Data Utility Maximization When Leveraging Crowdsensing in Machine Learning

Juan Li[†,‡], Jie Wu[‡], and Yanmin Zhu[†,1]
[†]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[‡] Department of Computer and Information Sciences, Temple University

***Abstract*—With the increasingly wide adoption of crowdsensing services, we can leverage the crowd to obtain labeled data instances for training machine learning models. In this paper, we focus on the critical problem that which data instances should be collected to maximize the performance of the trained model under the budget limit. Solving this problem is nontrivial because of the unclear relationship between the performance of the trained model and the data collection process, NP-hardness of the problem and the online arrival of workers. To overcome these challenges, we first propose a crowdsensing framework with multiple rounds of data collecting and model training. The framework is based on the stream-based batch-mode active learning. According to the framework, we come up with a novel data utility model to measure the contribution of a data batch to the performance of the learning model. The data utility model combines uncertainty and weighted density to measure the contribution of one instance. Finally, we propose an online algorithm to select a data batch in each round. The algorithm achieves fairness, computational efficiency and a competitive ratio 0.1218 when the ratio of the largest contribution of one data instance to the optimal offline total data utility is infinitely small. Through evaluations based on a real data set, we demonstrate the efficiency of our data utility model and our online algorithm.**

***Index Terms*—Crowdsensing, Machine learning, Data utility, Active learning, Budget constraint, Various cost, Online.**

I. Introduction

Because of the explosive increase in the number of mobile devices embedded with rich sensors, crowdsensing over mobile devices has become an appealing paradigm for collecting sensing data to train machine learning models [1] [2]. A crowdsensing system typically consists of a data collector and many workers (such as smartphone users). By recruiting many cheap workers, the data collector is able to collect sensing data for model training with a small cost.

For example, the government wants to learn how much time people in a specific city would spend on different activities, such as waling, jumping, running, sitting and standing. It can reflect the health condition of people in this city. The government has a large set of activity data Q collected by a background thread run on smartphones [3]. An activity data instance includes gyroscope data and accelerator data, which record information about human activities. The government needs labels (activity types) of Q to train a machine learning model which can perform human activity recognition on Q. Then the government can analyze the time that people spend on each activity. However, sensor data cannot be recognized by humans. Therefore, the government collects a new set of labeled activity data (much smaller than Q) from cheap workers. By the small set of labeled activity data, the government can train a machine learning model and use it to analyze Q.

When leveraging crowdsensing in machine learning, the data collector aims to train a learning model with the highest performance (accuracy, F1-score and so on). However, the budget is often limited. Data instances have different contributions to model training and different costs. *It is a critical problem that which data instances should be collected to maximize the performance of the trained model under a fixed budget.*

Challenges. It faces several major challenges to solve this problem. *First*, we have to consider the online setting. Due to the limited storage space on mobile devices, a worker cannot hold all data until selection decisions are made at the end time of the data collection process. The offline version of our problem is NP-hard. Online arrivals of workers make this problem even more complex.

Second, bridging the gap between the performance of the trained model and the process of selecting data is nontrivial. An explicit relationship between an instance and its contribution to the performance of the model is unknown for now.

Our approach. In response to the challenges mentioned above, we first propose our crowdsensing framework with multiple rounds of data collecting and model training. In each round, we select a batch of data instances online, enlarge the training set and update the learning model. To formulate the relationship between the performance of the learning model and the data collection process, we come up with a novel data utility model. It combines uncertainty and weighted density to measure the contribution of an instance. We next construct our data utility maximization problem for each round. This problem is NP-hard even in the offline scenario. Finally, we propose a multi-stage online algorithm to solve the data utility maximization problem. In each stage, the algorithm selects data instances according to their utility contributions per cost. If it reaches the threshold, the data is selected. The threshold is updated at the end of each stage by an offline algorithm. We find the optimal parameters (the piecewise point r and shrinkage factor δ') by optimizing the competitive ratio. The achieved competitive ratio is 0.1218 when the upper bound of the ratio between the utility contribution of an instance and the offline optimal data utility is infinitely small.

