in N$ has an (unknown) *competence level* $p_i^* \in [0, 1]$, where p_i^* is voter i's probability of choosing the correct answer. We denote our algorithm's estimated competency levels as p_i, omitting the star.

5 Our Algorithm

Let D denote the $n \times m$ data matrix of voter responses, where $D_{ij} = v_{ij}$ is the response of person i to question j. Let $V_{j\ell}$ denote the set of voters who respond with alternative a_ℓ for question j. We define the *consensus* c_j of question j as the highest number of voters who are in agreement about a particular answer choice:

$$c_j = \max_{\ell \in \{1 \ldots k\}} |V_{j\ell}|$$

Intuitively, consensus can be thought of as a proxy for question difficulty, with $c_j = n$ being the least difficult (everyone agrees), and $c_j = \lceil \frac{n}{k} \rceil$ being the most difficult (votes evenly split amongst all alternatives). The algorithm proceeds as follows:

1. Initialize competencies p_i for all $i \in N$
2. Rank questions $j \in M$ from most consensus to least consensus.
3. For each question j, we use a **weighted majority vote** to determine the estimated answer a_j^*, where the weight of voter i is her competency p_i
$$a_j^* = \arg\max_{a_\ell \in A_j} \sum_{i \in V_{j\ell}} p_i$$
4. Update voter competencies according to the following heuristics:
   - If voter i answers "correctly" (i.e. $v_{ij} = a_j^*$), increase p_i
   - Otherwise, decrease p_i

In our project, we explore the use of two different update frameworks.

The **penalization** update rule initializes all voter competencies to $p_i = 1$ for all $i \in N$. If $v_{ij} \neq a_j^*$, we penalize p_i and perform some update so the new value is less than the old value. In one variant of the rule, if i answers incorrectly, we update $p_i = \max(p_i - t, 0)$ for some constant $t \in (0, 1)$. If they answer correctly, we do nothing. This update rule is quite intuitive since it says that a voter is penalized each time she gets a question wrong, according to the estimated answer determined by the algorithm. Once the voter gets a certain number of questions wrong, her competency hits 0 and her vote no longer counts towards the weighted majority vote determination of the estimated answer.

In the **1/2-penalization** update rule, we halve the estimated competency of an individual if they are not in the weighted majority. If $v_{ij} \neq a_j^*$, we update $p_i = \frac{1}{2} p_i$. Otherwise, if they agree with the weighted majority, we do nothing.

The **beta-binomial** update rule initializes all voter competencies to $p_i = \frac{1}{2}$ for all $i \in N$. In this update rule, p_i is parameterized by α_i and β_i, such that $p_i = \frac{\alpha_i}{\alpha_i + \beta_i}$. If $v_{ij} = a_j^*$, update

$$\alpha_i = \alpha_i + \frac{c_j}{n}.$$

Otherwise, update

$$\beta_i = \beta_i + \left(1 - \frac{c_j}{n}\right).$$

We choose this update rule since it yields a convenient Bayesian interpretation where competencies are drawn from a Beta (α_i, β_i) distribution, with p_i as the posterior mean after each update. Additionally, for the layperson, this update rule can be loosely interpreted as the fraction of questions voter i has answered "correctly," weighted by the difficulty of each question. We heuristically chose to update α_i by c_j/n and β_i by $(1 - c_j/n)$ since this seemed to perform the best in the simulations. Intuitively, we can think of this as rewarding voters more for being correct on the initial easy questions to try to update the competencies quickly, and penalizing voters more for being incorrect on the later difficult questions to try to shrink the weights of lower competency voters towards 0.

Extending this idea is the additional heuristic we implement to find the "smart" crowd. Assuming that there is a population of high competency voters, it is intuitive to only consider the answers of these high competency voters on difficult questions and disregard other "noisy" voters. To improve our algorithm, we implement a cutoff that cumulatively eliminates the bottom percentile of lowest competency voters from the weighted majority vote for each question. As we see both on the simulation and real-world dataset results, this heurstic improves the performance of our algorithm.

6 Theoretical Analysis

We now attempt to give a few theoretical analyses of our model and algorithm. Consider the simplified setting where there are 2 alternatives in each social choice question, and voter competencies are either $p_i^* = 1$ (always correct) or $p_i^* = 0.5$ (fair coin toss to choose answer). Let S be the set of competent voters with $p_i^* = 1$. Assume that $|S| = s < \frac{n}{2}$, and $s > 2$, such that the competent voters are in the minority. Let $N \setminus S$ be the set of incompetent voters with $p_i^* = 0.5$ and $|N \setminus S| = n - s$.

