Crowdsourcing Quantitative Evaluation: Algorithms and Empirical Results*

Luca de Alfaro and **Michael Shavlovsky**
Computer Science Dept., University of California, Santa Cruz, CA 95064, USA
{luca, mshavlov}@ucsc.edu

Technical Report UCSC-SOE-14-03, School of Engineering, UC Santa Cruz

Abstract

We consider the problem of crowd-sourcing the quantitative evaluation of the quality of a set of items. Each item is examined and assigned a numerical grade by a small number human evaluators. Evaluators are naturally affected by personal biases, random mistakes, and unequal levels of dedication to the task: our goal is to compute consensus grades that are as precise and as free of biases as possible. We present two novel algorithms for this problem: one, nicknamed VariancePropagation, is related to belief propagation and maximum likelyhood; the other, nicknamed CostOfDisagreement, is based on iterated convex optimization. Both algorithms implement reputation systems that weigh more the input from more accurate evaluators.

On synthetic data, both algorithms far outperform simple aggregators such as average. To evaluate the performance of the algorithms in the real world, we used a dataset from CrowdGrader, a web based peer-grading tool, consisting in 11645 evaluations of 1797 submissions, collected over 22 homework assignments. As this dataset lacks a ground truth, we introduce and justify the use of *stability under subsampling* as a measure of algorithm precision that can be used in absence of a ground truth. On this real-world data, we identify a version of VariancePropagation that has superior performance to all other alternatives. We discuss the aspects and adaptations of the algorithms that make them well-suited to real-world use.

1 Introduction

Humans are the best judges of quality for many types of content. Examples are content produced via a creative or artistic process, such as creative-writing essays, photographs, non-trivial software applications, and content whose complexity precludes an easy understanding by machines. To produce quality estimates for large collections of such content, we need to rely on crowdsourcing: asking human evaluators to provide their judgement for individual items, and then merging the judgements into consensus estimates of quality. In this paper, we consider the question of how to best merge numerical quality estimates, such as grades, into reliable "consensus" estimates.

*This work was supported in part by the Google Research Award "Crowdsourced Ranking". The authors are listed in alphabetical order.

We consider a setting consisting of a set of items whose quality is to be evaluated, and a set of users that act as human evaluators. Each item is evaluated by a small set of users, which in our experiments numbers generally from 3-4 to a dozen at most. We assume that the evaluations are done in a non-anonymous fashion, so that we have stable identities for the users, and we assume that users typically perform more than one evaluation, making it possible to characterize their individual accuracy and error model. The goal is to reconstruct the best possible estimates of the qualities of the items. We introduce new algorithms for the computation of consensus quality estimates, and we evaluate the performance of these algorithms both on synthetic data, and on real-world data obtained via CrowdGrader (de Alfaro and Shavlovsky 2013), a web based peer-grading tool. As the peer-grading data does not come with a ground truth, one of the contributions of this paper will be to introduce accuracy measures for the algorithms that can be used also absent a ground truth. The crowdsourcing algorithms we introduce in this paper are based on reputation systems that measure the accuracy of each user, and that listen more to the input from more precise users.

The first algorithm, nicknamed VariancePropagation, is inspired by the algorithm of (Karger, Oh, and Shah 2011) for the crowdsourcing of boolean evaluations (such as spam/non-spam judgements), and is inspired also by expectation maximization techniques (Dempster et al. 1977; Dawid and Skene 1979; Raykar et al. 2010). The algorithm initially assumes that all users have the same a-priory accuracy, and then iteratively refines the estimates of item quality, and user accuracy. In each iteration, the algorithm computes a consensus estimate of the grade of each item, weighing the input from each user in according to the estimated accuracy of the user. The consensus estimats for the items are then used to refine our estimates of the accuracy of individual users.

Our second algorithm, nicknamed CostOfDisagreement, is based on the iterative solution of convex-optimization problems that aim at aligning the estimated grades to the available user input. At each iteration, CostOfDisagreement uses convex optimization to compute consensus grades for the items that minimize the total *cost of disagreement* with the users. The cost of disagreeing with a user is the product of two factors, which measure the amount of disagreement

of a user, and the reputation of a user. Initially, users all have the same reputation. After each iteration, once the convex-optimization problem is solved, we update the estimates of user reputation: users that have larger disagreement with the consensus grades are assumed to be less accurate and receive lower reputation, so that their input carries less weight in the next iteration.

We describe several variants of each of these algorithms that differ among other things in the details of how reputation is assigned to users and in the details of the error model adopted for users. We present an in-depth evaluation of these algorithms both on synthetic data, and on real-world data from peer-grading. On synthetic data, the algorithms far outperform simpler algorithms not based on user reptuation, such as average or median. Evaluating the performance of the algorithms on the real-world peer-grading data required overcoming one basic difficulty: we have no ground truth for the quality of the submitted solutions. Initially, we hoped that grades assigned by TAs or instructors could be used as ground truth. Aside from the relative sparsity of such data (people use peer grading to avoid grading all assignments, after all), the problem is that such control grades are often no more accurate than the grades crowdsourced from the students. We examined several computer science coding assignments, on topics we can be considered experts, and we noticed that often instructors and TAs, and ourselves, missed flaws that are detected by at least some of the student evaluators.

To overcome this difficulty, we introduce as quality criterion *stability under subsampling*, which measures how close are the consensus grades computed by the algorithms in runs in which some of the input data has been appropriately removed. The closer the consensus grades computed on pairs of independently subsampled input data, the greater the ability of the crowdsourcing algorithms to reconstruct accurate consensus grades with limited input. We validate this intuition both mathematically, and by showing that on synthetic data (where the ground truth is known), algorithm accuracy is very strongly related ($\rho \geq 0.9$) with stability under subsampling. Armed with this quality criterion, we present detailed results of the accuracy of VariancePropagation and CostOfDisagreement on our peer-grading dataset.