1. Corresponding author

978-1-5386-2542-2/18/$31.00 © 2018 IEEE

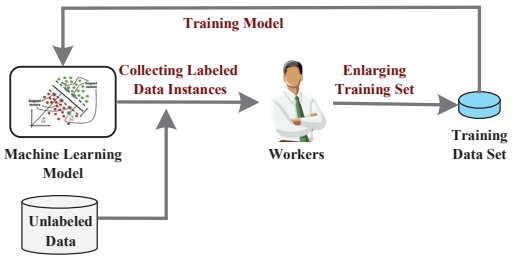

Fig. 1. The multi-round crowdsensing framework.

II. System Model and Problem Formulation

A. System Model

We first introduce the system model. A crowdsensing system consists of a data collector and many workers (e.g., smartphone users). The data collector has an unlabeled data set Q and needs a machine learning model to perform analysis on Q. The data collector needs to collect a set of labeled data from workers to train the machine learning model. To bootstrap the data collection process, a small set of labeled data serves as the initial training set D, and an initial machine learning model $f(\cdot)$ is trained with D. The data collector has a limited budget $\mathcal{B}$. The total cost of accepted instances should not exceed $\mathcal{B}$. The objective of the data collector is to collect a set of labeled data under the budget limit $\mathcal{B}$ to achieve the highest performance of the final trained model.

B. Crowdsensing Framework

We next introduce our crowdsensing framework with multiple rounds of data collecting and model training, as illustrated in Fig. 1. In round j, the data collector first collects a batch of labeled data under the budget in this round B_j, then enlarges the training set with the collected data instances and finally re-trains the machine learning model. The round continues recurrently until the total budget $\mathcal{B}$ is depleted.

The data collector can specify how to allocate the total budget $\mathcal{B}$ to each round. We provide a common budget allocation scheme adopted by most works about active learning [4]. In the common scheme, the total budget $\mathcal{B}$ is allocated equally to each round. Hence, the budget in each round is $B_j = \mathcal{B}/r$, where r is the number of rounds.

The interactions between the data collector and workers in each round are as follows. At the beginning of each round, the data collector announces the data type, such as the activity data sensed by the gyroscope and the accelerometer in smartphones. If necessary, the data collector would provide smartphone apps to help workers to collect data. Then, upon a worker arriving, the following interactions between the data collector and the arriving worker occur. First, the worker i submits a data instance x_i and the cost c_i for this instance. Second, the data collector decides whether or not to accept this instance and informs worker i of the decision. Third, worker i submits the label if the instance is accepted. Fourth, the data collector pays worker i according to the submitted cost.

C. Data Utility

To train a learning model with the highest performance, we should select data instances which increase the performance of the trained model most. We use data utility of a data batch to indicate its potential value for improving the performance of the model. In this paper, we make use of active learning strategies to construct the data utility model.

We adopt a data utility model, which is a combination of uncertainty and weighted density. Uncertainty measures how uncertain the learning model is about classifying an instance. It is the most intuitive and most widely used active learning strategy. However, it is prone to outliers. Therefore, we incorporate density. Density evaluates how representative a data instance is of data in Q, which serves as the test set. We modify the classic density measure [5] by further considering the importance of each data in Q.

1) Uncertainty: By uncertainty, we always choose the instances that are hardest (or most ambiguous) to the learning model. There are many ways to measure uncertainty of an instance, such as confidence-based, margin-based and entropy-based uncertainty measures [6]. For example, margin of an data instance x is defined as $P(\hat{y}_1|x,\Theta) - P(\hat{y}_2|x,\Theta)$. Label $\hat{y}_1$ and $\hat{y}_2$ are the first and second most likely predictions under the model $f(\Theta)$. The larger margin indicates less uncertainty.