6.1 Identifying Perfectly Competent Individuals

Suppose first that we know there are s people who are perfectly competent, and we'd like to identify them. What is the probability that we can perfectly identify the group S from the set of voter responses, without knowing the correct answers? The answer to this question contextualizes the analysis to follow on the convergence of our algorithm in this case, as we demonstrate that as $m \to \infty$ our bound increases past 1, which corroborates with our result of algorithmic convergence in this regime.

Formally, we ask, what's the probability that, given the responses of the N individuals, there is a unique group of size s with the same set of responses? Let this be $P(A)$. Note that $A = A_1 \cap A_2$, where A_1 is the event that no s people in $N \setminus S$ share answers, and A_2 is the event that no one in $N \setminus S$ answers perfectly correctly [1].

Theorem 1. *The probability that we uniquely identify the competent individuals S is given by:*

$$P(A) \leq 2\left(1 - \frac{1}{2^m}\right)^{(n-s)} F(s-1)^{2^m - 1}.$$

We put the proof in the appendix. To demonstrate that this bound fits our intuition, we plot the growth of this bound as a function of each of n, s, m, with fixed values for the other 2 in each case:

[1] Technically, we don't need to identify S in order to be 100% confident about knowing the correct responses. As long as there is only one set of answers with at least s people, we can uniquely identify the correct answer.

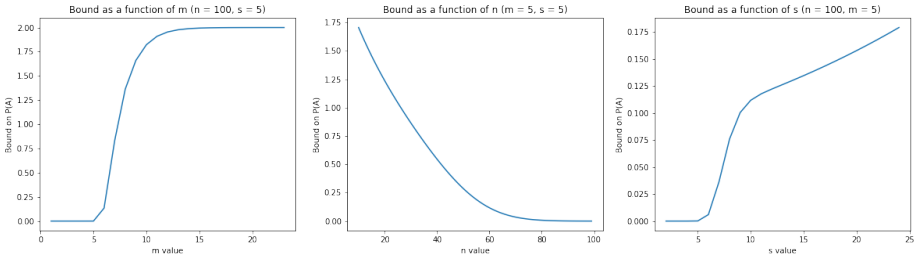

Figure 1: Upper Bound on $P(A)$, as a function of n, m, and s

These bounds fit our intuition. As m increases, the number of possible answer choices increases exponentially, so the chances that s random-guessing people match answers decreases and the bound on the probability that we can find S gets larger. As n increases, the number of random guessers increases, and the probability we can find S decreases since there is a greater chance s random-guessing people will converge on the same set of responses. Finally, as s increases, the probability at least s random guessers obtain the same answer decreases, so the probability we find S increases. Figure 1 shows example growths of the bound with respect to these three variables.

6.2 Convergence of our Algorithm

Now we explore the following theoretical analysis to better characterize the conditions under which our algorithm performs well, given the same settings in the previous section (assuming 2 answers per question, S competent individuals, and $N \setminus S$ random guessers, where $2 \leq s < \frac{n}{2}$). We assume the $1/2$-penalization rule for sake of analysis. All proofs are deferred to the appendix.

We define convergence to be the case where the proportion of questions the algorithm answers correctly approaches 1 as the number of questions grows large, or $m \to \infty$.

As before, let c_j be the consensus size (size of the unweighted majority). Let Q_j be the number of questions for which the consensus size is $n - j$. We show that Q_j tends to infinity:

Lemma 2. *Let $X \sim Bin(n, p)$, and fix $p > 0$ and $k > 0$ as constants. Then, as $n \to \infty$, we must have $P(n \leq k) \to 0$.*

Lemma 3. *$P(Q_j \leq k) \to 0$ as $m \to \infty$, for all k.*

Now let question j be the question with the jth highest consensus. We first prove that, when our algorithm sorts the questions in order of highest consensus to lowest consensus, it determines the correct answer on all questions with consensus value greater than or equal to $n - s + 1$.

Lemma 4. *If question j has consensus $c_j > n - s$, the algorithm will determine the correct answer on question j with probability 1.*

Finally, we extend this result to show that our algorithm will also choose the correct answers on all questions, including those with consensus values smaller than $n - s + 1$, in the limit as $m \to \infty$.

