

ABSTRACT

Title of dissertation: SECURITY AND TRUST
IN DISTRIBUTED COMPUTATION

Xiangyang Liu, Doctor of Philosophy, 2015

Dissertation directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

We propose three research problems to explore the relations between trust and security in the setting of distributed computation. In the first problem, we study trust-based adversary detection in distributed consensus computation. The adversaries we consider behave arbitrarily disobeying the consensus protocol. We propose a trust-based consensus algorithm with local and global trust evaluations. The algorithm can be abstracted using a two-layer structure with the top layer running a trust-based consensus algorithm and the bottom layer as a subroutine executing a global trust update scheme. We utilize a set of pre-trusted nodes, *headers*, to propagate local trust opinions throughout the network. This two-layer framework is flexible in that it can be easily extensible to contain more complicated decision rules, and global trust schemes.

The first problem assumes that normal nodes are homogeneous, i.e. it is guaranteed that a normal node always behaves as it is programmed. In the second and third problems however, we assume that nodes are heterogeneous, i.e, given a task, the probability that a node generates a correct answer varies from node

to node. The adversaries considered in these two problems are workers from the open crowd who are either investing little efforts in the tasks assigned to them or intentionally give wrong answers to questions.

In the second part of the thesis, we consider a typical crowdsourcing task that aggregates input from multiple workers as a problem in information fusion. To cope with the issue of noisy and sometimes malicious input from workers, trust is used to model workers' expertise. In a multi-domain knowledge learning task, however, using scalar-valued trust to model a worker's performance is not sufficient to reflect the worker's trustworthiness in each of the domains. To address this issue, we propose a probabilistic model to jointly infer multi-dimensional trust of workers, multi-domain properties of questions, and true labels of questions. Our model is very flexible and extensible to incorporate metadata associated with questions. To show that, we further propose two extended models, one of which handles input tasks with real-valued features and the other handles tasks with text features by incorporating topic models. Our models can effectively recover trust vectors of workers, which can be very useful in task assignment adaptive to workers' trust in the future. These results can be applied for fusion of information from multiple data sources like sensors, human input, machine learning results, or a hybrid of them. In the second subproblem, we address crowdsourcing with adversaries under logical constraints. We observe that questions are often not independent in real life applications. Instead, there are logical relations between them. Similarly, workers that provide answers are not independent of each other either. Answers given by workers with similar attributes tend to be correlated. Therefore, we propose a novel

unified graphical model consisting of two layers. The top layer encodes domain knowledge which allows users to express logical relations using first-order logic rules and the bottom layer encodes a traditional crowdsourcing graphical model. Our model can be seen as a generalized probabilistic soft logic framework that encodes both logical relations and probabilistic dependencies. To solve the collective inference problem efficiently, we have devised a scalable joint inference algorithm based on the alternating direction method of multipliers.

The third part of the thesis considers the problem of optimal assignment under budget constraints when workers are unreliable and sometimes malicious. In a real crowdsourcing market, each answer obtained from a worker incurs cost. The cost is associated with both the level of trustworthiness of workers and the difficulty of tasks. Typically, access to expert-level (more trustworthy) workers is more expensive than to average crowd and completion of a challenging task is more costly than a click-away question. In this problem, we address the problem of optimal assignment of heterogeneous tasks to workers of varying trust levels with budget constraints. Specifically, we design a trust-aware task allocation algorithm that takes as inputs the estimated trust of workers and pre-set budget, and outputs the optimal assignment of tasks to workers. We derive the bound of total error probability that relates to budget, trustworthiness of crowds, and costs of obtaining labels from crowds naturally. Higher budget, more trustworthy crowds, and less costly jobs result in a lower theoretical bound. Our allocation scheme does not depend on the specific design of the trust evaluation component. Therefore, it can be combined with generic trust evaluation algorithms.

Security and Trust in Distributed Computation

by

Xiangyang Liu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:

Professor John S. Baras, Chair/Advisor

Professor Gang Qu

Professor Charalampos Papamantou

Professor Tudor Dumitras

Professor Jennifer Golbeck

© Copyright by
Xiangyang Liu
2015

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever. First and foremost I would like to express my sincere gratitude to my advisor, Professor John S. Baras for giving me an invaluable opportunity to broaden my knowledge base and supporting me diving deep into some real-world problems. His deep insight on amazingly broad areas of research, limitless energy and enthusiasm on challenging problems has been the most important support for the past four years and will continue to be for years that will come. I would also like to thank Dr. Charalampos Papamantou, Dr. Gang Qu, Dr. Tudor Dumitras, and Dr. Jennifer Golbeck, for agreeing to serve on my thesis committee and for their invaluable discussions and input.

Sincere thanks to Xiangnan Weng, Peixin Gao, Ren Mao, Wentao Luan, who have enriched my graduate life in many ways and with whom I always had inspiring and fruitful discussions. Also I would like to thank Mrs. Kim Edwards for her great administrative support.

Lastly, I would like to acknowledge the support offered by AFOSR MURI FA-9550-10-1-0573, NSF CNS-1035655, NSF CNS-1018346 and by Maryland Procurement Office contract H98230-14-C-0127.

Table of Contents

List of Figures	v
1 Introduction	1
1.0.1 Security issues in distributed computation	1
1.0.2 Main contributions and thesis organization	3
1.0.2.1 Trust Models For Distributed Consensus With Adversaries	3
1.0.2.2 Worker Trust In Crowdsourcing With Adversaries	4
1.0.2.3 Trust-Aware Optimal Crowdsourcing With Budget Constraint	5
2 Trust Models For Distributed Consensus With Adversaries	6
2.1 Background on distributed consensus algorithms	6
2.2 Definitions of adversaries	10
2.3 Trust model	12
2.3.1 Local trust evaluation	14
2.3.1.1 Clustering-based Decision Rule	15
2.3.1.2 Distance-based Decision Rule	16
2.3.1.3 Consistency-based Decision Rule	17
2.3.2 Global trust evaluation	18
2.4 Trust-aware consensus algorithms	22
2.5 Theoretical analysis on security guarantees	24
2.5.1 Single-dimension decision rules	25
2.5.2 Multi-dimensional decision rules	30
2.5.3 Security performance for general trust graphs	32
2.6 Case study and performance evaluation	36
3 Worker Trust In Crowdsourcing With Adversaries	47
3.1 Enhancing data fusion using multi-dimensional trust	47
3.1.1 Motivation	47
3.1.2 Related work	49
3.1.3 Definitions	51

3.1.4	Multi-domain crowdsourcing model	52
3.1.5	Integration with features	56
3.1.6	Multi-domain crowdsourcing model with topic model	57
3.1.7	Experiments on real datasets	62
3.1.7.1	UCI datasets	62
3.1.7.2	Text Data	66
3.1.8	Proofs	67
3.1.8.1	Updates in MDC	69
3.1.8.2	Updates in MDFC	71
3.1.8.3	Updates in MDTC	72
3.1.9	Summary	74
3.2	Trust-aware crowdsourcing with domain knowledge	75
3.2.1	Motivation	75
3.2.2	Related work	77
3.2.3	Graphical model framework for trust-aware crowdsourceng with domain knowledge	77
3.2.4	Scalable inference algorithm based on ADMM	80
3.2.4.1	Definitions in probabilistic soft logic	81
3.2.4.2	Scalable ADMM-based inference	82
3.2.5	Case studies and experiments on real datasets	84
3.2.5.1	Affective Text Evaluation	86
3.2.5.2	Fashion Social Dataset Evaluation	88
3.2.6	Summary	91
4	Trust-Aware Optimal Crowdsourcing With Budget Constraint	93
4.1	Motivation	93
4.2	Related work	96
4.3	Problem setting	96
4.4	Trust-aware task allocation	100
4.4.1	Assumptions	101
4.4.2	Optimization Problem	102
4.5	Theoretical performance bound	106
4.6	Experimental results	109
4.6.1	Benchmark Algorithms	109
4.6.2	Experiment Setup on Galaxy Zoo Dataset	110
4.6.3	Analysis	112
4.7	Summary	114
6	Conclusions	115
	Bibliography	118

List of Figures

1.1	Distributed computation without supervisors.	2
1.2	Distributed computation with supervisors.	2
2.1	The adversary takes random vibration strategy.	30
2.2	Trust-aware consensus algorithm in situations with or without trust propagation when there are no adversaries in the network.	37
2.3	The adversary takes constant strategy.	38
2.4	The adversary takes random vibration strategy.	38
2.5	The adversary takes random noise strategy.	38
2.6	The adversary takes fixed noise strategy.	39
2.7	Sensor network of 7 nodes (sensors) with the centering shaded node as Byzantine adversary.	40
2.8	Network of 8 nodes (sensors) with the centering shaded node as Byzantine adversary and triangle node as header. The dotted links incident on the header node are not in the communication graph. Instead, they belong to the trust graph. Nodes 1 to 7 form a com- munication graph of connectivity 2.	41
2.9	Trust-aware consensus algorithm in sparse network Fig. 2.8 of con- nectivity 2.. Constant strategy.	42
2.10	Trust-aware consensus algorithm in sparse network Fig. 2.8 of con- nectivity 2.. Random vibration.	42
2.11	Trust-aware consensus algorithm in sparse network Fig. 2.8 of con- nectivity 2.. Random noise.	43
2.12	Trust-aware consensus algorithm in sparse network Fig. 2.8 of con- nectivity 2.. Fixed noise.	43
3.1	Multi-domain property of questions and workers in the test grading example. Q represents a question with concept vector $[0.7, 0.3]$ shown on the edges. Several workers with different two-dimensional trust vectors provide answers.	49

3.2	The graphical model for observed data provided by workers L , multi-domain expertise β , true labels R , domain variables C , and concept vectors λ . M is the total number of workers. N the number of questions. α is the hyperparameter of the Dirichlet prior distribution for λ and θ is the hyperparameter of the Beta prior distribution for β .	54
3.3	The graphical model for observed data provided by workers L , features x , multi-domain expertise β , true labels R , domain variables C , and parameter for domain distribution λ . μ , Σ , w , and δ are model parameters.	56
3.4	The graphical model for MDTC. L are observed answers from workers, w_{ik} is word k observed in question i , multi-domain expertise β , true labels R , domain variables C , parameter for domain distribution λ , topic distribution for word k in question i : z_{ik} , word distribution for domain l : ϕ_l .	60
3.5	Estimated worker reliability under different simulation settings on pima indians dataset. The estimated trust about workers' knowledge in Fig. 3.5(a), Fig. 3.5(b), and Fig. 3.5(c) are by MDFC and the results in Fig. 3.5(d) are by MDC.	65
3.6	Trust matrix about workers' knowledge over topics estimated by MDTC model.	68
3.7	Graphical Model of Trust-aware Crowdsourcing with Domain Knowledge (TCDK). z_i 's are true label variables, β_j 's are workers' trust variables, and l_{ij} 's are worker-provided answers. The black-dotted lines in the bottom layer encode probabilistic dependencies between variables and the red-dotted lines in the upper layer encode logical dependencies.	80
3.8	Estimated true labels for "cloth related" questions can be used for prediction of "fashion related" questions.	91
3.9	Estimated true labels for questions can be used for prediction of other questions using image similarity.	91
4.1	Total error probability of algorithms UA CQSA CA TAA TAAP on Galaxy Zoo dataset with budget ranging from 50 to 1500.	112
4.2	Total error probability of algorithm TAAP on Galaxy Zoo dataset under noise variance from 0 to 0.1 and the budget is from 50 to 1500.	113

Chapter 1: Introduction

1.0.1 Security issues in distributed computation

Distributed computation plays an important role in solving many real-world large-scale problems such as estimation of temperature using a network of sensors, image annotation by a pool of human workers on Amazon Mechanical Turk.

In the temperature estimation example, a network of sensors collaborate by executing a distributed consensus algorithm to reach an estimation of the average temperature of all nodes using only local information. This type of distributed computation does not have supervisors. Nodes in the network collaborate to accomplish a common task with local communication as illustrated in Fig. 1.1. Another application that fits these scenarios is a network of robots, each running a distributed consensus algorithm, collaborating to reach consensus on where to go for a rescue task. Each node in the network is supposed to run a local update algorithm by exchanging local information with its neighbors within communication range. In the existence of malicious nodes however, this is not the case. Communication links might be jammed [1] and nodes in the network might be hacked and behave arbitrarily [2,3].

In the image annotation example, there is a supervisor that is responsible for

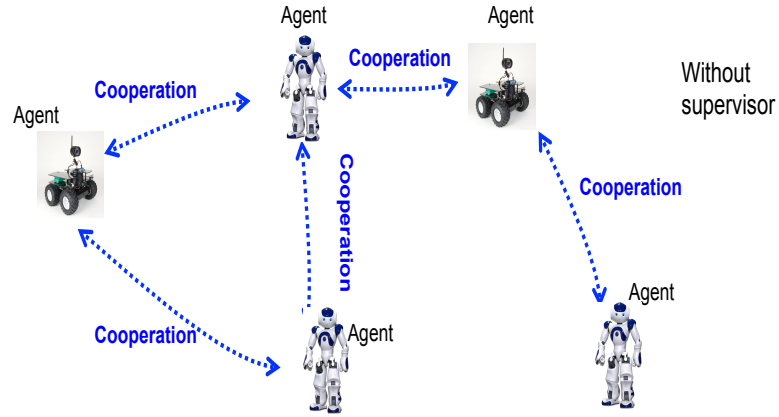


Figure 1.1: Distributed computation without supervisors.

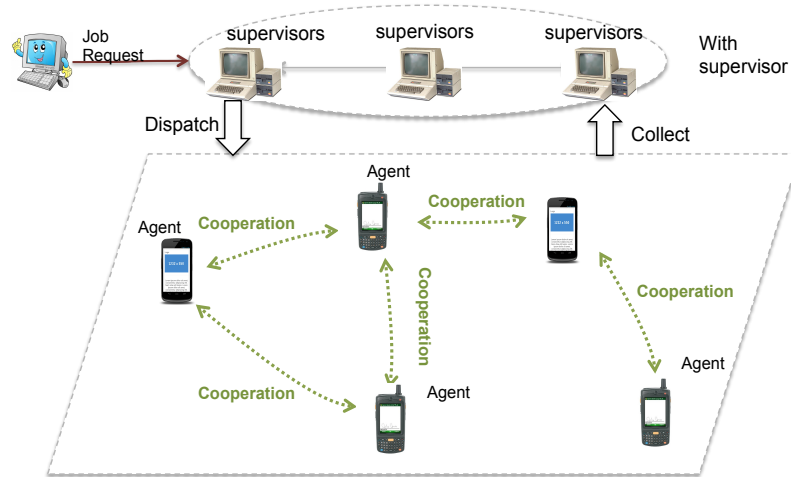


Figure 1.2: Distributed computation with supervisors.

distributing tasks to a group of workers who collaborate to give annotations to given images. Their answers are then sent back to the supervisor for information fusion. This process is illustrated in Fig. 1.2. Data contributed by workers are noisy and many times malicious. In real applications (crowdsourcing platforms), some workers want to make easy money by investing little effort in understanding the task assigned to them and giving random answers.

1.0.2 Main contributions and thesis organization

In this dissertation, we aim to answer the following questions: (1) How to develop a trust model of nodes and integrate it within a distributed consensus algorithm in the face of adversaries to effectively detect adversaries; (2) In distributed computation without supervisors (crowdsourcing) with malicious workers, how to develop a graphical model that estimates questions' true labels more accurately by taking into consideration workers' trust values; (3) How to allocate tasks to workers of different trust values in order to minimize error probability rate subject to a budget constraint. The first question assumes that normal nodes are homogeneous, i.e. it is guaranteed that a normal node always behaves as it is programmed. In the second and third questions however, we assume that nodes are heterogeneous, i.e, given a task, the probability that a node generates a correct answer varies from node to node. The adversaries considered in this problem are workers from the open crowd who are either investing little efforts in the tasks assigned to them or intentionally give wrong answers to questions. The main contributions of this dissertation are summarized below.

1.0.2.1 Trust Models For Distributed Consensus With Adversaries

To address the first question, we propose a trust model with various decision rules based on local evidences in the setting of Byzantine adversaries. Our trust-aware algorithm is flexible and can be easily extended to incorporate more complicated designs of trust models and decision rules. We provide theoretical se-

curity performance with respect to miss detection rate and false alarm rate under a regular trust graph assumption and provide a security performance bound under general trust graph assumptions. The theoretical results indicate that both error rates decrease exponentially with respect to the number of headers. Simulations show that our proposed trust-aware consensus algorithm can effectively detect various malicious strategies even in sparse networks where connectivity $< 2f + 1$, where f is the number of adversaries.

