The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

AdaFlock: Adaptive Feature Discovery for Human-in-the-Loop Predictive Modeling

Ryusuke Takahama
scouty Inc.*
r.takahama7591@gmail.com

Yukino Baba
Kyoto University
baba@i.kyoto-u.ac.jp

Nobuyuki Shimizu, Sumio Fujita
Yahoo Japan Corporation
{nobushim, sufujita}@yahoo-corp.jp

Hisashi Kashima
Kyoto University; RIKEN Center for AIP
kashima@i.kyoto-u.ac.jp

Abstract

Feature engineering is the key to successful application of machine learning algorithms to real-world data. The discovery of informative features often requires domain knowledge or human inspiration, and data scientists expend a certain amount of effort into exploring feature spaces. Crowdsourcing is considered a promising approach for allowing many people to be involved in feature engineering; however, there is a demand for a sophisticated strategy that enables us to acquire good features at a reasonable crowdsourcing cost. In this paper, we present a novel algorithm called *AdaFlock* to efficiently obtain informative features through crowdsourcing. AdaFlock is inspired by AdaBoost, which iteratively trains classifiers by increasing the weights of samples misclassified by previous classifiers. AdaFlock iteratively generates informative features; at each iteration of AdaFlock, crowdsourcing workers are shown samples selected according to the classification errors of the current classifiers and are asked to generate new features that are helpful for correctly classifying the given examples. The results of our experiments conducted using real datasets indicate that AdaFlock successfully discovers informative features with fewer iterations and achieves high classification accuracy.

1 Introduction

Feature engineering is the key to the successful design of machine learning classifiers for real-world data. The learning of a good predictive model is difficult if we fail to determine useful features for classification. The discovery of informative features often requires domain knowledge or human inspiration. Considerable efforts have been made in machine learning projects to explore feature spaces. Researchers in several fields have attempted to construct task-specific features by incorporating heuristics. For instance, in the computer vision area, SIFT (Lowe 2004) and SURF (Bay, Tuytelaars, and Gool 2006) were designed for general object recognition tasks, and different features were proposed for detailed tasks such as face detection (Viola and Jones 2004) or human body detection (Dalal and Triggs 2005).

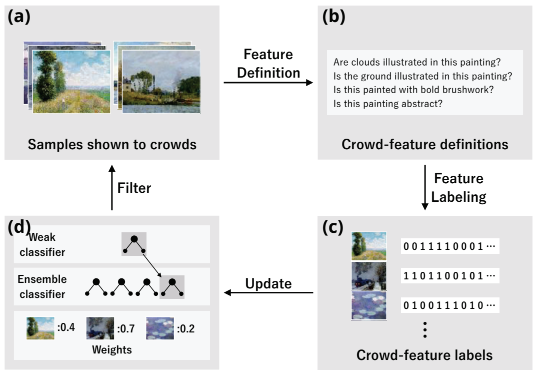


Figure 1: Overview of AdaFlock, which iteratively generates informative features by asking crowdsourcing workers to provide features that are helpful for correctly classifying the misclassified examples.

Although careful feature engineering for a particular task would be beneficial for improving classification performance, the broad exploration of features is quite laborious. The recent expansion of crowdsourcing platforms allows many people to be involved in feature engineering. Although it is difficult for non-expert crowdsourcing workers to provide correct answers for fine-grained classification tasks, they may have capabilities to determine informative features for classification (Branson et al. 2010).

Recently, a few methods for crowdsourced feature discovery have been proposed. Cheng and Bernstein developed an interactive platform called Flock (Cheng and Bernstein 2015), which asks crowdsourcing workers to compare paired examples and describe their differences. The collected descriptions are organized as features, such as “does this painting contain flowers?”; feature labels (i.e., “yes” or “no”) for each example are then generated using another set of crowdsourcing tasks. The paired examples are randomly chosen by Flock; however, features may be able to be generated in a more efficient manner by adaptively selecting the exam-

*This work has been done while the first author worked at JST, ERATO, Kawarabayashi Large Graph Project, Grant Number JPMJER1201, Japan

Table 1: Comparison between existing methods and our method

	Visual 20 questions game (Branson et al. 2010)	Flock (Cheng and Bernstein 2015)	Adaptive triple selection (Zou, Chaudhuri, and Kalai 2015)	**AdaFlock**
Crowdsourced feature-definition		✓	✓	✓
Crowdsourced feature-labeling	✓	✓	✓	✓
Adaptive example selection			✓	✓
Supervised learning		✓		✓

ples. Zou et al. proposed an adaptive method for selecting triples of examples to avoid the generation of overlapping features (Zou, Chaudhuri, and Kalai 2015). This method chooses examples based on feature labels of the previously discovered features. Because the method does not focus on supervised learning, the performance of a classifier is not considered during example selection.