Theorem 5. *For 2 alternatives and a heterogeneous population of voters with $p_i^* \in \{0.5, 1\}$, the algorithm under the 1/2-penalization rule converges to answer all m questions correctly as $m \to \infty$.*

7 Simulation

7.1 Beta-Binomial Update Rule

We conduct a simulation using $n = 100$ voters, $m = 300$ questions, and $k = 4$ alternatives over 20 trials. We implement the algorithm using the beta-binomial update rule with a 1% cutoff. Individual true competencies are drawn from a population competency distribution parameterized by a and b, such that $p_i^* \sim Beta(a, b)$ and the

a	b	$\bar{p^*}$	Beats Majority	Avg. Outperformance
0.5	1	1/3	20/20	39.95
1	2	1/3	20/20	38.2
2	5	2/7	19/20	102.5
3	10	3/13	1/20	-16.25

population mean competency level is $\bar{p^*} = \frac{a}{a+b}$. We are interested in cases where the simple majority does not perform well, so we examine distributions where $\bar{p^*} < \frac{1}{2}$. Our results are summarized in the table below.

For $\bar{p^*} > \frac{1}{k}$, our algorithm usually outperforms simple majority, and in at least one such case does so by a significant margin (a third of all questions). However, for $\bar{p^*} = \frac{3}{13} < \frac{1}{k}$, we see that the algorithm usually underperforms majority. But we also note that simple majority also does not perform well in this case, answering on average 36.55 questions correctly out of 300. Given these results, we hypothesize that the algorithm performs well in cases where $1/k < \bar{p^*} < 1/2$ because the simple majority will tend to miss questions when it is primarily composed of low-competence individuals and this effect is mitigated by the weighted majority vote that our algorithm employs. We can investiage this by graphing the average outperformance (which we define as the average number of questions our algorithm answers correctly, minus the average number of questions majority answers correctly, over 10 trials) for several values of $\bar{p^*}$, holding $a = 1$ constant and modifying b accordingly. Results are displayed in Figure 2.

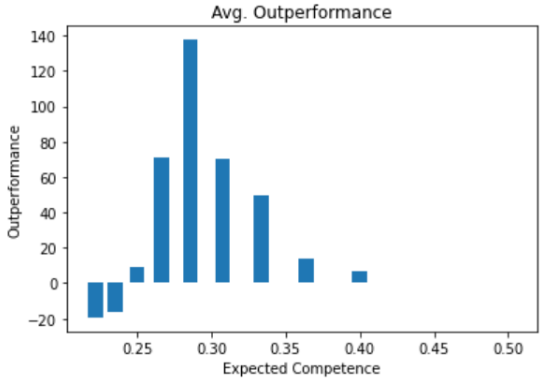

Figure 2: Average outperformance using the beta-binomial update rule

For high $\bar{p^*}$, both our algorithm and simple majority answer all questions correctly, so average outperformance is zero. As $\bar{p^*}$ decreases, our algorithm answers significantly more questions correctly than majority (peaking at 140, which is roughly half of all questions); but once $\bar{p^*}$ approaches $1/k$, we begin to significantly underperform majority. We hypothesize that this is due to the higher proportion of low-competence individuals in the population which causes the initial high-consensus questions to be incorrectly answer such that the algorithm rewards low-competence individuals.

7.2 Competency Estimate Accuracy

We also examine our algorithm's competency estimates and compare them to the true competencies in Figure 3. The graphs below demonstrate that our algorithm estimates competencies quite accurately in successful trials, which is surprising given that our competency updates are somewhat heuristic in nature. Importantly, the algorithm is able to differentiate between high-competence and low-competence individuals, which is consistent with the successful trials in which the algorithm outperforms the majority.

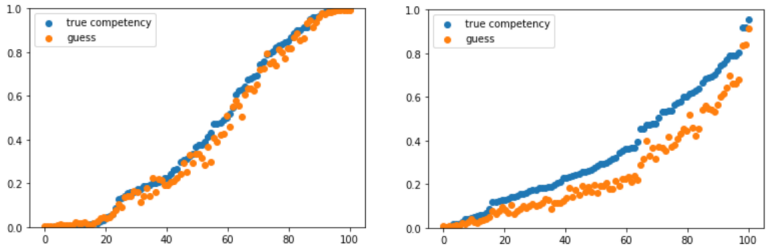

Figure 3: True competencies (in blue) versus our algorithm's estimated competencies (in orange). The left shows $p_i^* \sim Beta(1/2, 1/2)$; the right shows $p_i^* \sim Beta(1, 2)$