Our evaluation on real-world data is based on a peer-grading dataset consisting of 1797 items, 973 users, and 11645 grades, collected over 22 assignments. On this data, one variant of the VariancePropagation algorithm provides the best results, leading to a reduction in root mean square error of about 20% compared to simple average (which in our experiments, proved superior to simple median). The CostOfDisagreement algorithm, which performed almost as well as VariancePropagation on synthetic data, yields inferior results to our average-algorithm baseline on real data. This provides an interesting cautionary example of the difference between evaluation on real-world and synthetic data.

We believe that the performance difference between synthetic and real-world data is due to a combination of factors. In synthetic data, we can easily simulate the existence of groups of students with widely different grading accuracy, such as students who are completely uninterested in the class, or who exhibit borderline spammer behavior. The situation in our peer-grading data was quite different. Students came from real classes taught at universities, and shared somewhat similar background. CrowdGrader also provided a strong incentive to behave as accurate graders, since a good portion of a student's overall grade was determined by their grading accuracy. Thus, the students in our dataset may not differ as markedly in their grading accuracy, reducing the accuracy gains that can be obtained via a reputation system, but increasing the accuracy of the resulting grades.

Overall, our results indicate how practical and efficient crowdsourcing algorithms can lead to higher-accuracy consensus grades in a wide range of real-world situations.

2 Related work

The work that is the most closely related to ours is the work of (Piech et al. 2013) on crowdsourcing algorithms for MOOC peer-grading. In that work, the authors construct statistical models of submission quality, user bias, and user precision, and use them in order to obtain more accurate esimates of submission quality. The algorithms are based on Gibbs sampling to obtain the estimates. In order to evaluate the results, the authors set aside a small fraction of submissions to receive very many evaluations, on the order of hundreds. Taking an average of this large number of submissions provides the ground truth against which the algorithms can be evaluated. This approach is feasible in MOOCs because a small fraction of the total number of submissions can still contain enough submissions to be statistically significant, and the large number of evaluations for this small fraction of submissions represents a small portion of the total evaluation work. We cannot follow this approach, as in our classroom setting we would need to select a larger fraction of submissions to have a sufficiently large sample size for our ground truth, and asking for many evaluations for such submissions would draw away too large a share of the total reviewing work. We view the idea of using stability under subsampling as evaluation metric as one of the contributions of this work.

The authors of (Piech et al. 2013) report improvements of about 30% compared to the use of the median of peer grades. The results are difficult to compare with ours, due to the different nature of the datasets. In our peer evaluation dataset, the users are university students enrolled in traditional (not on-line) classes; as such, they are likely to be more uniform in skills and in motivation than students of a MOOC class, where different motivational levels (from mildly interested or just browsing, to very determined), skills levels, and attitudes towards coursework are to be found. When plotting the accuracy data for students in our peer-grading datasets, over 70% of the students were reasonably precise, and fairly undistinguishable in their grading performance. The best comparison would be obtained by running the algorithms on the same data.[1] The algorithms of (Piech et al. 2013) and those presented here are markedly different in structure:

[1] The code for the VariancePropagation algorithm is available at `https://github.com/lucadealfaro/vancouver`.

rather than Gibbs sampling, we use a relatively straightforward graph-based iteration to compute consensus grades.

The problem of computing reliable consensus grades in peer grading has been considered in (Goldin and Asley 2011; Goldin 2012). As in our work, (Goldin and Asley 2011; Goldin 2012) model the bias of individual students, even along the multiple-dimensions of a grading rubric consisting of several categories. Differently from our approach, they do not model individual student accuracies; careless students, or vandals, would be recognized only via their overall bias, if any. The problem of combining input from multiple raters is also considered in (Johnson 1996; Rogers, Girolami, and Polajnar 2010); the work focuses on raters who evaluate hundreds of submissions each, allowing for the development of sophisticated error models for each rater.

The structure of our VariancePropagation algorithm is related to the algorithm of (Karger, Oh, and Shah 2011) for the crowdsourcing of binary labels; both algorithms are in turn inspired by the belief-propagation approach (Pearl 1988). Indeed, we arrived at VariancePropagation by using (Karger, Oh, and Shah 2011) as starting point, and asking the question of how to best adapt the idea underlying the algorithm to quantitative evaluations, rather than binary labels. Our derivation of VariancePropagation is also informed by the expectation-maximizaton technique (Dawid and Skene 1979).

Other papers have proposed models for crowdsourcing binary labels. Dawid and Skene (Dawid and Skene 1979) developed an algorithm based on expectation maximization (EM). The algorithm is based on a statistical model with observable and hidden parameters. Observable parameters model quantities of interests (true labels). Hidden parameters model probabilities of user errors. The algorithm proceeds in iterations with two steps, Expectation and Maximization. In the Maximization step, estimates for observable variables are obtained using the maximum likelihood method. Then, in the Expectation step, hidden variables are estimated as expected values. There are few drawback of algorithms based on EM technique, as pointed out in (Karger, Oh, and Shah 2011). First, the output of an algorithm depends on the initial values of model parameters; and it can vary from one algorithm run to another. Second, there is no analysis to obtain performance guarantees. Later, Quang et al. (Liu, Peng, and Ihler 2012) developed a belief propagation algorithm by considered the problem of crowdsourcing binary labels as a standard inference problem in graphical models; the work generalizes (Karger, Oh, and Shah 2011). The problem of N-ary labels (or multiple-choice questions) is considered in (Karger, Oh, and Shah 2013) and (Bachrach et al. 2012).

3 Problem Setting

We consider the problem of crowdsourcing the evaluation of the quality of a set of items. We assume that we have fixed sets S of items and U of users. Each user is assigned a subset of items to evaluate: we model the evaluation assignment via a bipartite graph $G = (T, E)$, where $T = S \cup U$, and $E \subseteq S \times U$, so that $(s, u) \in E$ iff u evaluates s. For each $(s, u) \in E$, user u provides for item s a *grade* $g_{s,u} \in [0, G_{\max}]$ in some fixed grade range $[0, G_{\max}]$. We denote by $\partial u = \{s \in S \mid (s, u) \in E\}$ the set of items evaluated by user $u \in U$, and similarly, by $\partial s = \{u \in U \mid (s, u) \in E\}$ the set of users who evaluated item $s \in S$. Without loss of generality, we assume that every item and every user participates in at least one evaluation: that is, we assume that $\partial s \neq \emptyset$ and $\partial u \neq \emptyset$ for all $s \in S$ and $u \in U$. Items and users that do not participate in evaluations can simply be removed from the input. The goal of the crowdsourcing algorithms consists in computing consensus grades $\hat{g}_s$ for each item $s \in S$ that are as close as possible to the true quality of the items.