In this paper, we propose a new algorithm called *AdaFlock* to efficiently generate informative features through crowdsourcing. Our algorithm aims to obtain features helpful for improving the classification performance through iterations. AdaFlock is inspired by AdaBoost (Freund and Schapire 1997), which iteratively trains weak classifiers by increasing the weights of examples misclassified by the current classifiers. Analogously, at each iteration of AdaFlock, crowdsourcing workers are shown examples selected according to the classification errors of the current classifiers (Figure 1(a)). The workers are asked to generate features helpful for correctly classifying the given examples (Figure 1(b)). AdaFlock then asks crowdsourcing workers to label each example based on each feature definition (Figure 1(c)). A weak classifier is trained by using the obtained labels, and the Filter function (Bradley and Schapire 2007) is applied for resampling examples according to the classification errors of the current classifier (Figure 1(d)). The sampled examples are shown to workers of the next iteration.

Table 1 summarizes the difference between AdaFlock and the other crowdsourced feature discovery methods. Although the method proposed by Branson et al. (2010) requires predefined feature definitions, AdaFlock uses crowdsourcing for defining features. In contrast to Flock (Cheng and Bernstein 2015), AdaFlock adaptively generates features to improve the classification accuracy. While adaptive triple selection (Zou, Chaudhuri, and Kalai 2015) aims to obtain diverse features, the goal of AdaFlock is to obtain informative features to improve the classification accuracy.

We conducted experiments on a crowdsourcing platform by using image and movie classification datasets and observed that AdaFlock discovers various types of features not covered by Flock. Moreover, a classifier built by using AdaFlock outperforms the classifier built through Flock.

It would be worth noticing that one of the major drawbacks of crowdsourced feature discovery is scalability; this approach is not suitable for classifying a large number of examples because of the requirement of crowdsourced feature labeling. Practical situations where crowdsourced feature discovery is useful are (1) obtained interpretable features themselves are important for explaining predictions, and (2) it is desired to create an accurate classifier regardless of crowdsourcing costs.

The contributions of this paper are threefold:

- We address the problem of adaptive crowdsourced feature generation for supervised learning.
- We propose a novel algorithm called AdaFlock, which obtains informative features by sampling difficult examples for the current classifier.
- Through experiments conducted using actual crowdsourcing datasets, we confirm that AdaFlock achieves better classification performance than the existing methods.

2 AdaFlock

2.1 Problem Setting

We first show a formulation of our feature generation problem for predictive modeling. We focus on binary classification in this paper. Assume that there is a training dataset as $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $y_i \in \{-1, +1\}$. Given the training dataset, our goal is to build a classifier $H(x)$. Unlike typical binary classification problems, a feature vector of each x_i is not given here. Thus, we will concurrently generate feature vectors and train a classifier.

2.2 Overview

We propose a novel algorithm called AdaFlock to efficiently generate feature vectors through crowdsourcing and to train a classifier by using the obtained features. AdaFlock requests crowdsourcing workers to process *feature-definition* tasks and *feature-labeling* tasks. Examples of these two tasks are illustrated in Figure 3 and 4. In feature-definition tasks, workers are shown a small number of positive and negative examples and asked to describe the difference between the positive and negative examples. This approach is called analogical encoding, and its advantages for producing predictive features have been demonstrated in (Cheng and Bernstein 2015). AdaFlock instructs workers to write a description as a yes–no question; for example, "is the sky illustrated in this painting?" or "does this text contain any hard number?" In feature-labeling tasks, workers are given