2) Weighted Density: Choosing the instance hardest to the learning model might not always be the best strategy. The chosen instance may be a outlier. An alternative strategy would be to collect an instance representative of uncertain data instances in Q. This strategy is different from the classic density-based strategy, which assumes that all data instances in Q have the same importance. We emphasize most uncertain instances in Q.

The weighted density of an instance x is defined as $\sum_{x' \in Q} u(x') sim(x, x')$. $u(x')$ is uncertainty of x', serving as the weight. $sim(x, x')$ is the similarity between x and x'. When $sim(x, x')$ is larger, data x is more representative of x'. As a data instance (a graph, a sentence or a piece of sensor data) is usually represented as a vector, we can just adopt similarity measurements for vectors, such as Euclidean distance and cosine similarity.

Note that we cannot completely ignore uncertainty. When the learning model is highly confident of an instance, selecting that instance wastes budget because the corresponding label will most likely agree with the learning model. Therefore, defining the data utility of an instance x as its uncertainty $u(x)$ multiplied by its weighted density seems to be good.

Definition 1 (Data utility). *Given the unlabeled data set Q, the learning model $f(\cdot)$ obtained in the last round, data utility of a data batch S collected in the current round for improving the performance of the learning model is*

$$V(S) = \sum\nolimits_{x \in S} \left(u(x) * \sum\nolimits_{x' \in Q} u(x') sim(x, x')\right) \quad (1)$$

D. Problem Formulation

In a round, given the current trained model $f(\cdot)$, a set Q of unlabeled data instances and budget B, the general objective

of data collector is to increase the performance of $f(\cdot)$ to the maximum extent, which can be achieved by solving the data utility maximization problem defined as follows. Because we only focus on one round, we omit the subscript of B_j.

Definition 2 (**Data utility maximization problem**). *Let B denote the budget in a round. Under the constraint that the total cost of S does not exceed budget, we try to find a labeled data set S with the largest data utility.*

$$\begin{aligned} &max \ \ V(S) \\ &s.t. \sum_{x_i \in S} c_i \leq B \end{aligned} \tag{2}$$

Note that it is nontrivial to solve this problem. It is NP-hard even in the offline scenario.

III. Online Algorithm

A. Overview

In this section, we propose an online algorithm to solve our problem. In each round, our online algorithm (Alg. 1) proceeds in multiple stages. In each stage, it accepts data instances according to an efficiency threshold (to be defined later). The data instance whose efficiency is larger than the threshold is accepted. We propose another algorithm (Alg. 2) for updating the efficiency threshold at the end of each stage according to the sample set consisting of all instances arriving before.

Note that each stage is an accepting process as well as a sampling process, so all workers have an opportunity to be accepted. Some online algorithms based on the threshold reject workers arriving earlier and make use of them to compute the value of the threshold. These works are unfair for workers who come earlier. Since these workers have no chance to be accepted no matter how low the cost is or how high the utility contribution is. The unfairness encourages users to come late and even results in task starvation.

We introduce how to partition the total time T of a round into multiple stages. We recursively cut the total time T according to the proportion r, such as 0.3. More specifically, we first cut the total time T into two stages. The length of the first stage is rT and the remaining time, i.e., $(1-r)T$ constitutes the second stage. We then cut the first stage into two stages with length r^2T and $(1-r)rT$, respectively. We repeat this operation until the length of the shortest stage is 1. The time length of each stage is increasing. This is because the sample size is small at the beginning, and the efficiency threshold is not accurate. We have to update the threshold frequently.

The number of stages n depends on the piecewise point r. When r is smaller than 0.5, the shortest stage is the first stage with length $r^{n-1}T$. Set the length of the shortest stage to 1, i.e., $r^{n-1}T = 1$. We have $n = \left\lfloor log_r^{1/T} + 1 \right\rfloor$. If r is larger than or equal to 0.5, the shortest stage is the second stage with length $r^{n-2}T - r^{n-1}T$. Then, the number of stages is $\left\lfloor log_r^{1/((1-r)T)} \right\rfloor$. The i-th stage ends at time-slot $T_i = \lfloor r^{n-1}T \rfloor$. The budget that can be used before the end of the i-th stage is $(T_i/T)B$.