In the peer-grading CrowdGrader dataset, the set S consists of the solutions submitted to a homework assignment, and the set U corresponds to the students in the class. In the CrowdGrader datasets, we have $|S| \leq |U|$; the size of S can be smaller than that of U if some students fail to submit a solution, or if students are allowed to work in groups on the submissions. The graphs corresponding to CrowdGrader review assignments are essentially random graphs where all nodes in S have approximately the same degree, and all the nodes in U have approximately the same degree.

4 Algorithms

The simplest algorithm for computing consensus grades consists in averaging the grades each submission has received; we refer to this algorithm as AVG. To improve on AVG, we introduce two novel crowdsourcing algorithms for the computation of consensus grades: VariancePropagation and CostOfDisagreement.

4.1 The VariancePropagation algorithm

The VariancePropagation algorithm is based on the following variance minimization principle.

Proposition 1 (minimum variance estimator) *Suppose we have available uncorrelated estimates $\hat{X}_1, \ldots, \hat{X}_n$ of a quantify x of interest, where each $\hat{X}_i$ is a random variable with average x and variance v_i, for $1 \leq i \leq n$. We can obtain an estimate of x that has minimum variance by averaging $\hat{X}_1, \ldots, \hat{X}_n$ while giving each $\hat{X}_i$ a weight proportional to $1/v_i$, for $1 \leq i \leq n$. That is, the minimum variance estimator $\hat{X}$ of x can be obtained as:*

$$\hat{X} = \frac{\sum_{i=1}^{n} \hat{X}_i / v_i}{\sum_{i=1}^{n} 1/v_i} .$$

The variance of this estimator is $\mathrm{var}(\hat{X}) = \left(\sum_{i=1}^{n} \frac{1}{v_i} \right)^{-1}$.

Proof. Given two uncorrelated estimates $\hat{X}_1, \hat{X}_2$, with variances v_1, v_2, consider their linear combination $Y = \alpha_1 \hat{X}_1 + \alpha_2 \hat{X}_2$, with $\alpha_1 + \alpha_2 = 1$. By the Bienaymé formula, the variance of Y is given by $\alpha_1^2 v_1 + (1 - \alpha_1)^2 v_2$. If we take the derivative with respect to α_1, and set it to 0, we obtain $\alpha_1 v_1 = \alpha_2 v_2$, or $\alpha_1 \propto 1/v_1$ and $\alpha_2 \propto 1/v_2$. The general case for n estimates follows similarly. ∎

This observation suggests a crowdsourcing algorithm for grades: if we could somehow measure the variance v_i of

Algorithm 1 The VariancePropagation Algorithm.

Input: $G = ((S \cup U), E)$, $\{g_{s,u}\}_{(s,u)\in E}$, $K > 0$.
Output: Consensus grades g_s for $s \in S$.

1: {Initialization}
2: **for all** $u \in U$ **do** $v_u := 1$; $b_u := 0$
3: **for** iteration $k = 1, 2, \ldots, K$ **do**
4: {Update item estimates}
5: **for all** $s \in S$ **do**
6: $g_s := \text{WAvg}(\{(g_{s,u} - b_u, f(v_u)) \mid u \in \partial s\})$
7: $v_s := \left[\sum_{u\in\partial s} 1/v_u\right]^{-1}$
8: **end for**
9: {Update user estimates}
10: **for all** $u \in U$ **do**
11: $v_u := \text{WAvg}(\{((g_{s,u} - g_s)^2, 1/v_s) \mid s \in \partial u\})$
12: $b_u := \text{Avg}(\{(g_{s,u} - g_s) \mid s \in \partial u\})$
13: **end for**
14: **end for**

each student i, we could weigh the input provided by student i in proportion to $1/v_i$. The VariancePropagation algorithm, given as Algorithm 1, proceeds in iterative fashion, using consensus grades to estimate the grading variance of each user, and using the information on user variance to compute more precise consensus grades according to the above minimum variance estimator principle. At each iteration, the algorithm updates four families of estimates, for $u \in U$ and $s \in S$:

- estimates v_u of the grading variance of user u;
- estimates b_u of the grading bias of user u;
- estimates g_s of the grade of item s;
- estimates v_s of the variance with which g_s is known.

In the algorithm, we denote by $\text{Avg}(X)$ the average of a set X of estimates, and given a set $Z = \{(x_1, w_1), \ldots, (x_n, w_n)\}$ of pairs, we denote by $\text{WAvg}(Z)$ the weighed average of $x_1, \ldots, x_n$ computed according to weights $w_1, \ldots, w_n$, or

$$\text{WAvg}((x_1, w_1), \ldots, (x_n, w_n)) = \frac{\sum_{i=1}^{n} x_n\, w_n}{\sum_{i=1}^{n} w_n}\ .$$

The algorithm not only uses the estimated variance of users to compute better grade estimates, but it also uses the estimated variance of consensus grades to compute more accurate estimates of user variance (see line 11). This is novel, with respect to expectation maximization algorithms: the intuition is that, when computing the variance of a user, we should not give much credence to consensus values that are known only with considerable uncertainty.

Variants of VariancePropagation We consider variants of the Algorithm 1, according to the following choices.

User weight regularization. Proposition 1 directly suggest taking $f(v) = 1/v$ in line 6; this is our PURE choice below. An alternative choice consists in introducing a regularization factor.