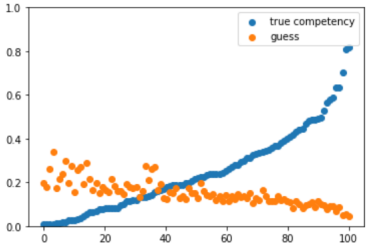

Figure 4: True competencies (in blue) versus our algorithm's estimated competencies (in orange) for $p^* \sim Beta(1, 3)$

We also examine competency estimates in trials where the algorithm fails to outperform majority in Figure 4. Not only are competencies poorly estimated (which is to be expected), but there also seems to be an inverse relationship between true competency and estimated competency. This supports the hypothesis earlier that for sufficiently low $\bar{p^*}$, our algorithm mistakes low-competence individuals for high-competence individuals and incorrectly rewards them for creating a large consensus around the wrong answer. This characterizes one of the weaknesses our algorithm.

7.3 Penalization Update Rule

We can also examine what happens if we switch to the penalization update rule with $t = 0.05$. Intuitively, this is a much harsher update rule than the beta-binomial rule: once an individual answers a question "incorrectly" as perceived by the algorithm, there is no opportunity to increase their competency again. Since there are 300 questions, we expect that the algorithm will eventually assign zero competency to most individuals. However, if we believe that our population individuals includes a few high-competence individuals and several low-competence individuals, this penalization rule may actually outperform majority for significantly lower $\bar{p^*}$. We simulate this outcome in Figure 5.

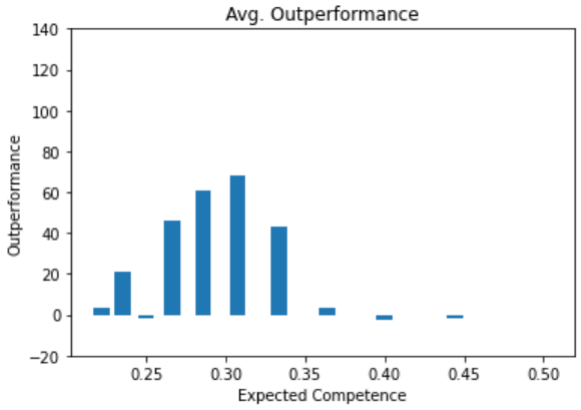

Figure 5: Average outperformance using the penalization update rule

Although the high range of average outperformance is significantly more muted than in the beta-binomial update rule, the penalization rule tends to perform much better for low $\bar{p^*}$; for example, for the extreme case when $\bar{p^*} = \frac{2}{9}$, the penalization rule gives an average outperformance of 3.7 questions, versus -19.4 for the beta-binomial rule. The intuition behind such improvement can be found by examining the competency estimates provided by the penalization rule in Figure 6.

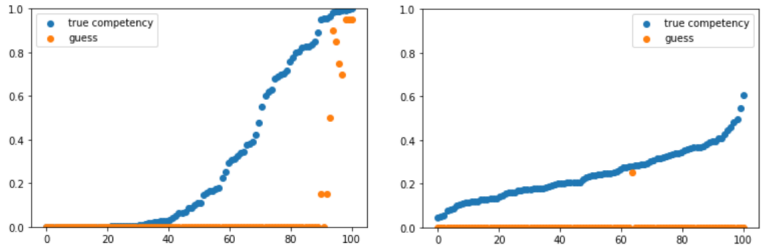

Figure 6: True competencies (in blue) versus our algorithm's estimated competencies (in orange). The left shows $p_i^* \sim Beta(1/5, 1/2)$ and the right shows $p_i^* \sim Beta(3, 10)$

In both cases, the algorithm eventually assigns zero competence to most individuals; however, the algorithm is able to find one or two individuals that have higher competency than the average individual. If the goal is to answer questions correctly, rather than estimate competency levels, the algorithm only needs to find the one or two people with high competence levels and ignore everyone else. Such improvement is the inspiration for our "finding the smart crowd" heuristic, which implements the same idea of zeroing out low competencies by iteratively cutting the bottom percentile of individuals from the weighted majority vote calculation for each question.