Algorithm 1: Selecting Instances

Input: Budget B, deadline $\mathcal{T}$, unlabeled data set Q, $\{T_i | i = 1 \cdots m\}$ and $\{B_i | i = 1 \cdots m\}$
Output: The collected data set $\mathcal{D}$.

```
(t, j, ρ, S, S') ← (1, 1, ϵ, ∅, ∅)
while t ≤ T do
    if there is an instance x_i arriving at time step t then
        if c_i ≤ V_i(S)/ρ ≤ B_j − Σ_{x_k∈S} c_k then
            S = S ∪ {x_i}
        end if
        S' = S' ∪ x_i
    end if
    if t = T_j then
        Update efficiency threshold ρ with S' and B_j.
        j = j + 1
    end if
    t = t + 1
end while
```

B. Selecting Data Instances

We first define the important concept of efficiency to be used by the online algorithm for determining which instances to select. The efficiency of an instance is the utility contribution per cost with respect to the current set of selected instances. More formally, it is defined as follows.

Definition 3 (**Efficiency**). *Given the current set of selected instances, denoted by S, the contribution to the data utility of a newly coming instance x_i is*

$$V_i(S) = V(S \cup x_i) - V(S) \tag{3}$$

Then, the efficiency of instance x_i is the contribution per cost, i.e., $V_i(S)/c_i$.

With the definition of efficiency, we next explain the main idea of Alg. 1. A new instance x_i is accepted if both the following two conditions hold: (1) the efficiency is not less than the efficiency threshold of the current stage, and (2) the remaining budget can cover the cost.

The details of the algorithm are shown in Alg. 1. The current stage is j. The accepted instance set is S. The sample set is S', which is used for updating the efficiency threshold. We initially set a small efficiency threshold ϵ, which is used for making decisions at the first stage. In each stage, when a new instance x_i arrives, if the efficiency is not less than the efficiency threshold, and we can afford the cost with the remaining budget, we add this instance into S. Otherwise, we reject this instance. We add all arriving instances into the sample set S'. When it is the end time of a stage, the efficiency threshold is updated by Alg. 2. At the same time, we move to the next stage.

C. Updating Efficiency Threshold

Let O_j denote the optimal data utility achieved under budget B_j with the sample set S' arriving before the end of stage j. Then, the efficiency threshold should be updated to O_j/B_j. However, as shown in section II-D, finding the optimal data utility is NP-hard. Therefore, we estimate the upper bound of O_j through a greedy algorithm.

Algorithm 2: Updating Efficiency Threshold

Input: Budget B_j, sample set S', unlabeled data set Q.
Output: The updated efficiency threshold ρ.

$R = \emptyset$
$x_k = argmax_{x_i \in S'}(V_i(R)/c_i)$
while $c_k \leq B_j - \sum_{x_i \in R} c_i$ **do**
  $R = R \cup \{x_k\}$
  $S' = S' \backslash \{x_k\}$
  if $S' \neq \emptyset$ **then**
    $x_k = argmax_{x_i \in S'}(V_i(R)/c_i)$
  else
    break;
  end if
end while
if $S' = \emptyset$ **then**
  $\rho = \frac{V(R)}{\delta B_j}$
else
  $\rho = \frac{V(R \cup x_k)}{\delta B_j}$
end if

The details of the algorithm are shown in Alg. 2. It adopts a greedy strategy, which always selects the instance with the largest efficiency with respect to the current set of selected instances, denoted by R. The algorithm stops when the remaining budget cannot cover the cost of the next candidate.