1.0.2.2 Worker Trust In Crowdsourcing With Adversaries

To answer the second question, we formulate a probabilistic model of crowdsourcing tasks with *multi-domain* characteristics and propose a novel inference method based on variational inference. Our model is very flexible and can be easily extended. In applications where each question comes with a feature vector, we further develop an extended model that handles questions with continuously-valued features. We further extend the model by combining a multi-domain crowdsourcing model with topic discovery based on questions' text descriptions and derive an analytical solution to the collective variational inference.

We assumed the true labels of different questions and the trust values of workers are independent. However, our domain knowledge tells us that there might be logical constraints between the true labels of questions as well as between the trust values of different workers. To incorporate domain knowledge, we formulate a novel trust-aware crowdsourcing with domain knowledge framework that combines domain

knowledge with a traditional crowdsourcing graphical model. Users can express high level domain knowledge without having to re-define the model and the framework can be used to integrate multiple data sources. We also develop a scalable joint inference algorithm for estimating true label variables and trust values of workers based on alternating consensus optimization. The inference algorithm can be easily scaled to multiple machines.

1.0.2.3 Trust-Aware Optimal Crowdsourcing With Budget Constraint

We answer the last question by formulating the problem of trust-aware task allocation in crowdsourcing and provide a principled way to solve it. Our formulation models the workers' trustworthiness and the costs depend on both the question and the worker group. Our method is ready to be extended to more complicated aggregation methods other than the weighted majority vote as well. The trust-aware task allocation scheme we propose can achieve total error probabilities bounded by $\frac{N}{2} - \mathcal{O}(\sqrt{B})$, where N is the number of tasks and B is the total budget. Different from [4], the exact performance bound of error probability also incorporates both trustworthiness of crowds and cost. More trustworthy crowds and less costly jobs result in lower guaranteed bound.

Chapter 2: Trust Models For Distributed Consensus With Adversaries

2.1 Background on distributed consensus algorithms

In distributed systems, nodes in the network are programmed to calculate given functions of nodes' values. However, due to message transmission delays, crashes, value domain errors, or even Byzantine behavior of malicious nodes, different nodes would probably have different views of the input vector of these parameters. Consensus requires that every correct agent in the network reach an agreement on some value. We will study the problem of consensus in the face of Byzantine adversaries.

The issue of consensus has been investigated for decades in the computer science, communication and control communities. There are mainly two types of failures discussed. One is crash failures and the other is Byzantine failures. A crash failure refers to the case when a process stops working, while in a Byzantine failure, a process may send arbitrary data to other processes. Byzantine failures are far more disruptive since they allow arbitrary behaviors.

In the computer science community, the consensus problem is to design a distributed protocol that allows processes to agree on a single decision value, which

should be one of the initial values. Based on different failure types, oracle-based consensus protocols [5] and condition-based approaches [6] have been proposed to achieve the goal. For asynchronous distributed systems, Chandra and Toeug [7] introduced the concept of Failure Detector (FD) and showed that with FD, consensus can be achieved and possible failure processes can be found. In [8] a probabilistic solution was applied to solve the consensus problem under Byzantine failures with the condition $f_b < \frac{1}{5}n$, where n is the total number of processes and f_b is the number of Byzantine failures. The leader-based consensus developed by Mostefaoui et al [5]. required $f_c < \frac{1}{2}n$ to reach consensus with crash failures, where f_c is the number of crash failures, and the time and message costs of the protocol can be reduced when $f_c < \frac{1}{3}n$. Later in [6], the leader oracle approach, the random oracle approach and the condition-based approach are combined to provided a hybrid consensus protocol. [9] connected Error-Correcting Codes (ECC) to the condition-based approach and showed that consensus can be solved despite f_c crash failures if and only if the condition can be mapped to a code whose Hamming distance is $f_c + 1$, and Byzantine consensus can be solved despite f_b Byzantine faults when the Hamming distance of the code is $2f_b + 1$.

Different from the consensus problem discussed in the computer science community, consensus in a network of connected agents means reaching an agreement regarding the states (or values of certain agent variables) of all (benign) agents that are used in computing certain functions (or function). Without considering failures, for a certain node, the consensus process can be as simple as using a weighted average of its neighbors' states [10]. Ren [11, 12] proposed update schemes for consensus

under dynamically changing directed interaction topologies, provided that the union of the directed interaction graphs have a spanning tree frequently enough as the system evolves. However the linear updating rules may not be resilient to misbehaving nodes. It was shown in [13] that a single misbehaving node can cause all agents to reach consensus on a wrong value, which potentially will result in a dangerous situation in physical systems. The framework for posing and solving consensus problems for multi-agent networked systems was analyzed in [13, 14], where key results on the theory and applications of consensus problems in networked systems are presented. [15] showed that the resilience of a partially connected network to Byzantine adversaries is characterized by its network topology, and a well-behaving agent can calculate any arbitrary function of all node values when the number of malicious nodes (f) is less than $1/2$ of the network connectivity, i.e the connectivity should be at least $2f+1$. [3] used system theory to reach a similar conclusion regarding network connectivity, requiring that the number of disjoint paths between any two nodes in the network should be greater than twice the number of adversaries. LeBlanc et al. developed the Adversarial Robust Consensus Protocol (ARC-P) [16, 17], which applied ideas from both the consensus algorithms resilient to Byzantine faults in distributed computing and the linear consensus protocols used for coordination of networked agents, and formulated the problem into a linear control problem where consensus could be reached among cooperative agents via agreement and validation conditions.

Applications of both kinds of consensus problems and formulations are appropriate for, and have been used in our earlier work, distributed filtering and estimation

in heterogeneous sensor networks with applications to power grids [18–20]. A Model-Based Systems Engineering framework for distributed heterogeneous sensor networks was presented in [21].

Network connectivity conditions in most previous works are hard to satisfy in reality. In addition, the robustness condition in [16] is itself hard to verify. These facts motivate us to design a Byzantine adversary detection scheme based on trust evaluation. We introduce the notion of *trust* in the context of consensus algorithms with Byzantine adversaries. Works related to the application of trust to distributed algorithms include [18,20] who embedded trust evaluation in distributed Kalman filtering (DKF) with applications to sensor networks in power grids, and [22] who proposed the RoboTrust algorithm in consensus algorithms. Our work differs from [22] in the following ways: 1) The trust model is different; and 2) Trust in [22] is established only by local evidence while our trust model also depends on second-hand evidence. 3) Trust propagation in evaluating global trust is resistant to malicious voting. In addition to the empirical results reported in our previous work [23], we provide a theoretical bound on the miss detection rate and false alarm rate for the result of trust propagation for both a regular trust graph assumption and a general trust graph assumption. Our contributions are as follows:

- We propose a trust model with various decision rules based on local evidences in the setting of Byzantine adversaries.
- Our trust-aware algorithm is flexible and can be easily extended to incorporate more complicated designs of trust models and decision rules.

- We provide theoretical security performance with respect to miss detection rate and false alarm rate under a regular trust graph assumption and provide security performance bound under a general trust graph assumption. The theoretical results indicate that both error rates decrease exponentially with respect to the number of headers.
- Simulations show that our proposed trust-aware consensus algorithm can effectively detect various malicious strategies even in sparse networks where connectivity $< 2f + 1$, where f is the number of adversaries.

2.2 Definitions of adversaries

Consider a communication network modeled by a directed graph $G(k) = (V, E(k))$, where V denotes the set of nodes in the network and $E(k)$ the set of edges at time k . If $e_{ij}(k) \in E(k)$, it means node i can hear node j 's message at time k , i.e. node j is a neighboring node of i at time k . Whether node i is able to receive node j 's message depends on their relative distance and j 's transmission power. A node can reach a larger portion of nodes in the network if it transmits messages with higher power. Let $N_i(k) = \{j | e_{ij}(k) \in E(k), j \neq i\}$ denote the set of neighbor nodes that node i can hear from at time k , and $N_i^+(k) = N_i(k) \cup \{i\}$ denote the extended neighbor set of node i at time k . We assume all nodes' transmission power is fixed. Therefore we have $N_i(k) = N_i, \forall k \geq 0$.

Let $X(k)$ denote the N -dimensional vector of all nodes' states at time k . The

nodes' beliefs evolve in time according to the dynamics:

$$x_i(k) = \sum_{j \in N_i} w_{ij}(k) x_j(k-1) + w_{ii}(k) x_i(k-1) \quad (2.1)$$

where $X(k) = [x_1(k), x_2(k), \dots, x_N(k)]^T$, and $x_i(k)$ denotes node i 's updated state at time k and $w_{ij}(k) \geq 0, j \neq i$ is the weight that node i puts on node j 's belief at the previous time instant for the calculation of its state update. $w_{ii}(k) \geq 0$ is the weight that node i puts on its own previous state. Coefficients are normalized and satisfy $\sum_{j \in N_i(k)} w_{ij}(k) + w_{ii}(k) = 1$. We denote by $W(k)$ the $N \times N$ matrix with elements $W_{ij}(k)$. Equation (2.1) is a standard module where all nodes are normal.

In a distributed environment, due to lack of central monitoring, nodes are subject to various attacks. In the worst case, some nodes might be hacked and do not function as they are originally programmed. We consider the following Byzantine adversary model:

Definition 2.2.1. *A Byzantine adversary may behave arbitrarily. It does not follow the prescribed distributed consensus update rule, i.e. at some time instant $k > 0$, a Byzantine adversary i sends incorrect message $v_i(k)$ other than $x_i(k)$ in equation (2.1) to its neighbors. We assume a broadcast model here, meaning adversaries send the same message to different neighbors. In addition, the adversary is assumed to have complete knowledge of the network topology, the states of all other nodes in the network, and the consensus update rule for all nodes.*

Next, we define a normal node's behavior and the information it can get access to.

Definition 2.2.2. *A normal node behaves according to the distributed consensus specification, i.e. it updates its states by combining the messages received from its neighbors using the specified coefficients. The normal node has access to just local information such as from its neighborhood, the messages sent by them, the coefficients it uses for updating states. Moreover, it does not know whether a neighbor is normal or malicious.*

From the above definitions, we can see that a normal node i has no way of determining whether one of its neighbors j is malicious or not since it can not get access to all the messages sent by its 2-hop neighbors $l \in N_j \setminus \{i\}$. To detect malicious nodes locally, we introduce the trust model and establish local trust between nodes based on first-hand evidence and define several decision rules that map from local evidence to local decisions. Often local evidence is not sufficient to reach useful decisions. We therefore define global trust based on both first-hand evidence and evidence of other nodes in the network. Our proposed trust-aware consensus algorithm takes global trust values as input.

2.3 Trust model

There are two possible connections from node i to node j . One is communication connection. If $j \in N_i(k)$, node j is said to be in the communication neighborhood of node i at time instant k . The other is trust connection. Denote the set of trust relations at time instant k as $E_c(k)$. A directed edge from node i to node j , denoted as $e_{ij}^c(k) \in E_c(k)$, represents the trust relation node i has towards node j .

We assume that if there is communication connection from node i to node j at time k , there must exist a trust relation $e_{i,j}^c(l), \forall l \geq k$ since node i receives messages from node j at time k which forms the direct evidence for i to evaluate j 's trustworthiness at the current iteration k and future iterations. However, if there exists trust relation $e_{ij}^c(k)$, the communication connection e_{ij} does not necessarily exist because the trust relation is possibly derived from indirect observations of other nodes in the network.

We associate a local trust value $c_{ij}(k) \in [0, 1)$ with the trust connection $e_{ij}^c(k) \in E_c(k)$. It represents the belief node i holds about j at time instant k based on its local interactions with node j . The value $c_{ij}(k)$ can be seen, in node i 's perspective, as the probability of node j behaving normally at time instant k . It depends on both the interaction between i and j at time k and history records i has on node j . We utilize the Beta reputation system [24] to model local trust values. Denote the probability that node j behaves normally at time instant k in node i 's perspective as $p_{ij}(k)$. $p_{ij}(k)$ is assumed to have beta distribution with parameter $\alpha_{ij}(k)$ and $\beta_{ij}(k)$:

$$\begin{aligned} f(p_{ij}(k)|\alpha_{ij}(k), \beta_{ij}(k)) \\ = \frac{\Gamma(\alpha_{ij}(k) + \beta_{ij}(k))}{\Gamma(\alpha_{ij}(k))\Gamma(\beta_{ij}(k))} p_{ij}(k)^{\alpha_{ij}(k)-1} (1 - p_{ij}(k))^{\beta_{ij}(k)-1} \end{aligned} \quad (2.2)$$

where Γ is Gamma distribution. Let $r_{ij}(k) = \alpha_{ij}(k) - 1$ and $s_{ij}(k) = \beta_{ij}(k) - 1$, to represent the number of events that node j is normal and the number of events j is malicious up to time k in the perspective of node i respectively. The local trust

value $c_{ij}(k)$ is defined to be the mean value of $p_{ij}(k)$, i.e.

$$c_{ij}(k) = \mathbb{E}[p_{ij}(k)] = \frac{r_{ij}(k) + 1}{r_{ij}(k) + s_{ij}(k) + 2} \quad (2.3)$$

Considering the time-varying behavior of node j , old records are less important than more recent observations. We introduce positive forgetting factors ρ_1 and ρ_2 such that

$$\begin{aligned} r_{ij}(k+1) &= \rho_1 r_{ij}(k) + f_{ij}(k+1) \\ s_{ij}(k+1) &= \rho_2 s_{ij}(k) + 1 - f_{ij}(k+1), \end{aligned} \quad (2.4)$$

where the trust decision $f_{ij}(k+1)$ equals to 1 if node i believes that node j behaves normally at time $k+1$ and equals 0 otherwise. In practice, we may choose $\rho_1 < \rho_2$ so that bad behaviors are remembered for longer period of time relative to good behaviors. Note that $f_{ij}(k+1)$ can also take fractional values from $[0, 1]$ which allows us to encode the uncertainty of local decisions. We consider two realizations of $f_{ij}(k+1)$: the local trust decision $I_{ij}(k+1)$ and the global trust decision $GI_{ij}(k)$. $I_{ij}(k+1)$ is obtained from direct local evidence while $GI_{ij}(k+1)$ is calculated by propagating local trust decisions from other nodes in the network. We will discuss each of them below.

2.3.1 Local trust evaluation

We now discuss the question of evaluating the local trust decision $I_{ij}(k)$ based on node i 's local observation about node j at time instant k , which involves the interplay of the consensus algorithm and the trust computation. To answer this

question, we have to examine what can be used as local evidence for node i to determine whether a neighbor j behaves normally or maliciously at time instant k . Essentially, we want to find a mapping from trust evidence to a binary decision. There are many choices of evidences available and the mapping can also be arbitrary. Denote the values $v_j(k)$ received from $j \in N_i(k)$ as the vector $r_i(k)$. We discuss three decision rules (mappings) below. One is based on clustering schemes, the second is based on the distance of the messages, and the third on the consistency of 2-hop messages.

2.3.1.1 Clustering-based Decision Rule

The motivation behind clustering-based decision rules is the observation that the malicious node's message tends to deviate from normal nodes' messages. Therefore the node whose message is far away from the rest is likely to be malicious. Formally, we define the deviation of a message sent by node j from the all other messages received by node i as

$$\text{dev}_{ij}(k) = \sum_{l \in N_i^+} \frac{|v_l(k) - v_j(k)|^2}{|N_i^+|} \quad (2.5)$$

The ranking of the deviation in equation (2.5) itself can not indicate whether node j is malicious or not because a normal node might be the one deviating the most when convergence is almost reached and all nodes' messages are close to each other.

Therefore we propose a decision rule based on relative ranking:

$$I_{ij}(k) = \begin{cases} 1 & \text{if } dev_{ij}(k) \leq Th_i * \text{median}(\{dev_{ij}(k)\}) \\ 0 & \text{otherwise} \end{cases} \quad (2.6)$$

where Th_i is the threshold used by node i and $\text{median}(\cdot)$ denotes the median of values within the bracket. The above decision rule reflects the heuristics that the node whose message is too far away from the median is deemed to be malicious.

