Environment Design in Human Computation

Chien-Ju Ho[1], **Yen-Ling Kuo**[2], **Jane Yung-jen Hsu**[2]
[1]UCLA, [2]National Taiwan University
cjho@cs.ucla.edu, {b94029, yjhsu}@csie.ntu.edu.tw

Abstract

Human computation is a new research area that takes advantage of human brain power to solve problems that computers cannot efficiently solve yet. In human computation systems, such as Games with A Purpose (GWAP) and crowdsourcing markets, users are given computationally hard tasks, and get rewards by accomplishing the tasks. In this work, we are interested in how to design a human computation system. In particular, we focus on the situation where developers cannot re-design the whole application but can only make limited changes to the systems. We model this situation as an environment design problem [10] with multiple agents. Under the assumption that agents do not interact, we define a m by n decision matrix, whose entry $[i, j]$ represents the utility value of historical decision agent i made in environment j. Our goal is to find the environment for each agent to maximize the system payoff while minimizing the number of entries needed. To achieve this goal, we propose two different approaches, agent type elicitation and collaborative filtering, under different conditions. We also derive the convergence bound on the number of entries needed for each method.

1 Introduction

Human computation aims to solve computationally-hard problems, e.g. image tagging or commonsense collection, by utilizing collective human brain power. There are a variety of applications available nowadays. Games with A Purpose (GWAP) [7] engage players in an online game and let them help solve tasks while having fun. Crowdsourcing markets, such as Amazon Mechanical Turk [1], provide platforms for workers to contribute their brain power in exchange for monetary rewards. Peer productions systems, e.g. Wikipedia or Yahoo! Answers, let online users construct knowledge base for common good.

Despite the impressive progress of developing applications to solve real-world problems [7], little study is conducted in theory to guide the system design of human computation. von Ahn and Dabbish [9] discussed the general design principles of Games with A Purpose. Some other researchers [5, 2, 4] analyzed the incentive structure of human computation systems in a game theoretic approach. While these projects addressed the design of the system mechanisms, many situations arise when the developers do not have full privilege to modify the systems. For example, developers on Mechanical Turk cannot change the way they interact with the workers. They can only make little modifications, such as the size of payments, the tasks chosen for worker, or the task descriptions, to encourage users complete the tasks quickly and accurately.

In this work, we focus on problems in which developers can only make limited changes to the system. The goal of the developers is to maximize their goal function, e.g. spending least money, taking least amount of time, or getting the best results, by finding the best system setting for each user. Some example problems include: how do developers on Mechanical Turk determine how much to pay for their tasks? How does the ESP gamer server [8] decide which images should be shown

[1]http://www.mturk.com

in the game and which players should be paired? In the following discussion, we call these settings the *environments* for the user. Our goal is to develop methods for finding the best environment for each user by observing the user responses to the environment change.

We model this problem as an environment design problem [10] in a multiagent setting. Given that users do not interact, we define a decision matrix which contains the utilities of decisions made by users in the environments. The fundamental assumption being made in this paper is that users will take actions similar to the actions taken by like-minded users or users with similar abilities. This assumption is modeled by introducing the concept of agent types and low rank matrix approximation. We analyze our model by first assuming the existence of the agent types. If the agent types are known a prior, the convergence is $\mathcal{O}(m/k)$ times faster than the algorithm proposed in single-agent environment design [10]. If the agent types are unknown, we propose an algorithm for agent type elicitation and show the logarithmic convergence. We then relax the assumption of agent types by proposing collaborative filtering approach and give a detailed discussion.

2 Related Work

The concept of environment design is first proposed by Zhang and Parkes [11]. They considered the case the interested party tries to influence the agent's behavior in a Markov Decision Process (MDP) setting. The interested party cannot force the agent behaviors but can only make limited changes to the reward function. They then extended their work to a more general framework [10] and provided a binary search elicitation algorithm. They showed the conditions when their algorithms can converge linearly and logarithmically. Inspired by these work, Huang et al. [3] applied environment design framework for automatic task design in Amazon Mechanical Turk.