When the algorithm stops, if $S' \neq \emptyset$, $V(R \cup x_k) \geq (1 - \frac{1}{e})O_j$, where x_k is the next candidate [7]. Therefore, the upper bound of O_j is $\frac{V(R \cup x_k)}{1-1/e}$. If $S' = \emptyset$, $V(R) = O_j$ is the exact upper bound. Finally, we set the updated efficiency threshold to be $\frac{V(R \cup x_k)}{\delta B_j}$ when $S' \neq \emptyset$ and $\frac{V(R)}{\delta B_j}$ when $S' = \emptyset$, where $\delta = (1 - 1/e)\delta'$. We call δ' as shrinkage factor and set $\delta' > 1$ to obtain a slight underestimate of the efficiency threshold for guaranteeing that enough instances are accepted and avoiding wasting budget. We find the best δ' to optimize the competitive ratio in the next section.

IV. Theoretical Analysis

We present a theoretical analysis to demonstrate that the proposed online mechanism has a competitive ratio.

Theorem 1. *Assume that the contribution of any instance is at most $\frac{1}{\lambda}$ of the offline optimal data utility. Then, when $\frac{1}{\lambda}$ is infinitely small, r is 0.4390, and δ' is 4.6048, the competitive ratio is 0.1218.*

Proof. We partition the whole time length T into two parts. The first part includes stage 1 to stage $n-1$ and the second part is the last stage.

Let R be the set of selected instances in the offline optimal solution. Define R_1 and R_2 as the subsets of R that appear in the first part and the second part, respectively. In the first part, we obtain the sample set S'. Thus, we have $R_1 = R \cap S'$, and $R_2 = R \cap U \backslash S'$, where U is the set of all arriving bids before the time slot T. Let R'_1 denote the set of selected instances by Alg.2 based on the sample set S' and the budget rB. Then, the efficiency threshold used in the last stage is $\rho = \frac{V(R'_1 \cup b_I)}{(1-1/e)\delta' rB} \geq \frac{O_{n-1}}{\delta' rB}$. Let R'_2 denote the set of selected instances by Alg. 1 in the last stage. Assume that the contribution of a instance is at most $V(R)/\lambda$, where λ is very large.

Since the costs and contributions of all users are i.i.d, they can be selected in the set R with the same probability. We have that $E(|R_1|) = r|R|$ and $E(|R_2|) = (1-r)|R|$. Because the contribution of each instance can be seen as an i.i.d random variable, we can obtain that $E(V(R_1)) = rV(R)$ and $E(V(R_2)) = (1-r)V(R)$. The expected sum of costs of instances in R_1 and R_2 are rB and $(1-r)B$, respectively. Since O_{n-1} is the optimal data utility achieved with the budget rB and the sample set S', it can be derived that: $E(O_{n-1}) \geq E(V(R_1)) = rV(R)$.

We study the achieved competitive ratio by comparing the data utilities achieved by the selected instance set R'_2 in the last stage and the intersection R_2 between the optimal instance set R and the set of arriving instances in the last stage. Consider two cases as follows.

Case 1: The sum of costs of instances in R'_2 is at least βB, $\beta \in (0, 1-r]$. In this case, since each selected instance has the efficiency at least $\rho \geq \frac{O_{n-1}}{\delta' rB}$, we have $V(R'_2) \geq \rho\beta B \geq \frac{\beta O_{n-1}}{\delta' r} \geq \frac{\beta}{\delta'} V(R)$.

Case 2: The sum of costs of instances in R'_2 is less than βB, $\beta \in (0, 1-r]$. There are two reasons leading to that instances in R_2 not being selected in R'_2. The first reason is the efficiencies of some instances in R_2 are less than ρ. Even if all instances in R_2 are such instances, their expected total cost is at most $(1-r)B$. The expected total loss due to such instances is at most $\rho \cdot (1-r)B \leq \frac{(1-r)O_{n-1}}{(1-1/e)\delta' r}$.

Furthermore, there is not enough budget to cover the costs of some instances in R_2 whose efficiencies are not less than ρ. It means that the cost of such a instance is larger than $(1-r-\beta)B$. Otherwise, selecting this instance would not lead to the total cost of R'_2 exceeding the stage budget $(1-r)B$. Therefore, the number of such instances in R_2 is at most $\frac{(1-r)B}{(1-r-\beta)B}$. Since the contribution of such a instance is at most $V(R)/\lambda$, the expected total loss due to such instances is at most $\frac{(1-r)V(R)}{(1-r-\beta)\lambda}$.