2.3.1.2 Distance-based Decision Rule

Denote the distance between node i 's state at time instant k , $x_i(k)$ and the value $v_j(k)$ as

$$d_{ij}(x_i(k-1), v_j(k-1)) = \|x_i(k-1) - v_j(k-1)\|_2 \quad (2.7)$$

$d_{ij}(x_i(k-1), v_j(k-1))$ measures the degree of disagreement of node j from node i at time $k-1$. We measure the degree of cooperation by $\Delta_{ij}(k)$ defined as:

$$\Delta_{ij}(k) = d_{ij}(x_i(k-1), v_j(k-1)) - d_{ij}(x_i(k-1), v_j(k)) \quad (2.8)$$

$\Delta_{ij}(k)$ measures the degree of cooperation in the sense that if node j cooperates (normal), its state is expected to be closer to the state of node i as the iteration goes on and that if node j does not cooperate (Byzantine), it may send value $v_j(k)$

that are far away from $x_i(k-1)$. The decision rule therefore can be specified as:

$$I_{ij}(k) = \begin{cases} 1 & \text{if } \Delta_{ij}(k) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.9)$$

2.3.1.3 Consistency-based Decision Rule

For node i , to verify the correctness of message $v_j(k)$, it needs more information than $v_j(k)$. We propose to augment the message sent from node j as follows. Several new notations will be introduced. The **inner message** of node j at time instant k is defined as the messages that node j collects from its neighbors and we denote it as $X_j^*(k) = \{x_{jl}^*(k-1), l \in N_j^+\}$ with $x_{jj}^*(k-1) = x_{jj}^c(k-1)$, the calculated state of node j at super-step $k-1$. The calculated state is defined in Table 2.1. Similarly, we define the **outer message** of node j at time instant k to be the messages broadcast by node j and we denote it as $X_j(k) = \{x_{jl}(k-1), l \in N_j^+\}$, where $x_{jj}(k-1)$ might not equal to $x_{jj}^c(k-1)$ for malicious nodes. For normal nodes, the inner message and the outer message are equal, i.e. $X_j(k) = X_j^*(k)$. However, for malicious nodes, they can choose to broadcast messages different from the inner message, i.e. $X_j(k) \neq X_j^*(k)$. The local decision node i makes about node j is not a scalar value. Instead, it consists of a set of values $I_{ij}^l(k), \forall l \in N_j^+$, where $I_{ij}^l(k)$ is node i 's local

decision about node j 's message $x_{jl}(k-1)$. Therefore the decision rule becomes

$$I_{ij}^l(k) = \begin{cases} 1, & x_{il}^*(k-1) = x_{jl}(k-1), l \in N_i \\ 0, & x_{il}^*(k-1) \neq x_{jl}(k-1), l \in N_i \\ 0.5, & l \notin N_i \end{cases} \quad (2.10)$$

where $x_{il}^*(k-1)$ denotes the inner message element acquired by node i about l 's state if node l can be heard by node i . When $l \notin N_i$, the local evidence available to node i is not sufficient to reach any decision. Therefore $I_{ij}^l(k) = 0.5$. We need to specify a function that maps from these atomic decisions $I_{ij}^l(k)$ to a single scalar decision about j . We choose the following mapping as in [25]:

$$I_{ij}(k) = \prod_{l \in N_j} I_{ij}^l(k) \quad (2.11)$$

Since $I_{ij}^l(k) \in [0, 1]$, it can be interpreted as the probability of node j behaving maliciously regarding x_{jl} . The mapping in equation (2.11) indicates that I_{ij} is closer to 1 only if all $I_{ij}^l(k)$'s are closer to 1 and that I_{ij} is closer to 0 if any $I_{ij}^l(k)$ is closer to 0. The aggregated decision toward node j at super step k , $I_{ij}(k)$, is then used to update node i 's local trust value toward node j , c_{ij} , according to equation (2.3).

2.3.2 Global trust evaluation

We utilize trust propagation in order to get global trust decisions for both the case of single-dimension decision rules such as clustering-based and distance-based

decision rules and multi-dimensional decision rules like consistency-based decision rule.

We first discuss the case of trust propagation for single-dimension decision rules. Node i maintains its local trust opinion $c_{ij}(k)$ about node j for $j \in N_i$ and the local trust decisions $I_{ij}(k)$'s calculated using various decision rules mentioned above. However, when node i wants to get a more accurate evaluation of $I_{ij}(k)$, it needs to rely on the observations of other nodes in the network, i.e. node i needs to aggregate local trust decisions $I_{lj}(k), l \in N_i, l \neq j$ through trust propagation (we use global trust evaluation and trust propagation interchangeably from now on).

Denote the global trust of node j in the perspective of node i as t_{ij} . [26] suggests to aggregate local trust values by weighing node i 's neighbors' local trust opinions about node j . Before the consensus algorithm advance to the next iteration, we want to obtain the equilibrium global trust opinion through the trust propagation subroutine, which is also an iterative process. Therefore, we assume $c_{ij}, \forall i, j$ remain constant during the iterations to obtain global trust and use τ to denote the iteration number of global trust values. It is a smaller time scale compared to consensus iteration number k . Omitting the time instant k for convenience, we have

$$t_{ij}^\tau = \begin{cases} 1 & \text{if } i = j \\ \frac{1}{z_i} \sum_{l \in N_i, l \neq j} c_{il} t_{lj}^{\tau-1} & \text{if } i \neq j \end{cases} \quad (2.12)$$

where the normalizing constant is $z_i = \sum_{l \in N_i, l \neq j} c_{il}$.

Note that in a distributed environment, only $c_{il}, l \in N_i$ are stored locally at

node i while t_{lj} is sent by node i 's neighbor l . However, node l might be malicious and lying about t_{lj} . Specifically, if node j is normal, node l intentionally reports to node i that $t_{lj} = 0$. Similarly, if node j is Byzantine, node l protects its peer by reporting $t_{lj} = 1$. To cope with this concern, we introduce a set of pre-trusted nodes in the network, called *headers*.

Definition 2.3.1. Headers are a set of pretrusted nodes besides V . They are equipped with more security measures and therefore are more reliable. The header's identity remains anonymous, i.e. neither a normal node in V nor an adversary knows if a given node is header or not. Therefore, a normal node can choose to trust or distrust a header since it does not know which node is a header.

The introduction of anonymous headers induces costs since they come with higher level of reliability. Therefore we might prefer to deploy headers in denser areas instead of sparse areas in order to make the most use of them. The problem of how to deploy headers optimally with fixed number of headers is beyond the scope of this dissertation.

Denote the trust that node i places on a header h as p_{ih} . Node i aggregates local trust from both its neighbors in N_i and headers that it can receive messages from. The global trust vector \vec{t}_i evolves according to:

$$t_{ij}^\tau = \begin{cases} 1 & \text{if } i = j \\ \frac{1}{z_i + b_H} \left(\sum_{l \in N_i, l \neq j} c_{il} t_{lj}^{\tau-1} + \sum_{h \in H} p_{ih} c_{hj} \right) & \text{if } i \neq j \end{cases} \quad (2.13)$$

where H is the set of headers, $b_H = \sum_{h \in H} p_{ih}$, $z_i = \sum_{m \in N_i, m \neq j} c_{il}$, and $z_i + b_H$ as

a whole serves as normalizing constant. Since node i does not know which node is a header, p_{ih} might be smaller than 1. When node i can not obtain messages from header h , $p_{ih} = 0$. In both cases (with and without trust propagation), the initial value of global trust t_{ij}^0 takes the following form:

$$t_{ij}^0 = \begin{cases} I_{ij} & \text{if node } i \text{ is normal} \\ 1 & \text{if both node } i \text{ and } j \text{ are malicious} \\ 0 & \text{if node } i \text{ is malicious and node } j \text{ is normal} \end{cases} \quad (2.14)$$

The local trust decisions in equation (2.14) are propagated via subroutine equation (2.12) or equation (2.13) with the help of headers. The global trust decision GI_{ij} is obtained by:

$$GI_{ij} = \begin{cases} 1 & \text{if } t_{ij} > \eta \\ 0 & \text{if otherwise} \end{cases} \quad (2.15)$$

where t_{ij} is the equilibrium global trust value when equation (2.13) converges and the real positive η is the threshold used to calculate the global trust decision. Larger value of η indicates that we are more likely to detect malicious nodes but at the same time we tend to make more false alarms. Then GI_{ij} is used to update trust values in equation (2.4).

Next we discuss the case of multi-dimensional decision rules. We get the global trust value $t_{ij,l}(k)$ for each dimension of the local trust decision $I_{ij}^l(k)$ via the same trust propagation process described for the single-dimension case. The global trust

decision GI_{ij}^l for dimension l is obtained by:

$$GI_{ij}^l = \begin{cases} 1 & \text{if } t_{ij,l} > \eta \\ 0 & \text{if otherwise} \end{cases} \quad (2.16)$$

where $t_{ij,l}$ is the global trust value at equilibrium for dimension l . The global trust decision GI_{ij} is calculated by:

$$GI_{ij} = \prod_{l=1}^{N_j} GI_{ij}^l \quad (2.17)$$

The introduction of trust propagation (global trust evaluation) in the trust model can incur high computational cost since it is an embedded iterative process. Therefore, we can choose to either activate this part or mute it wisely in practice. In this chapter, we give simulation results for both cases and leave the discussion of when to activate trust propagation and when to mute it as future work.

2.4 Trust-aware consensus algorithms

Given the trust model above, we obtain either the local or global trust decision $f_{ij}(k)$ which will then be used to update the trust value $c_{ij}(k)$ according to equation (2.4). We are now in the position to propose our trust-aware consensus algorithm. The state dynamics goes as follows:

$$x_i(k) = \frac{1}{A_i(k)} \sum_{j \in N_i} c_{ij}(k) v_j(k-1) \quad (2.18)$$

where $A_i(k) = \sum_{j \in N_i} c_{ij}(k)$ and $v_j(k-1)$ is the message sent by node j . If j is malicious, $v_j(k-1)$ is not necessarily $x_j(k-1)$. Using $c_{ij}(k)$ as the linear coefficient to combine message $v_j(k-1)$ is just one way to do it. The local trust value $c_{ij}(k)$ incorporates both the trust value history of node j in the local view of node i and the trust decisions made by other nodes in the network. There are other more complex choices.

Note that the trust model as well as the decision rules defined in the previous section become pluggable components to the trust-aware consensus algorithm because it only needs as input the trust dynamics given states, ignoring the details of the trust model design and decision rules. This makes the algorithm highly extensible to future developments of more complicated trust models and decision rules. The full algorithm consisting of the trust-aware consensus algorithm and the trust model based on cluster-based decision rule or distance-based decision rule is shown in Algorithm 1.

The trust-aware consensus algorithm with consistency-based decision rule based on augmented messages is shown in Algorithm 2. In practice, an ensemble of the various choices of decision rules will be used. Note that the actions within the for loops in Algorithm 1 and Algorithm 2 are executed in parallel instead of in a sequential way. Our trust-aware consensus algorithm is not restricted to the three decision rules in Section 2.3. We can readily place more delicate decision rules into the algorithm. Moreover, the model can incorporate more complex update schemes for trust compared to equation (2.4). Therefore, the trust-aware consensus algorithm is highly extensible. Table 2.1 shows the common notations used in this chapter.

2.5 Theoretical analysis on security guarantees

The trust propagation with the help of headers improves the accuracy of trust evaluation since trust evidence of multiple nodes in the network are utilized for the calculation of trust values in a distributed way. In what follows, we discuss how the connectivity of the graph and the number of headers affect the accuracy of trust evaluation within a specific iteration k .

The trust graph might be time varying and the edge weights c_{ij} may be heterogeneous, making the equilibrium decision values computationally difficult. Therefore, we will focus on special cases when the trust graph is regular and give some qualitative analysis on the cases when the trust graph is not regular. Therefore we assume that the trust graph is regular, i.e. the weights are the same across all edges, for the purpose of quantitative analysis. We use $|N_{h,j}|$ to denote the number of headers that have direct evidence of node j and can evaluate j 's trust with high accuracy and $|N_m|$ as the total number of adversaries in the network who collaborate to revert the trust evaluation in the propagation process. Specifically, all the adversaries will vote -1 for the target node at all times if it is honest and vote 1 if it is malicious. We assume that adversaries do not have power constraints so that they can reach the whole network to participate in every propagation process in order to maximize their damage.

2.5.1 Single-dimension decision rules

We analyse security guarantees of decision rules based on scalar-valued messages $v_j(k)$'s. Examples of such decision rules are previously-discussed clustering-based and distance-based decision rules. As in equation (2.13), we denote the trust opinion about node j by header h as $c_{hj} \in \{0, 1\}$ where $c_{hj} = 0$ means the header h considers node j to be malicious and evaluates its trust to be 0 and $c_{hj} = 1$ means the header considers the node to be normal. We also denote the true type of node j as $u_j \in \{0, 1\}$. Headers, with their high security measures, can also make mistakes in evaluating c_{hj} . Therefore we express the probability of making a wrong decision of by header h as:

$$p(c_{hj} = u_j) = 1 - \epsilon \tag{2.19}$$

where u_j is the true type of node j and ϵ is the probability of a header making a wrong trust decision. We assume in the above equation that the error probability is the same for all headers for simplicity. The problem of detecting whether node j is malicious or not by node i is reduced to a hypothesis problem specified in equation (2.15). In what follows, we derive the probability distribution of t_{ij} and give analytical results on miss detection rate under the assumption of regular trust graph.

Theorem 2.5.1. *For a regular trust graph, the miss detection rate and the false*

alarm rate are

$$\begin{aligned}
p(GI_{ij} = 1 | u_j = 0) &= \sum_{x=y_{hj}}^{|N_{hj}|} C_{|N_{hj}|}^x \epsilon^x (1 - \epsilon)^{|N_{hj}|-x} \\
p(GI_{ij} = 0 | u_j = 1) &= \sum_{x=|N_{hj}|-z_{hj}}^{|N_{hj}|} C_{|N_{hj}|}^x \epsilon^x (1 - \epsilon)^{|N_{hj}|-x}
\end{aligned} \tag{2.20}$$

where GI_{ij} is the node i 's global trust decision about node j after trust propagation, u_j is the hidden true type of node j , $C_{|N_{hj}|}^x$ is the number of x -combinations from a set of $|N_{hj}|$ distinct nodes and

$$\begin{aligned}
y_{hj} &= \lceil \eta |N_{hj}| - (1 - \eta) (|N_m| - 1) \rceil \\
z_{hj} &= \lfloor \eta (|N_{hj}| + |N_m|) \rfloor
\end{aligned} \tag{2.21}$$

Proof. The absorption probability ¹ is the same for every pair of nodes consisting of a node and the target node. Similar to [27], when the target node j is malicious, we have

$$t_{ij} = q \sum_{h \in N_{h,j}} c_{hj} + q(|N_m| - 1) \tag{2.22}$$

where q is the absorption probability from all nodes to the target node j and takes the following form:

$$q = \frac{1}{|N_{hj}| + |N_m| - 1} \tag{2.23}$$

And c_{hj} is the header h 's trust value on node j . Headers $h \in N_{h,j}$ estimates target

¹Absorption probability: in a Markov chain, the probability of being absorbed in one of the absorbing states when starting from a transient state

j 's type correctly with $1 - \epsilon$. We have:

$$\begin{aligned}
p(GI_{ij} = 1 | u_j = 0) &= p(t_{ij} > \eta | u_j = 0) \\
&= p\left(q \left(\sum_{h \in N_{h,j}} c_{hj} + |N_m| - 1\right) > \eta \mid u_j = 0\right) \\
&= p\left(\sum_{h \in N_{h,j}} c_{hj} > \frac{\eta}{q} - |N_m| + 1 \mid u_j = 0\right) \\
&= p\left(x > \frac{\eta}{q} - |N_m| + 1 \mid u_j = 0\right) \tag{2.24} \\
&= p(x > \eta(|N_m| + |N_{h,j}| - 1) - |N_m| + 1 \mid u_j = 0) \\
&= p(x \in \{y_{h,j}, y_{h,j} + 1, \dots, |N_{h,j}|\} \mid u_j = 0) \\
&= \sum_{x=y_{h,j}}^{|N_{h,j}|} C_{|N_{h,j}|}^x \epsilon^x (1 - \epsilon)^{|N_{h,j}|-x}
\end{aligned}$$

Similarly, when the target node j is normal node, we have

$$t_{ij} = q \sum_{h \in N_{h,j}} c_{hj} \tag{2.25}$$

And the absorption probability now becomes:

$$q = \frac{1}{|N_{h,j}| + |N_m|} \tag{2.26}$$

Then the false alarm rate can be derived as follows:

$$\begin{aligned}
p(GI_{ij} = 0 | u_j = 1) &= p(t_{ij} < \eta | u_j = 1) \\
&= p\left(x < \frac{\eta}{q} | u_j = 1\right) \\
&= p(x < \eta (|N_m| + |N_{hj}|) | u_j = 0) \\
&= p(x \in \{0, 1, \dots, z_{hj}\} | u_j = 1) \\
&= \sum_{x=0}^{z_{hj}} C_{|N_{hj}|}^x (1 - \epsilon)^x \epsilon^{|N_{hj}|-x} \\
&= \sum_{x=|N_{hj}|-z_{hj}}^{|N_{hj}|} C_{|N_{hj}|}^x \epsilon^x (1 - \epsilon)^{|N_{hj}|-x}
\end{aligned} \tag{2.27}$$

□

Corollary 2.5.1.1. *For a regular trust graph, the single-dimensional decision rule in equation (2.15) should satisfy the following:*

$$\begin{aligned}
|N_{hj}|^2 &> |N_m|(|N_m| - 1) \\
\frac{|N_m| - 1}{|N_{hj}| + |N_m| - 1} &< \eta < \frac{|N_{hj}|}{|N_{hj}| + |N_m|}
\end{aligned} \tag{2.28}$$

Otherwise, the miss detection rate and the false alarm rate will both deteriorate to 1.0.

