On Actively Teaching the Crowd to Classify *

Adish Singla
ETH Zürich
adish.singla@inf.ethz.ch

Ilija Bogunovic
ETH Zürich
ilijab@inf.ethz.ch

Gábor Bartók
ETH Zürich
bartok@inf.ethz.ch

Amin Karbasi
ETH Zürich
amin.karbasi@gmail.com

Andreas Krause
ETH Zürich
krausea@ethz.ch

Abstract

Is it possible to teach workers while crowdsourcing classification tasks? Amongst the challenges: (a) workers have different (unknown) skills, competence, and learning rate to which the teaching must be adapted, (b) feedback on the workers' progress is limited, (c) we may not have informative features for our data (otherwise crowdsourcing may be unnecessary). We propose a natural Bayesian model of the workers, modeling them as a learning entity with an initial skill, competence, and dynamics. We then show how a teaching system can exploit this model to interactively teach the workers. Our model uses feedback to adapt the teaching process to each worker, based on priors over hypotheses elicited from the crowd. Our experiments carried out on both simulated workers and real image annotation tasks on Amazon Mechanical Turk show the effectiveness of crowd-teaching systems.

1 Introduction

Crowdsourcing services, such as Amazon's Mechanical Turk platform[1] (henceforth MTurk), are becoming vital for outsourcing information processing to large groups of workers. Machine learning, AI, and citizen science systems can hugely benefit from the use of these services as large scale annotated data is often of crucial importance [1, 2, 3]. Data collected from such services however is often noisy, e.g., due to spamming, inexpert or careless workers [2]. As the accuracy of the annotated data is often crucial, the problem of tackling noise from crowdsourcing services has received considerable attention. Most of the work so far has focused on methods for combining labels from many annotators [4, 5, 6] or in designing control measures by estimating the worker's reliabilities through "gold standard" questions [1].

In this paper, we explore an orthogonal direction: *can we teach workers in crowdsourcing services in order to improve their accuracy?* That is, instead of designing models and methods for determining workers' reliability, can we develop intelligent systems that interact with workers to teach them to be more effective? There are numerous challenges in developing such systems, many of them attributed to the underlying complex nature of the learning process of human beings. Workers in crowdsourcing are often very diverse, coming from various backgrounds, with different skill sets and biases [4]. Also, while some concepts may be easy to learn by one set of workers, they could be very hard for others to pick up. All these factors, which affect the learning capability of the workers, are unknown to the teaching system. As the interaction with a worker is expensive, often the information that a teaching system can elicit from the worker in order to do more informed teaching is limited. Given these challenges, can we develop teaching systems that can interactively teach the workers and adapt the teaching as per the skills, competency, and dynamics of each worker?

*Under review by the International Conference on Machine Learning (ICML) 2014

[1] MTurk: https://www.mturk.com/mturk/welcome

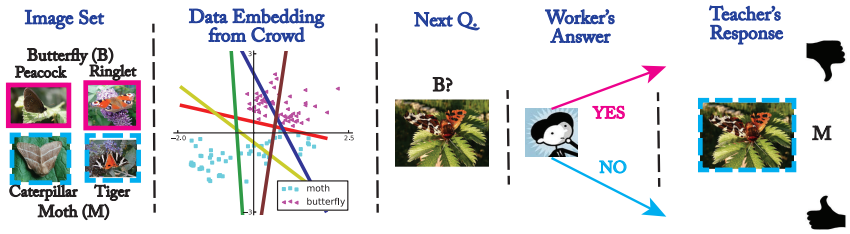


Figure 1: Illustration of the teaching process. Given a set of labeled images (labeled as "butterfly" or "moth", the knowledge of sub-species is not known), the teacher obtains an embedding of a small "teaching set" of images, as well as potential hypotheses used from a collection of workers. The teacher then uses this embedding and hypotheses in order to teach the rest of the crowd. The teacher sequentially provides an unlabeled example to the worker, who attempts an answer. The teacher then responds with the correct label, tracks the learner and steers him towards an accurate classifier.

To address the above questions, we provide the following contributions:

- We introduce a generic, Bayesian model formalizing the process of a worker learning to carry out a binary classification task. Our model is interactive, allowing for inferences about the learner's current hypothesis by requesting labels, and incorporating noisy responses. We further formalize assumptions about how the learner transits between hypotheses based on observed feedback.
- We propose a teaching algorithm that tracks the learner's progress and carefully and adaptively chooses which examples to teach. Under ideal conditions (no noise, etc.), our strategy provably has near-minimal cost.
- We carry out extensive experiments, on simulated workers, as well as on an actual image annotation task on MTurk. We demonstrate that our active teaching approach in fact improves classification performance over naive baselines.

2 Background and Teaching Process

We start with an overview of our approach, and then describe our learning domain and teaching protocol. As a running example, we consider the task of teaching to classify images, e.g., to distinguish butterflies from moths (see Figure 1).