Therefore, we have that

$$\begin{aligned} E(V(R'_2)) &\geq E(V(R_2)) - \frac{(1-r)O_{n-1}}{(1-1/e)\delta' r} - \frac{(1-r)V(R)}{(1-r-\beta)\lambda} \\ &\geq (1-r)V(R) - \frac{(1-r)O_{n-1}}{(1-1/e)\delta' r} - \frac{(1-r)V(R)}{(1-r-\beta)\lambda} \\ &\geq (1 - r - \frac{1-r}{(1-1/e)\delta' r} - \frac{1-r}{(1-r-\beta)\lambda})V(R) \end{aligned}$$

Consider case 1 and case 2, the ratio of $E(R'_2)$ to $E(R)$ is at least β/δ' if the following equation is satisfied.

$$(1 - \frac{1}{(1-1/e)\delta' r} - \frac{1}{(1-r-\beta)\lambda})(1-r) = \beta/\delta'$$

Therefore, we can obtain the optimal ratio by solving the following optimization problem when λ is infinitely large:

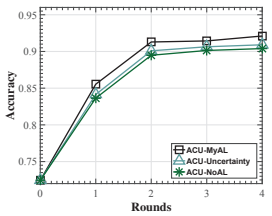

Fig. 2. Accuracy achieved in each round under different data utility models in two-class classification.

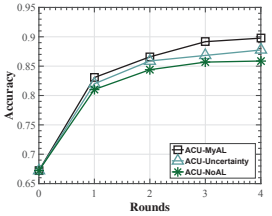

Fig. 3. Accuracy achieved in each round under different data utility models in multiclass classification.

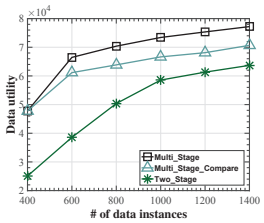

Fig. 4. Data utility vs. # of coming instances under different algorithms.

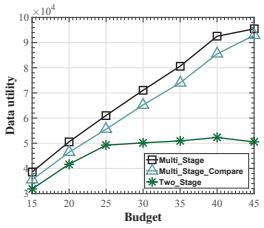

Fig. 5. Data utilities vs. budget under different online algorithms.

$$
\begin{aligned}
&max \quad \beta/\delta' \\
&s.t. \quad (1-\frac{1}{(1-1/e)\delta' r}-\frac{1}{(1-r-\beta)\lambda})(1-r)=\beta/\delta' \\
&s.t. \quad \beta \in (0, 1-r]
\end{aligned}
\tag{4}
$$

According to the optimization result, we achieve the competitive ratio 0.1218 when r is 0.4390 and δ' is 4.6048.

□

V. Evaluation

A. Methodology and Settings

We evaluate our data utility model and online algorithm based on the application of human activity recognition. Human activity recognition aims to identify the actions carried out by humans given a set of observations. The trace comes from the Human Activity Recognition Using Smartphones Dataset in the UCI Machine Learning Repository [8]. The dataset was collected from 30 people. Each person performed six activities (walking, walking upstairs, walking downstairs, sitting, standing, lying down) wearing a smartphone (Samsung Galaxy S II) on the waist. The labels were manually annotated by analyzing video recordings. The sensor signals were pre-processed by applying noise filters and then sampled in fixed-width sliding windows. As a result, 561 features in time and frequency domain are selected.

We perform two-class and multiclass classification on this dataset, respectively. In the two-class classification problem, the classes are walking downstairs and walking upstairs, and we choose 20 most relevant features by checking the mutual information between each feature and each class. We have implemented a logistic regression classifier for this problem. For the multiclass classification problem, all 561 features are used and a one-vs-one SVM classifier with linear kernels is applied. In both problems, the uncertainty measure is based on the margin of an data instance, and the similarity between two instances is defined as the cosine similarity.