- PURE: $f(v) = 1/v$, as called for by Proposition 1;
- ATT: $f(v) = 1/(\bar{v} + v)$, where $\bar{v} = \frac{1}{2\,|U|} \sum_{u\in U} v_u$ is half the average estimated variance of users.

When PURE is used, VariancePropagation can over-estimate the accuracy of some users. This can feed a vicious circle, where these users gain growing influence over the consensus estimates, and are thus considered increasingly accurate. In ATT, the regularization factor limits the weight given to precise uses.

Debiasing. We consider versions of VariancePropagation with and without the debiasing step:

- DEBIAS: the debiasing step 12 is performed;
- NODEB: the debiasing step 12 is not performed.

Debiasing user-provided grades is a seemingly obvious step for obtaining more precise estimates of consensus grades. Indeed, (Piech et al. 2013) attributes much of the increase in precision in their computed consensus grades to debiasing, while also stating that no usable variance estimates could be obtained in their setting. In our evaluation over peer-grading data, debiasing proves useful, but to a lesser extent, perhaps because VariancePropagation already derives benefits from the modeling of user variance, which is not considered in (Piech et al. 2013).

Discarding extremal grades. We consider two choices for the aggregation function WAvg in line 11. The first choice consists in using weighted average as aggregator. The second choice consists in using, in place of WAvg, a version MWAvg of weighted average that discards the highest and the lowest grade before computing the weighted average. This is a common technique for outlier rejection, used e.g. to score Olympic competitions. The drawback lies in the fact that many items do not have outlier grades, and unnecessarily discarding some grades reduces the precision of the resulting estimate.

4.2 The CostOfDisagreement Algorithm

The CostOfDisagreement algorithm is inspired by the empirical observation that people are often better judges of relative quality than of absolute quality. Thus, the CostOfDisagreement algorithm pays attention not to the absolute grades provided by the users, but to the *grade differences* among items graded by the same user. The second idea used in the CostOfDisagreement algorithm consists in keeping track of the total cost of disagreement with each user, defined below, and to give less weight to the opinion of users who have a higher total disagreement with the consensus grades, as these must be the less reliable users. More precisely, assume that a user u provides grades $g_{s,u}$ and $g_{t,u}$ for two items s, t, with $g_{s,u} \geq g_{s,t}$. The user $u \in U$ is thus indicating a perceived difference of $\delta_{u:s,t} = g_{s,u} - g_{t,u} \geq 0$ in the grades of $s, t \in S$. If the algorithm chooses the estimated grades x_s, x_t for s and t, the user disagrees with the algorithm by an amount

$$d_{u:s,t}(\mathbf{x}) = (x_s - x_t) - \delta_{u:s,t} = (x_s - x_t) - (g_{s,u} - g_{t,u}) \quad (1)$$

where $\mathbf{x} = [x_s]_{s \in S}$ is the vector of all grades. We translate this disagreement into a *cost of disagreement* with user u over s, t, given by $g(d_{u:s,t}(\mathbf{x}))$, where g is a convex function. To compute the total cost of disagreement, we consider the set of comparisons

$$\begin{aligned} \mathcal{C} = &\{(u,s,t) \mid (s,u) \in E, (t,u) \in E, g_{s,u} - g_{t,u} > 0\} \\ &\cup \{(u,s,t) \mid (s,u) \in E, (t,u) \in E, s \prec t, g_{s,u} = g_{t,u}\} , \end{aligned} \tag{2}$$

where $\prec$ is an arbitrary strict ordering over items. Thus, in the definition of $\mathcal{C}$, we orient the comparisons, so that $d_{u:s,t} \geq 0$ for all $(u,s,t) \in \mathcal{C}$. This ensures that, for all $s, t \in S$ and $u \in U$, at most one of (u,s,t) or (u,t,s) is included in $\mathcal{C}$, so that each user comparison of two items is counted once only. The orientation requiring $\delta_{u:s,t} \geq 0$ allows the use of cost functions g that are not symmetrical with respect to 0. We also let $\mathcal{C}_u = \{(u',s,t) \mid (u',s,t) \in \mathcal{C}, u = u'\}$ be the subset of comparisons involving user u.

The CostOfDisagreement algorithm, presented as Algorithm 2, first assigns unit reputation to each user, that is, $r_u = 1$ for all $u \in U$. The algorithm then proceeds in a series of iteration. At each iteration, the algorithm computes via convex optimization the grade vector $\mathbf{x}$ that minimizes the total cost of disagreement, weighed by user reputations, given by:

$$\sum_{(u,t,s) \in \mathcal{C}} r_u \, g(d_{u:s,t}(\mathbf{x})) \, . \tag{3}$$

The algorithm then updates the reputation r_u of each user $u \in U$ via

$$r_u := \frac{1}{1 + \sum_{(u,t,s) \in \mathcal{C}_u} g(d_{u:s,t}(\mathbf{x}))}$$

so that users with higher disagreement with the consensus grades will receive lower reputation. Convergence is quite fast: 20 iterations are sufficient in our experience. Since the algorithm is based on grade differences, the estimated grades computed are invariant with respect to translation. The algorithm computes the consensus grades by optimally aligning via a linear transformation the estimated grades with the grades provided by the users.

The objective function (3) is convex with respect to $\mathbf{x}$ because we have $r_u > 0$ for all $u \in U$, and $g(d_{u:s,t}(\mathbf{x}))$ is convex as a composition of linear and convex functions. Thus, we can perform Step 7 of Algorithm 2 via convex optimization methods. In our implementation, we use gradient descent using Newton's method, which yields the global minimum in very few iterations, typically 4 or 5. We have experimented with several alternative cost-of-disagreement

Algorithm 2 CostOfDisagreement

Input: $G = ((S \cup U), E)$, $\{g_{s,u}\}_{(s,u) \in E}$, $K > 0$.
Output: Consensus grades g_s for $s \in S$.

