Fast Dawid-Skene: A Fast Vote Aggregation Scheme for Sentiment Classification

Vaibhav B Sinha, Sukrut Rao, Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad
Telangana, India
cs15btech11034@iith.ac.in, cs15btech11036@iith.ac.in, vineethnb@iith.ac.in

ABSTRACT

Many real world problems can now be effectively solved using supervised machine learning. A major roadblock is often the lack of an adequate quantity of labeled data for training. A possible solution is to assign the task of labeling data to a crowd, and then infer the true label using aggregation methods. A well-known approach for aggregation is the Dawid-Skene (DS) algorithm, which is based on the principle of Expectation-Maximization (EM). We propose a new simple, yet effective, EM-based algorithm, which can be interpreted as a 'hard' version of DS, that allows much faster convergence while maintaining similar accuracy in aggregation. We show the use of this algorithm as a quick and effective technique for online, real-time sentiment annotation. We also prove that our algorithm converges to the estimated labels at a linear rate. Our experiments on standard datasets show a significant speedup in time taken for aggregation - upto ∼8x over Dawid-Skene and ∼6x over other fast EM methods, at competitive accuracy performance. The code for the implementation of the algorithms can be found at https://github.com/GoodDeeds/Fast-Dawid-Skene.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing**; • **Computing methodologies** → **Machine learning**; **Supervised learning**; **Online learning settings**; • **Information systems** → *Information systems applications*;

KEYWORDS

crowdsourcing, vote aggregation, expectation maximization, supervised learning

ACM Reference Format:
Vaibhav B Sinha, Sukrut Rao, Vineeth N Balasubramanian. 2018. Fast Dawid-Skene: A Fast Vote Aggregation Scheme for Sentiment Classification. In *Proceedings of 7th KDD Workshop on Issues of Sentiment Discovery and Opinion Mining (KDD WISDOM 2018).* ACM, New York, NY, USA, 8 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDD WISDOM 2018, August 2018, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Supervised learning has been highly effective in solving challenging tasks in sentiment analysis over the last few years. However, the success of supervised learning for the domain in recent years has been premised on the availability of large amounts of data to effectively train models. Obtaining a large labeled dataset is time-consuming, expensive, and sometimes infeasible; and this has often been the bottleneck in translating the success of machine learning models to newer problems in the domain.

An approach that has been used to solve this problem is to crowdsource the annotation of data, and then aggregate the crowdsourced labels to obtain ground truths. Online platforms such as Amazon Mechanical Turk and CrowdFlower provide a friendly interface where data can be uploaded, and workers can annotate labels in return for a small payment. With the ever-growing need for large labeled datasets and the prohibitive costs of seeking experts to label large datasets, crowdsourcing has been used as a viable option for a variety of tasks, including sentiment scoring [3], opinion mining [39], general text processing [35], taxonomy creation [2], or domain-specific problems, such as in the biomedical field [1, 13], among many others.

In recent times, there is a growing need for a fast and real-time solution for judging the sentiment of various kinds of data, such as speech, text articles, and social media posts. Given the ubiquitous use of the internet and social media today, and the wide reach of any information disseminated on these platforms, it is critical to have a efficient vetting process to ensure prevention of the usage of these platforms for anti-social and malicious activities. Sentiment data is one such parameter that could be used to identify potentially harmful content. A very useful source for identifying harmful content is other users of these internet services, that report such content to the service administrators. Often, these services are set up such that on receiving such a flag, they ask other users interacting with the same content to classify whether the content is harmful or not. Then, based on these votes, a final decision can be made, without the need for any human intervention. Some such works include: crowdsourcing the sentiment associated with words [21], crowdsourcing sentiment scoring for online media [3], crowdsourcing the classification of words to be used as a part of lexicon for sentiment analysis [14], crowdsourcing sentiment judgment for video review [10], crowdsourcing for commodity review [39], and crowdsourcing for the production of word level annotation for opinion mining tasks [32]. However, with millions of users creating and adding new content every second, it is necessary that this decision be quick, so as to keep up with and effectively address all

flags being raised. This indicates a need for fast vote aggregation schemes that can provide results for a stream of data in real time.

The use of crowdsourced annotations requires a check on the reliability of the workers and the accuracy of the annotations. While the platforms provide basic quality checks, it is still possible for workers to provide incorrect labels due to misunderstanding, ambiguity in the data, carelessness, lack of domain knowledge, or malicious intent. This can be countered by obtaining labels for the same question from a large number of annotators, and then aggregating their responses using an appropriate scheme. A simple approach is to use majority voting, where the answer which the majority of annotators choose is taken to be the true label, and is often effective. However, many other methods have been proposed that perform significantly better than majority voting, and these methods are summarized further in Section 2.

Despite the various recent methods proposed, one of the most popular, robust and oft-used method to date for aggregating annotations is the Dawid-Skene algorithm, proposed by [6], based on the Expectation Maximization (EM) algorithm. This method uses the M-step to compute error rates, which are the probabilities of a worker providing an incorrect class label to a question with a given true label, and the class marginals, which are the probabilities of a randomly selected question to have a particular true label. These are then used to update the proposed set of true labels in the E-step, and the process continues till the algorithm converges on a proposed set of true labels (further described in Section 3.3).

In this work, we propose a new simple, yet effective, EM-based algorithm for aggregation of crowdsourced responses. Although formulated differently, the proposed algorithm can be interpreted as a 'hard' version of Dawid-Skene (DS) [6], similar to Classification EM [5] being a hard version of the original EM. The proposed method converges upto 7.84x faster than DS, while maintaining similar accuracy. We also propose a hybrid approach, a combination of our algorithm with the Dawid-Skene algorithm, that combines the high rate of convergence of our algorithm and the better likelihood estimation of the Dawid-Skene algorithm as part of this work.

2 RELATED WORK

The Expectation-Maximization algorithm for maximizing likelihood was first formalized by [8]. Soon after, Dawid and Skene [6] proposed an EM-based algorithm for estimating maximum likelihood of observer error rates, which became very popular for crowdsourced aggregation and is still considered by many as a baseline for performance. Many researchers, to this day, have worked on analyzing and extending the Dawid-Skene methodology (henceforth, called DS), of which we summarize the more recent efforts below. The work on crowdsourced data aggregation have not been confined only for sentiment analysis or opinion mining tasks, instead most of the methods are generic and can easily used for sentiment analysis and opinion mining tasks.