2.1 Overview

We first provide a high-level intuition of our approach. Suppose we wish to teach the crowd to label a large set of images. How can this be done without having access to all labels already (in which case crowdsourcing would be useless)? We suppose we have ground truth labels for a small "teaching set" of examples. The hypothesis is that if we can teach a worker to classify this teaching set well, she can generalize to new images. In our approach, we first ask a small set of workers to each label the entire teaching set. From those labels, we use an existing Bayesian model [4] to infer a set of candidate features as well as hypotheses (linear classifiers) that the crowd may be using. Now, having access to the true labeling, as well as the crowd's prior distribution over hypotheses, we use a novel teaching algorithm (described below) to provide a sequence of training examples to new workers in order to steer them towards the true hypothesis.

2.2 The domain and the teaching protocol

Let $\mathcal{X}$ denote a set of examples (images), called the *teaching set*. We use (x, y) to denote a labeled example where $x \in \mathcal{X}$ and $y \in \{-1, 1\}$. We denote by $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$ a class of hypotheses. Each element of $\mathcal{H}$ is a function $h : \mathcal{X} \mapsto \mathbb{R}$. Then, the label assigned to x by hypothesis h is $\mathrm{sgn}(h(x))$. For now, let us assume that $\mathcal{X}$ and $\mathcal{H}$ are known to both the teacher and the learner. In our image classification example, each image may be given by a set of features x, and each hypothesis $h(x) = w_h^T x$ could be a linear function (inferred, for example, using the method of [4] on the teaching set, as described in Section 2.1). In Section 5, we discuss the concrete hypothesis spaces used in our crowdsourcing task, and how we can infer them from observing workers' labels alone (i.e., without having to explicitly compute any features).

The teacher has access to the labels of all the examples in $\mathcal{X}$. We consider the *realizable* setting where $\mathcal{H}$ contains a hypothesis h^* (known to the teacher, but not the learner) for which $\mathrm{sgn}(h^*(x)) = y(x)$ for all $x \in \mathcal{X}$. The goal of the teacher is to teach the correct hypothesis h^* to the learner. The basic assumption behind our approach is that if we can teach the workers to classify $\mathcal{X}$ correctly, then

they will be able to generalize to new examples (for which we neither have ground truth labels nor features). We will verify this assumption experimentally in Section 5. In the following, we review existing approaches to teaching classifiers, and then present our novel teaching method.

2.3 Existing teaching models

Here, we briefly review existing work on algorithmic teaching classifiers. In the seminal work, [7] consider a *non-interactive* model: The teacher reveals a sequence of labeled examples, and the learner discards any inconsistent hypotheses (i.e., for which $h(x) \neq y$ for any example (x, y) shown). For a given hypothesis class, the *Teaching Dimension* is the smallest number of examples required to ensure that all inconsistent hypotheses are eliminated. [7] relate this notion of complexity to other notions in learning theory such as the VC dimension. More recent work [8, 9, 10] considers models of *interactive* teaching. Alternative notions of teaching complexity are introduced [9, 10], which capture "rational" learners that reason about the teacher's intent. [8] consider a setting where, after the teacher shows each labeled example (x, y), the learner reveals his current hypothesis h. In case $h(x) \neq y$, the learner adopts an alternative hypothesis h'. [8] prove that under some assumptions on the learner's transition model, the teaching complexity (i.e., number of labels required) can be reduced. One example is the "lazy learner model" of [8], in which hypotheses are endowed by a metric, and the learner is assumed to deterministically move to the closest consistent hypothesis. While these approaches provide intriguing insights on the complexity of teaching, in practical settings they are limited. First, the learner's response is considered to be noise free. Second, the assumptions on the learner's transition model (capturing learning progress) are strong (e.g., the "lazy learner" model is deterministic in nature). Lastly, the assumption that the learner's hypothesis is fully revealed is hard to achieve in practice. In this paper, we introduce a Bayesian model relaxing these assumptions. Very recently, [11] study a similar problem of teaching workers to classify images. The authors empirically investigate a variety of mechanisms on a set of human subjects for a synthetically generated data set. We tackle the same problem by designing a principled Bayesian teaching framework, inspired by the models of teaching and learning.

2.4 Our model: Teaching with Partial Feedback

We assume that the teaching process is a sequential interaction between the teacher and the learner. See Figure 1 for an illustration. At the beginning, the learner adopts an initial hypothesis $h_0 \in \mathcal{H}$. In every round t, the teacher shows an (unlabeled) example $x_t \in \mathcal{X}$ to the learner. Then, the learner probabilistically guesses (using his current hypothesis h_t) the label l_t of x_t and reveals the guess l_t to the teacher. Finally, the learner is presented with the true label y_t, upon which he may or may not adopt a different hypothesis. We will formalize our assumptions about the learner's transition using a probabilistic model. Note that while the learner shares l_t, he does not disclose the hypothesis used for guessing, hence we call our model "teaching with partial feedback". Based on the history of examples and feedback labels, the teacher decides which example to show next, or alternatively, to stop the teaching sequence. We will formalize the learner's model in Section 3, and our teaching approach in Section 4.