Other settings are as follows. The cost of a worker for labeling one instance confirms the truncated normal distribution from $[\$0.05, \$0.15]$. The number of rounds is 4. Each data point reported below is the average result of 20 independent runs under the same setting.

B. Efficiency of Data Utility Model

In this subsection, we compare our data utility model with other two models to show the efficiency and advantage of our model. More specifically, we compare the accuracy of the trained model under different data utility models. The first compared data utility model only considers uncertainty, called Uncertainty. The second one is selecting instances randomly, called NoAL. To eliminate the effect of various costs of data collecting, we adopt the simplified budget constraint that the number of selected instance is limited in each round. The algorithm used for instance selecting also has effect on the accuracy of the trained model. To remove this effect, we adopt the offline approximate-optimal algorithm to choose instances.

In Fig. 2, we report the accuracy of the trained model in each round under the three data utility models in the two-class classification problem. In each round, the number of coming instances is 150 and we select 15 instances. The initial training set has 5 instances. The unlabeled data set Q (the test data set) consists of 368 instances. The accuracy of the trained model increases in each round, because more data are obtained. Our data utility model performs best. Uncertainty is worse because it does not consider density. NoAL is the worst. The accuracy achieved by our data utility model in the second round is higher than that achieved by NoAL in the last round. That is to say, our data utility model can save at least $3/4$ times the budget spent by NoAL.

We report the accuracy of the trained model achieved for the multiclass classification problem in Fig. 3. In each round, the number of coming instances is 600 and we select 80 instances. The initial training set has 40 instances. The set Q consists of 1201 instances. Our strategy still performs best.

C. Comparison with Other Algorithms

We evaluate our online algorithm with other two algorithms called Two-Stage [9] and Multi-Stage-Compare [10]. Tow-Stage observes the utility contribution of the uploaded data and does not accept any one in the first stage. In the second stage, it accepts a data instance if its efficiency reaches the efficiency threshold computed according to the instances arriving in the first stage. The competitive ratio is $min(0.0208, 0.2083-1/\lambda)$ [9]. The shrinkage factor is 6 and the piecewise point is 0.5. Multi-Stage-Compare is similar with our algorithm. The difference is that it applies another offline algorithm to update the efficiency threshold [11]. What's more, the different piecewise

point and shrinkage factor are adopted. The competitive ratio of Multi-Stage-Compare is about 0.026 when λ is infinitely large [10]. It is much smaller than ours. We focus on one round of the multiclass classification problem.

In Fig. 4, we study the data utilities achieved by the three online algorithms when the number of coming instances increases from 400 to 1400. The budget is \$30. The data utilities increase as the number of coming users becomes larger because better data instances can be selected among more candidates. Our algorithm performs best. Two-Stage is worst. More candidates can decrease the performance gap between Two-Stage and the two algorithms with multiple stages.

We report the data utilities achieved by the three online algorithms, when the budget increases from \$15 to \$45 in Fig. 5. Our algorithm is still the best. The performance gap between Two-Stage and the other two algorithms increases with the larger budget. This is because Two-Stage only accepts data instances in the second stage. Therefore, the candidate set is not enough.

VI. Related Work

Data Collecting by Crowdsensing. There are many works about data collecting by crowdsensing [12] [13] [14] [15]. For example, Yang et al. [14] focus on truth estimation on one Point of Interest, e.g., estimating the noise level on one PoI. They try to derive the quality of each submitted data by calculating the distance from the data to the center of all data instances. The estimated truth is the weighted center of all data instances, where the weight of a data instance is the quality.

Nonetheless, these works can hardly be applied to our problem. The collected data instances are not for training models. Therefore, no one bridges the gap between the performance of the trained model and the data collection process.

There is one work collecting data by crowdsensing for training learning models [16]. However, they consider a simpler setting where the cost of each data instance is same. It is not realistic in the real world. At the same time, they update the learning model upon getting a new data instance, which is very time-consuming. Moreover, they do not consider density of each data.