A new model, GLAD, was proposed in [38], that could simultaneously infer the true label, the expertise of the worker, and the difficulty of the problem, and use this to improve on the labeling scheme. [27] improved upon DS by jointly learning the classifier while aggregating the crowdsourced labels. However, the efforts of [38] were restricted to binary choice settings; and in the case of [27], they focused on classification performance, which is however not the focus of this work.

[15] presented improvements over DS to recover from biases in labels provided by the crowd, such as cases where a worker always provides a higher label than the true label when labels are ordinal. More recently, [20] analyzed and characterized the tradeoff between the cost of obtaining labels from a large group of people per data point, and the improved accuracy on doing so, as well as the differences in adaptive vs non-adaptive DS schemes.

In addition to these efforts, there has also been a renewed interest in recent years to understand the rates of convergence of the Dawid-Skene method. [11] obtained the convergence rates of a projected EM algorithm under the homogeneous DS model, which however is a constrained version of the general DS model. [40] proposed a two-stage algorithm which uses spectral methods to offset the limitations of DS to achieve near-optimal rate convergence. [33] recently proposed a permutation-based generalization of the DS model, and derived optimal rates of convergence for these models. However, none of these efforts have explicitly focused on increasing the speed of convergence, or making Dawid-Skene more efficient in practice. The work in [22] is the closest in this regard, where they proposed an EM-based Iterative Weighted Majority Voting (IWMV) algorithm which experimentally leads to fast convergence. We use this method for comparison in our experiments.

In addition to methods based on Dawid-Skene, other methods for vote aggregation have been developed, such as using Gaussian processes [30] and online learning methods [37]. The scope of the problem addressed by Dawid-Skene has also been broadened, to allow cases such as when a data point may have multiple true labels [9]. (In this work, we show how our method can be extended to this setting too.) For ensuring reliability of the aggregated label, a common approach is to use a large number of annotators, which may however increase the cost. To mitigate this, work has also been done to intelligently assign questions to particular annotators [18], reduce the number of labels needed for the same accuracy [37], consider the biases in annotators [36] and so on. Recent work on vote aggregation also includes deep learning-based approaches, such as [1, 12, 29]. A survey of many earlier methods related to vote aggregation can be found in the work of [26] and [34]. Moreover, a benchmark collection of methods and datasets for vote aggregation is defined in [34], which we use for evaluating the performance of our method.

While many new methods have been developed, the DS algorithm still remains relevant as being one of the most robust techniques, and is used as a baseline for nearly every new method. Inspired by [5], our work proposes a simple EM-based algorithm for vote aggregation, that provides a similar performance as Dawid-Skene but with a much faster convergence rate. We now describe our method.

3 PROPOSED ALGORITHM

We propose an Expectation-Maximization (EM) based algorithm for efficient vote aggregation. The E-step estimates the dataset annotation based on the current parameters, and the M-step estimates the parameters which maximize the likelihood of the dataset. Starting from a set of initial estimates, the algorithm alternates between the

M-step and the E-step till the estimates converge. Although formulated using a different approach to the aggregation problem, we call our algorithm Fast Dawid-Skene (FDS), because of its similarity to the DS algorithm (described in Section 3.3).

3.1 Preliminaries

For convenience, we use the analogy of a question-answer setting to model the crowdsourcing of labels. The data shown to the crowd is viewed as a question, and the possible labels as choices of answers from the crowd worker/participant. Let the questions (data points, problems) that need to be answered be $q = \{1, 2, 3, \ldots, Q\}$ and the annotators (participants, workers) labeling them be $a = \{1, 2, 3, \ldots, A\}$. The task requires the participants to label each question by selecting one of the predefined set of choices (options), $c = \{1, 2, 3, \ldots, C\}$, which has the same length across all questions. A participant is said to answer a given question when s/he chooses an option as the answer for that question. A participant need not answer all the questions, and in fact, for a large pool of questions, it is reasonable to assume that a participant might be invited to answer only a small subset of all the questions. Each question is assumed to be answered by at least one participant (ideally, more). We also assume that the choice selected by a participant for a question is independent of the choice selected by any other participant. This assumption holds for real-world applications that use contemporary crowdsourcing methods, where participants generally do not know each other, and are often physically and geographically separated, and thus do not influence each other. Besides, while answering a question, the participants have no knowledge of the choices chosen by previous participants in these settings.

3.2 The Fast Dawid-Skene Algorithm

We now derive the proposed Fast Dawid-Skene (FDS) algorithm under the assumption that each question has only one correct choice, and that a participant can select only one choice for each question. (In Section 6, we show how our method can be extended to relax this assumption.) Our goal is to aggregate the choices of the crowd for a question and to approximate the correct choice. Consider the question q. Let the K participants that answered this question be $\{q_1, q_2, \ldots, q_K\}$. The value of K may vary for different questions. Let the choices chosen by these K participants for question q be $\{c_{q_1}, c_{q_2}, \ldots, c_{q_K}\}$, and the correct (or aggregated) answer to be estimated for the question q be Y_q. We define the answer to the question q to be the choice $c \in \{1, 2, \ldots, C\}$ for which $P\left(Y_q = c | c_{q_1}, c_{q_2}, \ldots, c_{q_K}\right)$ is maximum. Using Bayes' theorem and the independence assumption among participants' answers, we obtain:

$$\begin{aligned} & P(Y_q = c | c_{q_1}, c_{q_2}, \ldots, c_{q_K}) \\ &= \frac{P(c_{q_1}, c_{q_2}, \ldots, c_{q_K} | Y_q = c) P(Y_q = c)}{\sum_{c=1}^{C} P(c_{q_1}, c_{q_2}, \ldots, c_{q_K} | Y_q = c) P(Y_q = c)} \\ &= \frac{\left(\prod_{k=1}^{K} P(c_{q_k} | Y_q = c)\right) P(Y_q = c)}{\sum_{c=1}^{C} \left(\prod_{k=1}^{K} P(c_{q_k} | Y_q = c)\right) P(Y_q = c)} \end{aligned} \tag{1}$$

Let T_{qc} be the indicator that the answer to question q is choice c. Using our formulation:

$$T_{qc} = \begin{cases} 1 & c = \underset{j \in \{1,2,\ldots,C\}}{\arg\max} \; P(Y_q = c | c_{q_1}, c_{q_2}, \ldots, c_{q_K}) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

These T_{qc}s serve as the proposed answer sheet.