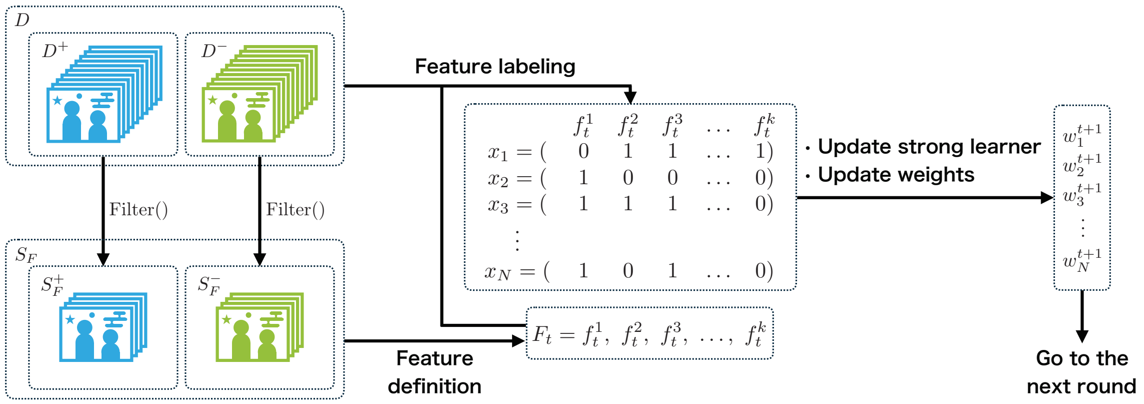
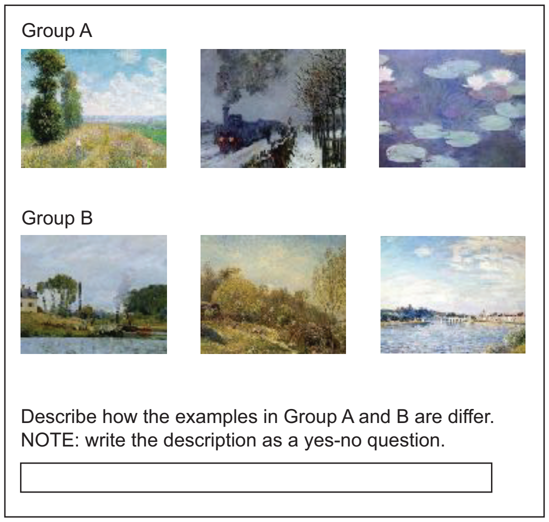


Figure 2: Overview of AdaFlock algorithm. At the t-th iteration, given the weights of examples, AdaFlock first resamples a small number of positive and negative examples, $S_F^+ \subseteq D^+$ and $S_F^- \subseteq D^-$. AdaFlock shows S_F^+ and S_F^- to crowdsourcing workers and requests them to process feature-definition tasks to obtain F_t. Feature-labeling tasks are then posted, in which workers are asked to label each object x_i based on each feature definition $f_t^j \in F_t$. AdaFlock next trains a weak leaner by using the obtained labels, and update the current leaner and the weight of each example, w_i^{t+1}.



Group A

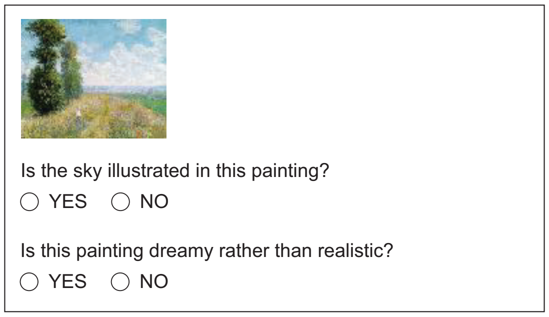
Group B

Describe how the examples in Group A and B are differ.
NOTE: write the description as a yes-no question.

Figure 3: Example of a feature-definition task, where workers are shown a small number of positive and negative examples and asked to describe the difference between them. Workers are instructed to write a description as a yes-no question.

an example and asked to choose a "yes" or "no" label for each question collected in the feature-definition tasks.

AdaFlock iteratively queries feature-definition tasks and feature-labeling tasks to discover informative features through crowdsourcing. The design of AdaFlock is motivated by AdaBoost, which iterates the following three steps to obtain an ensembled classifier: (a) train a weak learner on the current data distribution; (b) calculate classification errors of the current classifier; and (c) update the data distribution by increasing the weights of the misclassified examples. AdaFlock follows these steps by querying the crowdsourcing workers.



Is the sky illustrated in this painting?
◯ YES ◯ NO

Is this painting dreamy rather than realistic?
◯ YES ◯ NO

Figure 4: Example of a feature-labeling task, in which workers are given an example and asked to provide a binary label for each question collected in the feature-definition tasks.

In step (a), AdaFlock uses crowdsourcing to obtain new features based on the given data distribution, and trains a classifier by using the obtained features. The data distribution is derived from the weights of the examples according to the classification errors in the previous iteration. The crowdsourcing workers are required to define features helpful for correctly classifying the misclassified examples in the previous iteration. In other words, we intend for the workers to consider data distribution when they define features; however, data distribution is less interpretable for humans. We follow the solution proposed by Pareek and Ravikumar (2013) and apply the filtering approach to resample a small number of examples. The sampled examples are shown to the workers.