1: {Initialization}
2: Compute $\mathcal{C}$ according to (2)
3: **for all** $u \in U$ **do** $r_u := 1$
4: **for all** $(u,s,t) \in \mathcal{C}$ **do** $\delta(u:s,t) := g_{s,u} - g_{t,u}$
5: {Computes estimated grades}
6: **for** iteration $k = 1, 2, \ldots, K$ **do**
7: $\quad \mathbf{x} := \arg\min \sum_{(u,s,t) \in \mathcal{C}} r_u g(x_s - x_t - \delta_{u:s,t})$
8: $\quad$ **for all** $u \in U$ **do**
9: $\quad\quad r_u := [1 + \sum_{(u,s,t) \in \mathcal{C}_u} g(x_s - x_t - \delta_{u:s,t})]^{-1}$
10: $\quad$ **end for**
11: **end for**
12: {Optimally aligns estimated grades}
13: **for all** $s \in S$ **do** $\bar{g}_s := (\sum_{u \in \partial s} g_{s,u})/|\partial s|$
14: $a^*, b^* = \arg\min(\sum_{s \in S} a x_s^K + b - \bar{g}_s)^2$
15: **for all** $s \in S$ **do** $g_s = a^* x_s + b^*$

functions g, specifically:

$$g(x) = x^2 \tag{4}$$

$$g(x) = \frac{x^2}{1 + |x|} \tag{5}$$

$$g(x) = \begin{cases} \frac{x^2}{1+|x|} & \text{if } x \geq 0 \\ x^2 & \text{if } x < 0 \end{cases} \tag{6}$$

$$g(x) = \begin{cases} x^2 & \text{if } x \geq 0 \\ \frac{x^2}{1+|x|} & \text{if } x < 0 \end{cases} \tag{7}$$

Choice (4) yields aggregation by average. Choice (5) yields an aggregation that behaves like average for close data, and as median for far away data: essentially, this combines median's ability to disregard outliers with average's ability to combine accurate data. Finally, choices (6) and (7) constitute asymmetrical aggregators. Let $s, t \in S$ be two items evaluated by a user $u \in U$ that preferred s to t, i.e., $g_{s,u} > g_{t,u}$. Choice (6) yields an aggregator where the cost of believing that s is markedly better than u stated is low, whereas the cost of believing that s is markedly worse than stated is high. In particular, in this choice, believing that user u has mis-ordered the items is high. Choice (7) yields an aggregator where the cost of believing that s is markedly even better than u stated is high, while the cost of believing that s is markedly worse than stated by u, including worse than t, is low. These asymmetrical aggregators are useful in allowing different partial rankings being easily merged into a single ranking.

5 Results on Synthetic Data

In our evaluation on synthetic data, we construct assignments involving 50 users and 50 submissions, with each user reviewing 6 items; these numbers are similar to those occurring in the actual peer-grading assignments. The true quality q_s of each item s we assumed was normal-distributed

		CostOfDisagreement , $g(x)$ equals			
	AVG	x^2	(5)	(6)	(7)
$k=1$	0.285	0.066	0.047	0.062	0.048
$k=2$	0.68	0.247	0.175	0.277	0.164
$k=3$	1.145	0.536	0.46	0.582	0.404

		VariancePropagation			
		NODEB		DEBIAS	
	AVG	PURE	ATT	PURE	ATT
$k=1$	0.285	0.018	0.108	0.022	0.147
$k=2$	0.68	0.121	0.42	0.148	0.562
$k=3$	1.145	0.432	0.717	0.548	0.785

Table 1: Root Mean Square Error of the AVG, CostOfDisagreement, and VariancePropagation algorithms on synthetic data with **unbiased** users. Values are computed for different shape factors k of the Gamma distribution which governs the user error model.

		CostOfDisagreement , $g(x)$ equals			
	AVG	x^2	(5)	(6)	(7)
$k=1$	0.337	0.072	0.052	0.067	0.052
$k=2$	0.695	0.239	0.168	0.233	0.163
$k=3$	1.261	0.577	0.474	0.643	0.417

		VariancePropagation			
		NODEB		DEBIAS	
	AVG	PURE	ATT	PURE	ATT
$k=1$	0.337	0.133	0.198	0.024	0.177
$k=2$	0.695	0.285	0.248	0.153	0.216
$k=3$	1.261	0.606	0.793	0.581	0.854

Table 2: Root Mean Square Error of the AVG, CostOfDisagreement, and VariancePropagation algorithms on synthetic data **with user bias**. Values are computed for different shape factors k of the Gamma distribution which governs the user error model.

with standard deviation 1; this assumption is non-critical, as this parameter does not affect the evaluation per se. We let the grade $g_{s,u}$ assigned by user u to item s be equal to $q_s + \Delta_{s,u}$, where q_s is the true quality of s, and $\Delta_{s,u}$ is normally-distributed with mean b_u and variance v_u. We assumed that the variances $\{v_u\}_{u \in U}$ of the users were distributed according to a Gamma distribution with scale 0.4, and shape factors $k = 1, 2, 3$. We considered two cases of user bias b_u. In the first case, every user is unbiased, i.e. $b_u = 0$ for $u \in U$. In the second case, every user has bias distributed according to a normal distribution with mean 0 and standard deviation 0.4. The code used for the VariancePropagation algorithm can be found at `https://github.com/lucadealfaro/vancouver`.

The results are summarized in Tables 1, 2. For each shape factor k, and each of the algorithm variants, we report the Root Mean Squared Error between the ground truth q_i and the consensus grades g_i. Each entry in the table is the average over 100 runs. Table 1 compares the algorithms performance on simulated data with unbiased user. The VariancePropagation algorithm without the debias step (line 12 of Algorithms 1) and PURE option shows the smallest error; it reduces the the error by a factor of 15, 5 and 3 compared with simple average AVG, according to the shape factors $k = 1, 2, 3$ of the Gamma distribution of user variance. Table 2 compares the algorithm on simulated data with user bias. The VariancePropagation algorithm with the debias step has uniformly smaller error than the algorithm without debiasing; a variant of VariancePropagation with PURE choice of function $f(\cdot)$ has the smallest error for shape factors 1 and 2.

Figures 1a, 1b, and 1c illustrate the ability of the VariancePropagation and CostOfDisagreement algorithms to identify the less precise users, and to listen less to their input.