To determine the correct (or aggregated) choice for a question q, we need the values of $P(c_{q_k} | Y_q = c)$ for all k and c, which however is not known given only the choices from the crowd annotators. However, if the correct choices are known for all the questions, we can compute these parameters. Let q_k be the annotator a. To compute the parameters, we first define the following sets:

$$S_a^{(c)} = \{i \mid Y_i = c \wedge a \text{ has answered question } i\}$$

and

$$T_{c_a}^{(c)} = \{i \mid Y_i = c \wedge a \text{ has answered } c_a \text{ on question } i\}$$

Then, we have:

$$P(c_a | Y_q = c) = \frac{\left|T_{c_a}^{(c)}\right|}{\left|S_a^{(c)}\right|} \tag{3}$$

where $|\cdot|$ denotes the cardinality of the set. Also, $P(Y_q = c)$ can be defined as:

$$P(Y_q = c) = \frac{\text{Number of questions having answer as } c}{\text{Total number of questions}} \tag{4}$$

The above quantities can be estimated if we have the correct choices, and conversely, the correct choices can be obtained using the above quantities. We hence use an Expectation-Maximization (EM) strategy, where the E-step calculates the correct answer for each question, while the M-step determines the maximum likelihood parameters using equations 3 and 4. There are no pre-calculated values of parameters to begin with, and so in the first E-step, we estimate the correct choices using majority voting. We continue applying the EM steps until convergence. We use the total difference between two consecutive class marginals being under a fixed threshold as the convergence criterion. We discuss the convergence criterion in more detail in Section 4. The proposed algorithm is summarized below in Algorithm 1.

Algorithm 1 The Fast Dawid-Skene Algorithm

Input Crowdsourced choices of Q questions by A participants (annotators) from C choices
Output Proposed true choices - T_{qc}

1: Estimate Ts using majority voting.
2: **repeat**
3: *M-step:* Obtain the parameters, $P(c_a | Y_q = c)$ and $P(Y_q = c)$ using Equations 3 and 4
4: *E-step:* Estimate Ts using the parameters, $P(c_a | Y_q = c)$ and $P(Y_q = c)$, and with the help of Equations 2 and 1.
5: **until** convergence

3.3 Connection to Dawid-Skene Algorithm

The Dawid-Skene algorithm [6] was one of the earliest EM-based methods for aggregation, and still remains popular and competitive to newer approaches. In this subsection, we briefly describe the Dawid-Skene methodology, and show the connection of our approach to this method.

As defined in [6], the maximum likelihood estimators for the DS method are given by:

$$\hat{\pi}_{cl}^{(a)} = \frac{\text{number of times participant } a \text{ chooses } l \text{ when } c \text{ is correct}}{\text{number of questions seen by participant } a \text{ when } c \text{ is correct}}$$

and $\hat{p_c}$, which is the probability that a question drawn at random has a correct label of c. Let $n_{ql}^{(a)}$ be the number of times participant a chooses l for question q. Let $\{T_{qc} : q = 1, 2, \ldots, Q\}$ be the indicator variables for question q. If choice m is true, for question q, $T_{qm} = 1$ and $\forall j \neq m$, $T_{qj} = 0$. Given the assumptions made in Section 3.1, when the true responses of all questions are available, the likelihood is given by:

$$\prod_{q=1}^{Q}\prod_{c=1}^{C}\left\{p_c \prod_{a=1}^{A}\prod_{l=1}^{C}\left(\pi_{cl}^{(a)}\right)^{n_{ql}^{(a)}}\right\}^{T_{qc}} \tag{5}$$

where $n_{ql}^{(a)}$ and T_{qc} are known. Using equation 5, we obtain the maximum likelihood estimators as:

$$\hat{\pi}_{cl}^{(a)} = \frac{\sum_q T_{qc} n_{ql}^{(a)}}{\sum_l \sum_q T_{qc} n_{ql}^{(a)}} \tag{6}$$

$$\hat{p_c} = \frac{\sum_q T_{qc}}{Q} \tag{7}$$

We then obtain using Bayes' theorem:

$$p(T_{qc} = 1|\text{data}) = \frac{\prod_{a=1}^{A}\prod_{l=1}^{C}(\pi_{cl}^{(a)})^{n_{ql}^{(a)}} p_c}{\sum_{r=1}^{C}\prod_{a=1}^{A}\prod_{l=1}^{C}(\pi_{rl}^{(a)})^{n_{ql}^{(a)}} p_r} \tag{8}$$

The DS algorithm is then defined by using equations 6 and 7 to obtain the estimates of ps and πs in the M-step, followed by using equation 8 and the estimates of ps and πs to calculate the new estimates of Ts in the E-step. These two steps are repeated until convergence (when the values don't change over an iteration).

A close examination of the DS and proposed FDS algorithms shows that our algorithm can be perceived as a 'hard' version of DS. The DS algorithm derives the likelihood assuming that the correct answers (which are ideally binary-valued) are known, but uses the values for T_{qc} (which form a probability distribution over the choices) directly as obtained from equation 8. Instead, in our formulation, we always have T_{qc} as either 0 or 1 after each E-step. Our method is similar to the well-known Classification EM proposed in [5], which shows that a 'hard' version of EM significantly helps fast convergence and helps scale to large datasets [16]. We show empirically in Section 4 that this subtle difference between DS and FDS ensures that changes in the answer sheet dampens down quickly, and allows our method to converge much faster than DS with comparable performance. A careful implementation for both FDS and DS provides a solution in $O(QACn)$ time under the assumption that there is only one correct choice for each question, where n is the number of iterations required by the algorithm to converge. As the cost per iteration of FDS would be similar to DS by the nature of its formulation, this implies that the speedup of our algorithm is proportional to the ratio of the number of iterations required to converge by the two algorithms, which we also confirm experimentally.