In step (b), AdaFlock computes the weighted accuracy of the weak learner and incorporates the learner into the current classifier. The weights are calculated based on the prediction accuracies for all the training samples. In the step (c), the data distribution is updated so that the examples misclassified by the current classifier are more likely to be sampled in step (a) of the next iteration. AdaFlock repeats these steps

Algorithm 1 AdaFlock

```
1: input Crowd oracle $\mathcal{O}$; training dataset $D$;
       termination parameters $\delta_t$ and $\epsilon$;
       number of examples for each iteration $m$;
       number of feature definitions at each iteration $k$;
       number of iterations $T$
2: Set weights $w_i^1 = 1/N, i = 1, \ldots, N$
3: $D^+ = \{(x, y) \mid (x, y) \in D,\ y = +1\}$
4: $D^- = \{(x, y) \mid (x, y) \in D,\ y = -1\}$
5: for $t = 1, \ldots, T$ do
6:     $S_F^+ = \phi, S_F^- = \phi$
7:     while $|S_F^+| \leq \frac{m}{2}$ and $|S_F^-| \leq \frac{m}{2}$ do
8:         Sample an object $(x, y)$ using FILTER$(w^t, \delta_t)$
9:         if $y = +1$ and $|S_F^+| \leq \frac{m}{2}$ then
10:            Add $(x, y)$ to $S_F^+$
11:        else if $y = -1$ and $|S_F^-| \leq \frac{m}{2}$ then
12:            Add $(x, y)$ to $S_F^-$
13:    $S_F = S_F^+ + S_F^-$
14:    Use $\mathcal{O}$ to get features $F_t = f_t^1, \ldots, f_t^k$ for $S_F$
                                          ▷ feature-definition task
15:    Use $\mathcal{O}$ to label $D$ based on features $F_t$
                                          ▷ feature-labeling task
16:    Train a classifier $h_t$
           by using labels for $D$ based on $F_t$
17:    Set edge
           $\gamma_t \leftarrow \sum_{i \in D} \mathbf{1}[y_i = h_t(x_i)] w_i^t / \sum_{i \in D} w_i^t - 0.5$
18:    Set weight $\alpha_t \leftarrow 0.5 \log(0.5 + \gamma_t) / (0.5 - \gamma_t)$
19:    Update $H_t(x) \leftarrow H_{t-1}(x) + \alpha_t h_t(x)$
20:    Update $w_i^{t+1} \leftarrow 1/(1 + \exp(y_i H_t(x_i)))$
21: return strong learner $H_T(x) = \text{sign}(\sum_t \alpha_t h_t(x))$
22:
23: function FILTER$(w^t, \delta_t)$
24:    $r \leftarrow$ Number of calls to FILTER so far on round $t$
25:    $\delta_t' \leftarrow \frac{\delta_t}{r(r+1)}$
26:    $(D', w') \leftarrow$ Permutation of $(D, w^t)$
27:    for $i = 0;\ i < \frac{2}{\epsilon}\ln(1/\delta_t');\ i = i + 1$ do
28:        $(x, y) \leftarrow$ the $i$-th element of $D'$
29:        return $(x, y)$ with propability $w_i'$
30:    return $H_t(x)$
```

several times to output the final classifier.

2.3 Algorithm

Algorithm 1 shows the detailed procedure of AdaFlock, which is illustrated in Figure 2. We first divide the training dataset into the positive examples $D^+ = \{(x, y) \mid (x, y) \in D,\ y = +1\}$ and the negative examples $D^- = \{(x, y) \mid (x, y) \in D,\ y = -1\}$. At each round t, for step (a), AdaFlock resamples a small number of positive and negative examples, $S_F^+ \subseteq D^+$ and $S_F^- \subseteq D^-$, respectively. The Filter function is called to collect examples until the number of examples in both S_F^+ and S_F^- exceed $\frac{m}{2}$, where m is the parameter given to AdaFlock. The Filter function performs random sampling without replacement and obtains a sample, (x, y), and its assigned weight, w. With the probability of w, the Filter function returns the sample which is then added to S_F^+ or S_F^-. Otherwise the sample is rejected and the Filter function draws another sample.

Next, AdaFlock requests crowdsourcing workers to process feature-definition tasks. In these tasks, the examples in S_F^+ and S_F^- are shown to the workers, who are then asked to define the features of round $F_t = f_t^1, \ldots, f_t^k$, where k is the number of features of each round. AdaFlock then generates feature-labeling tasks, in which the workers are asked to provide binary labels $\boldsymbol{x}_{it} = x_{it}^1, \ldots, x_{it}^k$ for each object $x_i \in X$ and features F_t. By using $\{(\boldsymbol{x}_{it}, y_i)\}_{i=1}^N$, AdaFlock trains a weak learner h_t.

