The Importance of Being Earnest in Crowdsourcing Systems

Alberto Tarable\textsuperscript{1}, Alessandro Nordio\textsuperscript{1}, Emilio Leonardi\textsuperscript{1,2}, Marco Ajmone Marsan\textsuperscript{1,2,3}

\textsuperscript{1}CNR-IEIIT
\textsuperscript{2}Politecnico di Torino
\textsuperscript{3}IMDEA Networks Institute

Cavalese, Jan 14-th, 2015
Crowdsourcing systems

- provide a significant new type of work organization paradigm
Crowdsourcing systems

- provide a significant new type of work organization paradigm
- solve a wide range of complex problems in a scalable way
Crowdsourcing systems

- provide a significant new type of work organization paradigm
- solve a wide range of complex problems in a scalable way
- integrate a large number of human and/or computer efforts
The key characteristic of crowdsourcing systems is that:
Introduction

The key characteristic of crowdsourcing systems is that:

- a *requester* structures his problem in a set of *tasks*
The key characteristic of crowdsourcing systems is that:

- a *requester* structures his problem in a set of *tasks*
- it assigns every task to a set of *workers*
The key characteristic of crowdsourcing systems is that:

- a requester structures his problem in a set of tasks
- it assigns every tasks to a set of workers
- workers provide unreliable answers, (for simplicity answers are assumed to be binary)
The key characteristic of crowdsourcing systems is that:

- a *requester* structures his problem in a set of *tasks*
- it assigns every task to a set of *workers*
- workers provide *unreliable answers*, (for simplicity answers are assumed to be binary)
- the correct task *solution* is obtained from answers through a *decision* rule
Assumptions

- $T$ binary tasks whose outcome is represented by i.i.d. uniform random variables (RV's) $\tau_1, \tau_2, \ldots, \tau_T$ over $\{\pm 1\}$, i.e., $\mathbb{P}\{\tau_t = \pm 1\} = \frac{1}{2}$, $t = 1, \ldots, T$
Assumptions

- $T$ binary tasks whose outcome is represented by i.i.d. uniform random variables (RV’s) $\tau_1, \tau_2, \ldots, \tau_T$ over $\{\pm 1\}$, i.e., $\mathbb{P}\{\tau_t = \pm 1\} = \frac{1}{2}$, $t = 1, \ldots, T$

- $W$ workers, each one modeled as a binary symmetric channel (BSC); i.e., providing a wrong answer with probability $p_{tw}$ and a correct answer with probability $1 - p_{tw}$
Normally every task is assigned to $K$ randomly chosen workers (uniform assignment). Better performance can be achieved by designing smarter assignment schemes and decision rules!
Normally

- every task is assigned to $K$ randomly chosen workers (uniform assignment)
- task decision is made by adopting a simple majority rule

Better performance can be achieved by designing smarter assignment schemes and decision rules!
Normally

- every task is assigned to $K$ randomly chosen workers (uniform assignment)
- task decision is made by adopting a simple majority rule

Better performance can be achieved by designing smarter assignment schemes and decision rules!
The performance of crowdsourcing systems has been explored, but...
Motivations

The performance of crowdsourcing systems has been explored, but most previous works assume no a-priori information about worker reliability (worker reputation - or earnestness) at the scheduler.
Motivations

The performance of crowdsourcing systems has been explored, but

- most previous works assume no a-priori information about worker reliability (worker reputation - or earnestness) at the scheduler

- smart decision rules exploit redundancy and correlation in the pattern of answers to infer an a-posteriori reliability estimate for every worker
The performance of crowdsourcing systems has been explored, but

- most previous works assume no a-priori information about worker reliability (worker reputation - or earnestness) at the scheduler
- smart decision rules exploit redundancy and correlation in the pattern of answers to infer an a-posteriori reliability estimate for every worker

Understanding the potential impact of a-priori information about worker reliability is extremely important
We provide the first systematic analysis of the potential benefits deriving from a-priori knowledge about the reputation of workers.
We provide the first systematic analysis of the potential benefits deriving from a-priori knowledge about the reputation of workers

- workers can be grouped into classes, each one composed of workers with similar accuracy and skills
We provide the first systematic analysis of the potential benefits deriving from a-priori knowledge about the reputation of workers:

- workers can be grouped into classes, each one composed of workers with similar accuracy and skills
  - each worker belongs to one of $K$ classes, $C_1, C_2, \ldots, C_K$
  - each class is characterized, for each task, by an average error probability $\pi_{tk}$, known to the requester

Two extreme scenarios are possible:

- **Full Knowledge**: the error probability of each worker in $C_k$ is deterministically equal to $\pi_{tk}$ for task $t$ (zero variance case)
- **Hammer-Spammer (HS)**: perfectly reliable and completely unreliable users coexists within the same class (maximum variance case)
We provide the first systematic analysis of the potential benefits deriving from a-priori knowledge about the reputation of workers

- workers can be grouped into classes, each one composed of workers with similar accuracy and skills
  - each worker belongs to one of $K$ classes, $C_1, C_2, \ldots, C_K$
  - each class is characterized, for each task, by an average error probability $\pi_{tk}$, known to the requester
- two extreme scenarios are possible:
  - **Full Knowledge**: the error probability of each worker in $C_k$ is deterministically equal to $\pi_{tk}$ for task $t$ (zero variance case)
  - **Hammer-Spammer (HS)**: perfectly reliable and completely unreliable users coexists within the same class (maximum variance case)
An allocation is a set of assignments of tasks to workers; a generic allocation corresponds to a set $G$ of pairs $(t, w)$ with $t \in \{1, \cdots, T\}$ and $w \in \{1, \cdots, W\}$. 

$O$ is the complete allocation set ($O$ is the set composed of all the $T \cdot W$ pairs $(t, w)$) 

We impose the following constraints: 

- A given task $t$ can be assigned at most once to a given worker $w$ 
- No more than $r_w$ tasks can be assigned to worker $w$ 
- The total number of assignments cannot be larger than $C$
An allocation is a set of assignments of tasks to workers; a generic allocation corresponds to a set $G$ of pairs $(t, w)$ with $t \in \{1, \ldots, T\}$ and $w \in \{1, \ldots, W\}$.

$O$ is the complete allocation set ($O$ is the set composed of all the possible $T \cdot W$ pairs $(t, w)$).
An allocation is a set of assignments of tasks to workers; a generic allocation corresponds to a set $\mathcal{G}$ of pairs $(t, w)$ with $t \in \{1, \ldots , T\}$ and $w \in \{1, \ldots , W\}$

$\mathcal{O}$ is the complete allocation set ($\mathcal{O}$ is the set composed of all the possible $T \cdot W$ pairs $(t, w)$)

We impose the following constraints:
An *allocation* is a set of assignments of tasks to workers; a generic allocation corresponds to a set $\mathcal{G}$ of pairs $(t, w)$ with $t \in \{1, \cdots, T\}$ and $w \in \{1, \cdots, W\}$.

$O$ is the complete allocation set ($O$ is the set composed of all the possible $T \cdot W$ pairs $(t, w)$).

We impose the following constraints:

- a given task $t$ can be assigned at most once to a given worker $w$
An *allocation* is a set of assignments of tasks to workers; a generic allocation corresponds to a set \( \mathcal{G} \) of pairs \((t, w)\) with \( t \in \{1, \cdots, T\} \) and \( w \in \{1, \cdots, W\} \).

\( \mathcal{O} \) is the complete allocation set (\( \mathcal{O} \) is the set composed of all the possible \( T \cdot W \) pairs \((t, w)\)).

We impose the following constraints:

- a given task \( t \) can be assigned at most once to a given worker \( w \)
- no more than \( r_w \) tasks can be assigned to worker \( w \)
An allocation is a set of assignments of tasks to workers; a generic allocation corresponds to a set $G$ of pairs $(t, w)$ with $t \in \{1, \cdots, T\}$ and $w \in \{1, \cdots, W\}$

$\mathcal{O}$ is the complete allocation set ($\mathcal{O}$ is the set composed of all the possible $T \cdot W$ pairs $(t, w)$)

We impose the following constraints:

- a given task $t$ can be assigned at most once to a given worker $w$
- no more than $r_w$ tasks can be assigned to worker $w$
- the total number of assignments cannot be larger than $C$
Greedy task assignment

The task assignment we propose to approximate the optimum behavior is a simple greedy algorithm that starts from an empty assignment \( G^{(0)} = \emptyset \), and at every iteration \( i \) adds to \( G^{(i-1)} \) the individual assignment \( (t, w)^{(i)} \), so as to maximize an objective function \( P() \):
Greedy task assignment

The task assignment we propose to approximate the optimum behavior is a simple greedy algorithm that starts from an empty assignment \( G^{(0)} = \emptyset \), and at every iteration \( i \) adds to \( G^{(i-1)} \) the individual assignment \( (t, w)^{(i)} \), so as to maximize an objective function \( P() \):

\[
(t, w)^{(i)} = \arg \max_{(t, w) \in O \setminus G^{(i-1)}, (G^{(i-1)} \cup \{(t, w)\}) \in \mathcal{F}} P(G^{(i-1)} \cup \{(t, w)\})
\]

The algorithm stops when no assignment can be further added to \( G \) without violating the cost constraint \( C \).
Several choices are possible for the objective function $P()$: 