3.4 Theoretical Guarantees for Convergence

In this subsection, we establish guarantees for convergence. We prove that if we start from an area close to a local maximum of the likelihood, we are guaranteed to converge to the maximum at a linear rate. For the analysis of our algorithm's convergence, we first frame it in a way similar to the Classification EM algorithm as proposed by [5]. Classification EM introduces an extra C-step (Classification step) after the E-step. This is the step that assigns each question a single answer, thus doing a 'hard' clustering of questions based on options instead of the 'soft' clustering by DS.

To continue with the proof we will use the notation used for DS. The term $P(c_{q_k}|Y_q = c)$ for FDS is replaced by $\pi_{cc_{qk}}^{q_k}$ and the term $P(Y_q = c)$ for FDS is replaced by p_c. $n_{ql}^{(a)}$ used by DS would be either 1 or 0 for the setting considered.

Having established the analogy, we restate the algorithm in CEM form (Algorithm 2).

Algorithm 2 The Fast Dawid-Skene Algorithm

Input Crowdsourced choices of Q questions by A participants (annotators) from C choices
Output Proposed true choices - T_{qc}

1: Estimate Ts using majority voting. This essentially does the first E and C step.
2: **repeat**
3: *M-step:* Obtain the parameters, πs and ps using Equations 3 and 4
4: *E-step:* Estimate Ts using the parameters, π and p, and with the help of Equation 1.
5: *C-step:* Assign Ts using the values obtained in the E-step and Equation 2.
6: **until** convergence

We prove the convergence of the CEM algorithm similar to [5]. For the proof, let us first form partitions. We form C partitions out of all the questions based on their correct answer in a step.

$$P_c = \{q|Y_q = c\} \tag{9}$$

In the CEM approach, each question can belong to only one partition. Now, we define the CML (Classification Maximum Likelihood) criterion:

$$C_2(P, p, \pi) = \sum_{c=1}^{C}\sum_{q\in P_c}\log\left(p_c f(q, \pi_c)\right) \tag{10}$$

In the above equation, $\pi_c = \{\pi_{cj}^{(a)}|\forall j \in \{1 \ldots C\} \text{ and a } \in \{1 \ldots A\}\}$ and

$$f(q, \pi_c) = \prod_{a=1}^{A}\prod_{l=1}^{C}\left(\pi_{cl}^{(a)}\right)^{n_{ql}^{(a)}} \tag{11}$$

To prove convergence, we define a few more notations. Note that we begin the algorithm by first doing a majority vote. This assigns

each question to a class and forms the first partition. We denote this partition as P^0. We then proceed to the M-step and estimate π and p. Let us denote this first set of parameters by π^1 and p^1. The next EC step gives the next partition, P^1. Thus, the algorithm continues to calculate $(P^m, p^{m+1}, \pi^{m+1})$ from (P^m, p^m, π^m) in the M step. Then, in the EC step, it calculates $(P^{m+1}, p^{m+1}, \pi^{m+1})$ from $(P^m, p^{m+1}, \pi^{m+1})$.

Theorem 3.1. *For the sequence (P^m, p^m, π^m) obtained by FDS, the value of $C_2(P^m, p^m, \pi^m)$ increases and converges to a stationary value. Under the assumption that ps and πs are well defined, the sequence (P^m, p^m, π^m) converges to a stationary point.*

Proof. To prove the above theorem we prove that $C_2(P^{m+1}, p^{m+1}, \pi^{m+1}) \geq C_2(P^m, p^m, \pi^m) \forall m > 1$.
Note that equations 3 and 4 maximize the likelihood given the values of T and n (as shown by [6]), i. e. T is known, and so πs and ps obtained by the M-step maximize the likelihood. We need to show that maximizing the likelihood is the same as maximizing the CML criterion, C_2. In the case of hard clustering, for each q, only one class, c, can have T_{qc} as 1; all other classes will have T_{qc} as 0. With this observation, we can rewrite the CML criterion as:

$$C_2(P, p, \pi) = \sum_{c=1}^{C} \sum_{q \in P_c} \log(p_c f(q, \pi_c)) \tag{12}$$

$$= \log \left\{ \prod_{q=1}^{Q} \prod_{c=1}^{C} (p_c f(q, \pi_c))^{T_{qc}} \right\} \tag{13}$$

$$= \log \left\{ \prod_{q=1}^{Q} \prod_{c=1}^{C} \left(p_c \prod_{a=1}^{A} \prod_{l=1}^{C} \left(\pi_{cl}^{(a)} \right)^{n_{ql}^{(a)}} \right)^{T_{qc}} \right\} \tag{14}$$

Thus, maximizing maximum likelihood is equivalent to maximizing C_2. So, we have that after the M step, $C_2(P^m, p^{m+1}, \pi^{m+1}) \geq C_2(P^m, p^m, \pi^m)$.

Now, we consider the EC step. Observe that for each question q, we choose the answer as the option c' for which $p'_c f(q, \pi'_c) \geq p_c f(q, \pi_c)$ for all c (By definition of the criterion for the C-step). Thus, $\log p_c f(q, \pi_c)$ increases individually for each question, and so cumulatively, $C_2(P^{m+1}, p^{m+1}, \pi^{m+1}) \geq C_2(P^m, p^{m+1}, \pi^{m+1})$. Combining the two inequalities, we obtain,

$$C_2(P^{m+1}, p^{m+1}, \pi^{m+1}) \geq C_2(P^m, p^m, \pi^m) \tag{15}$$

This proves that C_2 increases at each step. Since the number of questions are finite and so the number of partitions as well are finite; the value of C_2 must converge after a finite number of iterations. On convergence, we obtain $C_2(P^{m+1}, p^{m+1}, \pi^{m+1}) = C_2(P^m, p^{m+1}, \pi^{m+1}) = C_2(P^m, p^m, \pi^m)$ for some m. By definition of the C-step, the first equality implies that $P^{m+1} = P^m$. Also under the assumption that ps and πs are well defined, we have that $p^m = p^{m+1}$ and $\pi^{m+1} = \pi^m$. This proves the convergence to a stationary point. □

To prove the rate of convergence, we define M to be the set of matrices $U \in \mathbb{R}^{C \times Q}$ of nonnegative values. The matrices are defined such that the summation of values in each column is 1 and the summation along each row is nonzero.
Consider the criterion to be maximized as:

$$C'_2(U, p, \pi) = \sum_{c=1}^{C} \sum_{q=1}^{Q} u_{qc} \log(p_c f(q, \pi_c)) \tag{16}$$

With the above definitions, proposition 3 of [5] guarantees a linear rate of convergence for FDS to a local maximum from a neighborhood around the maximum.