6 Results on Peer-Grading Data

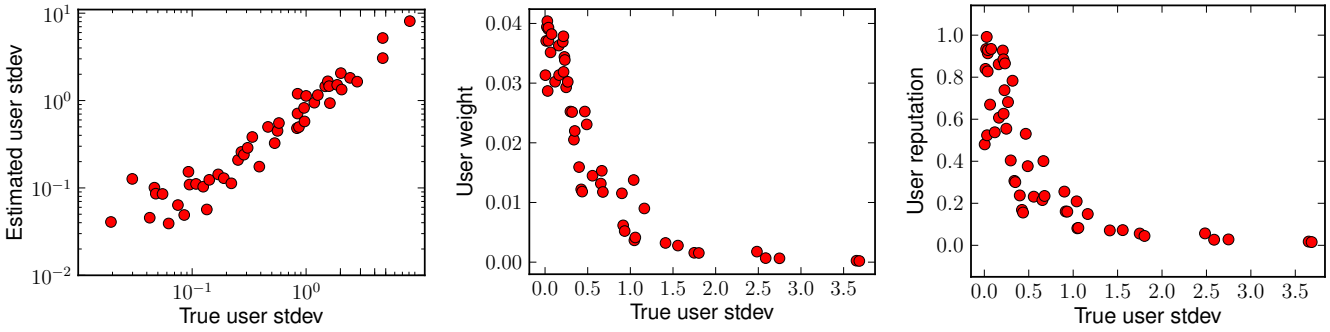
The chief difficulty in evaluating the performance of algorithms for crowd-sourcing evaluations on real-world data consists in the lack of ground truth, and this applies to the peer grading data we have available. Initially, we thought we could construct a sufficient ground truth by manually grading a fraction of the the assignments. This approach was not practical for two reasons. First, we were able to grade only the few assignments that fell in our domain of expertise. Second, upon comparing our grades with the grades assigned by students participating in the peer grading, we realized that our grades were generally no more precise than those of the students, as we often missed flaws that some of the students spotted.

In the MOOCs considered in (Piech et al. 2013), a reasonable approximation of a ground truth was constructed by selecting a small fraction of the submissions, having them evaluated by hundreds of users, and using the average evaluations obtained as ground truth. While this approach is feasible in MOOCs involving many thousands of submissions and participants, it was not feasible in our setting, where the class size varied from 50 to 300 students. Singling out even only a dozen submissions to receive 20 reviews each would have called for 240 reviews, or almost 50% of the reviewing effort in a class with 100 submissions and 5 reviews per submission (typical values for our peer-grading dataset). Using 50% of the reviewing effort purely for research reasons would not have been acceptable to the users of CrowdGrader.

To enable a meaningful study of algorithm accuracy, we introduce the notion of the *stability under subsampling* of algorithms, and we provide theoretical and experimental justifications for its use as an estimator of algorithm accuracy.

6.1 Stability under subsampling

The intuition behind stability under subsampling as a criterion for algorithm precision is as follows. If we had a very large number of grades for each item, the use of sophisticated crowdsourcing algorithms would be unnecessary: taking the average of the grades received by each item would suffice. The goal of the crowdsourcing algorithms consid-



(a) Estimated vs. actual user standard deviation according to VariancePropagation with NODEB, PURE.

(b) User weight $f(v_u)$ vs. actual user standard deviation according to VariancePropagation with NODEB, PURE.

(c) User reputation vs. actual user standard deviation according to CostOfDisagreement with loss function (7).

Figure 1: Ability of VariancePropagation and CostOfDisagreement algorithms to characterize user precision, and give less weight to the input from less precise users. In all figures, the shape factor is $k = 2$, and users have zero bias.

ered in this paper is to provide precise estimates even when the number of grades per item is modest. Thus, a reasonable approach to estimate the quality of a crowdsourcing algorithm consists in asking: *If we delete some information, how closely do the estimates obtained approximate the estimates obtained under full information?*

To describe formally our notion of stability under subsampling, we denote by h an algorithm for computing consensus grades. The input of h is the graph $G = (S \cup U, E)$ of reviews, where each $(s, u) \in E$ is labeled with the grade $g_{s,u}$ assigned by u to s. The output of h is a vector of grades $\mathbf{x} = h(G)$, where $\mathbf{x} = [x_s]_{s \in S}$ and x_s is a consensus grade for item s. The process for computing the stability under subsampling of algorithm h is given in Algorithm 3. The process computes the average stability, measured over K iterations of a subsampling procedure. In each iteration, we randomly select a subset $S' \subset S$ consisting of $\alpha \cdot |S|$ items. We then construct two graphs G_1 and G_2 from G: each of these graphs is obtained by deleting one review, selected at random, among the reviews of items in S'.

Once the graphs G_1 and G_2 have been obtained, we compute the grades $\mathbf{x}^{(1)} = h(G_1)$ and $\mathbf{x}^{(2)} = h(G_2)$, and we compute the average square difference δ_k between grades $[x_s^{(1)}]_{s \in S'}$ and $[x_s^{(2)}]_{s \in S'}$ (line 5). The *stability* of the algorithm is the square root of the average δ_k over all iterations $k = 1, \ldots, K$. We perform the subsampling only at a subset $S' \subset S$ of items, rather than for all items, so that the algorithms can use the full information at the items $S \setminus S'$ to infer the accuracy of the users, and use that information for computing optimal consensus values for the items in S'. Subsampling from all items would reduce the ability of the algorithms to infer user precision, yielding a less precise estimate of their true precision. We sub-sample by removing just one of the reviews, in order to alter as little as possible the original graph, as crowd-sourcing algorithm performance is sensitive to node degree distribution. We provide the following justification for the connection between stability under subsampling, and algorithm error. For an algorithm h and a graph G, indicate by $h_s(G)$ the grade for item s in a graph G, and let g_s be the true grade of s, as usual.

Algorithm 3 Instability under subsampling.