In this work, we focus more on utilizing the collective information from multiple users in human computation systems. In work on k-implementation [6], they studied the problem where the interested party provides non-negative rewards for the different strategy profiles to implement the desired outcome. While this work is closely related to multiagent environment design, their work is different from ours in the following perspectives. First, they modeled an one-shot game where agents' interactions are considered. We focus more on the repeated-game setting where the agent's past decisions are used to elicit agent parameters. Besides, we do not consider interactions of agents. They change the environment via adding non-negative reward to the payoff function in game. We discuss a more general framework for changing the environments.

3 Model of Environment Design in Human Computation

Our model is a direct extension of Zhang and Parkes's work [10]. An environment design problem in human computation consists of agents $\{i : i \in N\}$, environments $\{e : e \in \varepsilon\}$, agent decision space $\{x : x \in X\}$, agent model parameters $\{\theta_i : \theta_i \in I\}$, agent decision function $f : I \times \varepsilon \rightarrow 2^X$, and a utility function $u : X \rightarrow R$. The goal of the problem is to maximize the total utility value by finding the best environment for each agent i using the past histories of the agent's behaviors.

Take developers on Mechanical Turk for example, agents are the workers, environments are the possible modifications developers can make, agent decision space is the set of possible actions workers can take, agent model parameters are the private information of workers, and the utility function describes how desirable the workers actions are to the developers.

In addition to the model, we define a m by n matrix called "decision matrix", where m is the number of agents and n is the number of environments. The (i, j)th entry of the matrix represents the utility value of the actions taken by agent i in environments j. The higher the utility value is, the more desirable this action is to the developer.

4 Analysis

Clearly, without any assumptions about the agents, we cannot do any better than the one-agent environment design problem. The fundamental assumption being made in this work is that agents will take actions similar to the actions that like-minded agents or agents with similar abilities take.

4.1 Agent type elicitation

We first assume that agents fall into a relatively small set of "types" compared to the number of agents. Agents of the same type will take the same actions in all environments. In the following discussion, the number of agents is denoted as m, and the number of agent types is denoted as k.

If the types of agents are known a priori, the agent behaviors of the same type can be used together to elicit agent model parameters. Therefore, the convergence speed is trivially $O(m/k)$ times faster than the algorithm proposed in [10]. For example, if the environment denotes the language translation tasks on Mechanical Turk, and the agent type denotes the languages workers speak, it is possible to get the agent type information a prior by asking users to answer a brief survey or to pass a qualification test. However, it is usually not possible to know the types of agents in advance. Therefore, we propose an algorithm using pre-testing rounds to first elicit agent types and then use the agent type information to help speed up the elicitation of model parameters. The algorithm for agent type elicitation is stated as followed:

1. Pick environment set E, where $|E| = r$.
2. For each agent, present him/her all the environments in E and observe his/her decisions.
3. Classify agents.

It is clear that the performance of the algorithm highly depends on the choice of the environment set E. In the following definition, we provide a measure for how different the agent types are given an environment set.

Definition 1 *(p-separable) The agent types are called* p-separable *in environment set E if for any two agents of different types, the fraction of the environments in E that they choose the actions with the same utility value is less then p.*

The smaller value of p means agents are less likely to take the same actions in the environment set. Therefore it is easier to distinguish different types of agents in the environments. In real-word application, the developers usually have some prior knowledge about the environments, e.g. task types in Mechanical Turk. Therefore, they could choose some representative environments (minimizing p) to speed up the convergence of classifying agents.

Lemma 1 [2] *If the agent types are* p-separable *on the environment set E and $|E| = r$, the probability of eliciting the wrong agent type after observing r environments would be less than $(k-1)p^r$ if $p^r \leq \frac{3}{k-3}$.*

Assume there are $k = 10$ types of agents, and the environment set E is 0.5-separable for agents. Then the probability of wrongly classifying the agents are less than 1% if $r = 10$.

4.2 Collaborative filtering

In this section, we relax the constraint of agent type assumption and propose a collaborative filtering approach. If we look into the decision matrix, which records the utility values of past agent actions in the environments, the problem we are solving is to find the environment with highest utility value for each agent, given only part of the entries in the matrix. This is actually an *environment recommendation* problem in which the developers aim to find the best environment for each user to maximize the utility value.