3.5 Hybrid Algorithm

While the proposed FDS method is quick and effective, by using the softer marginals, DS can obtain better likelihood values (which we found in some of our experiments too). A comparison of the likelihood values over multiple datasets (described in Section 4) is provided in Table 2. To bring the best of both DS and FDS, we propose a hybrid version, where we begin with DS, and at each step, we keep track of sum of the absolute values of the difference in class marginals (p_cs). When this sum falls below a certain threshold, we switch to the FDS algorithm and continue (Algorithm 3). Our empirical studies showed that this hybrid algorithm can maintain high levels of accuracy along with faster convergence (Section 4). We however observe that a similar likelihood to DS does not necessarily translate to better accuracy, and in fact FDS outperforms Hybrid on some datasets.

Algorithm 3 The Hybrid Algorithm

Input Crowdsourced choices for Q questions by A participants given C choices per question, threshold γ
Output Aggregated choices: T_{qc}
1: Estimate Ts using majority voting.
2: **repeat**
3: *M-step:* Obtain parameters, $\hat{\pi}_{cl}^{(a)}$ and $\hat{p}_c$ using equations 6 and 7
4: *E-step:* Estimate Ts using parameters, $\hat{\pi}_{cl}^{(a)}$ and $\hat{p}_c$ using equation 8.
5: **until** $\sum_c |p_c^t - p_c^{t-1}| < \gamma$
6: **repeat**
7: EM steps of Algorithm 1 (FDS)
8: **until** convergence

4 EXPERIMENTAL RESULTS

We validated the proposed method on several publicly available datasets for vote aggregation, and the results are presented in this section. We first describe the datasets, competing methods used for comparison and the performance metrics used before presenting the results.

Datasets: We used seven real-world datasets to compare the performance of the proposed method against other methods. These include *LabelMe* [28, 31], *SentimentPolarity (SP)* [25, 30], *DAiSEE* [7, 17], and four datasets from the SQUARE benchmark [34]: *Adult2* [15], *BM* [24], *TREC2010* [4], and *RTE* [35].

Many of the datasets had varying number of annotators per data point. For uniformity, we set a threshold for each dataset, and all data points with fewer annotators than the threshold were

	# qns	# options (per qn)	Maximum # of annotators (per qn)	Speedup of FDS over DS in Time (Iterations)	Speedup of FDS over IWMV in Time (Iterations)	Speedup of Hybrid over DS in Time (Iterations)
Adult2	305	4	9	6.61(7.87)	1.32(1.15)	2.30(2.43)
BM	1000	2	5	2.69(4.51)	1.70(1.02)	1.49(2.03)
TREC2010	3670	4	5	7.84(8.64)	6.09(2.93)	4.39(4.59)
DAiSEE	4628	4	10	6.57(7.37)	4.40(2.04)	4.11(4.37)
LabelMe	589	8	3	7.55(8.59)	0.54(1.14)	5.15(5.47)
RTE	800	2	10	3.14(4.95)	2.63(1.24)	1.88(2.24)
SP	4968	2	5	3.00(3.95)	2.78(0.94)	2.40(2.54)

Table 1: Datasets Used and Speedup of FDS and Hybrid

removed. In our experiments, we studied the performance of all the methods by varying the number of annotators from one till the threshold, by taking a random subset of all annotators for a data point at each step (We maintained the same random seed across the methods, and conducted multiple trials to verify the results presented herewith). Also, the *TREC2010* dataset has an 'unknown' class, which we removed for our experiments. Table 1 lists the size, the number of classes, and the number of annotators in each dataset.

Baseline Methods: A total of six aggregation algorithms were used in our experiments for evaluation - Majority Voting (MV), Dawid-Skene (DS) [6], IWMV [22], GLAD [38], proposed Fast Dawid-Skene (FDS), and the proposed hybrid algorithm. IWMV is among the fastest methods using EM for aggregation under general settings. [22] compared IWMV against other well-known aggregation methods, including [27], [19] and [23], and showed that IWMV gives an accuracy comparable to these algorithms but does so in a much lesser time. We hence compare our performance to IWMV in this work. GLAD [38], another popular method, was proposed only for questions with two choices, and we hence use this method for comparison only on the binary label datasets in our experiments.

Performance Metrics: For each experiment, the following metrics were observed: the accuracy of the aggregated results (against provided ground truth), time taken and number of iterations needed for empirical convergence. For DS, FDS, and Hybrid, the negative log likelihood after each iteration was also observed. For MV, only the accuracy was observed. The experiments were conducted on a 4-core system with Intel Core i5-5200U 2.20GHz processors with 8GB RAM.

Results: The results of our experiments are presented in Figure 1 and Table 2. Table 1 shows the speedup in time and number of iterations needed to converge of FDS over DS and IWMV and of Hybrid over DS, averaged over all observations with varying number of annotators.