3 Model of the Learner

We now introduce our model of the learner. In particular, we formalize our probabilistic assumptions about how the learner "guesses" the label l_t when given unlabeled example x_t, using his hypothesis h_t. We also formalize our probabilistic transition model. The behavior of the learner is governed by the following parameters, discussed in detail below:

- A (not necessarily symmetric) distance function $\mathcal{D} : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}_+$,
- a "reliability" parameter $\alpha \geq 0$ (models the learner's competence), and
- a "eagerness" parameter $\beta \geq 0$ (models the dynamics, i.e., the ability to learn quickly).

While our algorithm needs to know these parameters, our experiments demonstrate robustness against misspecification (see, c.f., Fig. 4(c)).

Learner's "guess": If the learner's current hypothesis is h_t and the example shown is x_t, the likelihood that the learner outputs the label $l_t = +1$ is given w.l.o.g. by the following logistic model:

$$P(l_t = +1 \mid x_t, h_t) = \frac{1}{1 + e^{-\alpha \cdot h_t(x_t)}} .$$

Thus, the absolute value of the hypothesis $|h_t(x_t)|$ is the confidence of the hypothesis in the answer. The above expression implies that $\alpha = 0$ results in a uniform random guesser, while the limit $\alpha = \infty$

corresponds to a deterministic learner whose label l_t is always the same as implied by the current hypothesis, i.e., $l_t = \text{sgn}(h_t(x_t))$. [2]

Transition model: First, we wish to capture the fact that learners are unlikely to dramatically change their hypotheses, and prefer gradual change. We consider a learner who probabilistically "jumps" between hypotheses according to their distance, in case the learner made an incorrect guess. More precisely, if the learner's guessed label l_t agrees with the provided example (x_t, y_t), i.e., $l_t = y_t$, the learner does not change his hypothesis: $P(h_{t+1} \mid h_t, x_{1:t}, y_{1:t}, l_t) = 1$ for $h_{t+1} = h_t$ if $l_t = y_t$. If $l_t \neq y_t$, however, he chooses to abandon the current hypothesis h_t and jumps to another hypothesis h_{t+1}, with a probability depending on the observed history and the distance between h_t and h_{t+1}. Concretely, we model the transition probability w.l.o.g. to be proportional to

$$T(h_{t+1} \mid h_t) = \frac{1}{1 + e^{\mathcal{D}(h_{t+1}, h_t)/\beta}}.$$

For finite $\beta > 0$, the learner more likely jumps to hypotheses that are close (according to $\mathcal{D}$) than far. The magnitude controls the distance traveled. Here again, the extreme values of β represent extreme behaviors: if $\beta = 0$, then the learner is not willing to move from the current hypothesis, while $\beta = \infty$ corresponds to jumping to an arbitrary hypothesis independent of the distance.

Secondly, we assume that the probability of jumping to any particular hypothesis h' also depends on the "evidence" that the learner has accrued about the correctness of h' so far. Specifically, after every round, the learner calculates a probability distribution over hypotheses, and the transition probability is also proportional to this distribution. Formally, the learner computes the belief $Q_t(h')$ about whether a hypothesis h' is correct, based on the history of labeled examples:

$$Q_t(h' \mid x_{1:t}, y_{1:t}) \propto \prod_{\tau=1}^{t} P(y_\tau \mid x_\tau, h') = \prod_{\tau=1}^{t} \frac{1}{1 + e^{-\alpha \cdot y_\tau \cdot h'(x_\tau)}}.$$

Combining both "distance" and "correctness" factors, the final distribution P from which the learner samples when $l_t \neq y_t$ is:

$$P(h_{t+1} \mid h_t, x_{1:t}, y_{1:t}, l_t) \propto T(h_{t+1} \mid h_t) Q_t(h_{t+1} | x_{1:t}, y_{1:t}) = \frac{1}{1 + e^{\mathcal{D}(h_{t+1}, h_t)/\beta}} \prod_{\tau=1}^{t} P(y_\tau \mid x_\tau, h).$$

4 Active Teaching Algorithm

Given the model of the learner and a prior over the learner's initial hypothesis $P'(h_0)$, how should the teacher choose examples to encourage the learner to quickly adopt an accurate hypothesis? By performing inference in the model, the teacher can track the learner's current hypothesis. By carefully showing examples, she can control the learner's progress by forcing (probabilistic) transitions of the learner's hypothesis towards the correct one.

Notice that, according to our assumptions, the learner follows a Markov process, whose transitions are controlled by the examples shown. Unfortunately the learner's state (his current hypothesis, together with the set of labeled examples he has seen) is only partially observable. While the teaching process could be modeled as a Partially Observable Markov Decision Process (POMDP) with exponentially large state space, solving for the optimal policy of presenting examples is intractable. Instead of attempting to find the optimal teaching policy, we will consider simple greedy heuristics, and demonstrate their effectiveness in our experiments (Section 5).