Since we model the problem in a standard format of collaborative filtering, any collaborative filtering algorithms can be applied to solve this problem. However, in this work, we are more curious about if we can design an algorithm which can achieve high accuracy of recommendations with few matrix entries by actively choosing the environments agents are experiencing.

Given the assumption that agents take actions similar to the actions taken by like-minded agents, it is implied that the decision matrix has a good low rank approximation [3]. In the following discussion,

[2] The proofs are omitted due to the limitation of space and will be available in a longer version of the paper.

[3] A matrix has a good low rank approximation means it can be approximated by using a small number of singular values in SVD decomposition

we first talk about the work in [1] which also assumed low rank matrix. They provided an algorithm to complete the low rank matrix and derived the convergence bound. We then interpret how their work can be applied in our problem settings. In their algorithm, they require to random sample c columns and r rows of the matrix A. Give these sampled entries, they can provide a prediction matrix $\hat{A}$, where the expected error between A and $\hat{A}$ satisfies the following lemma.

Lemma 2 *([1]) Let $\sigma_t, t = 1, \ldots, \rho$ denote the singular value of A. Then, the algorithm provided above shows the error satisfies*

$$E(\|A - \hat{A}\|_F^2) \leq \sum_{t=k+1}^{\rho} \sigma_t^2 + (\sqrt{\frac{k}{c}} + \frac{k}{r})\|A\|_F^2$$

In the above lemma, picking $\mathcal{O}(k/\epsilon)$ rows and $\mathcal{O}(k/\epsilon^2)$ column bounds the expectation of relative error to $\lambda + \epsilon$, where λ is the relative error between the actual matrix and the low-rank matrix. In our setting, if we are able to find some volunteers to test all the possible environments (r rows) and ask the remaining agents to perform test rounds (c columns), Lemma 2 can be used to provide an error bound. While asking a small number of agents to test all the environments would sometimes be infeasible, we could still get some intuitions about how good the approximation could be given the low rank assumption. Developing new algorithms to avoid the "sampling all environment" constraints would be an interesting future research direction.

5 Conclusion

In this work, we extend the environment design problem to settings with multiple agents in the context of human computation systems. We propose two approaches under different assumptions and give detailed discussion. The formulation and algorithms provide solutions for developers in human computation systems to find the environment settings maximizing their goal functions. Future work should continue to explore the aspects of agent interactions, integrations to the mechanism design of human computation, and applying the result to real-world applications.

References

[1] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proceedings of STOC'02*, pages 82–90, New York, NY, USA, 2002. ACM.

[2] C.-J. Ho and K.-T. Chen. On formal models for social verification. In *Proceedings of HCOMP '09*, pages 62–69, New York, NY, USA, 2009. ACM.

[3] E. Huang, H. Zhang, D. C. Parkes, K. Z. Gajos, and Y. Chen. Toward automatic task design: a progress report. In *Proceedings HCOMP '10*, pages 77–85, New York, NY, USA, 2010. ACM.

[4] S. Jain, Y. Chen, and D. C. Parkes. Designing incentives for online question and answer forums. In *Proceedings of EC'09*, pages 129–138, New York, NY, USA, 2009. ACM.

[5] S. Jain and D. C. Parkes. A game-theoretic analysis of games with a purpose. In *WINE 2008*, pages 342–350, December 2008.

[6] D. Monderer and M. Tennenholtz. k-implementation. In *Proceedings of EC'03*, pages 19–28, New York, NY, USA, 2003. ACM.

[7] L. von Ahn. Games with a purpose. *IEEE Computer Magazine*, 39(6):92–94, 2006.

[8] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of CHI'04*, pages 319–326. ACM Press, 2004.

[9] L. von Ahn and L. Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[10] H. Zhang, Y. Chen, and D. C. Parkes. A general approach to environment design with one agent. In *Proceedings of IJCAI'09*, Pasadena, CA, USA, 2009. ACM.

[11] H. Zhang and D. C. Parkes. Value-based policy teaching with active indirect elicitation. In *Proceedings of AAAI'08*. AAAI Press, 2008.