In step (b), similar to the HumanBoost algorithm (Pareek and Ravikumar 2013), AdaFlock computes γ_t, which is the weighted accuracy of the weak learner h_t:

$$\gamma_t = \frac{\sum_{i \in D} \mathbf{1}[y_i = h_t(x_i)] w_i^t}{\sum_{i \in D} w_i^t} - 0.5, \tag{1}$$

where $\mathbf{1}[\cdot]$ returns 1 if the condition is true and 0 otherwise. The weighted accuracy γ_t is then used for determining α_t, which is the ensemble weight of h_t:

$$\alpha_t = \frac{0.5 \log(0.5 + \gamma_t)}{0.5 - \gamma_t}. \tag{2}$$

Furthermore, step (c) is also performed similar to that in the HumanBoost algorithm. Here, AdaFlock updates the ensemble classifier H_t by using α_t and h_t:

$$H_t(x) = H_{t-1}(x) + \alpha_t h_t(x). \tag{3}$$

A new weight w_i^{t+1} is then assigned to each example $(x_i, y_i) \in D$ by using the outputs of H_t:

$$w_i^{t+1} = \frac{1}{1 + \exp(y_i H_t(x_i))}. \tag{4}$$

The assigned weight w_i^{t+1} is less than $1/2$ if H_t correctly classifies the example and greater than $1/2$ otherwise. When $y_i H_t(x_i)$ decreases, w_i^{t+1} becomes closer to one so that this example is more likely to be selected for feature-definition tasks in the next iteration.

AdaFlock repeats these steps T times and outputs the final classifier $H_T(x) = \text{sign}(\sum_t \alpha_t h_t(x))$.

AdaFlock has one termination condition; if the Filter function rejects an example several times, AdaFlock terminates the process and returns the current classifier H_t. This condition is derived from a theoretical guarantee of classification errors. By following HumanBoost, AdaFlock ensures that the error rate becomes sufficiently small after rejecting several samples. Let us define the error rate of H_t over target distribution D as $\text{err}_t = \Pr_{(x,y) \leftarrow D}[H_t(x) \neq y]$ and denote ϵ a desired error rate. It is guaranteed that $\text{err}_t < \epsilon$ with probability at least $1 - \delta_t'$ when the Filter function rejects $n \geq \frac{2}{\epsilon} \ln\left(\frac{1}{\delta_t'}\right)$ samples.

In addition, based on a proposition regarding the number of iterations and the error rate for AdaBoost (Freund and Schapire 1997), we gets the number of iterations T that is sufficient to achieve $\text{err}_t < \epsilon$ as $T = \frac{1}{2\gamma_*^2} \ln \frac{1}{\epsilon}$ where the error rate of each weak learner h_t is ensured to be less than $1/2$ and $\gamma_* = \min_t |\gamma_t|$.

When we apply the final classifier for a new object x^*, we must query feature-labeling tasks for the example by using $\{F_t\}_t$ to create $\{\boldsymbol{x}_t^*\}_t$. The output $h_t(x^*)$ of each weak learner is used to compute the output of the final classifier $H_T(x^*) = \text{sign}\left(\sum_t \alpha_t h_t(x^*)\right)$.

3 Experiments

3.1 Experimental Setup

Datasets We prepared two datasets for conducting our experiments: *Paintings* and *Smiles*. The paintings dataset contains paintings by two impressionist painters: Claude Monet and Alfred Sisley. We obtained 200 paintings by Monet[1] and 200 paintings by Sisley[2]. Each picture was cropped from each side by 30 pixels to remove signatures. The smiles dataset contains videos of spontaneous or posed smiles of enjoyment. We randomly selected 200 spontaneous smile videos and 200 posed smile videos from the UvA-NEMO Smile Database (Dibeklioğlu, Salah, and Gevers 2012).[3]

Both datasets have 400 examples each. We used 200 examples for training, from which 100 were positive and the others were negative; the remaining 200 examples were used for testing. We prepared two different training-testing splits for each dataset and the methods were performed on each split (i.e., trial).

Procedure We fix the number of iterations to a rather large value (i.e., $T = 10$) to investigate the change of the accuracies according to the number of iterations. We set the number of selected examples at each iteration to $m = 20$ and the number of feature definitions at each iteration to $k = 10$ because we consider these settings are convenient for workers. We assumed that δ_t and ϵ are sufficiently small values. We employed a decision tree as a weak learner.