8 Real-World Datasets

We evaluate the algorithm performance on an empirical dataset of ~ 8000 multiple-choice question responses from the Good Judgment Project [4]. We implemented the algorithm using the beta-binomial update rule where the bottom 20% percentile of lowest competence voters where cumulatively cut from the weighted majority vote on each question.

1. **Raven's Advanced Progressive Matrices**
   Cognitive pattern matching test (8 answer choices)
   - Algorithm: 12 out of 12 correct
   - Simple Majority: 11 out of 12 correct
2. **General Political Knowledge**
   Ex. Who is the U.S. National Security Adviser? (4 answer choices)
   - Algorithm: 15 out of 15 correct
   - Simple Majority: 12 out of 15 correct
3. **Cluster Political Knowledge**
   Categorical questions on China, global economy, Iran, and Russia (4 answer choices)
   - Algorithm: 30 out of 32 correct
   - Simple Majority: 26 out of 32 correct

For the Raven's Advanced Progressive Matrices test, our algorithm gets the "most difficult" (least consensus) question correct while the simple majority fails on this question. This also holds for the General Political Knowledge test for which our algorithm also answers the top three "most difficult" (least consensus) questions while the simple majority fails on these questions. The Cluster Political Knowledge test is composed of 32 questions in 4 categories on China,

global economy, Iran and Russia. Interestingly, we might expect that running the algorithm separately on each of the different question categories would allow us to identify the "experts" in each of these categories. However, doing so reduces the algorithm's performance and it only gets the same number questions correct as the simple majority. We hypothesize that this may be due to the small number of questions (8 instead of 32) such that the algorithm is not able to effectively update competencies. Additionally, since the question categories are very correlated, it makes sense that an expert in one category would be an expert in all of the categories, and thus we see that the algorithm performs well on the 32 questions as whole.

Overall, these empirical results are reassuring as they show the viability of how our intuitive algorithm can outperform the simple majority vote. Additionally, the assumptions that there are populations of more competent voters and enough easy questions such that our algorithm updates competencies in the correct direction seem to hold empirically as well.

9 Conclusion

In conclusion, we consider a model in which the ground-truth answers and voter competencies are not known, and voters vote on multiple issues. Given a set of voter responses, we propose an intuitive algorithm which iteratively estimates voter competencies and uses these in a weighted majority vote to compute the estimated ground-truth answers.

In our theoretical findings, we consider the setting in which there are a few perfectly competent individuals and others who are randomly guessing, and we give an upper bound to the theoretical limit on the probability of identifying these individuals. Furthermore, we show that, in this scenario, our algorithm achieves a perfect score in estimating the ground truth answers as the number of questions tends to infinity.

In the simulation results, we characterize conditions where our algorithm performs well and show that the algorithm is able to outperform the simple majority and accurately estimate voter competencies. We also evaluate the algorithm on empirical datasets to show that these conditions are generally held in the real world and that our algorithm is thus able to perform well on many practical instances.

Future directions for our empirical work include benchmarking our algorithm on datasets against existing information aggregation models. We would also like to evaluate the performance of our algorithm on social choice datasets and compare these to the simple majority outcomes. For future theoretical work, we would like to give more theoretical bounds for our algorithm's performance and variance under more general scenarios, including more realistic distributions of competencies.

10 References

[1] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver, "How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing," *arXiv preprint arXiv:1206.6386*, 2012.

[2] R. Herbrich, T. Minka, and T. Graepel, "Trueskill," in *NIPS*, 2006.

[3] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi, "Community-based bayesian aggregation models for crowdsourcing," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14, Seoul, Korea: Association for Computing Machinery, 2014, pp. 155–164, ISBN: 9781450327442. DOI: 10.1145/2566486.2567989. [Online]. Available: https://doi.org/10.1145/2566486.2567989.

[4] G. J. Project, *GJP Data*, version V1, 2016. DOI: 10.7910/DVN/BPCDH5. [Online]. Available: https://doi.org/10.7910/DVN/BPCDH5.

[5] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

11 Appendix

We provide proof sketches and proofs to the lemmas and theorems stated above.