The case of $\alpha = \infty$, $\beta = \infty$ and no feedback. We first consider the special case where the learner always deterministically predicts, and – if incorrect – moves to a consistent hypothesis uniformly at random. In this case, every candidate training example is inconsistent with a subset $S_x \subseteq \mathcal{H}$ of the hypothesis space: $S_x = \{h \in \mathcal{H} : \text{sgn}(h(x)) \neq \text{sgn}(h^*(x))\}$. Thus, a natural teaching strategy is to attempt to greedily eliminate as much mass of inconsistent hypotheses as possible. Formally, having taught examples $x_1, \ldots, x_t$, let $\mathcal{H}_t = \{h \in \mathcal{H} : \text{sgn}(h(x_\tau)) = \text{sgn}(h^*(x_\tau)) \ \forall 1 \leq \tau \leq t\}$ be the remaining set of consistent hypotheses. In this setting, a natural approach is to greedily teach:

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} P'(\mathcal{H}_t \cap S_x), \tag{1}$$

where, for a subset $\mathcal{H}' \subseteq \mathcal{H}$, $P'(\mathcal{H}') = \sum_{h \in \mathcal{H}'} P'(h)$ is the amount of teacher's prior probability mass associated with the hypotheses in $\mathcal{H}'$. We stop teaching once all inconsistent hypotheses are

[2] Logistic model is w.l.o.g. as for alternate likelihood functions one can simply rescale all hypotheses $h \in \mathcal{H}$.

eliminated, i.e., for all $h \in \mathcal{H}_t$ and all $x \in \mathcal{X}$ it holds that $\text{sgn}(h(x)) = \text{sgn}(h^*(x))$. How well does this greedy algorithm perform? It turns out that the set function $F : 2^{\mathcal{X}} \to \mathbb{R}$ defined by $F(A) = 1 - P'\left(\mathcal{H} \bigcap_{x \in A} S_x\right) = P'\left(\bigcup_{x \in A} S_x\right)$ is a weighted set coverage function, and as such *monotone and submodular*. This means that whenever $A \subseteq B \subset \mathcal{X}$, and $x \in \mathcal{X}$, it holds that $F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B)$. For such functions, it is known that the greedy algorithm (implemented in (1)) provides a near-optimal solution [12] to the problem of finding a set $A \subseteq \mathcal{X}$ of minimum size such that $F(A) = F(\mathcal{X})$, which is satisfied iff all inconsistent hypotheses are eliminated. Moreover, under reasonable complexity theoretic assumptions, no efficient algorithm provides better solutions [13].

The general case What to do in the more general setting where α and β are arbitrary? Here, hypotheses are not simply eliminated, but just become less likely. Further, we would like to take the learner's guesses into account as partial feedback. To address these challenges, the teacher performs inference in her Bayesian model of the learner. In particular, the teacher starts with a prior $P'_0(h_0)$ of the learner's initial hypothesis. She then tracks the learner's belief utilizing the learner's transition and guessing model, as described in Section 3:

$$P'_t(h \mid x_{1:t}, l_{1:t}) \propto \sum_{h' \in \mathcal{H}} P'_{t-1}(h' \mid x_{1:t-1}, l_{1:t-1}) \cdot P(h|h', x_{1:t-1}, y_{1:t-1}, l_{t-1}) \cdot P(l_t \mid x_t, h_t)\,.$$

How should examples be chosen? We propose the following greedy rule:

$$x_{t+1} = \operatorname*{arg\,max}_{x \in \mathcal{X} \setminus \{x_1, \ldots, x_t\}} \mathbb{E}_{h \sim P'_t} \Big| P(l = +1 \mid h, x) - P(l = +1 \mid h^*, x) \Big|\,.$$

In words, the teacher picks the example for which the probability of obtaining guess +1 (w.l.o.g.) maximally differs from the probability of obtaining the same guess under the correct hypothesis, in expectation w.r.t. her current belief. This greedy rule prefers examples where the learner (according to the teacher's current belief) strongly disagrees with the correct hypothesis h^*. As we find in our experiments, it generally prefers examples about which the correct hypothesis h^* has high confidence (i.e., $P(l = +1 \mid h^*, x)$ close to 0 or 1), while maximizing the chance the learner disagrees. Thus, it trades "clarity" (simplicity) of the examples with their ability to disambiguate. As a further justification, it can be seen that in the case of $\alpha = \infty$ (where $P(l = +1 \mid h, x) \in \{0, 1\}$) and $\beta = \infty$ (where feedback does not help), this greedy rule will always pick examples where the probability that the learner disagrees with the teacher are maximized. Hence, it makes identical decisions to the greedy rule (1), and thus enjoys the same performance guarantees in that setting (i.e., $\alpha = \beta = \infty$).