We used Lancers,[4] a commercial crowdsourcing platform in Japan, for processing our experiments. In each AdaFlock iteration, feature-definition and feature-labeling tasks were posted to the crowdsourcing platform. We hired k workers for the feature-definition tasks of each iteration. Each worker was shown three positive and three negative examples randomly selected from S_F^+ and S_F^-, respectively, and was asked to provide one feature definition; thus, we collected k definitions in each round. We obtained a total of kT feature definitions. In the feature-labeling tasks, we asked three crowdsourcing workers to label each object–feature pair, and used the majority voting rule to decide the label. We paid 0.30 USD for defining one feature and 0.05 USD for providing one label for a object–feature pair. For each trial, we spent 630 USD for completing both feature-definition and feature-labeling tasks of $T = 10$ iterations in total.

[1]http://www.claudemonetgallery.org/

[2]http://www.alfredsisley.org/

[3]http://www.uva-nemo.org

[4]http://www.lancers.jp/

Baselines We prepared three baseline methods to investigate the efficiency of AdaFlock:

- **Crowd prediction**: This method directly asked crowdsourcing workers to classify the samples. Specifically, three crowdsourcing workers were assigned to each of the samples, and their majority voting results were used as the final label.
- **Off-the-shelf features**: We prepared computed features that were not specially designed for a given classification task and trained an AdaBoost classifier by using the features. For the painting dataset, we applied the pre-trained Inception-v3 model (Szegedy et al. 2015) for the images to obtain feature vectors with $1{,}008$ dimensions. We did not prepare off-the-shelf-features for the Smiles dataset because it is not straight forward to extract reasonable features for videos.
- **StandardFlock**: We collected features by using the Flock algorithm (Cheng and Bernstein 2015). We first posted feature-definition tasks to generate kT features. In the feature-definition tasks, each worker was shown three positive and three negative examples randomly selected from the training datasets and was asked to provide a feature definition. We then requested feature-labeling tasks for all the examples and the feature definitions. Unlike the original Flock, we did not apply clustering to organize the obtained feature definitions. Therefore, AdaFlock and this implementation of Flock (called *StandardFlock*) differ only in adaptive selection of examples for feature-definition tasks. An AdaBoost classifier was trained by using the obtained feature labels.

For each trial of each dataset, StandardFlock and AdaFlock require 630 USD for crowdsourcing. We spent 12 USD for crowd prediction tasks, and Off-the-shelf-features does not require crowdsourcing costs.

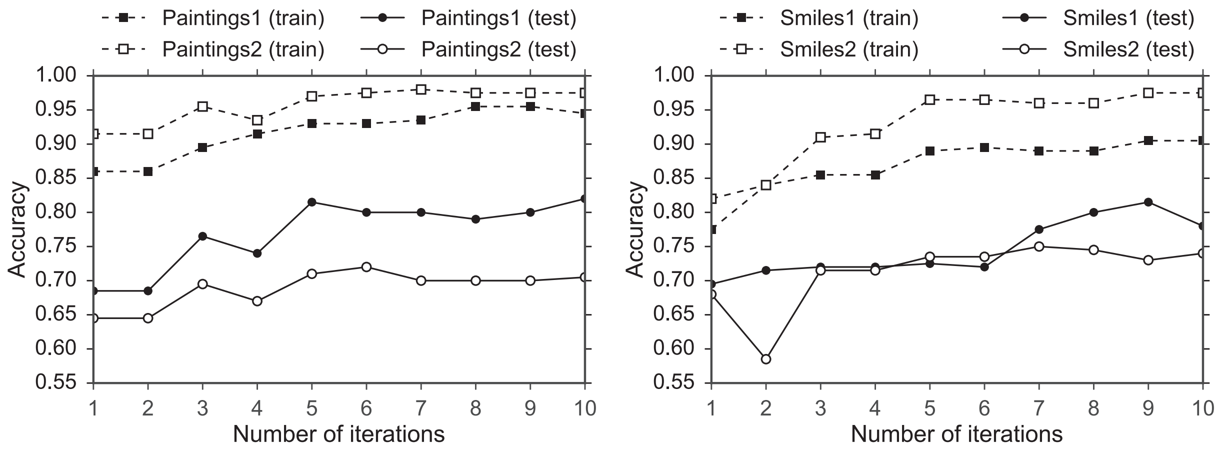
3.2 Results

Figures 5a and 5b show the training and testing accuracies at each iteration of AdaFlock, respectively. It is observed that the training accuracies successfully improve with increasing iterations in both the datasets. The testing accuracies increase with increasing iterations as well. It is remarkable that the testing accuracy of the first trial of the paintings dataset at the fifth iteration achieves a 10% improvement over that at the previous iteration. At the fifth iteration, AdaFlock discovered a feature of "is the ground illustrated in this painting?" which was strongly correlated with the true labels and contributed to improve the testing accuracy of AdaFlock. A similar result is observed in the smiles dataset; a feature of "Is the duration of the smile short?" was discovered at an iteration by AdaFlock, which contributed to an 8% improvement in the testing accuracies.