Proof. For the miss detection rate to take a value strictly less than 1, we need to

have $y_{hj} > 0$ which is equivalent to:

$$\begin{aligned} \eta|N_{hj}| - (1 - \eta)(|N_m| - 1) &> 0 \Leftrightarrow \\ \eta &> \frac{|N_m| - 1}{|N_{hj}| + |N_m| - 1} \end{aligned} \tag{2.29}$$

Similarly, for the false alarm rate to take a value strictly less than 1, we need to have $z_{hj} < |N_{hj}|$ which is equivalent to:

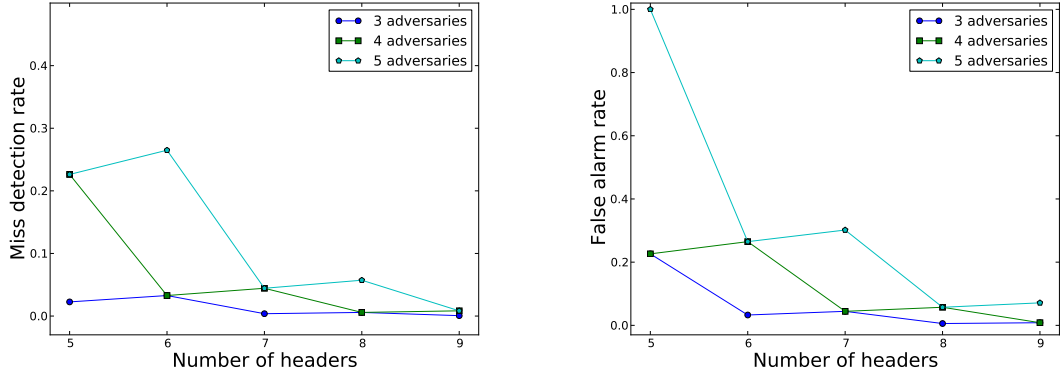
$$\begin{aligned} \eta(|N_{hj}| + |N_m|) &< |N_{hj}| \Leftrightarrow \\ \eta &< \frac{|N_{hj}|}{|N_{hj}| + |N_m|} \end{aligned} \tag{2.30}$$

Combining equation (2.29) and equation (2.30), we get:

$$\frac{|N_m| - 1}{|N_{hj}| + |N_m| - 1} < \eta < \frac{|N_{hj}|}{|N_{hj}| + |N_m|}$$

Letting $\frac{|N_m| - 1}{|N_{hj}| + |N_m| - 1} < \frac{|N_{hj}|}{|N_{hj}| + |N_m|}$, we get the first inequality in equation (2.28). \square

Fig. 2.1(a) shows that with increasing number of headers, the miss detection rate decreases. When the number of headers is six and the number of adversaries is more than three, increasing the number of headers further will not bring much gain however. This is practically beneficial because by deploying a small number of expensive headers, we already have low enough miss detection rate (lower than 0.05 as shown in the figure). Also, when the number of adversaries increases, we need more headers for the same miss detection rate. Next we examine the false alarm rate in Fig. 2.1(b). We observe that for the same number headers and adversaries,



(a) the decision threshold is $\eta = 0.5$, the probability of making a wrong decision by a header is $\epsilon = 0.05$.

(b) $\eta = 0.5$, $\epsilon = 0.05$.

Figure 2.1: The adversary takes random vibration strategy.

the false alarm rate is always worse than the miss detection. This is because when evaluating the trust of a normal node, one more adversaries participate in the trust propagation than when evaluating a malicious node. The figures in Fig. 2.1 serve as a guidance on how many headers we need to deploy given targeted miss detection rate and false alarm rate.

2.5.2 Multi-dimensional decision rules

We analyse the security guarantee of multi-dimensional decision rules. Example rules are consistency-based decision rule. At each iteration k , node i needs to give trust decision on node j 's true type z_j . We have the following theorem regarding the miss detection rate and the false alarm rate:

Theorem 2.5.2. *For a regular trust graph, the miss detection rate and the false*

alarm rate of multi-dimensional decision rules are:

$$\begin{aligned}
p(GI_{ij} = 1|u_j = 0) &= \prod_{l \in N_j} \sum_{x=y_{hjl}}^{|N_{hjl}|} C_{|N_{hjl}|}^x \epsilon^x (1 - \epsilon)^{|N_{hjl}|-x} \\
p(GI_{ij} = 0|u_j = 1) &= \prod_{l \in N_j} \sum_{x=|N_{hjl}|-z_{hjl}}^{|N_{hjl}|} C_{|N_{hjl}|}^x \epsilon^x (1 - \epsilon)^{|N_{hjl}|-x}
\end{aligned} \tag{2.31}$$

where $|N_{hjl}|$ is the number of headers that have direct evidence of node j 's l -th dimension message and can evaluate node j 's trust regarding dimension l . And we have:

$$\begin{aligned}
y_{hjl} &= \lceil \eta |N_{hjl}| - (1 - \eta) (|N_m| - 1) \rceil \\
z_{hjl} &= \lfloor \eta (|N_{hjl}| + |N_m|) \rfloor
\end{aligned} \tag{2.32}$$

Proof. We have:

$$\begin{aligned}
p(GI_{ij} = 1|u_j = 0) &= p(GI_{ij,l} = 1, \forall l \in N_j|u_j = 0) \\
&= \prod_{l \in N_j} p(GI_{ij,l} = 1|u_j = 0) \\
&= \prod_{l \in N_j} \sum_{x=y_{hjl}}^{|N_{hjl}|} C_{|N_{hjl}|}^x \epsilon^x (1 - \epsilon)^{|N_{hjl}|-x}
\end{aligned} \tag{2.33}$$

The first equality holds because different dimensions of augmented messages are evaluated independently. The same derivation process applies to the false alarm rate. \square

2.5.3 Security performance for general trust graphs

The previous discussions assume that the trust graph is regular which makes the absorption probability uniform across all headers. This makes the decision rules equivalent to majority vote with uniform weights. In a general trust graph however, the absorption probability q_{ik} should be different for different source node i and node k pairs. Therefore at equilibrium, the absorption probabilities satisfy:

$$\sum_{h \in N_{h_j}} q_{ih} + \sum_{k \in N_m} q_{ik} = 1 \quad (2.34)$$

where q_{ih} is the absorption probability from source node i to a header node h and q_{ik} is the absorption probability from source node i to a malicious node k .

Theorem 2.5.3. *For a general trust graph, the miss detection rate for evaluating whether target node j is malicious by source node i to is upper bounded by:*

$$\begin{aligned} p(GI_{ij} = 1 | u_j = 0) &\leq \exp \left(-\frac{2 \left(\eta - \epsilon \sum_{h \in N_{h_j}} q_{ih} - \sum_{k \in N_m - \{j\}} q_{ik} \right)^2}{\sum_h q_{ih}^2} \right) \\ p(GI_{ij} = 0 | u_j = 1) &\leq \exp \left(-\frac{2 \left(\eta - \epsilon \sum_{h \in N_{h_j}} q_{ih} \right)^2}{\sum_h q_{ih}^2} \right) \end{aligned} \quad (2.35)$$

where η is the decision threshold used in equation (2.15), ϵ is the probability of making a mistake by a header in evaluating target node j .

Proof. When the target node j 's true type is $u_j = 0$, malicious nodes in $N_m - \{j\}$ collude to vote 1 for j in the trust propagation. Therefore the equilibrium global

trust becomes:

$$t_{ij} = \sum_{h \in N_{hj}} q_{ih} c_{hj} + \sum_{k \in N_m} q_{ik} \quad (2.36)$$

According to the Hoeffding-Azuma concentration bound in [28], we have:

$$\begin{aligned} p(GI_{ij} = 1 | u_j = 0) &= p(t_{ij} \geq \eta | u_j = 0) \\ &= p\left(\sum_{h \in N_{hj}} q_{ih} c_{hj} + \sum_{k \in N_m} q_{ik} \geq \eta | u_j = 0\right) \\ &= p\left(\sum_{h \in N_{hj}} q_{ih} c_{hj} \geq \eta - \sum_{k \in N_m} q_{ik} | u_j = 0\right) \\ &= p\left(\sum_{h \in N_{hj}} q_{ih} c_{hj} - \sum_{h \in N_{hj}} q_{ih} \mathbb{E} c_{hj} \geq \eta - \sum_{k \in N_m} q_{ik} - \sum_{h \in N_{hj}} q_{ih} \mathbb{E} c_{hj} | u_j = 0\right) \\ &= p\left(\sum_{h \in N_{hj}} q_{ih} c_{hj} - \epsilon \sum_{h \in N_{hj}} q_{ih} \geq \eta - \sum_{k \in N_m} q_{ik} - \epsilon \sum_{h \in N_{hj}} q_{ih} | u_j = 0\right) \\ &\leq \exp\left(-\frac{2\left(\eta - \epsilon \sum_{h \in N_{hj}} q_{ih} - \sum_{k \in N_m - \{j\}} q_{ik}\right)^2}{\sum_h q_{ih}^2}\right) \end{aligned} \quad (2.37)$$

Similarly, when the target node j 's true type is $u_j = 1$, malicious nodes in N_m collude to vote 0 for j in the trust propagation. The equilibrium global trust becomes:

$$t_{ij} = \sum_{h \in N_{hj}} q_{ih} c_{hj} \quad (2.38)$$

And we have:

$$\begin{aligned}
p(GI_{ij} = 0|u_j = 1) &= p(t_{ij} \leq \eta|u_j = 1) \\
&= p\left(\sum_{h \in N_{h_j}} q_{ih} c_{hj} \leq \eta|u_j = 1\right) \\
&= p\left(\sum_{h \in N_{h_j}} q_{ih} c_{hj} - \sum_{h \in N_{h_j}} q_{ih} \mathbb{E} c_{hj} \leq \eta - \sum_{h \in N_{h_j}} q_{ih} \mathbb{E} c_{hj}|u_j = 1\right) \\
&= p\left(\sum_{h \in N_{h_j}} q_{ih} c_{hj} - (1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih} \leq \eta - (1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih}|u_j = 1\right) \\
&= p\left((1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih} - \sum_{h \in N_{h_j}} q_{ih} c_{hj} \geq (1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih} - \eta|u_j = 1\right) \\
&\leq \exp\left(-\frac{2\left(\eta - (1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih}\right)^2}{\sum_h q_{ih}^2}\right)
\end{aligned} \tag{2.39}$$

□

To get an intuitive idea of how the error rates vary in equation (2.35) with respect to different absorption probabilities corresponding to different trust graphs, we derive the miss detection rates under some assumptions about the trust graph. We omit the discussion of false alarm rates because the derivation is similar to that of the miss detection rates.

Corollary 2.5.3.1. *If in the ideal case the absorption probabilities from source node i to any malicious node $k \in N_m$ satisfy $q_{ik} = 0$ and that $q_{ih} = q_{ih'}, \forall h, h' \in N_{h_j}$, then the miss detection rate is upper bounded by:*

$$p(GI_{ij} = 1|u_j = 0) \leq \exp(-2|N_{h_j}|(\eta - \epsilon)^2) \tag{2.40}$$

Proof. If $q_{ik} = 0, \forall k \in N_m$, we have $\sum_{h \in N_{h_j}} q_{ih} = 1$. Plugging this into the first equation in equation (2.35):

$$\begin{aligned}
p(GI_{ij} = 1 | u_j = 0) &\leq \exp\left(-\frac{2(\eta - \epsilon)^2}{\sum_{h \in N_{h_j}} q_{ih}^2}\right) \\
&= \exp\left(-\frac{2(\eta - \epsilon)^2}{\sum_{h \in N_{h_j}} \frac{1}{|N_{h_j}|^2}}\right) \\
&= \exp(-2|N_{h_j}|(\eta - \epsilon)^2)
\end{aligned} \tag{2.41}$$

The second equality holds because $q_{ih} = \frac{1 - \sum_{k \in N_m} q_{ik}}{|N_{h_j}|} = \frac{1}{|N_{h_j}|}$ □

Corollary (2.5.3.1) gives us a straightforward conclusion that the upper bound of the miss detection rate decreases exponentially with respect to the increase of the number of headers $|N_{h_j}|$. We generalize corollary (2.5.3.1) by the following corollary:

Corollary 2.5.3.2. *If the absorption probabilities from source node i to any malicious node $k \in N_m$ satisfies $\sum_{k \in N_m} q_{ik} \leq \sigma$ where σ is a positive real number such that $\sigma < \frac{\eta - \epsilon}{1 - \epsilon}$ and that $q_{ih} = q_{ih'}, \forall h, h' \in N_{h_j}$, then the upper bound of the miss detection rate becomes:*

$$p(GI_{ij} = 1 | u_j = 0) \leq \exp\left(-2|N_{h_j}| \left(1 - \epsilon - \frac{1 - \eta}{1 - \sigma}\right)^2\right) \tag{2.42}$$

Proof. If $\sum_{k \in N_m} q_{ik} \leq \sigma$, we have $\sum_{h \in N_{h_j}} q_{ih} \geq 1 - \sigma$. Plugging this into the first

equation in equation (2.35):

$$\begin{aligned}
p(GI_{ij} = 1|u_j = 0) &\leq \exp\left(-\frac{2\left(\eta - \epsilon \sum_{h \in N_{h_j}} q_{ih} - \left(1 - \sum_{k \in N_m} q_{ik}\right)\right)^2}{\sum_{h \in N_{h_j}} q_{ih}^2}\right) \\
&= \exp\left(-\frac{2\left(\eta - 1 + (1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih}\right)^2}{\sum_{h \in N_{h_j}} q_{ih}^2}\right) \\
&= \exp\left(-2|N_{h_j}| \frac{\left(\eta - 1 + (1 - \epsilon) \sum_{h \in N_{h_j}} q_{ih}\right)^2}{\left(\sum_{h \in N_{h_j}} q_{ih}\right)^2}\right) \\
&= \exp\left(-2|N_{h_j}| \left(1 - \epsilon - \frac{1 - \eta}{\sum_{h \in N_{h_j}} q_{ih}}\right)^2\right)
\end{aligned} \tag{2.43}$$

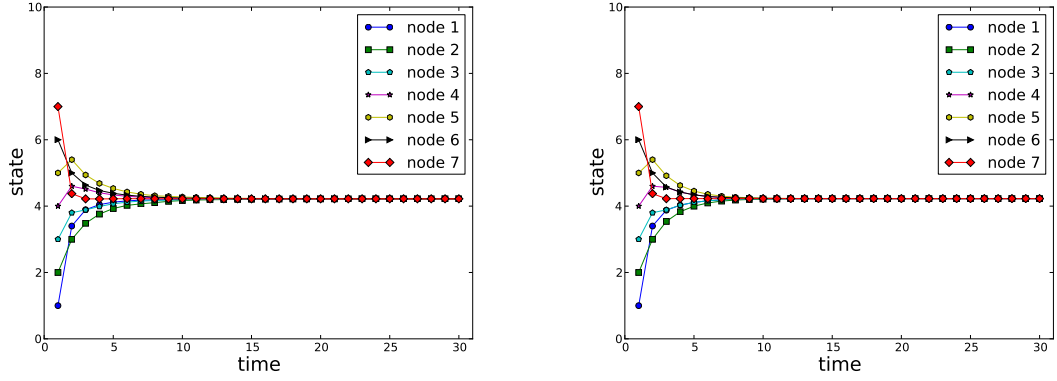
Since $\sum_{h \in N_{h_j}} q_{ih} \geq 1 - \sigma > 1 - \frac{\eta - \epsilon}{1 - \epsilon} = \frac{1 - \eta}{1 - \epsilon}$, we have $1 - \epsilon - \frac{1 - \eta}{\sum_{h \in N_{h_j}} q_{ih}} > 1 - \epsilon - \frac{1 - \eta}{1 - \sigma} > 0$.

By plugging in the inequality $\sum_{h \in N_{h_j}} q_{ih} \geq 1 - \sigma$, we obtain the upper bound in corollary (2.5.3.2). \square

From the above corollary, we not only know that the miss detection rate decreases exponentially as the number of headers increases, but also observe that when σ decreases, the miss detection rate also decreases. This result is very intuitive in that lower absorption probabilities on malicious nodes results in lower miss detection rate.