Input: $G = ((S \cup U), E)$, $\{g_{s,u}\}_{(s,u) \in E}$, and algorithm h for computing consensus grades, a subsampling method R, a number of iterations $K > 0$, and a subsampling fraction $0 < \alpha < 1$.

Output: instability under subsampling of algorithm h.

1: **for** $k = 1, 2, \ldots, K$ **do**
2: Randomly choose S' with $|S'| = \lfloor \alpha \cdot |S| \rfloor$ items.
3: Construct G_1 and G_2 from G by independently choosing, and removing, one review for each item in S
4: $\mathbf{x}^{(1)} := h(G_1)$, $\mathbf{x}^{(2)} := h(G_2)$
5: $\delta_k = \sqrt{\left(\sum_{s \in S'} (\mathbf{x}_s^{(1)} - \mathbf{x}_s^{(2)})^2\right) / |S'|}$
6: **end for**
7: Output $\sum_k \delta_k / K$

Proposition 2 *Consider the case where the number of users U and reviews E in graph G is large enough that vectors $h(G_1)$ and $h(G_2)$ can be assumed to be independent random vectors. If algorithm h computes unbiased grades, i.e., if $\mathrm{E}\left[h_s(G) - g_s\right]^2 = 0$, then the instability under subsampling is proportional to algorithm error in the subsampled graphs.*

Proof. We have

$$\mathrm{E}\left[\left(h_s(G_1) - h_s(G_2)\right)^2\right] = \mathrm{E}\left[\left(h_s(G_1) - g_s + g_s - h_s(G_2)\right)^2\right]$$
$$= \mathrm{E}\Big[\left(h_s(G_1) - g_s\right)^2 + \left(h_s(G_2) - g_s\right)^2 - 2\left(h_s(G_1) - g_s\right)\left(h(G_2) - g_s\right)\Big]$$
$$= 2\mathrm{E}[\left(h_s(G_1) - g_s\right)^2] - 2\left(\mathrm{E}\left[h_s(G_1)\right] - g_s\right)^2.$$

Thus, if squared bias is zero, we have that the contribution of s to the instability under subsampling $\mathrm{E}[\left(h_s(G_1) - h_s(G_2)\right)^2]$ is proportional to the mean square error at s. ∎

Gamma shape factor	Correlation
$k = 1$	0.99
$k = 2$	0.91
$k = 3$	0.9

Table 3: Correlation between instability under subsampling $[\delta_h]_{h\in\mathcal{A}}$ and error $[e_h]_{h\in\mathcal{A}}$.

Algorithm		Code
AVG		0
CostOfDisagreement	$g(x) = x^2$	1
CostOfDisagreement	$g(x)$ as in (5)	2
CostOfDisagreement	$g(x)$ as in (6)	3
CostOfDisagreement	$g(x)$ as in (7)	4
VariancePropagation	NODEB WAvg	5
VariancePropagation	NODEB MWAvg	6
VariancePropagation	DEBIAS WAvg	7
VariancePropagation	DEBIAS MWAvg	8

Table 4: Algorithm codes used as captions in Figure 2.

The connection between stability under subsampling and algorithm precision can also be verified experimentally under synthetic data. In Table 3 we give the correlation between the measured instability under subsampling $[\delta_h]_{h\in\mathcal{A}}$, and the error $[e_h]_{h\in\mathcal{A}}$, for the algorithms belonging to the set $\mathcal{A}$ of algorithms considered in this paper (AVG, and the various versions of VariancePropagation and CostOfDisagreement). The synthetic assignment model is as described in Section 5, and we averaged results on 11 synthetically generated assignments without user bias. The correlation is not 1 chiefly because the error in the subsampled graphs is greater (due to less data being available) than the error in the complete graph.

6.2 Results

In order to make data comparable across assignments, we normalized the grading scale of all assignments to $[0, 10]$. The results are reported in Table 5, where we give the instability of the algorithms relative to the AVG baseline. The table presents the data by grouping assignments into bins, according to the average number of reviews per item. We combine the values for different assignments in the same bin via geometric average. The data is presented graphically for the best-performing subset of the algorithms in Figure 2.

The VariancePropagation algorithm with DEBIAS, WAvg and ATT options performs best. In particular, debiasing user input seems to be important, a finding that confirms the results of (Piech et al. 2013). While multiple versions of VariancePropagation improve on the AVG baseline, we also note how CostOfDisagreement fares generally worse than the baseline. Given how superior CostOfDisagreement is to AVG on synthetic data (see Section 5), this offers a cautionary example of the dangers of relying too much on synthetic evaluations, where the user models may not be a good match for the actual behavior of students using a peer-grading tool.

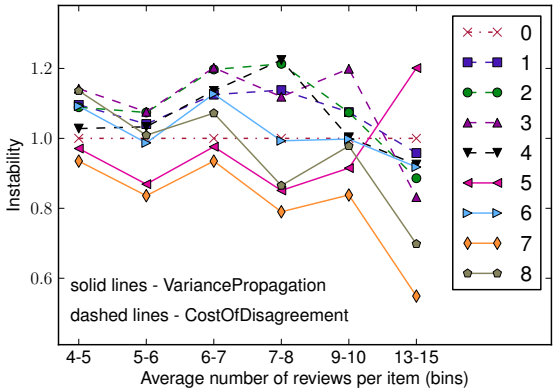


Figure 2: Instability under subsampling of crowdsourcing algorithms, relative to AVG.

Bins: average number of reviews per item	Assignments statistics Number of assignments	submissions	users
$[4, 5)$	4	219	207
$[5, 6)$	6	650	681
$[6, 7)$	4	373	473
$[7, 8)$	3	374	504
$[9, 10)$	2	103	190
$[13, 15)$	3	78	284
$[4, 15)$	22	1797	2339

Table 6: Statistics for the assignments used in Table 5.