Table 2 shows the testing accuracy of AdaFlock at the end of all iterations in each trial, and those of the baseline methods. AdaFlock outperforms all the baseline methods in most cases. On the paintings dataset, as the performance of the crowd prediction indicates, although crowdsourcing workers did not demonstrate high performances for classifying the pictures, they generated informative features through

Table 2: Comparison of the testing accuracies for each trial of each dataset. AdaFlock outperforms the baseline methods in most cases.

Dataset	Trial	Crowd prediction	Off-the-shelf features	StandardFlock	AdaFlock
Paintings	1	0.560	0.755	0.755	0.820
Paintings	2	0.580	0.735	0.745	0.705
Smiles	1	0.720	-	0.715	0.780
Smiles	2	0.715	-	0.720	0.740



a Paintings

b Smiles

Figure 5: Training and testing accuracies at each AdaFlock iteration for each trial of each dataset. The number after each dataset name indicates the trial number. Both the training and the testing accuracies increase with increasing iterations.

AdaFlock. In contrast, on the smiles dataset, whereas the crowd prediction reaches a high level of accuracy, the accuracies of the classifier with StandardFlock are on a similar level. Even in such a case, AdaFlock performs better then both baseline methods.

Additionally, AdaFlock achieves the same level of accuracy as that of StandardFlock in less than or equal to five iterations in three out of the four cases. This indicates that AdaFlock successfully generated more informative features than the baselines at reasonable crowdsourcing costs.

AdaFlock shows a lower accuracy than StandardFlock only in the second trial of the painting dataset. As Figure 5a shows, in this trial, the training accuracy reached to high level in the first few iterations; this indicates that AdaFlock discovered appropriate features for the training samples at these iterations but they slightly overfitted. In such a case, showing more examples in the feature-definition tasks would be efficient to let the crowdsourcing workers generate more general features.

We next investigate the features discovered by AdaFlock and StandardFlock. Table 3 lists the representative features discovered by AdaFlock. It is observed that AdaFlock successfully discovered several informative features that were not discovered by StandardFlock. It is remarkable that AdaFlock generated features related to the ground (e.g., "is the ground illustrated in this painting?") for the paintings dataset, which were strongly correlated with the true labels but were not discovered by StandardFlock. As this finding indicates, AdaFlock successfully discovers informative features through adaptive selection of difficult examples.

Additionally, we examine the number of redundant features generated by AdaFlock. Considering a feature is redundant if the feature has the same meaning with one of the previously discovered features, we find that 23.5% and 44% of the AdaFlock features are redundant in the painting and smiles dataset, respectively. We can reduce costs for crowdsourced feature generation by avoiding such redundant features and incorporating a procedure for addressing this issue into AdaFlock would be our future work; for example, as AdaFlock is an iterative approach, we can show the features obtained in the previous iterations to workers and ask them to provide features that are not yet collected.

4 Related Work

There have been a few studies addressing crowdsourcing feature discovery. Branson et al. proposed a method for crowdsourced feature labeling (Branson et al. 2010). Although ordinary crowdsourcing workers are not able to recognize detailed categories, such as "Myrtle Warbler" or "Thruxton Jackaroo," which are species of birds, the workers may be able to identify some attributes of examples. Based on this assumption, the authors proposed a framework in which each worker was asked to answer predefined questions, such as "is the belly white?" The answers for each ex-

Table 3: List of generated features for the paintings dataset and the smiles dataset . Features assigned with a high positive or negative correlation coefficient with the true labels of the test samples are shown. Correlation coefficient values are given in brackets. The features not discovered by StandardFlock are bold-faced.

Dataset	Features
Paintings	Is the vast sky illustrated in this painting?; Are clouds illustrated in this painting?; **Is the ground illustrated in this painting?**; Are people in this painting drawn from a distance?; Are any flowers illustrated in this painting?; **Is this painting dreamy rather than realistic?**; Is this painted with bold brushwork?; **Is this painting abstract?**
Smiles	Does the person keep smiling for more than three seconds?; Does the person shake slightly?; Is the person's shoulder shaking while they are laughing? ; Does the person shake their faces while the corners of their mouths turning up?; Is the duration of smile short?; **Does the person suddenly stop smiling?**; Does the person turn serious shortly after they stop smiling?; **Does the person lean back in a seat while they are laughing?**

ample were used as feature labels. It was demonstrated that the combination of these feature labels and computed visual features was helpful for improving the image classification accuracies.