When should we stop teaching? In general there are several options, for example the teacher would stop when the majority of the probability mass of P'_t is on one hypothesis, or the learner's expected error on the unlabeled examples $\mathcal{X}$ (according to the teacher's belief P'_t) is below some threshold ε. In our experiments, we choose to stop after a predefined number of iterations.

5 Experiments

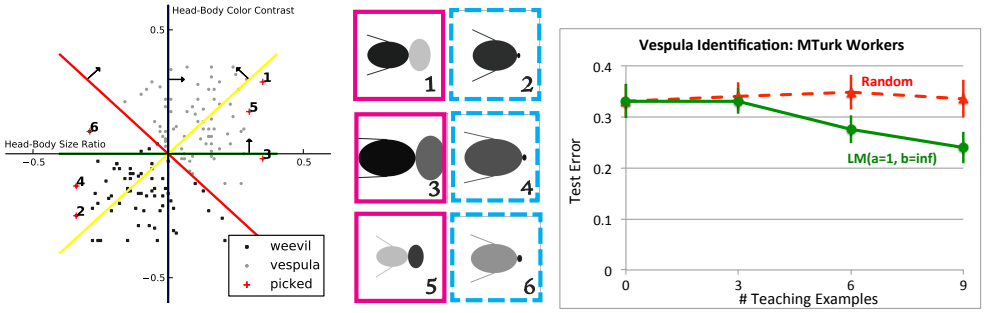
In our experiments, we consider two classification tasks, on synthetic and real images. While the feature space and hypothesis class is known for synthetic images, we illustrate how to automatically obtain a crowd-based embedding of the data for real images. We present results from simulated workers, as well as an actual annotation task on the Mechanical Turk platform.

5.1 Data set I – Synthetic Images

We first explored the use of synthetic images, in order to ensure that workers have no prior knowledge, and allow for more controlled experimentation.

Vespula vs Weevil - Data $\mathcal{X}$: We generated synthetic images of insects belonging to two species: *Weevil* and *Vespula*. The task is to classify whether a given image contains a Vespula insect or not. The images were generated by varying body size and color as well as head size and color. A given image x_i can be distinguished based on the following two-dimensional feature vector $x_i = [x_{i,1} = f_1, x_{i,2} = f_2]$: i) f_1: the head/body size ratio, ii) f_2: head/body color contrast. Fig. 2(a) shows the embedding of this data set in a two-dimensional space based on these two features and Fig. 2(b) shows sample images of the two species. A total of 100 images per species were generated by sampling the features f_1 and f_2 from two bivariate Gaussian distributions: ($\mu = [0.10, 0.13]$, $\Sigma = [0.12, 0; 0, 0.12]$) for Vespula and ($\mu = [-0.10, -0.13]$, $\Sigma = [0.12, 0; 0, 0.12]$) for Weevil.

Hypothesis Class and Distance Function: Since we know the exact feature space of $\mathcal{X}$, we can use any parametrized class of functions $\mathcal{H}$ on $\mathcal{X}$. In our experiments, we use a class of linear functions



(a) $\mathcal{X}$ and $\mathcal{H}$ (b) Picked Examples (c) MTurk Accuracy

Figure 2: **Vespula Identification**. **i)** Fig. 2(a) shows the 2-D embedding of synthetic images for *Weevil* and *Vespula* species for the features: head/body size ratio (f_1) and head/body color contrast (f_2), normalized around origin. **ii)** Fig. 2(b) illustrates that Weevil have short heads with color similar to their body, whereas Vespula are distinguished by their big and contrasting heads. **iii)** Fig. 2(a) shows four classes of hypotheses in $\mathcal{H}$ a) use both features (*correctly*) (target), b) use only feature f_1 (*correctly*), c) use only feature f_2 (*correctly*), d) use both features (*but* f_1 *is used incorrectly*). **iv)** Fig. 2(b) illustrates the order of examples picked by our algorithm for showing to synthetic as well as MTurk workers. **v)** Fig. 2(c) shows the performance on the task of Vespula identification for our algorithm compared to random teacher on MTurk workers as we vary the length of teaching phase.