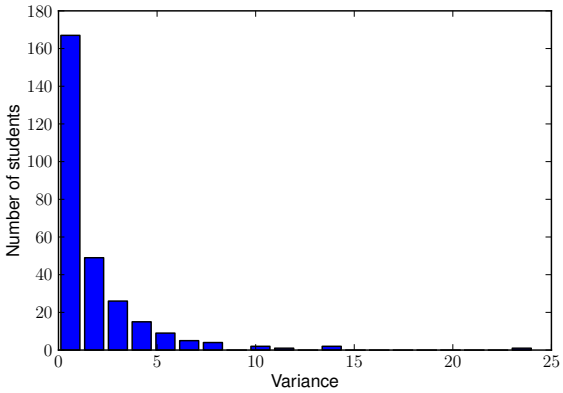


Figure 3: Histogram of reviewer variance in an assignment with grade range $[0, 10]$.

Bins: average number of reviews per item	AVG	CostOfDisagreement , $g(x)$ equals				VariancePropagation with ATT option			
						NODEB		DEBIAS	
		x^2	(5)	(6)	(7)	WAvg	MWAvg	WAvg	MWAvg
$[4, 5)$	1	1.096	1.089	1.142	1.028	0.971	1.092	0.935	1.136
$[5, 6)$	1	1.041	1.074	1.076	1.033	0.869	0.987	0.836	1.009
$[6, 7)$	1	1.125	1.197	1.202	1.136	0.976	1.127	0.935	1.072
$[7, 8)$	1	1.138	1.213	1.119	1.224	0.851	0.993	0.79	0.865
$[9, 10)$	1	1.075	1.074	1.199	1.003	0.915	0.998	0.838	0.978
$[13, 15)$	1	0.958	0.886	0.832	0.925	1.201	0.918	0.549	0.698
$[4, 15)$	1	1.07	1.088	1.088	1.056	0.948	1.021	0.816	0.968

Table 5: Instability under subsampling of algorithms, relative to the instability of AVG as baseline. Each row is obtained by computing the geometric average of the results for assignments with average number of reviews per item as indicated in the bins. Statistics for the assignments are provided in Table 6.

7 Conclusions

We introduced two alternative algorithms for crowdsourcing estimates of item quality, VariancePropagation and CostOfDisagreement. On synthetic data, both algorithms perform better than the Avg aggregator, cutting the average error by factors of 3 or more, depending on the user model chosen. On real-world data obtained from classroom peer-grading, both algorithms perform better than the Avg aggregator. The edge goes to a version of VariancePropagation that characterizes users both in terms of their grading variance, and in terms of their overall positive or negative bias.

On peer-grading data, the average reduction in error, compared to Avg, is about 20%. This reduction is lower than that observed in the MOOC experiment of (Piech et al. 2013). We believe that this can be explained at least in part by considering the different natures of the datasets. Our peer-grading data consisted of students enrolled in university classes that were taught in-person by instructors. In such classes, students have more uniform skills and background than in MOOCs, which mix very proficient and motivated students with students that are just superficially curious about the subject. Furthermore, CrowdGrader provided a strong incentive to students to provide precise evaluations, as part of the overall grade of each student was tied to the student's precision as grader. These factors likely made the students involved in our peer-grading datasets more uniform in accuracy than the users in the MOOC datasets of (Piech et al. 2013). The accuracy distribution for students of one class is given in Figure 3; this (typical) distribution confirms the underlying precision of the students, and the relative rarity of wholly imprecise users. Crowdsourcing algorithms based on reputation systems leverage the non-uniform accuracy of reviewers: after all, if all students were equally accurate, then Avg would be the optimal aggregator. The more uniform accuracy of the students involved in peer grading may have meant less scope for improvement for reputation-based algorithms.

Acknowledgments

References

[Bachrach et al. 2012] Bachrach, T.; Minka, J.; Guiver, J.; and Graepel, T. 2012. How to grade a test without knowing the answers — A bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *29th Annual International Conference on Machine Learning (ICML 2012)*.

[Dawid and Skene 1979] Dawid, A. P., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics* 20–28.

[de Alfaro and Shavlovsky 2013] de Alfaro, L., and Shavlovsky, M. 2013. Crowdgrader: A tool for crowdsourcing the evaluation of homework assignments. *SIGCSE 2013*.

[Dempster et al. 1977] Dempster, A. P.; Laird, N. M.; Rubin, D. B.; et al. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society* 39(1):1–38.

[Goldin and Asley 2011] Goldin, I. M., and Asley, K. D. 2011. Peering inside peer review with bayesian models. In *Proceedings of the 15th International Conference on Artificial Intelligence in Education, AIED 2011*, 90–97. Springer-Verlag.

[Goldin 2012] Goldin, I. M. 2012. Accounting for peer reviewer bias with bayesian models. In *Proceedings of the Workshop on Intelligent Support for Learning Groups at the 11th International Conference on Intelligent Tutoring Systems*.

[Johnson 1996] Johnson, V. E. 1996. On bayesian analysis of multi-rater ordinal data: An application to automated essay grading. *Journal of the Americal Statistical Association* 91:42–51.

[Karger, Oh, and Shah 2011] Karger, D. R.; Oh, S.; and Shah, D. 2011. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 1953–1961.

[Karger, Oh, and Shah 2013] Karger, D. R.; Oh, S.; and Shah, D. 2013. Efficient crowdsourcing for multi-class labeling. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, 81–92. ACM.

[Liu, Peng, and Ihler 2012] Liu, Q.; Peng, J.; and Ihler, A. T. 2012. Variational inference for crowdsourcing. In *NIPS*, 701–709.

[Pearl 1988] Pearl, J. 1988. *Probabilistic reasoning in in-*

telligent systems: networks of plausible inference. Morgan Kaufmann.

[Piech et al. 2013] Piech, C.; Huang, J.; Chen, Z.; Do, C.; Ng, A.; and Koller, D. 2013. Tuned models of peer assessment in moocs. *arXiv preprint arXiv:1307.2579*.

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

[Rogers, Girolami, and Polajnar 2010] Rogers, S.; Girolami, M.; and Polajnar, T. 2010. Semi-parametric analysis of multi-rater data. *Statistics and Computing* 20(3):317–334.