2.6 Case study and performance evaluation

We consider a sensor network with 7 nodes (sensors) shown in Fig. 2.7. The shaded node is a Byzantine adversary and all the other nodes are normal nodes. In the simulation, each normal node uses both cluster-based decision rule and distance-

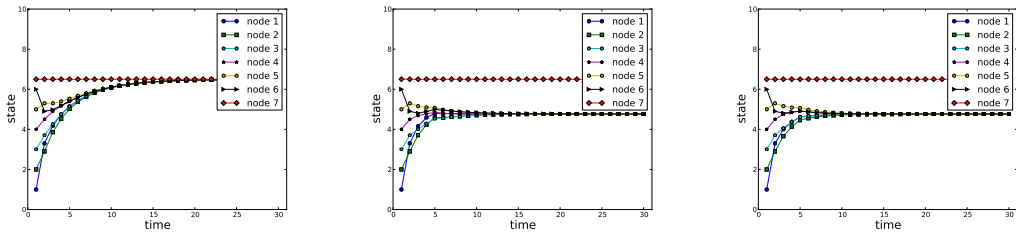


(a) No trust propagation and no adversaries. (b) No adversary. Use trust propagation.

Figure 2.2: Trust-aware consensus algorithm in situations with or without trust propagation when there are no adversaries in the network.

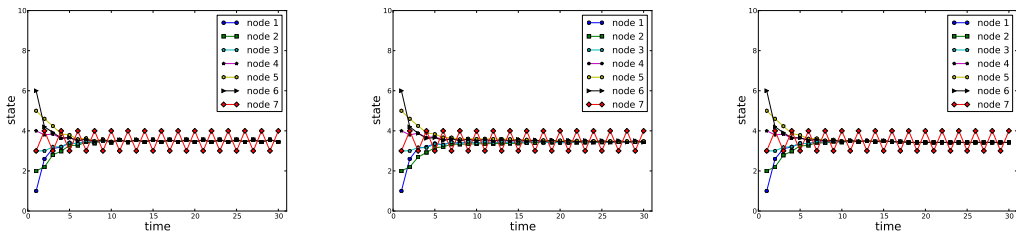
based decision rule to generate local decisions. The consistency-based decision rule is evaluated theoretically in [25]. We assume the probability of choosing either one is set to 0.5. In practice, each node (sensor) can have its own parameters for the probabilities of randomly choosing decision rules. For simulation purposes, we consider the following 4 malicious strategies adopted by the Byzantine adversary:

1. *Remain constant*: the adversary, disregarding the update rule in equation (2.18), holds a constant value.
2. *Random vibration*: the adversary switches between several values randomly at each iteration.
3. *Random noise*: the adversary adds a random noise to the state calculated if it is a normal node.
4. *Fixed noise*: the adversary adds a fixed input to the state calculated if it is a normal node.



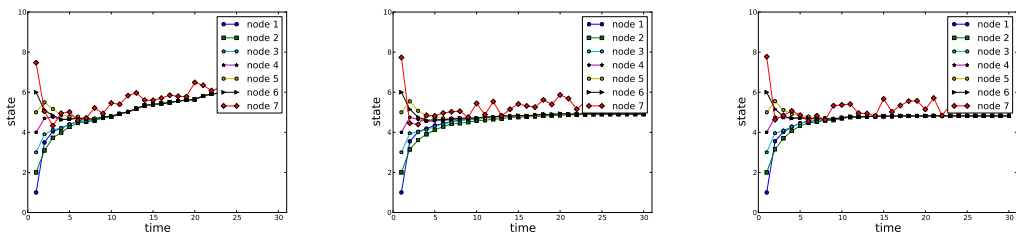
(a) Node 7 is adversary. No trust and adversary adopts main constant strategy. (b) Node 7 is adversary. No trust and adversary adopts remain constant strategy. (c) Node 7 is adversary. Use trust and adversary adopts remain constant strategy.

Figure 2.3: The adversary takes constant strategy.



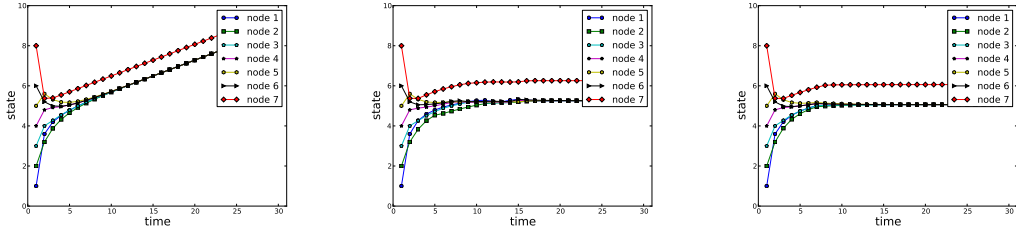
(a) Node 7 is adversary. No trust and adversary adopts random vibration strategy. (b) Node 7 is adversary. No trust and adversary adopts random vibration strategy. (c) Node 7 is adversary. Use trust and adversary adopts random vibration strategy.

Figure 2.4: The adversary takes random vibration strategy.



(a) Node 7 is adversary. No trust and adversary adopts random noise strategy. (b) Node 7 is adversary. No trust and adversary adopts random noise strategy. (c) Node 7 is adversary. Use trust and adversary adopts random noise strategy.

Figure 2.5: The adversary takes random noise strategy.



(a) Node 7 is adversary. No trust. Adversary adopts fixed noise strategy. (b) Node 7 is adversary. Adversary trust propagation. Adversary adopts fixed noise strategy. (c) Node 7 is adversary. Use trust. Adversary adopts fixed noise strategy.

Figure 2.6: The adversary takes fixed noise strategy.

The simulation results are shown in Fig. 2.2, Fig. 2.3, Fig. 2.4, Fig. 2.5, and Fig. 2.6. First look at Fig. 2.2(a) and Fig. 2.2(b). When no adversaries exist, the use of global trust (trust propagation) can speed up convergence. When node 7 is set to be adversary and it adopts remain constant strategy, all nodes can still reach convergence as shown in Fig. 2.3(b). However, all nodes are dragged to closer to the constant input of node 7 because local evidence of nodes are not sufficient to reach good decision at the beginning of the consensus iterations and it takes a period of time before all other nodes can detect the adversary and exclude it from consensus updates. This problem is mitigated when we invoke global trust and take advantage of trust decisions made by other nodes in the network as shown in Fig. 2.3(c). Nodes can detect an adversary much faster than in Fig. 2.3(b) and all normal nodes remain relatively unaffected by node 7. The effects of earlier detection also happens for use of trust propagation when the adversary adopts random vibration strategy as in Fig. 2.4(b) and Fig. 2.4(c). Detection of an adversary with random noise and fixed noise is similar. Local evidences are still sufficient to detect this malicious behavior as shown in Fig. 2.5(b) and Fig. 2.6(b). And with global trust, adversaries

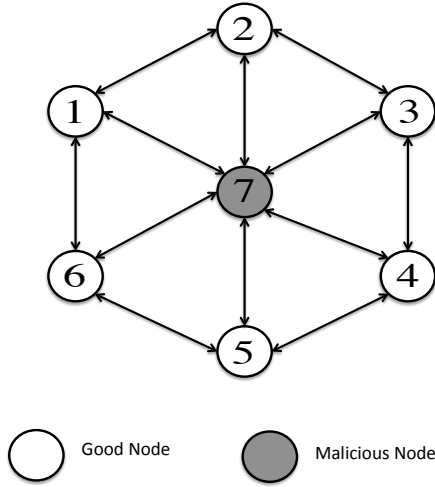


Figure 2.7: Sensor network of 7 nodes (sensors) with the centering shaded node as Byzantine adversary.

are detected at an earlier stage as shown in Fig. 2.5(c) and Fig. 2.6(c).

Next we present the performance of trust-aware consensus algorithm in an even sparser sensor network shown in Fig. 2.8. The communication graph contains seven nodes numbering from 1 to 7 plus a triangle node. The triangle node is a header node and the links connecting the header node and others are not part of the communication graph. However, the dotted links are in the trust graph. Therefore network connectivity of the communication graph in this example network is 2, rendering connectivity-based approaches in most of previous works invalid because $\text{connectivity} < 2f + 1$. The header node does not participate in consensus iterations but is involved in the global trust evaluation process in equation (2.13). The dotted lines indicate that node 5 and node 6 can obtain trust decisions from header. We assume header node can provide trust decisions about node 1 and 4 and since header is anonymous in the eye of normal nodes, normal nodes do not necessarily trust

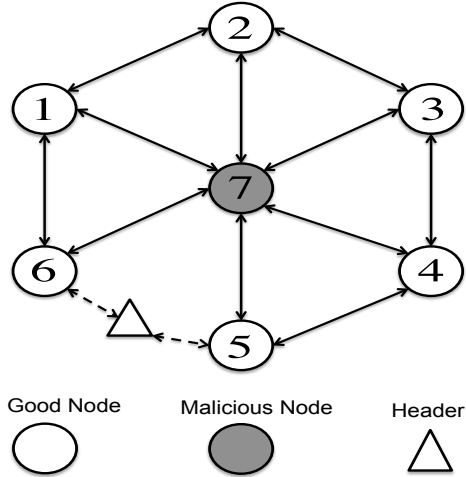
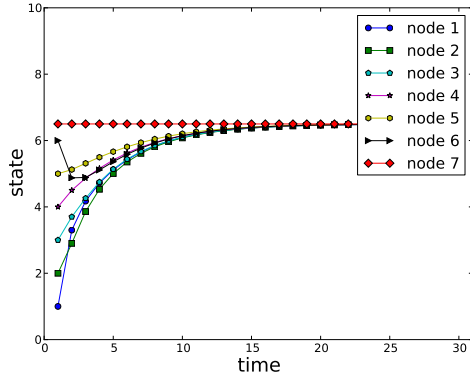
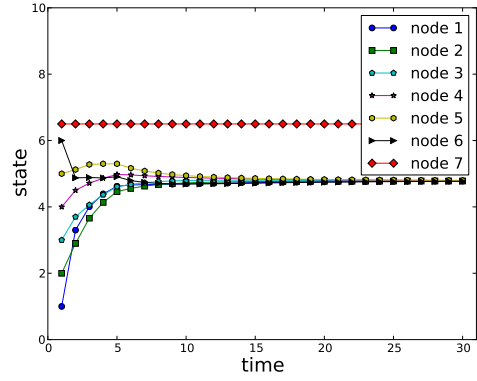


Figure 2.8: Network of 8 nodes (sensors) with the centering shaded node as Byzantine adversary and triangle node as header. The dotted links incident on the header node are not in the communication graph. Instead, they belong to the trust graph. Nodes 1 to 7 form a communication graph of connectivity 2.

headers. Therefore we set the local trust value from node 5 and 6 toward header to be 0.5. The results are shown in Fig. 2.9, Fig. 2.10, Fig. 2.11, and Fig. 2.12. We observe that even in this sparse network, normal nodes can still detect node 7 and reach consensus eventually under each of the adversary strategies.

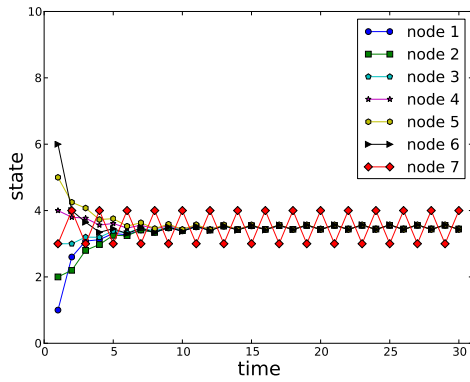


(a) Adversary takes constant strategy. No Trust.

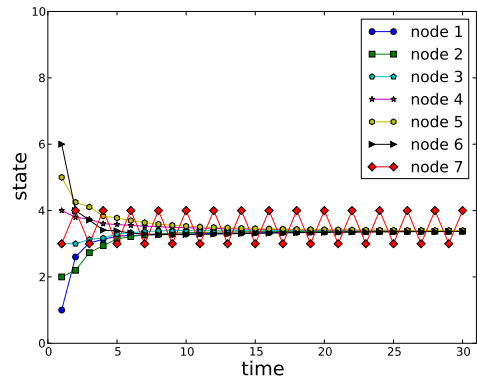


(b) Adversary takes constant strategy.

Figure 2.9: Trust-aware consensus algorithm in sparse network Fig. 2.8 of connectivity 2.. Constant strategy.

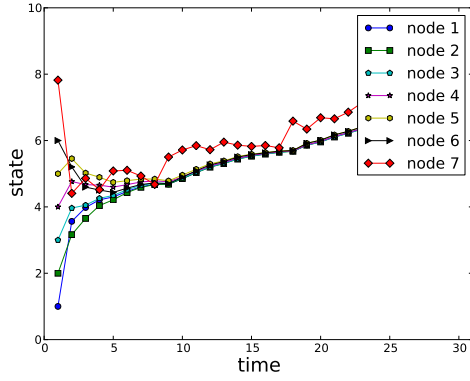


(a) Adversary takes random vibration strategy. No trust.

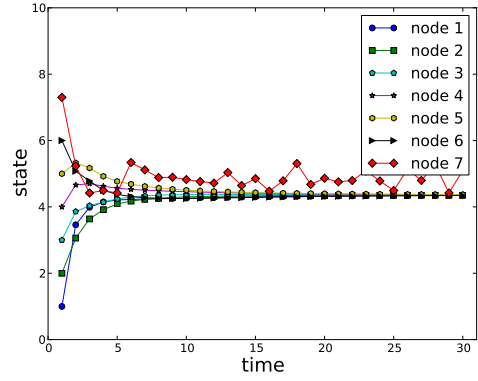


(b) Adversary takes random vibration strategy.

Figure 2.10: Trust-aware consensus algorithm in sparse network Fig. 2.8 of connectivity 2.. Random vibration.

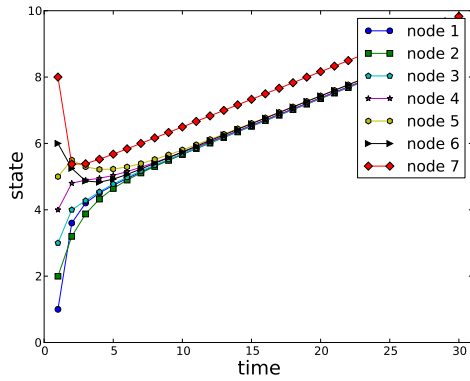


(a) Adversary takes random noise strategy. No trust.

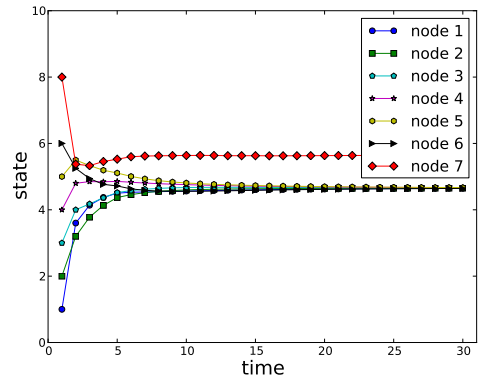


(b) Adversary takes random noise strategy.

Figure 2.11: Trust-aware consensus algorithm in sparse network Fig. 2.8 of connectivity 2.. Random noise.



(a) Adversary takes fixed noise strategy. No trust.



(b) Adversary takes fixed noise strategy.

Figure 2.12: Trust-aware consensus algorithm in sparse network Fig. 2.8 of connectivity 2.. Fixed noise.