1. $P_1 = 1 - \frac{1}{T} \sum_t P_e(t)$
2. $P_2 = 1 - \max_t P_e(t)$
3. $P_3 = \sum_{t=1}^T I(a_t; \tau_t)$
Several choices are possible for the objective function $P()$:

- $P_1 = 1 - \frac{1}{T} \sum_t P_{e,t}$
- $P_2 = 1 - \max_t P_{e,t}$
- $P_3 = \sum_{t=1}^{T} l(a_t; \tau_t)$
Majority rule: $\hat{r}_t(a_t) = \text{sgn}(\sum_w a_{tw})$
Decision Rules

- Majority rule: \( \hat{\tau}_t(a_t) = \text{sgn} \left( \sum_w a_{tw} \right) \)

- MAP rule: \( \hat{\tau}_t(a_t) = \text{sgn} \left( \sum_w a_{tw} \sigma_{k(w)} \right) \) with \( \sigma_{k(w)} = \log \frac{1 - \pi_{w,k(w)}}{\pi_{t,k(w)}} \)

Majority rule: $\hat{\tau}_t(a_t) = \text{sgn} \left( \sum_w a_{tw} \right)$

MAP rule: $\hat{\tau}_t(a_t) = \text{sgn} \left( \sum_w a_{tw} \sigma_k(w) \right)$ with $\sigma_k(w) = \log \frac{1 - \pi_{w,k(w)}}{\pi_{t,k(w)}}$

Low Rank Approximation (LRA) rule [1]:

$$\hat{\tau}_t(a_t) = \text{sgn} \left( \sum_w a_{tw} v_w \right)$$

where $v_w$ are the components of the leading right singular vector associated with the matrix of answers $[a_{tw}]$

“Majority voting” + “Uniform allocation” → “Majority”
Results: Considered Algorithms

- “Majority voting” + “Uniform allocation” → “Majority”
- “LRA” + “Uniform allocation” → “LRA uniform”
Results: Considered Algorithms

- “Majority voting” + “Uniform allocation” → “Majority”
- “LRA” + “Uniform allocation” → “LRA uniform”
- “LRA” + “Greedy allocation” → “LRA greedy”
Results: Considered Algorithms

- “Majority voting” + “Uniform allocation” → “Majority”
- “LRA” + “Uniform allocation” → “LRA uniform”
- “LRA” + “Greedy allocation” → “LRA greedy”
- “MAP” + “Greedy allocation” → “MAP greedy”
Results: a first scenario

- Number of i.i.d tasks:  $T = 100$
- 3 classes of workers:  $\pi_{t1} = 0.1$, $\pi_{t2} = 0.2$, $\pi_{t3} = 0.5$
- Number of workers per class:  $W_1 = 30$, $W_2 = 120$, and $W_3 = 150$
- Maximum number of tasks per worker:  $r_w = 20$
\( \beta \) is the average number workers per task
Hammer-Spammer

![Graph showing the average error probability (Pe) against β for different algorithms: LRA uniform, Majority, LRA greedy, and MAP greedy. The graph plots the error probability on a logarithmic scale from $10^{-5}$ to $10^0$. The β values range from 2 to 20.](image-url)
Results: a second scenario

- Two groups of 50 tasks each
- Error probabilities for the tasks in group 1 and 2 are given by
  \[
  \begin{align*}
  \pi_{t_11} &= 0.1, \pi_{t_12} = 0.25, \pi_{t_13} = 0.5 \\
  \pi_{t_21} &= 0.5, \pi_{t_22} = 0.25, \pi_{t_23} = 0.1 
  \end{align*}
  \]
- Number of workers per class: \( W_1 = 40, W_2 = 120, \) and \( W_3 = 40 \)
- Maximum number of tasks per worker: \( r_w = 20 \)
Several other results in the paper!
Main findings of our paper are:
Conclusions

Main findings of our paper are:

- even largely inaccurate estimates of workers’ reputation during task assignment → large improvements of system performance
Conclusions

Main findings of our paper are:

- even largely inaccurate estimates of workers’ reputation during task assignment → large improvements of system performance
- a simple optimal task-independent MAP decision rule is proposed for the case of full knowledge of workers’ reputation
Conclusions

Main findings of our paper are:

- even largely inaccurate estimates of workers’ reputation during task assignment → large improvements of system performance
- a simple optimal task-independent MAP decision rule is proposed for the case of full knowledge of workers’ reputation
- when workers’ reputation estimates are significantly inaccurate, the best performance can be obtained by combining our proposed task assignment algorithm with advanced decision rules such as LRA
Many, many thanks!
Many, many thanks! Questions?