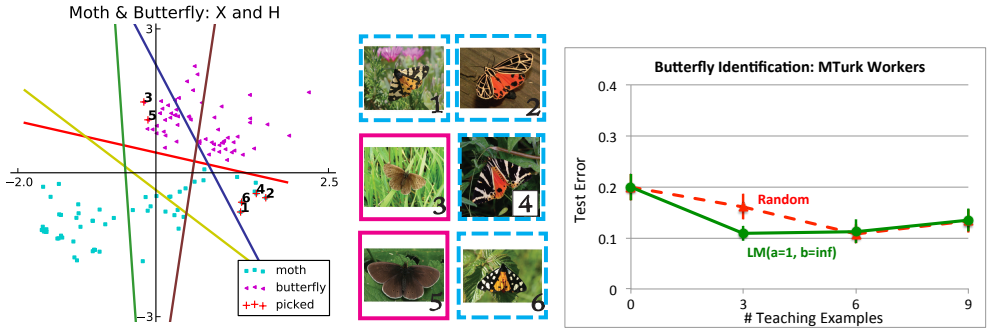
for $\mathcal{H}$. The target hypothesis is represented in red in Fig. 2(a). Instead of considering every possible linear hypothesis, we restrict ourselves to a specific subset. As the workers are shown a sample image of *Vespula* before starting the identification task, we restrict $\mathcal{H}$ to specific clusters based on the cues that workers would reasonably have. In this experimentation, we used four clusters of hypotheses, as illustrated in Fig. 2(a): *a)* use both features (correctly) *b)* use only feature f_1 (correctly), *c)* use only feature f_2 (correctly), *d)* use feature f_2 correctly but feature f_1 wrongly. The distance function $\mathcal{D} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ was modeled by measuring the number of data points on which the two hypothesis disagree and $h : \mathcal{X} \mapsto \mathbb{R}$ is computed given the parameters and features.

5.2 Data set II – Real Images

Butterfly vs Moth - Data $\mathcal{X}$: As our second dataset, we used a collection of 200 real images of four species of butterflies and moths from publicly available images[3], two species belonging to moths and two belonging to butterflies: a) Caterpillar Moth, b) Tiger Moth, c) Ringlet Butterfly, d) Peacock butterfly. On this dataset, the task is to classify whether a given image contains a butterfly or not. While Peacock Butterfly and Caterpillar Moth are clearly distinguishable as butterfly and moths, Tiger Moth and Ringlet Butterfly are visually hard to classify correctly. Fig. 3(a) shows these four species. We used 160 of these images (40 per sub-species) for teaching phase and the remaining 40 (10 per sub-species) for the testing phase.

Hypothesis Class and Distance Function: A Euclidean embedding of $\mathcal{X}$ for such an image set is not readily available. Human-perceptible features for such real images may be difficult to compute. In fact, this challenge is one major motivation for using crowdsourcing in image annotation. However, several techniques do exist that allow estimating such an embedding from a small set of images and limited number of crowd labels. In particular, we used the approach of [4] as a preprocessing step and requested labels for our teaching set consisting of 160 training images from a set of 60 workers. Using these crowd labels, we inferred a 2-D embedding of the data, as well as a prior distribution over linear hypotheses that the workers in crowd may have been using, as shown in Fig. 3(a). Note that these hypotheses capture various idiosyncrasies (termed "schools of thought" by [4]) in the workers' annotation behavior – i.e., some workers were more likely to classify certain moths as butterflies and vice versa. The distance function $\mathcal{D} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ was again obtained by considering the number of labels for which a given pair of hypotheses disagree. We emphasize that – crucially – this embedding is *not* required for test images. Neither the workers nor the system used any information about sub-species in the images. Given this embedding, we used the framework as discussed in case of synthetic images of Weevil and Vespula.

[3]Imagenet: `http://www.image-net.org/`



(a) $\mathcal{X}$ and $\mathcal{H}$ (b) Picked Examples (c) MTurk Accuracy

Figure 3: **Butterfly Identification**. **i)** Fig. 3(a) shows the 2-D embedding of images for the *Moth* and *Butterfly* data set obtained using the approach of [4]. Also, the hypothesis for a small set of workers are displayed, as inferred by the model of [4] **ii)** Fig. 3(b) illustrates the order of examples picked by our algorithm for showing to simulated as well as MTurk workers. **iii)** Fig. 3(c) shows the performance on the task of Butterfly identification for our algorithm compared to random teacher on MTurk workers as we vary the length of teaching phase.

5.3 Results on Simulated Learners

Next, we measure the performance of our algorithms on the two datasets above on a set of 50 simulated learners corresponding to different parameters α and β. We carry out the following experiments in order to understand the value and need for the three main components of our model: i) reliability α, ii) eagerness β and ii) feedback obtained by teacher. Additionally, we want to understand how robust our teaching algorithm is against misspecified parameters, as the actual values of α, β of the learner in general will not be known by the teacher.

Metrics and baselines: The performance metric that we use is the average classification error of the learners on a test data set, as we vary the length of teaching phase. In addition to the specific teacher models used for the experiment, we also compare our algorithms against two baselines: a) *Random* teacher (which picks the examples randomly), b) *SetCover* teacher (which is equivalent to $LM(\alpha = \infty, \beta = \infty)$, i.e., assuming a perfectly accurate, eager learner).