The Secretary Problem. In the online scenario, our data utility maximization problem is actually a submodular secretary problem with a knapsack constraint. The secretary problem is a kind of problems where applicants arrive online in a random order. A decision about each particular applicant is to be made immediately after the interview. Generally, once rejected, an applicant cannot be recalled. In our work, a data instance is an applicant. Our secretary problem has a knapsack constraint.

There are two related works [9] [10] solving this problem. They are the compared algorithms in the evaluation. Two-Stage is unfair and Multi-Stage-Compare has a much lower competitive ratio than our work.

VII. Conclusion

In this paper, we have studied the data utility maximization problem under the budget constraint when leveraging crowdsensing in machine learning. We propose the multi-round crowdsensing framework. We come up with a novel data utility model to bridge the gap between the performance of the trained model and the collected instances. We further design a fair online algorithm and achieve a non-trivial competitive ratio. The experimental results demonstrate that the performance of the trained model is significantly improved by maximizing the data utility using our algorithm.

Acknowledgements

This research is supported in part by 973 Program (No. 2014CB340303), NSFC (No. 61772341, 61472254, 61572324 and 61170238), NSF grants (CNS 1757533, CNS1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672). This work is also supported by the Program for Changjiang Young Scholars in University of China, the Program for China Top Young Talents, and the Program for Shanghai Top Young Talents.

References

[1] Stephanie R. Debats, Lyndon D. Estes, David R. Thompson, and Kelly K. Caylor, "Integrating active learning and crowdsourcing into large-scale supervised landcover mapping algorithms", in *PeerJ PrePrints*, 2017.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges.", *IEEE Communications Magazine*, vol. 49, pp. 32–39, 2011.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine", in *Ambient Assisted Living and Home Care*, 2012.

[4] Barzan Mozafari, Purnamrita Sarkar, Michael J. Franklin, Michael I. Jordan, and Samuel Madden, "Scaling up crowd-sourcing to very large datasets: A case for active learning", in *PVLDB*, 2014.

[5] Burr Settles, "Active learning", *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[6] Yifan Fu, Xingquan Zhu, and Bin Li, "A survey on instance selection for active learning", *Knowledge and Information Systems*, vol. 35, pp. 249–283, 2013.

[7] Thibaut Horel, "Notes on greedy algorithms for submodular maximization", 2015, https://thibaut.horel.org/submodularity/notes/02-12.pdf.

[8] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones.", in *ESANN*, 2013.

[9] Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Morteza Zadimoghaddam, "Submodular secretary problem and extensions", *ACM Trans. Algorithms*, vol. 9, no. 4, pp. 32:1–32:23, 2013.

[10] Dong Zhao, Xiang-Yang Li, and Huadong Ma, "Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully", *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 647–661, 2016.

[11] Y. Singer, "Budget feasible mechanisms", in *IEEE FOCS*, 2010.

[12] Jacopo De Benedetto, Paolo Bellavista, and Luca Foschini, "Proximity discovery and data dissemination for mobile crowd sensing using lte direct", *Computer Networks*, vol. 129, pp. 510–521, 2017.

[13] Jing Wang, Jian Tang, Guoliang Xue, and Dejun Yang, "Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems", *Computer Networks*, vol. 115, pp. 100–109, 2017.

[14] Shuo Yang, Fan Wu, Shaojie Tang, Xiaofeng Gao, Bo Yang, and Guihai Chen, "On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing", *IEEE Journal on Selected Areas in Communications*, vol. 35, pp. 832–847, 2017.

[15] Kai Han, Chi Zhang, and Jun Luo, "Taming the uncertainty: Budget limited robust crowdsensing through online learning", *IEEE/ACM Transactions on Networking*, vol. 24, pp. 1462–1475, 2016.

[16] Qiang Xu and Rong Zheng, "When data acquisition meets data analytics: A distributed active learning framework for optimal budgeted mobile crowdsensing", in *IEEE Infocom*, 2017.