Algorithm 1: Trust-Aware Consensus Algorithm

Input: initial states $x_i(0), \forall i \in V$ and initial local trust

$$c_{ij}(0), \forall i \in V \setminus \mathcal{F}, \forall j \in V \cup H, \eta$$

Receive messages from neighbors

// Calculate local trust decisions $I_{ij}(k+1)$

for $\forall i \in V \setminus \mathcal{F}$ **do**

for $\forall j \in N_i$ **do**

 calculate $I_{ij}(k+1)$

end

end

// Trust decision propagation

repeat

for $\forall i \in V \setminus \mathcal{F}$ **do**

for $\forall j \in N_i$ **do**

 // The local trust values $c_{ij}(k)$ remain constant within
 trust iterations.

 update $t_{ij}^\tau(k)$ according to equation (2.13)

end

end

until $|t_{ij}^\tau(k) - t_{ij}^{\tau-1}(k)| < \epsilon;$

Calculate global trust decisions $GI_{ij}(k+1)$ using equation (2.15)

// Update trust values c_{ij}

for $\forall i \in V \setminus \mathcal{F}$ **do**

for $\forall j \in N_i$ **do**

$$r_{ij}(k+1) = \rho_1 r_{ij}(k) + GI_{ij}(k+1)$$

$$s_{ij}(k+1) = \rho_2 s_{ij}(k) + 1 - GI_{ij}(k+1)$$

$$c_{ij}(k) = \mathbb{E}[p_{ij}(k)] = \frac{r_{ij}(k)+1}{r_{ij}(k)+s_{ij}(k)+2}$$

end

end

Update state x_i based on updated c_{ij} according to equation (2.18)

Repeat until convergence

Algorithm 2: Trust-Aware Consensus Algorithm Based on Consistency

Input: initial states $x_i(0), \forall i \in V$ and initial local trust
 $c_{ij}(0), \forall i \in V \setminus \mathcal{F}, \forall j \in V \cup H, \eta$
 Receive augmented message vectors from neighbors
for $\forall i \in V \setminus \mathcal{F}$ **do**
 for $\forall j \in N_i$ **do**
 | compute $I_{ij}^l(k+1)$ according to equation (2.10)
 end
end
 // Trust decision propagation
repeat
 for $\forall i \in V \setminus \mathcal{F}$ **do**
 for $\forall j \in N_i$ **do**
 for $\forall l \in N_j^+$ **do**
 | // The local trust values $c_{ij}(k)$ remain constant
 | within trust iterations.
 | update $t_{ij,l}^\tau(k)$ according to equation (2.13)
 end
 end
 end
until $|t_{ij,l}^\tau(k) - t_{ij,l}^{\tau-1}(k)| < \epsilon$;
 Calculate global trust decisions $GI_{ij,l}(k+1)$ using equation (2.16)
 Calculate global trust decisions $GI_{ij}(k+1) = \prod_{l \in N_j^+} GI_{ij,l}(k+1)$ using
 equation (2.17)
 // Update trust values c_{ij}
for $\forall i \in V \setminus \mathcal{F}$ **do**
 for $\forall j \in N_i$ **do**
 | $r_{ij}(k+1) = \rho_1 r_{ij}(k) + GI_{ij}(k+1)$
 | $s_{ij}(k+1) = \rho_2 s_{ij}(k) + 1 - GI_{ij}(k+1)$
 | $c_{ij}(k) = \mathbb{E}[p_{ij}(k)] = \frac{r_{ij}(k)+1}{r_{ij}(k)+s_{ij}(k)+2}$
 end
end
 Update state x_i based on updated c_{ij} according to equation (2.18)
 Repeat until convergence

Table 2.1: Commonly Used Notations

Notation	Definition
k	the time step at the top layer (communication graph)
τ	the time step at the bottom layer (trust graph) which is a smaller time scale compared to k
V	set of nodes in the network (nodes are the same in both layers)
$x_j^c(k)$	the calculated state of node j according to equation (2.18)
$x_j(0)$	the initial state of node j
$X_j^*(k)$	the messages that node j hears from its neighbors $N_j^+(k-1)$
$\bar{X}_j(k)$	the messages that node j broadcast about what it hears and what it calculates
$f_{ij}(k)$	the trust decision about node j by node i . It takes values from $\{0, 1\}$. It is calculated either from purely local evidence or from both local and second-hand evidence
$I_{ij}(k)$	the local trust decision about node j by node i
$GI_{ij}(k)$	the global trust decision about node j by node i through trust propagation
$t_{ij}^\tau(k)$	the global trust value held by node i toward j at iteration τ for k th round of consensus algorithm
$t_{ij}(k)$	the equilibrium global trust value held by node i toward j at iteration k for consensus algorithm. It is used to calculate global trust decision $GI_{ij}(k)$
$c_{ij}(k)$	the local trust value node i has about node j at iteration k for consensus algorithm. Either $I_{ij}(k)$ or $GI_{ij}(k)$ is used to update it. It incorporates both the trust value history of node j and the trust decisions from other nodes

Chapter 3: Worker Trust In Crowdsourcing With Adversaries

3.1 Enhancing data fusion using multi-dimensional trust

3.1.1 Motivation

In a crowdsourcing task, in order to estimate the true labels of questions, each question is distributed to the open crowd and is answered by a subset of workers. The answers from workers are then aggregated, taking into account the reliability of workers, to produce final estimates of true labels. Example questions are image label inference with multiple annotators' input, topic-document pair relevance inference with crowd's judgements, Bayesian network structure learning given experts' partial knowledge, and test grading without knowing answers. Most past research ignores the multi-domain property present in the questions above. For example in test grading without golden truth, bio-chemistry questions require knowledge in both biology and chemistry. Some are more related to biology while others are more related to chemistry. Similarly, workers also exhibit such multi-domain characteristics: people are good at different subjects. The above observations motivate our modeling of multi-domain characteristics for both questions and trust in workers' knowledge and the design of principled methods for aggregating knowledge input

from various unreliable sources with different expertise in each domain.

In this problem, we propose to model each question by a *concept vector*, which is a real random vector where the value in a particular dimension indicates its association in that dimension [29]. Back to the test grading example, each bio-chemistry question is represented by a two-dimensional hidden concept vector with the first dimension being chemistry and the second dimension being biology. So a concept vector $[0.7, 0.3]$ means the question is more associated with chemistry. Note that the concept vector can be far more general than this. In the case of identifying causal relationships between entities, reasoning ability and past experience are two dimensions of the concept vector. Each worker is modeled also by a trust vector, which is a real random vector with each dimension representing the trustworthiness of the worker in that dimension. The multi-domain property of questions and workers for the biology-chemistry example is illustrated in Fig. 3.1. Our goal is to better estimate the true labels of question Q by fusing answers from multiple unreliable workers with varying trust values in each of the domains. Note that the concept vectors of questions and the trust vectors of workers are both hidden. We therefore propose a probabilistic model that incorporates questions' concept vectors, workers' trust vectors, answers submitted by workers and design an inference algorithm that jointly estimates the true label of questions along with concept vectors and trust vectors. The inference algorithm is based on a variational approximation of posterior distributions using a factorial distribution family. In addition, we extend the model by incorporating continuously-valued features. In applications where each question is associated with a short text description, each dimension of the concept

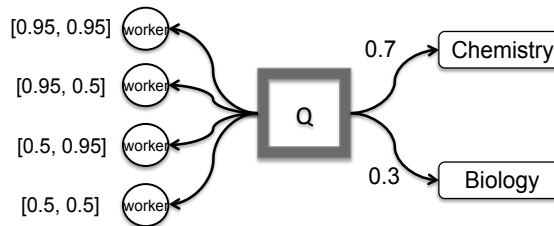


Figure 3.1: Multi-domain property of questions and workers in the test grading example. Q represents a question with concept vector $[0.7, 0.3]$ shown on the edges. Several workers with different two-dimensional trust vectors provide answers.

vector corresponds to a topic. Therefore we further propose an extended model that integrates topic discovery.

Our contributions are as follows:

- We formulate a probabilistic model of crowdsourcing tasks with *multi-domain* characteristics and propose a novel inference method based on variational inference.
- Our model is very flexible and can be easily extended. In applications where each question comes with a feature vector, we further develop an extended model that handles questions with continuously-valued features.
- We further extend the model by combining a multi-domain crowdsourcing model with topic discovery based on questions' text descriptions and derive an analytical solution to the collective variational inference.

3.1.2 Related work

There are a lot of works on how to leverage trust models to better aggregate information from multiple sources. Conflicts between information provided by

different sources were used to revise trust in the information [30]. Trust was also used as weights of edges in the sensor network and was integrated into distributed Kalman filtering to more accurately estimate the state of a linear dynamical system in a distributed setting [20]. Local evidence was leveraged to establish local trust between agents in a network and those local trusts were then used to isolate untrustworthy agents during sensor fusion [23].

In the context of crowdsourcing tasks to the open crowd, many works develop models for aggregating unreliable input from multiple sources to more accurately estimate true labels of questions. [31] combined multiple weak workers' input for constructing a Bayesian network structure assuming each worker is equally trustworthy. Workers' trust was considered to improve accuracy in aggregating answers in [32–35].

A model that jointly infers label of image, trust of each labeler and difficulty of image is proposed in [36]. However, they model questions and workers using scalar variables and they use the Expectation-Maximization inference algorithm, which has long been known to suffer from many local optima difficulties. Another work that went a step further based on signal detection theory is [37], where they assume each question comes with a feature set and models each worker by a multidimensional classifier in an abstract feature space. Our model can handle more general cases without such an assumption and when text information is available for each question, each dimension of a question becomes interpretable. Moreover, it is difficult to find analytical solutions to posterior distributions of hidden variables in [37]. An approach in the spirit of test theory and item-response theory (IRT) was proposed

in [38] and they relied on approximate message-passing for inference. Their model is not as flexible and extensivle as our model because they have to redesign their model to incorporate rich metadata associated with each question.

3.1.3 Definitions

We assume there are M workers available and N questions whose true labels need to be estimated. We use R_i to denote the true label variable of question i , where $R_i \in \{0, 1\}$. Each question is answered by a subset of workers M_i and we denote the answer of question i given by worker j by $l_{ij} \in \{0, 1\}$. The set of questions answered by worker j is denoted by N_j .

The multi-domain characteristics of question i are represented by a concept vector λ_i , a D -dimensional real-valued random vector, where D is the total number of domains. To simulate a probability distribution, we further require $\lambda_{il} \in [0, 1]$, $l = 1, \dots, D$ and $\sum_{l=1}^D \lambda_{il} = 1$, where λ_{il} denotes the l th dimension of the concept vector. We impose a Dirichlet prior distribution for concept vector λ_i with hyperparameter $\alpha = \{\alpha_l\}_{l=1}^D$, where α_l denotes the soft counts that specify which domain a question falls into a priori.

Workers contribute to the estimation of the true label of questions by providing their own guesses. However, workers' inputs may not be reliable and sometimes even malicious. In multi-domain crowdsourcing tasks, different workers may be good at different domains. The multi-dimensional characteristics of a worker is described by a D -dimensional trust vector $\beta_j = \{\beta_{j1}, \dots, \beta_{jl}, \dots, \beta_{jD}\}$, where β_{jl} denotes j -th

worker’s trust value in domain l and it takes either a continuous or a discrete value. In the discrete case, the inference is generally NP-hard and message-passing style algorithms are used. We consider the continuous case only where $\beta_j \in [0, 1]^D, \forall j$. Higher value of β_{jl} indicates that worker j is more trustworthy in domain l . The true value of β_{jl} is usually unknown to the crowdsourcing platform. It has to be estimated from answers provided by workers. We assume a Beta prior distribution for β_{il} with hyper-parameter $\theta = \{\theta_0, \theta_1\}$, where $\theta_0 > 0$ is the soft count for worker j to behave maliciously and $\theta_1 > 0$ is the soft count for worker j to behave reliably. This interpretation resembles the Beta reputation system [24] that models beliefs of workers.

We aim to estimate the true labels of questions and trust vectors of workers from answers provided by workers.

3.1.4 Multi-domain crowdsourcing model

We describe the generating process for the Multi-Domain Crowdsourcing (MDC) Model in this section.

1. For each question $i \in \{1, \dots, N\}$,
 - (a) draw the domain distribution $\lambda_i | \alpha \sim \text{Dir}(\alpha)$;
 - (b) draw domain $C_i | \lambda_i \sim \text{Discrete}(\lambda_i)$;
2. For each question i , draw the true label $R_i \sim \text{Uniform}(0, 1)$;
3. For each worker $j \in \{1, \dots, M\}$ and domain $l \in \{1, \dots, D\}$, draw the trust

value $\beta_{jl} \sim \text{Beta}(\theta)$;

4. For each question-worker pair (i, j) , draw observed answer $l_{ij} \sim F(R_i, \beta_j, C_i)$

In step 1, the domain for question i is then drawn according to a discrete distribution with parameter λ_i , i.e. generating $C_i = l$ with probability λ_{il} . In step 3, we profile each worker by a vector β_j with β_{jl} drawn from a Beta distribution. In step 4, the observed answer of question i provided by worker j is drawn according to an output distribution F , a Bernoulli distribution. We will specify the form of this output distribution in the following paragraph.

The generating process is illustrated in Fig. 3.2. The joint probability distribution is

$$p(L, R, \beta, C, \lambda) = \prod_{i=1}^N p(r_i) p(\lambda_i | \alpha) p(C_i | \lambda_i) \cdot \prod_{j=1}^M p(\beta_j) \prod_{l=1}^D p(l_{ij} | r_i, C_i = l, \beta_j) \quad (3.1)$$

where N is the total number of questions, M is the total workers, and D is the total number of domains. $p(l_{ij} | r_i, C_i = l, \beta_j)$ is the output distribution F in Fig. 3.2 and is the likelihood of worker j 's answer given its expertise vector and the domain variable of question i , and the true label. $p(r_i)$, and $p(\beta_j)$ are prior distributions. F can be compactly expressed as:

$$p(l_{ij} | r_i, C_i = l, \beta_j) = \beta_{jl}^{\mathbb{1}\{l_{ij}=r_i\}} (1 - \beta_{jl})^{\mathbb{1}\{l_{ij} \neq r_i\}} \quad (3.2)$$

where $\mathbb{1}\{l_{ij} = r_i\}$ is an indicator function taking the value of 1 if the observed label given by worker j to question i is equal to the ground truth. We assume a non-

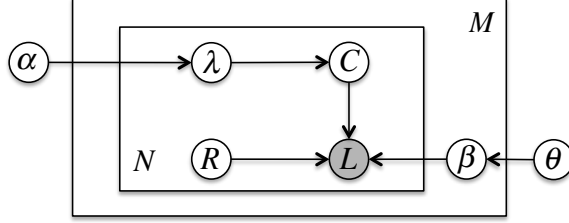


Figure 3.2: The graphical model for observed data provided by workers L , multi-domain expertise β , true labels R , domain variables C , and concept vectors λ . M is the total number of workers. N the number of questions. α is the hyperparameter of the Dirichlet prior distribution for λ and θ is the hyperparameter of the Beta prior distribution for β .

informative prior for true label $p(r_i = 1) = p(r_i = 0) = \frac{1}{2}$.

In order to estimate the questions' true labels $r_i, i = 1, \dots, N$ and workers' trust vectors $\beta_j, j = 1, \dots, M$, their posterior distributions need to be computed. However, the computation of posterior distributions involves integrating out a large number of variables, making the computation intractable. We propose to use a variational approximation of the posterior distribution of variables in equation (3.1) with a factorized distribution family:

$$q(R, \beta, C, \lambda) = \prod_i q(r_i) \prod_i q(\lambda_i | \tilde{\alpha}_i) \prod_i q(C_i) \prod_{j,l} q(\beta_{jl} | \tilde{\theta}_{jl}) \quad (3.3)$$

The optimal forms of these factors are obtained by maximizing the following lower bound of the log likelihood of observed labels $\ln p(L)$:

$$\ln p(L) \geq \mathbb{E}_q \ln p(L, R, \beta, C, \lambda) - \mathbb{E}_q \ln q(R, \beta, C, \lambda) \quad (3.4)$$

We show inference details in Algorithm (3). Updates for each factor are derived in the Appendix. Upon convergence of Algorithm (3), we obtain the approximate

posterior distributions of the questions' true labels $\{r_i\}$'s and of the workers' trust vectors $\{\beta_j\}$'s.

Algorithm 3: Multi-Domain Crowdsourcing

Input: initial values of hyperparameters α, θ

Output: approximate posterior $q(R, \beta, C, \lambda)$

Do the following updates repeatedly until convergence.

1) First update $q(\beta_j), \forall j = 1, \dots, M, l = 1, \dots, D$, sequentially, in the following way:

$$\beta_{jl} \sim \text{Beta} \left(\tilde{\theta}_{jl0}, \tilde{\theta}_{jl1} \right) \quad (3.5)$$

where $\tilde{\theta}_{jl0} = \theta_{jl0} + \sum_{i \in N_j} q(C_i = l)q(R_i \neq l_{ij})$ and

$\tilde{\theta}_{jl1} = \theta_{jl1} + \sum_{i \in N_j} q(C_i = l)q(R_i = l_{ij})$.

2) Then update $q(r_i), \forall i = 1, \dots, N$, sequentially, in the following way:

$$\begin{aligned} \ln q(r_i) \propto \ln p(r_i) + \\ \sum_{j \in M_i} \sum_{l=1}^D q(C_i = l) \left[\delta_{ij} \left(\psi(\tilde{\theta}_{jl1}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) + (1 - \delta_{ij}) \left(\psi(\tilde{\theta}_{jl0}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) \right] \end{aligned} \quad (3.6)$$

where $\psi(\cdot)$ is digamma function. Then normalize $q(r_i), r_i \in \{0, 1\}$ to make them valid probabilities.

3) Then update $q(\lambda_i)$:

$$q(\lambda_i) \sim \text{Dir} \left(\{\tilde{\alpha}_{il}\}_{l=1}^D \right) \quad (3.7)$$

where $\text{Dir}(\cdot)$ is Dirichlet distribution and $\tilde{\alpha}_{il} = \alpha_l + q(C_i = l)$.