11.1 Proof of Theorem 1

Proof. We observe that $P(A_2) = \left(1 - \frac{1}{2^m}\right)^{(n-s)}$. To calculate $P(A_2|A_1)$, we note that the remaining $2^m - 1$ sets of answers that are not fully correct are distributed uniformly with probability $\frac{1}{2^m-1}$ each. Thus, we can imagine this as a balls-and-bins problem, with $2^m - 1$ bins and $n - s$ balls. We desire the probability that no more than $s - 1$ balls fall into any one bin. Using the Poisson Approximation [5], we can bound this probability above by:

$$P(A_2|A_1) \leq 2F(s-1)^{2^m-1},$$

where F is the CDF function of a Poisson random variable with rate parameter $\frac{n-s}{2^m-1}$. Thus, we have that:

$$P(A) \leq 2\left(1 - \frac{1}{2^m}\right)^{(n-s)} F(s-1)^{2^m-1},$$

as desired. □

11.2 Proof of Lemma 2

Proof. The proof follows by Hoeffding's inequality. See this link on Wikipedia. □

11.3 Proof of Lemma 3

Proof. We note that Q_j follows a Binomial distribution with size parameter m and probability parameter p, where $p = \frac{\binom{N-S}{j}}{2^{N-S}}$. Then this lemma follows directly from Lemma 2. □

11.4 Proof of Lemma 4

Proof. We will use an inductive approach. Consider the base case where $c_1 = n$. This means that all voters agree on the same answer, which must be the correct answer since the competent voters are part of the total population. By strong induction, for all $j < k$, assume that $c_j > n - s$ and the algorithm determines the correct answer on question j. For the inductive step, we want to show that if $c_k > n - s$, then the algorithm will determine the correct answer on question k. In order for this to occur, we need the correct answer to have the largest weight such that it wins the weighted majority vote (WMV). Let X be the set of incompetent voters who vote correctly on question k, and Y be

the set of incompetent voters who vote incorrectly on question k. Note that $Y = N \setminus (S \cup X)$, so $|Y| = n - c_k$. The WMV determines the correct answer when the following condition holds:

$$\sum_{i \in S} p_i + \sum_{i \in X} p_i > \sum_{i \in Y} p_i.$$

Note that by the inductive assumption, the algorithm has only answered questions correctly thus far, so $p_i = 1$ for all $i \in S$. Additionally, by the condition that $c_k > n - s$, we know that we can upper bound the RHS (since all $p_i \leq 1$) as follows:

$$\sum_{i \in Y} p_i < s.$$

Since the competencies are always positive, $\sum_{i \in X} p_i > 0$ and the inequality holds as desired:

$$s + \sum_{i \in X} p_i > s > \sum_{i \in Y} p_i.$$

□

11.5 Proof of Theorem 5

Proof. By Lemma 4, we know that the algorithm will answer the first j questions correctly for all j such that $c_j > n - s$. Let k be the first question where $c_k \leq n - s$. We wish to show that the algorithm still determines the correct answer on question k.

Let us first consider the estimated competencies of all voters at this step of the algorithm. For all $i \in S$, since the algorithm has answered all previous questions correctly, $p_i = 1$. Now consider the number of questions an incompetent individual has answered incorrectly, which is directly related to their competence. As before, let Q_i be the number of questions with consensus value $n - i$. Within this set of questions, let $w_i \leq Q_i$ be the number of questions that an incompetent person x in $N \setminus S$ incorrectly answers, such that person x has competence (after updating on questions $1, \ldots, k-1$)

$$p_x = 2^{-\sum_{i=1}^{s-1} w_i}.$$

Then we note that, by Lemma 2, we have that:

$$P\left(\sum_{i=1}^{s-1} w_i < \log\left(\frac{n-s}{s}\right)\right) \leq P\left(w_1 < \log\left(\frac{n-s}{s}\right)\right) \to 0,$$

because w_1 is distributed Binomial with size parameter Q_1 and probability parameter $\frac{1}{n-1}$, and Q_1 goes to ∞ with probability 1 by Lemma 3. Thus, we have:

$$\begin{aligned} P\left(\sum_{i=1}^{s-1} w_i < \log\left(\frac{n-s}{s}\right)\right) &= P\left(\frac{s}{n-s} < 2^{-\sum_{i=1}^{s-1} w_i}\right) \\ &= P\left(\frac{s}{n-s} < p_x\right) \to 0. \end{aligned}$$

This implies that, with probability 1, each person $x \in N \setminus S$ will have competency less than $\frac{s}{n-s}$ by the time the algorithm reaches the questions for which the consensus is at most $n - s$. In the weighted majority vote, the competent voters will always outweight the incompetent voters, as the sum of the weights of the other voters is less than s with probability 1. This completes the proof of our theorem.

□