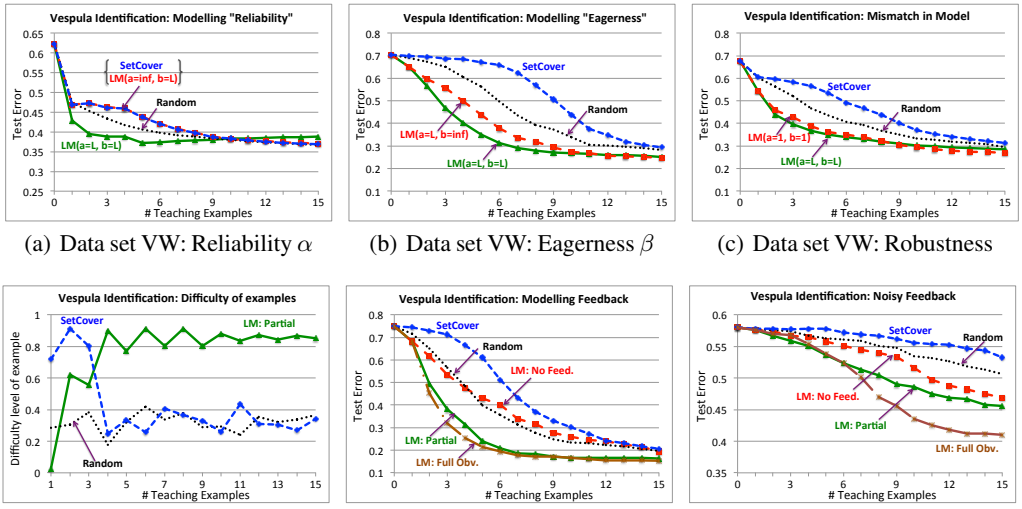
Is modeling the reliability α of the learner important? Fig. 4(a) shows the performance of a teacher using learner model $LM(\alpha = \infty, \beta = L)$ (i.e., teacher assumes a noiseless learner, though it has the correct specification of β). This is then compared to $LM(\alpha = L, \beta = L)$, which correctly models the noise α for the learner. The results clearly demonstrate the value of modeling the noise of the learner.

Is modeling the dynamics β of learner important? Fig. 4(b) compares the performance of a teacher which does not model eagerness ($\beta = \infty$) to the teacher using the correct specification of β of the learner. Correctly modeling the dynamics clearly helps the teaching process.

How robust is the teaching for a mismatched α and β? In real-world annotation tasks, α, β parameters of the learner are not known. In this experiment, we simulated a distribution over learners with different values of α, β. Fig. 4(c) shows that a conservative teacher (using $\alpha = 1$, $\beta = 1$) performs much better than a *Random* or *SetCover* teacher. And, the performance of this teacher is not much worse than that of the teacher which (unrealistically) uses the exact specification of the learner.

On difficulty of teaching. Fig. 4(d) shows the difficulty of examples picked by different algorithms during the process of teaching, where difficulty is measured in terms of probability of making a false guess by the learner. The *SetCover* teacher starts with very difficult examples assuming that the learner is perfect. Our teaching algorithm starts with easy examples, followed by more difficult ones, as also illustrated in the experiments in Fig. 2(b). Recent results of [11] show that such curriculum-based learning (where the difficulty level of teaching is increased with time) indeed is a useful teaching mechanism. Note that our Bayesian teaching process inherently incorporates this behavior, without requiring explicit heuristic choices. Also, the transition of *SetCover* to easier examples is just an artifact as *SetCover* randomly starts selecting examples once it (incorrectly) infers that the learner has adopted the target hypothesis.

Can limited feedback improve teaching? Fig. 4(e) illustrates the usefulness of partial feedback when the learner is reliable and lazy ($\alpha = 3$, corresponding to 95% accuracy when $h(x) = 1$, and $\beta = 0.2$, allowing "jumps" to only the nearest hypotheses). It compares the performance of our teaching



(a) Data set VW: Reliability α (b) Data set VW: Eagerness β (c) Data set VW: Robustness

(d) Data set VW: Difficulty (e) Data set VW: Feedback (f) Data set VW: Noisy Feedback

Figure 4: **(i)** Fig. 4(a) and 4(b) illustrate the importance of modelling reliability and eagerness in the teaching process, by comparing it to a teacher who ignores these aspects by setting $\alpha = \infty$ (assuming noiseless learner) and setting $\beta = \infty$ (assuming an eager learner with unrestricted ability to jump to any hypothesis). **(ii)** Fig. 4(c) illustrates how a conservative teacher ($\alpha = 1, \beta = 1$) performs much better than SetCover or random teacher, even if there is a model mismatch w.r.t. the α, β parameters used by the learner. **(iii)** Fig. 4(d) illustrate that difficulty level of the examples picked by our algorithm increases in the teaching process. **(iv)** Fig. 4(e) illustrate the usefulness of partial feedback when the learner is not very noisy ($\alpha = 3$) and has restricted ability to learn ($\beta = 0.2$). **(v)** Fig. 4(f) show that feedback can hurt when the learner is noisy ($\alpha = 0.5$).

algorithm *LM-Partial* (with correctly specified α, β) to another two variants: *a) LM-NoFeed* ignores the partial labeling feedback obtained from the worker, *b) LM-FullObv* is the unrealistic teacher that has full knowledge of the current hypothesis of the learner all the time. The results illustrate that surprisingly, partial feedback alone is enough to accurately track the worker and teach effectively compared to the unrealistic *LMFullObv* model. However, as the teaching process continues, *LM-Partial* accumulates more and more error, hence finally deviating away from *LM-FullObv*.