4) Then update $q(C_i = l)$:

$$\begin{aligned} \ln q(C_i) \propto \psi(\tilde{\alpha}_{il}) - \psi \left(\sum_{k=1}^D \tilde{\alpha}_{ik} \right) \\ + \sum_{j \in M_i} \left[q(r_i = l_{ij}) \left(\psi(\tilde{\theta}_{jl1}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) + q(r_i \neq l_{ij}) \left(\psi(\tilde{\theta}_{jl0}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) \right] \end{aligned} \quad (3.8)$$

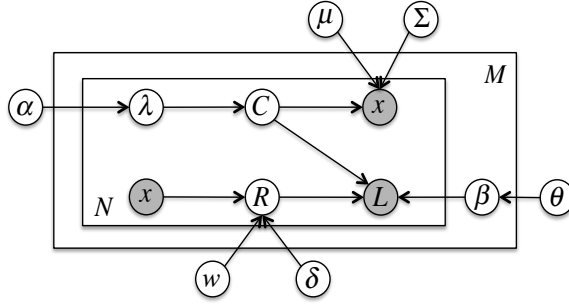


Figure 3.3: The graphical model for observed data provided by workers L , features x , multi-domain expertise β , true labels R , domain variables C , and parameter for domain distribution λ . μ , Σ , w , and δ are model parameters.

3.1.5 Integration with features

Algorithm (3) ignores features of questions. In most cases we do have features associated with questions. These features help us better estimate both the questions' true labels and the workers' trust vectors. Our proposed model MDC can be easily extended to incorporate question features. The extended graphical model is shown in Fig. 3.3, where x denotes the features observed. We call this extended model MDFC. Intuitively, the features associated with questions allow us to better estimate the questions' concept vectors and the workers' trust vectors so that true labels of questions can be more accurately inferred.

Let's assume question i 's feature vector x_i is a K -dimensional real-valued vector. The likelihood of feature x_i , given domain variable C_i , is modeled by a multivariate Gaussian distribution with μ_l ' as the K -dimensional mean vector of the l -th domain and Σ_l as the $K \times K$ covariance matrix:

$$\ln p(x_i | C_i = l) \propto -\frac{1}{2} (x_i - \mu_l)^\top \Sigma_l^{-1} (x_i - \mu_l) - \frac{1}{2} \ln |\Sigma_l|, \quad (3.9)$$

where $|\Sigma_l|$ denotes the determinant of the covariance matrix of the l -th domain. The conditional distribution of the true label variable R_i , given feature variable x_i , can take various forms. We use the logistic regression model:

$$p(r_i = 1|x_i) = (1 + \exp(-w^\top x_i - \delta))^{-1} \quad (3.10)$$

where w is the regression coefficient and δ is the intercept for the regression model.

The inference and parameter estimation of MDFC differs from Algorithm (3) in three ways: first, the update of $q(C_i)$ includes an extra term $\ln p(x_i|C_i = l)$; second, the update of $q(r_i)$ includes an additional term $p(r_i|x_i)$; third, there is an additional M-step to estimate model parameters μ_l 's, Σ_l 's, w , and δ given current approximate posteriors. The details of variational inference and model parameter estimation of MDFC is similar to that of MDTC and are shown in the Appendix.

3.1.6 Multi-domain crowdsourcing model with topic model

In many crowdsourcing applications, we can often get access to questions' text descriptions. Given the text description, we can use the latent Dirichlet allocation to extract topic distribution of a question [39]. The advantage of topic models over the Gaussian mixture model in Section 3.1.5 is that the domains (topics) are of low dimensions and are easier to interpret. For example, using topic models, a question might be assigned to the domain of sports while another question assigned to music domain. For a crowdsourcing platform, it needs to profile a worker's trust in all these interpretable topics instead of some latent unexplainable domain. We call this

Algorithm 4: Multi-Domain Crowdsourcing With Features

Input: initial values of hyperparameters α, θ

Output: approximate posterior $q(R, \beta, C, \lambda)$

E-step and M-step are repeated until convergence

E-step: Given current estimation of model parameters μ_l 's, Σ_l 's, w , and δ :

Do the following updates repeatedly until convergence.

1) First update $q(\beta_j), \forall j = 1, \dots, M, l = 1, \dots, D$, sequentially, in the following way:

$$\beta_{jl} \sim \text{Beta} \left(\tilde{\theta}_{jl0}, \tilde{\theta}_{jl1} \right) \quad (3.11)$$

where $\tilde{\theta}_{jl0} = \theta_{jl0} + \sum_{i \in N_j} q(C_i = l)q(R_i \neq l_{ij})$ and

$\tilde{\theta}_{jl1} = \theta_{jl1} + \sum_{i \in N_j} q(C_i = l)q(R_i = l_{ij})$.

2) Then update $q(r_i), \forall i = 1, \dots, N$, sequentially, in the following way:

$\ln q(r_i) \propto \ln p(r_i)$

$$\begin{aligned} &+ \sum_{j \in M_i} \sum_{l=1}^D q(C_i = l) \left[\delta_{ij} \left(\psi(\tilde{\theta}_{jl1}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) + (1 - \delta_{ij}) \left(\psi(\tilde{\theta}_{jl0}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) \right] \\ &- \log(1 + \exp(-w^T x - \delta)) \end{aligned} \quad (3.12)$$

where $\psi(\cdot)$ is digamma function. Then normalize $q(r_i), r_i \in \{0, 1\}$ to make them valid probabilities.

3) Then update $q(\lambda_i)$:

$$q(\lambda_i) \sim \text{Dir} \left(\{\tilde{\alpha}_{il}\}_{l=1}^D \right) \quad (3.13)$$

where $\text{Dir}(\cdot)$ is Dirichlet distribution and $\tilde{\alpha}_{il} = \alpha_l + q(C_i = l)$.

4) Then update $q(C_i = l)$:

$$\begin{aligned} \ln q(C_i = l) &\propto \psi(\tilde{\alpha}_{il}) - \psi \left(\sum_{k=1}^D \tilde{\alpha}_{ik} \right) \\ &+ \sum_{j \in M_i} \left[q(r_i = l_{ij}) \left(\psi(\tilde{\theta}_{jl1}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) + q(r_i \neq l_{ij}) \left(\psi(\tilde{\theta}_{jl0}) - \psi(\tilde{\theta}_{jl1} + \tilde{\theta}_{jl0}) \right) \right] \\ &- \frac{1}{2} (x_i - \mu_l)^T \Sigma_l^{-1} (x_i - \mu_l) - \frac{1}{2} \ln |\Sigma_l| (x_i - \mu_l) \end{aligned} \quad (3.14)$$

Algorithm 4: Multi-Domain Crowdsourcing With Features (Continued)

M-step: Given current approximate posterior distributions, obtain the estimates of μ_l 's, Σ_l 's, w , and δ by maximizing the expectation of the logarithm of the posterior:

$$\begin{aligned} \mu_l^{new} &= \frac{\sum_{i=1}^N q(C_i = l) x_i}{\sum_{i=1}^N q(C_i = l)} \\ \Sigma_l^{new} &= \frac{\sum_i q(C_i = l) (x_i - \mu_l^{new}) (x_i - \mu_l^{new})^T}{\sum_{i=1}^N q(C_i = l)} \\ \frac{\partial Q}{\partial w} &= \sum_{i=1}^N [q(r_i = 1)\sigma(w^T x_i + \delta) - q(r_i = 0)\sigma(-w^T x_i - \delta)] x_i \\ \frac{\partial Q}{\partial \delta} &= \sum_{i=1}^N [q(r_i = 1)\sigma(w^T x_i + \delta) - q(r_i = 0)\sigma(-w^T x_i - \delta)] \\ w^{new}, \delta^{new} &= \arg \max_{w, \delta} \mathbb{E} \ln p(L, R, \beta, C, \lambda | \{\mu_l^{new}\}_{l=1}^D, \{\Sigma_l^{new}\}_{l=1}^D, w, \delta) \\ &\text{using } L\text{-BFGS quasi-Newton method} \end{aligned} \tag{3.15}$$

extended model with topic discovery MDTC and we will exploit the topic discovery of questions in the experiments section.

Each topic corresponds to one domain of a question. The learned topic distribution can then be used as a damping prior for domain variable C . We show that our MDC is flexible to incorporate topics models and it is an easy extension to jointly infer topic distribution and the true labels of questions and the workers' trust vectors in equation (3.1).

In addition to obtaining posterior probability distributions for R, β, C, λ , we can also obtain the posterior distribution for the topic distribution for the k -th word in the i -th question z_{ik} , and the word distribution for l -th topic ϕ_l simultaneously. Denote n_{iw} as the number of occurrences of word w in question i and η_{iwl} as the probability that the word w in question i is associated with domain l . The variational

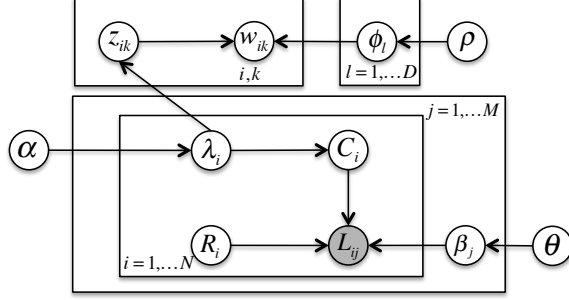


Figure 3.4: The graphical model for MDTC. L are observed answers from workers, w_{ik} is word k observed in question i , multi-domain expertise β , true labels R , domain variables C , parameter for domain distribution λ , topic distribution for word k in question i : z_{ik} , word distribution for domain l : ϕ_l .

inference process differs from Algorithm (3) in the following ways:

1. The λ_i 's have a Dirichlet posterior distribution with parameter $\alpha_l + q(C_i = l) + \sum_w n_{iw} \eta_{iwl}$ where $\sum_w n_{iw} \eta_{iwl}$ is the additional term introduced by topic discovery.
2. The update of $q(z_{iw} = l) = \eta_{iwl}$ follows:

$$\ln \eta_{iwl} \propto \mathbb{E}_q \ln p(z_{iw} = l | \lambda_i) + \mathbb{E}_q \ln \phi_{lw} \quad (3.16)$$

where $\phi_{lw} = p(w_{ik} = w | \phi, z_{ik} = l)$.

3. The ϕ_l 's have a Dirichlet posterior distribution with parameter $\tilde{\Upsilon}_l$ as follows:

$$\tilde{\Upsilon}_{lw} = \Upsilon + \sum_i n_{iw} \eta_{iwl} \quad (3.17)$$

where Υ is the hyper-parameter of the Dirichlet prior distribution.

Algorithm 5: Multi-Domain Crowdsourcing With Topic Model

Input: initial values of hyperparameters α, θ

Output: approximate posterior $q(R, \beta, C, \lambda)$

Do the following updates repeatedly until convergence.

1) First update $q(\beta_j), \forall j = 1, \dots, M, l = 1, \dots, D$, sequentially, in the following way:

$$\beta_{jl} \sim \text{Beta} \left(\tilde{\theta}_{j10}, \tilde{\theta}_{j11} \right) \quad (3.18)$$

where $\tilde{\theta}_{j10} = \theta_{j10} + \sum_{i \in N_j} q(C_i = l)q(R_i \neq l_{ij})$ and

$\tilde{\theta}_{j11} = \theta_{j11} + \sum_{i \in N_j} q(C_i = l)q(R_i = l_{ij})$.

2) Then update $q(r_i), \forall i = 1, \dots, N$, sequentially, in the following way:

$\ln q(r_i) \propto \ln p(r_i) +$

$$\sum_{j \in M_i} \sum_{l=1}^D q(C_i = l) \left[\delta_{ij} \left(\psi(\tilde{\theta}_{j11}) - \psi(\tilde{\theta}_{j11} + \tilde{\theta}_{j10}) \right) + (1 - \delta_{ij}) \left(\psi(\tilde{\theta}_{j10}) - \psi(\tilde{\theta}_{j11} + \tilde{\theta}_{j10}) \right) \right] \quad (3.19)$$

where $\psi(\cdot)$ is digamma function. Then normalize $q(r_i), r_i \in \{0, 1\}$ to make them valid probabilities.

3) Then update $q(\lambda_i)$:

$$q(\lambda_i) \sim \text{Dir} \left(\{\tilde{\alpha}_{il}\}_{l=1}^D \right) \quad (3.20)$$

where $\text{Dir}(\cdot)$ is Dirichlet distribution and $\tilde{\alpha}_{il} = \alpha_l + q(C_i = l) + \sum_w n_{iw} \eta_{iwl}$.

4) Then update $q(C_i = l)$:

$$\begin{aligned} \ln q(C_i) \propto & \psi(\tilde{\alpha}_{il}) - \psi \left(\sum_{k=1}^D \tilde{\alpha}_{ik} \right) \\ & + \sum_{j \in M_i} \left[q(r_i = l_{ij}) \left(\psi(\tilde{\theta}_{j11}) - \psi(\tilde{\theta}_{j11} + \tilde{\theta}_{j10}) \right) + q(r_i \neq l_{ij}) \left(\psi(\tilde{\theta}_{j10}) - \psi(\tilde{\theta}_{j11} + \tilde{\theta}_{j10}) \right) \right] \end{aligned} \quad (3.21)$$

5) Then update $q(\phi_l)$:

$$q(\phi_l) \sim \text{Dir} \left(\{\tilde{\Upsilon}_{lw}\} \right) \quad (3.22)$$

where $\tilde{\Upsilon}_{lw} = \Upsilon + \sum_i n_{iw} \eta_{iwl}$.

6) Then update $q(z_{iw})$:

$$\ln p(z_{iw} = l) = \ln \eta_{iwl} = \psi(\tilde{\alpha}_{il}) - \psi \left(\sum_{k=1}^D \tilde{\alpha}_{ik} \right) + \psi(\tilde{\Upsilon}_{lw}) - \psi \left(\sum_{w'} \tilde{\Upsilon}_{lw'} \right) \quad (3.23)$$

For each i, w , normalize $\{\eta_{iwl}\}_{l=1}^D$ to make them valid probabilities.

3.1.7 Experiments on real datasets

In this section, we compare our proposed models MDC, MDFC, and MDTC with crowdsourcing models with single dimensional trust (SDC) and show that our models have superior performance on both the UCI dataset and scientific text dataset. In addition, our models can effectively recover the workers' trust vectors which can be used to match the right workers to a given task in the future. The models we consider for comparison are listed as follows:

1. MDC: our proposed multi-domain crowdsourcing model without features.
2. MDFC: extended model of MDC with continuously-valued features.
3. MDTC: another extended model of MDC that combines topic model given text descriptions associated with questions.
4. MV: the majority vote as the baseline algorithm.
5. SDC: the state-of-the-art in [35]. We call this algorithm SDC because it is equivalent to MDC when each worker is represented by only a scalar variable (single domain in our case)

3.1.7.1 UCI datasets

We conducted experiments on the **pima** dataset from UCI Machine Learning Repository¹ [40]. Each data instance corresponds to a 8-dimensional feature of an anonymous patient. The dataset consists of 768 data instances and we ask

¹<http://archive.ics.uci.edu/ml/datasets.html?sort=nameUp&view=list>

the following question for each instance: should the patient be tested positive for diabetes. Since there are no worker-provided labels in this dataset, we simulate workers with varying reliability in different domains. We adopt k-means clustering to cluster the data into two clusters (domains). Therefore, each worker is profiled by a two-dimensional random vector. Details of the simulated workers are shown in Table 3.1. Type 1 workers are malicious in both domains, answering questions correctly with probability 0.5, type 2 workers answer questions in domain 0 correctly with probability 0.95 and answer those in domain 1 correctly with probability 0.5 while type 3 workers answer questions in domain 0 correctly with probability 0.5 and answer questions in domain 1 correctly with probability 0.95, and type 4 workers are good at questions in both domains and answer questions correctly with probability 0.95. In order to show that our model MDC and MDFC works with increasing number of workers that are not trustworthy, we simulated several groups of worker settings with increasing number of type 1 workers.

We compare MDC with MV and SDC when no features are included and compare MDFC with MV and SDC when features are incorporated. We use accuracy as the evaluation criterion and report results in Table 3.2, where the first column denotes worker settings.

When features are used, MDFC results in lowest error rates. When features are omitted, MDC and SDC perform nearly equally well. This could be explained by that when features are not utilized to infer domain distributions for questions, the estimated domain distributions by MDC might be inconsistent with the truth. However, MDC is still very attractive because it can still effectively estimate the

Table 3.1: Worker settings for UCI datasets

worker type	domain 0	domain 1
type 1	0.5	0.5
type 2	0.95	0.5
type 3	0.5	0.95
type 4	0.95	0.95

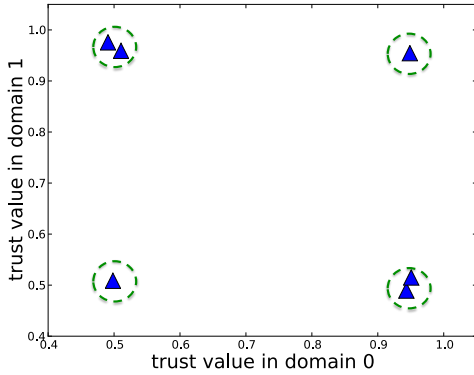
workers’ reliability in different domains as shown in Fig. 3.5(d). This can be useful for task assignment adaptive to workers’ trust in the future. Specifically, upon arrival of a new task, we can use the estimated profile of workers to match the question belonging to a particular domain to a worker that is trustworthy in that domain. For example, consider the case when we need to know the true label of a new question that belongs to a certain domain. Then we can match the question with workers that have the highest reliability in that domain.