	FDS	DS	Hybrid
Adult2	1283.75	1153.09	1154.97
BM	2110.16	2094.76	2100.32
TREC2010	13109.26	12180.84	12346.91
DAiSEE	39968.08	36178.16	36350.61
LabelMe	1714.50	1655.94	1660.06
RTE	3741.61	3679.63	3680.32
SP	12472.00	12433.70	12440.70

Table 2: Negative Log Likelihood at convergence of FDS, DS and Hybrid methods

Performance Analysis of Fast Dawid-Skene: The results show that FDS gives similar accuracies when compared to DS, Hybrid, GLAD, and IWMV, and a significant improvement over MV, on most datasets except for the BM and LabelMe datasets. In LabelMe, the aggregation accuracy is not at par with DS or Hybrid but is still significantly higher than MV and comparable to IWMV. In the BM dataset, the accuracies of FDS and IWMV are slightly lower than MV but both are comparable to each other. In terms of time taken, we notice that apart from the LabelMe dataset, FDS performs much better than DS, Hybrid, IWMV and GLAD all through. In the case of LabelMe, IWMV outperforms in terms of speed but the margin is very small (around 0.1 sec). This leads us to infer that in general, FDS gives comparable accuracies to other methods while taking significantly lesser time.

Performance Analysis of the Hybrid Method: The goal of the Hybrid algorithm is to converge to a similar likelihood as DS in much lesser time. From the experiments (especially Table 2), we see that this is indeed the case - the log likelihood of the Hybrid algorithm is close to that of DS and consistently better than FDS. This naturally leads to accuracies almost similar to those obtained by DS, as is confirmed in the results. The total time taken for convergence is much lower for Hybrid as compared to DS. Moreover, the time taken for convergence by Hybrid is consistently low and does not deviate as much as IWMV. While IWMV outperforms Hybrid with respect to time in a few datasets, the proposed Hybrid outperforms IWMV on accuracy on those datasets. These observations support Hybrid to be an algorithm which performs with accuracies similar to DS in a much lesser time consistently over datasets.

Implementation Details: We discuss two important implementation details of the proposed methods in this section: *initialization* and *stopping conditions*. As argued in [6], a symmetric initialization of the parameters (all $P(Y_q = c)$s to be $1/C$) corresponds to a start from a saddle point, from where the EM algorithm faces difficulty in converging. Instead, a good initialization is to start with the majority voting estimate. While performing majority voting, it could often happen that there is a tie between two or more options with the highest number of votes. In such situations, we randomly choose an option among those which received the highest votes[1]. We maintained the same random seed for all methods which required this decision.

The ideal convergence criterion would be when the answer sheet proposed by an algorithm stops changing. This condition is met within a few iterations for FDS and Hybrid, but DS does not converge using this criterion in a reasonable number of steps. For example, in case of the *DAiSEE* dataset, DS did not converge even after 100 iterations (as compared to ≤ 10 for FDS). To address this issue, we set the convergence criterion as the point when the difference in class marginals is less than 10^{-4}. We do not include the changes in participant error rates in the final convergence criterion because we observed that its fluctuations could lead to stopping prematurely. Similarly, the criterion for switching from DS to FDS in the Hybrid algorithm is the point when the change in class marginals is less

[1]We also tried a variant, in which the option with the highest running class marginal was used to break ties. But this variant did not perform as well as the randomized majority voting across all methods. We also ran many trials with different random seeds, and found the results to almost the same as those presented.

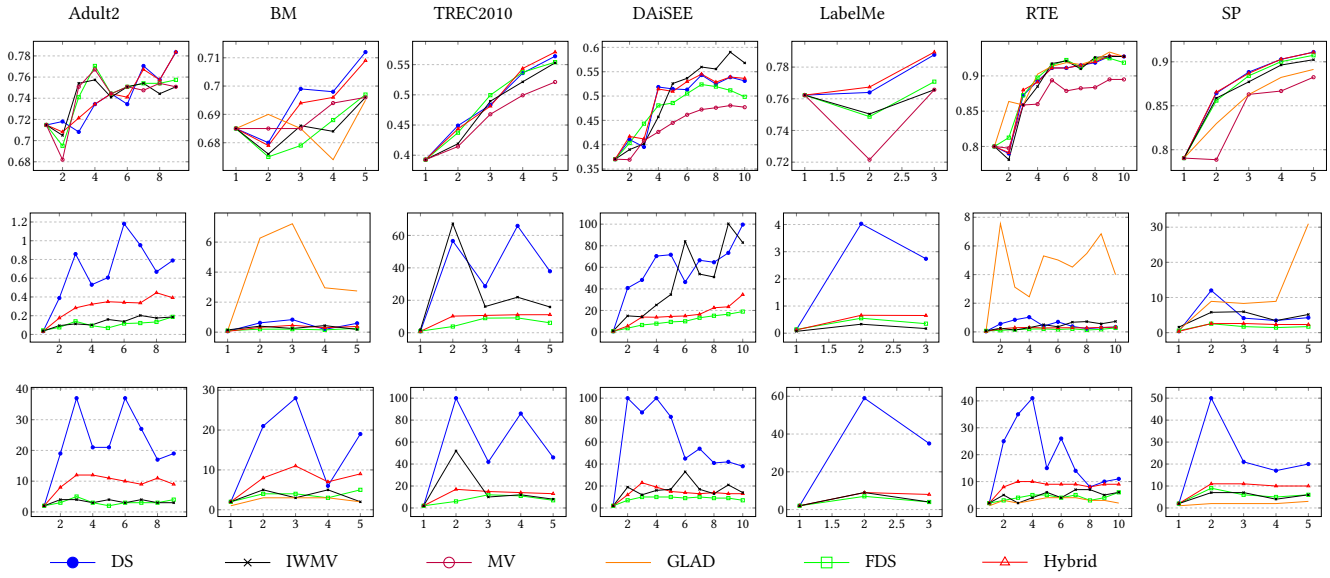

Figure 1: Experimental results: *(Row 1:)* Accuracy of different methods across the considered datasets; *(Row 2:)* Time taken in seconds to converge; and *(Row 3:)* Number of iterations to converge. X-axis denotes the varying number of annotators studied for each dataset.

than 0.005 (which happened approximately between 45-75% of total iterations across the datasets).