The above-mentioned framework requires pre-defined questions (i.e., feature definitions). A method of utilizing crowdsourcing for both feature definition and feature labeling was proposed (Cheng and Bernstein 2015). This system, named Flock, shows a pair of positive and negative examples to workers and asks them to describe how the examples differ. The collected descriptions are organized as features, and then feature labels for each example are generated by another set of crowdsourcing tasks. The features are used for learning a decision tree, and Flock has a function of detecting weak parts of the tree and asking crowdsourcing workers to generate new features for examples at the nodes.

After collecting the feature descriptions, Flock performs clustering and selects a representative feature of each cluster to remove redundant features. Zou, Chaudhuri, and Kalai proposed an algorithm called adaptive triple selection, which adaptively selects examples that are helpful for obtaining diverse features (Zou, Chaudhuri, and Kalai 2015). This method asks crowdsourcing workers to define a feature that is common to two out of three given examples. Feature-labeling tasks based on this feature are then requested. Using the collected labels, the algorithm selects examples for the next feature-definition task. The examples are chosen because they have the same feature labels, so that we can avoid obtaining the same feature again.

Like AdaFlock, Flock and the adaptive triple selection use crowdsourcing for both feature definition and feature labeling; however, Flock does not apply any adaptive method to select examples. Although the adaptive triple selection chooses examples that are suitable for generating diverse features, this method is not limited to applying for supervised learning, so the ground truth labels are not incorporated into the selection. AdaFlock calculates the classification errors at each iteration by using the ground truth labels to select the examples that are difficult for the current classifier and to obtain informative features that improve the classification accuracy.

Pareek and Ravikumar proposed a boosting algorithm called HumanBoost, which considers a crowdsourcing worker as a weak learner and iteratively trains the workers (Pareek and Ravikumar 2013). HumanBoost uses the sampling method of FilterBoost to select important examples, and the sampled examples are shown to the workers when they learn classification rules. In contrast to HumanBoost, AdaFlock does not consider a worker as a weak learner; AdaFlock uses crowdsourcing workers for feature definition and labeling, and the classifiers are trained by using ordinary machine learning methods.

5 Conclusion

In this paper, we proposed AdaFlock, an algorithm for generating informative features through crowdsourcing. AdaFlock iteratively obtains helpful features for improving the classification performance. Inspired by AdaBoost, at each iteration of AdaFlock, crowdsourcing workers are shown examples selected according to the classification errors of previous classifiers. They are then asked to generate features useful for correctly classifying the given examples. We apply the FilterBoost approach to sample the examples based on the classification errors. We conducted experiments on a commercial crowdsourcing platform, and the results indicate that AdaFlock successfully discovers informative and interpretable features.

At each iteration of AdaFlock, the positive and negative examples are independently selected based on the classification error of each example; however, the combination of examples would be an important factor for eliciting informative features from crowdsourcing workers. Incorporating such a mutual influence into the algorithm would be the focus of our future work. Combination of AdaFlock and machine learning methods is an interesting research direction that should be pursued. We will consider to replace human labeling process with machine learning methods which use raw image features. If the labeling for a feature is difficult for machines, we can perform additional run of AdaFlock for the sub-classification task. This is another promising future work.

References

Bay, H.; Tuytelaars, T.; and Gool, L. V. 2006. SURF: Speeded Up Robust Features. In *Proceedings of the 9th European Conference on Computer Vision (ECCV)*, 404–417. Springer.

Bradley, J. K., and Schapire, R. E. 2007. FilterBoost: Regression and Classification on Large Datasets. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 185–192.

Branson, S.; Wah, C.; Schroff, F.; Babenko, B.; Welinder, P.; Perona, P.; and Belongie, S. 2010. Visual Recognition with Humans in the Loop. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, 438–451. Springer.

Cheng, J., and Bernstein, M. S. 2015. Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW)*, 600–611.

Dalal, N., and Triggs, B. 2005. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 886–893.

Dibeklioğlu, H.; Salah, A. A.; and Gevers, T. 2012. Are You Really Smiling at Me? Spontaneous Versus Posed Enjoyment Smiles. In *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, 525–538. Springer.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Lowe, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60(2):91–110.

Pareek, H., and Ravikumar, P. 2013. Human Boosting. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 338–346.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the inception architecture for computer vision. *CoRR* abs/1512.00567.

Viola, P., and Jones, M. J. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision* 57(2):137–154.

Zou, J. Y.; Chaudhuri, K.; and Kalai, A. 2015. Crowdsourcing Feature Discovery via Adaptively Chosen Comparisons. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 198–205.