Does feedback always help? We hypothesize that noisy feedback could lead to wrong inferences by the teacher and may not help or could even hurt the performance of the teaching process. In the worst case, one can think of an adversarial learner who always provides wrong feedback about his responses. Teaching to such adversarial learners is hopeless. Results in Fig. 4(f) use a learner with $\alpha = 0.5$ (compared to $\alpha = 3$ as used in Fig. 4(e)). This reduces the reliability rate of learner to 0.62 instead of 0.95 when $h(x) = 1$. The results indeed illustrate this aspect, where the teacher that uses partial feedback from a noisy learner tends to start performing worse over time.

5.4 Results on Real Image Annotation Tasks on Amazon Mechanical Turk

Next, we measure the performance of our algorithms when deployed on the actual MTurk platform. To simplify our experimental setup, we used a variant of our algorithm that models reliability, but does not take into account eagerness and feedback by using $\alpha = 1$, $\beta = \infty$. We simulated workers offline on the above two datasets and use the sequence produced by the algorithm to teach the workers on MTurk. Here, we varied the number of teaching examples from 0, 3, 6 and 9. Teaching is followed by a phase of testing examples without providing feedback, for which we report the classification error. We did this for both the random teacher and our algorithm. A total of 210 workers participated in the Vespula identification task with testing phase of 15 examples, and 420 workers in the Butterfly identification task with testing phase of 8 examples. This corresponds to running a particular algorithm for a particular number of examples for 30 workers for Vespula and 60 workers for Butterfly identification. We seek to understand the following two questions:

Does teaching help? Fig 2(c) and Fig. 3(c) show that teaching by our algorithm indeed significantly improves the accuracy of the workers in both the tasks. Random teaching in Fig 2(c) does not seem to help in the Vespula task, possible since examples are selected that may be confusing to the workers. Fig. 2(b) and Fig 3(b) demonstrate the sequence of images that were picked by our algorithm and shown to a particular worker.

Does our teaching algorithm perform better than simply showing random examples? Both Fig 2(c) and Fig. 3(c) demonstrate that our algorithm significantly outperforms random teaching. As the length of the teaching phase increases, we observe diminishing returns, and random teaching catches up with the performance of our algorithm on the Butterfly task. This is consistent with offline simulations when random teaching eventually catches up with an intelligent algorithm when allowed enough teaching examples. For harder tasks, we expect random to take longer to catch up.

6 Conclusions

We proposed a Bayesian model of the workers' learning process in crowdsourcing classification tasks. Our model of the learner captures some natural aspects such as idiosyncratic prior knowledge, reliability (to capture competence) and eagerness (to capture the ability to learn quickly). We then developed a teaching system that exploits this model to teach the workers interactively, in an adaptive manner. Our model generalizes existing models of teaching and introduces new practical features including handling noise and incorporating partial feedback. Our extensive experiments on simulated workers as well as on real annotation tasks on the Mechanical Turk platform demonstrate the effectiveness of our active teaching approach. More generally, our approach goes beyond solving the problem of teaching workers in crowdsourcing services. With the recent growth of online education and tutoring systems [4], such intelligent teaching systems can be envisioned to aid in supporting personalized online education [14, 15].

References

[1] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

[2] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *First IEEE Workshop on Internet Vision*, 2008.

[3] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 389:1179–1189, 2008.

[4] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Proc. Neural Information Processing Systems (NIPS)*, 2010.

[5] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *Proceedings Neural Information Processing Systems (NIPS)*, 2011.

[6] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294, 2013.

[7] S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50:303–314, 1992.

[8] F. J. Balbach and T. Zeugmann. Recent developments in algorithmic teaching. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, pages 1–18, 2009.

[9] S. Zilles, S. Lange, R. Holte, and M. Zinkevich. Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12:349–384, 2011.

[10] T. Doliwa, H. U. Simon, and S. Zilles. Recursive teaching dimension, learning complexity, and maximum classes. In *Proceedings of the 21st international conference on Algorithmic learning theory*, pages 209–223, 2010.

[11] S. Basu and J. Christensen. Teaching classification boundaries to humans. In *AAAI*, 2013.

[12] G.L. Nemhauser, L.A. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[13] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45:314–318, 1998.

[14] D. S Weld, E. Adar, L. Chilton, R. Hoffmann, and E. Horvitz. Personalized online education a crowdsourcing challenge. *Workshop on Human Computation*, 2012.

[15] S. Dow, E. Gerber, and A. Wong. A pilot study of using crowds in the classroom. In *CHI*, 2013.

[4] c.f., `https://www.coursera.org/`