5 ONLINE VOTE AGGREGATION

Online aggregation of crowdsourced responses is an important setting in today's applications, where data points may be streaming in large data applications. We consider a setting in which we have access to an initial set of questions and have obtained the proposed answer key using FDS. We also have $P(Y = c)$ and $P(c_a|Y = a)\,\forall\, c, a$ at this time. When we receive a new question and the answers from multiple participants for this new question, we first estimate the answer for this question directly using majority voting. We then update the parameters using the M-step in Algorithm 1. After the M-step, we run the E-step only for this question to re-obtain the aggregated choice. To update the new knowledge which we have regarding the new participants, we run the M-step for one last time. We conducted experiments on the *SP* dataset[2], and observed almost the same accuracy for online FDS as offline FDS (Table 4) for different number of annotators. Table 3 shows the results for the max number of annotators (= 5).

	DS	FDS	Hybrid
Accuracy	90.94%	90.60%	90.64%
Time taken to converge (s)	4.40	3.76	4.09
# Iterations to converge	26	4	5

Table 3: Online Vote Aggregation on *SP* dataset.

Accuracy	2	3	4	5
FDS	85.59%	88.41%	90.02%	90.74%
Online FDS	83.57%	88.06%	89.90%	90.60%

Table 4: Online FDS vs FDS for varying number of annotators.

6 EXTENSION TO MULTIPLE CORRECT OPTIONS

The proposed FDS method can be extended to solve the aggregation problem under different settings. We describe an extension below, using the same notations as in Section 3.1.

In real-world machine learning settings such as multi-label learning, a data point might belong to multiple classes, which would result in more than one true choice per question. For such cases, we now assume that participants are allowed to choose more than one choice for each question. Our Algorithm 1 originally assumes that every question has exactly one correct choice. To overcome this limitation, we can make a simple modification in how we interpret questions when multiple options are correct. We assume that every (question, option) pair is a separate binary classification problem, where the label is true if the option is chosen for that question, and false otherwise. This transforms a task with Q questions and C options each to a task with QC questions and two options each. This is valid because the correctness of an option is independent of the correctness of all other options for that question in this setting. We ran experiments using this model on the Affect Annotation Love dataset *(AffectAnnotation)* used in [9] (which was specifically developed for this setting) on FDS, and compared our performance with DS and Hybrid. Our results are summarized in Table 5 (annotators=5, averaged over five subsets), showing the significantly improved results of FDS over DS. Hybrid attempts to follow DS in the likelihood estimation, and thus does not perform as well as FDS in this case. Besides, our results for FDS also performed better

[2]More results, including on other datasets, on https://sites.google.com/view/fast-dawid-skene/

than the methods proposed in [9], which showed a best accuracy of ≈ 92% on this dataset.

	DS	FDS	Hybrid
Accuracy	88.66%	94.14%	89.26%
Time taken to converge (s)	0.44	0.057	0.14
# Iterations to converge	29.6	2	5.8

Table 5: Multiple Correct Options setting on *AffectAnnotation* data.

7 CONCLUSION

In this paper we introduced a new EM-based method for vote aggregation in crowdsourced data settings. Our method, Fast Dawid-Skene (FDS), turns out to be a 'hard' version of the popular Dawid-Skene (DS) algorithm, and shows up to 7.84x speedup over DS and up to 6.09x speedup over IWMV in time taken for convergence. We also propose a hybrid variant that can switch between DS and FDS to provide the best in terms of accuracy and speed. We compared the performance of the proposed methods against other state-of-the-art EM algorithms including DS, IWMV and GLAD, and our results showed that FDS and the Hybrid approach indeed provide very fast convergence at comparable accuracies to DS, IWMV and GLAD. We proved that our algorithm converges to the estimated labels at a linear rate. We also showed how the proposed methods can be used for online vote aggregation, and extended to the setting where there are multiple correct answers, showing the generalizability of the methods.

REFERENCES

[1] S. Albarqouni *et al.* 2016. AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images. *IEEE Transactions on Medical Imaging* 35 (2016), 1313–1321.

[2] J. Bragg *et al.* 2013. Crowdsourcing Multi-Label Classification for Taxonomy Creation. In *HCOMP*.

[3] Anthony Brew, Derek Greene, and Pádraig Cunningham. 2010. Using Crowdsourcing and Active Learning to Track Sentiment in Online Media. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 145–150. http://dl.acm.org/citation.cfm?id=1860967.1860997

[4] C. Buckley, M. Lease, and M. D. Smucker. 2010. Overview of the TREC 2010 Relevance Feedback Track (Notebook). In *The Nineteenth Text Retrieval Conference (TREC) Notebook*.

[5] G. Celeux and G. Govaert. 1992. A classification EM algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis* 14, 3 (1992), 315–332.

[6] A. P. Dawid and A. M. Skene. 1979. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *J. Royal Stat. Soc. Series C* 28, 1 (1979), 20–28. http://www.jstor.org/stable/2346806

[7] Arjun D'Cunha, Abhay Gupta, Kamal Awasthi, and Vineeth Balasubramanian. 2016. Daisee: Towards user engagement recognition in the wild. *arXiv preprint arXiv:1609.01885* (2016).

[8] A. Dempster *et al.* 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Royal Stat. Soc. Series B* 39, 1 (1977), 1–38. http://www.jstor.org/stable/2984875

[9] L. Duan, S. Oyama, H. Sato, and M. Kurihara. 2014. Separate or joint? Estimation of multiple labels from crowdsourced annotations. *Expert Systems with Applications* 41, 13 (2014), 5723 – 5732. https://doi.org/10.1016/j.eswa.2014.03.048

[10] Ahmad Elsehnawy, Steele Carter, and Daniele Braga. 2016. It's Not Just What You Say, But How You Say It: Muiltimodal Sentiment Analysis Via Crowdsourcing.

[11] C. Gao and D. Zhou. 2013. *Minimax Optimal Convergence Rates for Estimating Ground Truth from Crowdsourced Labels*. Technical Report.

[12] Alex Gaunt, Diana Borsa, and Yoram Bachrach. 2016. Training Deep Neural Nets to Aggregate Crowdsourced Responses. *AUAI*. https://www.microsoft.com/en-us/research/publication/training-deep-neural-nets-aggregate-crowdsourced-responses/

[13] M. Y. Guan, V. Gulshan, A. Dai, and G. Hinton. 2017. Who Said What: Modeling Individual Labelers Improves Classification. *CoRR* abs/1703.08774 (2017). arXiv:1703.08774 http://arxiv.org/abs/1703.08774

[14] Giannis Haralabopoulos and Elena Simperl. 2017. Crowdsourcing for Beyond Polarity Sentiment Analysis A Pure Emotion Lexicon. *CoRR* abs/1710.04203 (2017). arXiv:1710.04203 http://arxiv.org/abs/1710.04203

[15] P. Ipeirotis, F. Provost, and J. Wang. 2010. Quality Management on Amazon Mechanical Turk. In *ACM SIGKDD Workshop on Human Computation (HCOMP '10)*. New York, NY, USA, 64–67. https://doi.org/10.1145/1837885.1837906

[16] F-X Jollois *et al.* 2007. Speed-up for the expectation-maximization algorithm for clustering categorical data. *Journal of Global Optimization* 37, 4 (2007), 513–525.

[17] Aditya Kamath, Aradhya Biswas, and Vineeth Balasubramanian. 2016. A crowdsourced approach to student engagement recognition in e-learning environments. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1–9.

[18] D. R. Karger, S. Oh, and D. Shah. 2011. *Budget-optimal crowdsourcing using low-rank matrix approximations*. 284–291. https://doi.org/10.1109/Allerton.2011.6120180

[19] D. R. Karger, S. Oh, and D. Shah. 2011. Iterative Learning for Reliable Crowdsourcing Systems. In *NIPS*. 1953–1961. http://papers.nips.cc/paper/4396-iterative-learning-for-reliable-crowdsourcing-systems.pdf

[20] A. Khetan and S. Oh. 2016. Achieving budget-optimality with adaptive schemes in crowdsourcing. In *NIPS*. 4844–4852. http://papers.nips.cc/paper/6124-achieving-budget-optimality-with-adaptive-schemes-in-crowdsourcing.pdf

[21] Svetlana Kiritchenko and Saif M. Mohammad. 2017. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best-Worst Scaling. *CoRR* abs/1712.01741 (2017). arXiv:1712.01741 http://arxiv.org/abs/1712.01741

[22] H. Li and B. Yu. 2014. Error rate bounds and iterative weighted majority voting for crowdsourcing. *CoRR* abs/1411.4086 (2014).

[23] Q. Liu *et al.* 2012. Variational Inference for Crowdsourcing. In *NIPS*. 692–700. http://papers.nips.cc/paper/4627-variational-inference-for-crowdsourcing.pdf

[24] B. Mozafari *et al.* 2012. Active Learning for Crowd-Sourced Databases. *CoRR* abs/1209.3686 (2012).

[25] B. Pang and L. Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL*. 115–124. https://doi.org/10.3115/1219840.1219855

[26] N. Quoc Viet Hung, N. T. Tam, and L. N. Tran. 2013. An Evaluation of Aggregation Techniques in Crowdsourcing. In *WISE*. 1–15.

[27] V. C. Raykar *et al.* 2010. Learning From Crowds. *JMLR* 11 (Aug. 2010), 1297–1322. http://dl.acm.org/citation.cfm?id=1756006.1859894

[28] F. Rodrigues, M. Lourenco, B. Ribeiro, and F. C. Pereira. 2017. Learning Supervised Topic Models for Classification and Regression from Crowds. *IEEE Trans. on PAMI* 39, 12 (Dec. 2017), 2409–2422. https://doi.org/10.1109/TPAMI.2017.2648786

[29] F. Rodrigues and F. Pereira. 2017. Deep learning from crowds. *CoRR* abs/1709.01779 (2017). arXiv:1709.01779 http://arxiv.org/abs/1709.01779

[30] F. Rodrigues *et al.* 2014. Gaussian Process Classification and Active Learning with Multiple Annotators. In *ICML*. II–433–II–441. http://dl.acm.org/citation.cfm?id=3044805.3044941

[31] B. Russell, A. Torralba, K. Murphy, and W. Freeman. 2008. LabelMe: A Database and Web-Based Tool for Image Annotation. *IJCV* 77, 1 (01 May 2008), 157–173. https://doi.org/10.1007/s11263-007-0090-8

[32] A. B. Sayeed *et al.* 2011. Crowdsourcing Syntactic Relatedness Judgements for Opinion Mining in the Study of Information Technology Adoption. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 69–77. http://dl.acm.org/citation.cfm?id=2107636.2107646

[33] N. B. Shah, S. Balakrishnan, and M. J. Wainwright. 2016. A Permutation-based Model for Crowd Labeling: Optimal Estimation and Robustness. *arXiv preprint arXiv:1606.09632* (06 2016).

[34] A. Sheshadri and M. Lease. 2013. Square: A benchmark for research on computing crowd consensus. In *HCOMP*.

[35] R. Snow *et al.* 2008. Cheap and Fast—but is It Good?: Evaluating Non-expert Annotations for Natural Language Tasks. In *EMNLP*. 254–263. http://dl.acm.org/citation.cfm?id=1613715.1613751

[36] F. Wauthier and M. Jordan. 2011. Bayesian Bias Mitigation for Crowdsourcing. In *NIPS*. 1800–1808. http://papers.nips.cc/paper/4311-bayesian-bias-mitigation-for-crowdsourcing.pdf

[37] P. Welinder and P. Perona. 2010. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. *IEEE CVPR Workshops* (2010), 25–32.

[38] J. Whitehill *et al.* 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *NIPS*. 2035–2043. http://papers.nips.cc/paper/3644-whose-vote-should-count-more-optimal-integration-of-labels-from-labelers-of-unknown-expertise.pdf

[39] H. Wu *et al.* 2015. Combining Machine Learning and Crowdsourcing for Better Understanding Commodity Reviews. In *AAAI'15*. AAAI Press, 4220–4221. http://dl.acm.org/citation.cfm?id=2888116.2888337

[40] Y. Zhang *et al.* 2014. Spectral Methods meet EM: A Provably Optimal Algorithm for Crowdsourcing. In *NIPS*. 1260–1268. http://papers.nips.cc/paper/5431-spectral-methods-meet-em-a-provably-optimal-algorithm-for-crowdsourcing.pdf