In Fig. 3.5, we show that both MDC and MDFC can effectively estimate workers’ trust values in both domains considered. Each triangle stands for a worker’s trust profile (a two-dimensional real-valued trust vector) and the dotted circle is used to cluster workers whose estimated trust values are close to each other. Taking a closer look at Fig. 3.5(a), we see that one worker is clustered close to $(0.51, 0.51)$, two workers close to $(0.95, 0.5)$, two workers close to $(0.5, 0.96)$, and one worker close to $(0.95, 0.95)$. This estimation of trust vectors is consistent with the worker setting $(1, 2, 2, 1)$. Workers’ trust vectors can also be effectively estimated in other worker settings in Fig. 3.5(b), Fig. 3.5(c), and Fig. 3.5(d).

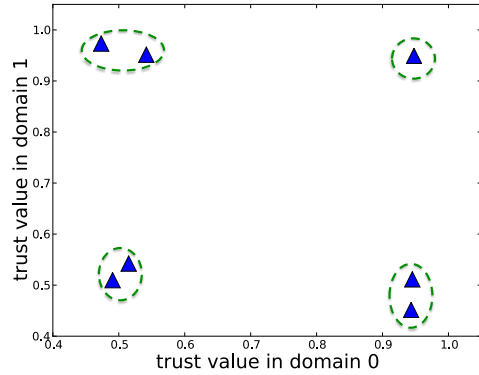
Table 3.2: Error rates of various methods on UCI dataset Pima Indians.

pima dataset	MV	SDC	MDFC	MDC
(1, 2, 2, 1)	0.098	0.040	0.009	×
(2, 2, 2, 1)	0.103	0.042	0.009	×
(3, 2, 2, 1)	0.150	0.042	0.008	×
(1, 2, 2, 1), NF	0.098	0.040	×	0.039
(2, 2, 2, 1), NF	0.103	0.042	×	0.043
(3, 2, 2, 1), NF	0.150	0.042	×	0.041

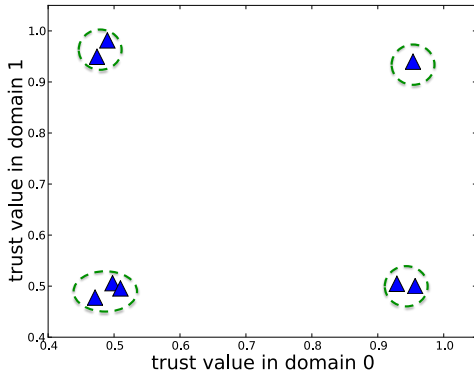
In the expression (1, 2, 2, 1), the four numbers from left to right mean: there are 1 worker of type 1, 2 workers of type 2, 2 workers type 3, and 1 worker of type 4. NF means omitting the features in pima dataset.



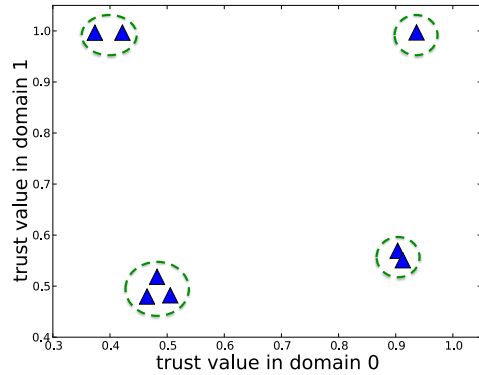
(a) Estimated mean value of trust about workers' knowledge given worker setting (1,2,2,1).



(b) worker setting (2,2,2,1)



(c) worker setting (3,2,2,1)



(d) worker setting (3,2,2,1) and no features available

Figure 3.5: Estimated worker reliability under different simulation settings on pima indians dataset. The estimated trust about workers' knowledge in Fig. 3.5(a), Fig. 3.5(b), and Fig. 3.5(c) are by MDFC and the results in Fig. 3.5(d) are by MDC.

3.1.7.2 Text Data

To evaluate MDTC, we tested our model on 1000 sentences from the corpus of biomedical text with each sentence annotated by 5 workers [41]. Each worker answers whether a given sentence contains contradicting statements (Polarity). Each sentence has the scientific text along with the labels provided by 5 experts. However, since the labels provided by experts are almost consensus and the naive majority vote algorithm gives ground truth answers, we need to simulate workers of varying trust of knowledge in different topics. When the number of topics (domains) is D , we simulate D workers in total, where worker j answers topic j close to perfectly (probability of right guess 0.97) and answers questions in topics other than j nearly randomly (probability of right guess 0.64). For each simulation setting, we repeat 30 times and report the mean error rate.

To the best of our knowledge, there is no comparative model that integrates topics models into a probabilistic crowdsourcing framework in the literature, therefore we compare the performance of MDTC with MDC that ignores topic information and with the baseline majority vote algorithm. The mean error rates are reported in Table 3.3. We can see that in all experiments with the number of topics ranging from 4 to 14, MDTC gives the lowest error rate, outperforming MDC by over 50%. This strongly demonstrates the power of MDTC over other models that do not take into account text information.

To further show that MDTC can effectively recover the reliability of workers in different topics, we plot, for each worker, the mean value of trust for each worker

Table 3.3: Error rates of various methods on Text:Polarity. $T4$ denotes the assumption of 4 topics.

scientific text	MV	MDC	MDTC
$T4$	0.181	0.095	0.044
$T6$	0.160	0.089	0.037
$T8$	0.141	0.082	0.034
$T10$	0.125	0.074	0.032
$T12$	0.116	0.069	0.032
$T14$	0.100	0.064	0.032

in each of the eight topics ($T8$) as a heatmap in Fig. 3.6. The x-axis denotes topic index and the y-axis denotes worker index. The intensity of the color in the j -th row and l -th column denotes the trust value of worker j in l -th dimension. We can see that the diagonal blocks have more intense color than others, which is consistent with the simulation setting where each worker $j \in \{0, 1, \dots, 7\}$ is trustworthy in topic j and is not reliable in topics other than j . The estimated trust vectors of workers in all eight topics can be very useful in the following scenario: if for example a new question with concept vector $[0.93, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]$ is added, we probably want to match this question with a worker whose trust vector has high value in the first dimension. The representative words (top 10 words with the highest probability in a particular topic) in all eight topics are shown in Table 3.4.

3.1.8 Proofs

This section derives the approximate posteriors in MDC, MDTC and MDTC using variational inference.

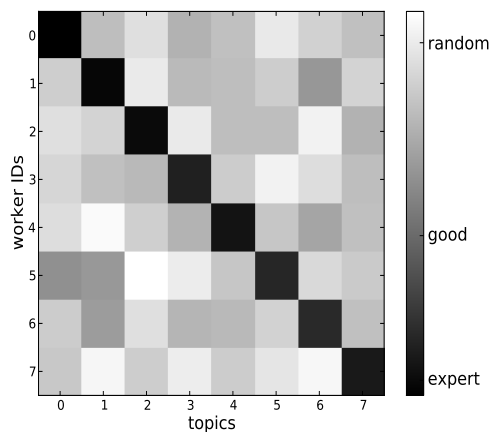


Figure 3.6: Trust matrix about workers' knowledge over topics estimated by MDTC model.

Table 3.4: representative words in topics on scientific text

topic0	ins, protein, cells, express, activity, mutant, resulted, similar, human, rna
topic1	ins, cells, binding, two, presence, day, method, study, acids, reporter
topic2	ins, binding, process, protein, cells, quot, factor, structure, dna, splice
topic3	ins, cells, blotting, protein, using, analysis, western, express, antibodies, demonstrate
topic4	ins, signal, wnt, cells, activity, resulted, using, protein, pathway, regulation
topic5	ins, system, two, sequences, suggest, cloning, data, effects, transcripts, different
topic6	ins, activity, cells, dna, binding, forms, gene, required, phosphorylation, receptor
topic7	ins, min, cells, containing, activity, described, incubated, protein, mms, buffer

3.1.8.1 Updates in MDC

Update each factor $q(\beta_{jl})$ by variational approach, $q(\beta_{jl})$ has the following form:

$$\begin{aligned}
\ln q(\beta_{jl}) &\propto \ln p(\beta_{jl}|\theta_{j10}, \theta_{j11}) + \sum_{i \in N_j} \mathbb{E}_q \ln p(l_{ij}|r_i, C_i = l, \beta_j) \\
&= \left(\theta_{j11} + \sum_{i \in N_j} q(C_i = l)q(R_i = l_{ij}) \right) \ln \beta_{jl} \\
&\quad + \left(\theta_{j10} + \sum_{i \in N_j} q(C_i = l)q(R_i \neq l_{ij}) \right) \ln (1 - \beta_{jl})
\end{aligned} \tag{3.24}$$

We can see that the above posterior of $q(\beta_{jl})$ has Beta distribution $\text{Beta}(\tilde{\theta}_{jl})$ with parameter $\tilde{\theta}_{jl} = [\tilde{\theta}_{j10}, \tilde{\theta}_{j11}]$, where $\tilde{\theta}_{j10} = \theta_{j10} + \sum_{i \in N_j} q(C_i = l)q(R_i \neq l_{ij})$ and $\tilde{\theta}_{j11} = \theta_{j11} + \sum_{i \in N_j} q(C_i = l)q(R_i = l_{ij})$.

Update each factor $q(r_i)$ the optimal approximate posterior of $q(r_i)$ takes the form:

$$\begin{aligned}
\ln q(r_i) &\propto \ln p(r_i) + \sum_{j \in M_i} \mathbb{E}_q \ln p(l_{ij}|r_i, C_i, \beta_j) \\
&= \ln p(r_i) + \sum_{j \in M_i} \sum_{l=1}^D q_{il} \left[\delta_{ij} \mathbb{E}_q \ln \beta_{jl} + (1 - \delta_{ij}) \mathbb{E}_q \ln (1 - \beta_{jl}) \right]
\end{aligned} \tag{3.25}$$

where $q_{il} = q(C_i = l)$. The expectation of logarithmic beta variables

$$\mathbb{E}_q \ln \beta_{jl} = \psi(\tilde{\theta}_{j11}) - \psi(\tilde{\theta}_{j11} + \tilde{\theta}_{j10})$$

and

$$\mathbb{E}_q \ln(1 - \beta_{jl}) = \psi(\tilde{\theta}_{j10}) - \psi(\tilde{\theta}_{j11} + \tilde{\theta}_{j10})$$

where $\psi(\cdot)$ is digamma function. Then $q(r_i)$ is normalized in order to be a valid probability.

Update each factor $q(\lambda_i)$ assume λ_i takes a Dirichlet prior with parameter $\{\alpha_l\}_{l=1}^D$. We have

$$\begin{aligned} \ln q(\lambda_i) &\propto \ln p(\lambda_i) + \mathbb{E}_q \ln p(C_i | \lambda_i) \\ &= \ln p(\lambda_i) + \sum_{l=1}^D q(C_i = l) \ln \lambda_{il} = \ln \prod_{l=1}^D \lambda_{il}^{\tilde{\alpha}_{il}-1} \end{aligned} \quad (3.26)$$

where $\tilde{\alpha}_{il} = \alpha_l + q(C_i = l)$. It is evident that the posterior $q(\lambda_i)$ also has a Dirichlet distribution with parameters $\{\tilde{\alpha}_{il}\}_{l=1}^D$.

Update each factor $q(C_i)$ We have

$$\begin{aligned} \ln q(C_i = l) &\propto \mathbb{E}_{q(\lambda_i)} \ln p(C_i = l | \lambda_i) + \mathbb{E}_q \ln p(l_{ij} | r_i, C_i = l, \beta_j) \\ &= \mathbb{E}_{q(\lambda_i)} \ln \lambda_{il} \\ &+ \sum_{j \in M_i} \left[q(r_i = l_{ij}) \mathbb{E}_q \ln \beta_{jl} + q(r_i \neq l_{ij}) \mathbb{E}_q \ln(1 - \beta_{jl}) \right] \end{aligned} \quad (3.27)$$

where

$$\mathbb{E}_{q(\lambda_i)} \ln \lambda_{il} = \psi(\tilde{\alpha}_{il}) - \psi\left(\sum_{k=1}^D \tilde{\alpha}_{ik}\right)$$

3.1.8.2 Updates in MDFC

The updates in MDC are divided into two steps: E-step and M-step. In E-step, we obtain the approximate posterior distributions for different random variables in our model given current estimates of model parameters μ_l 's, Σ_l 's, w , and δ . In M-step, the model parameters are obtained given current posterior approximations. E-step and M-step are iterated until convergence.

E-step Since the updates of posterior distributions of β_j 's and λ_i 's are the same as those in MDC, we just show the updates of $q(C_i)$'s and $q(r_i)$'s below:

For $q(C_i)$, besides the terms in equation (3.27), it has an extra term:

$$\begin{aligned} \ln p(x_i|C_i = l, \mu_l, \Sigma_l) = & -\frac{1}{2}(x_i - \mu_l)^T \Sigma_l^{-1} (x_i - \mu_l) - \\ & \frac{1}{2} \ln |\Sigma_l| (x_i - \mu_l) \end{aligned} \quad (3.28)$$

where $(*)^T$ denotes the transpose of the term inside the parenthesis. For $q(r_i)$, besides the terms in equation (3.25), it has an additional term:

$$\begin{aligned} p(r_i|x_i) = & \sigma(w^T x_i + \delta)^{\mathbb{1}\{r_i=1\}} (1 - \sigma(w^T x_i + \delta))^{\mathbb{1}\{r_i=0\}} \\ = & \sigma(w^T x_i + \delta)^{\mathbb{1}\{r_i=1\}} \sigma(-w^T x_i - \delta)^{\mathbb{1}\{r_i=0\}} \end{aligned} \quad (3.29)$$

where $\sigma(*)$ denotes the sigmoid function and $\mathbb{1}\{r_i = 1\}$ denotes the indicator function that equals to 1 if $r_i = 1$ and equals to 0 if not. The second equality in equation (3.29) holds because for the sigmoid function we have $\sigma(-z) = 1 - \sigma(z)$.

M-step In order to estimate the model parameters μ_l 's, Σ_l 's, w , and δ , we adopt alternating optimization by optimizing one set of the parameters while fixing the others. The objective function is the expectation of the logarithm of the likelihood function $Q = \mathbb{E}_q \ln p(L, R, \beta, C, \lambda | \mu, \Sigma, w, \delta)$ given current approximate posteriors q . Then we have:

$$\begin{aligned}
\mu_l^{new} &= \frac{\sum_{i=1}^N q(C_i = l) x_i}{\sum_{i=1}^N q(C_i = l)} \\
\Sigma_l^{new} &= \frac{\sum_i q(C_i = l) (x_i - \mu_l^{new}) (x_i - \mu_l^{new})^T}{\sum_{i=1}^N q(C_i = l)} \\
\frac{\partial Q}{\partial w} &= \sum_{i=1}^N [q(r_i = 1) \sigma(w^T x_i + \delta) - q(r_i = 0) \sigma(-w^T x_i - \delta)] x_i \\
\frac{\partial Q}{\partial \delta} &= \sum_{i=1}^N [q(r_i = 1) \sigma(w^T x_i + \delta) - q(r_i = 0) \sigma(-w^T x_i - \delta)]
\end{aligned} \tag{3.30}$$

To obtain the optimal values of w and δ , we derive the first order derivatives $\frac{\partial Q}{\partial w}$ and $\frac{\partial Q}{\partial \delta}$ and use the L-BFGS quasi-Newton method [42].

3.1.8.3 Updates in MDTC

The updates for the parameters of the variational posterior distribution for C_i , β_{jl} , and r_i remain the same since no additional dependencies for those variables are introduced as shown in Fig. 3.4. We derive the posteriors for λ_i , z_{iw} , and ϕ_l .

Update each factor $q(\lambda_i)$ We have

$$\begin{aligned} \ln q(\lambda_i) &\propto \exp \left\{ \ln p(\lambda_i) + \mathbb{E}_q \ln p(C_i | \lambda_i) + \sum_w \mathbb{E}_q \ln p(z_{iw} | \lambda_i) \right\} \\ &\propto \prod_{l=1}^D \lambda_{il}^{\tilde{\alpha}_{il}-1} \end{aligned} \quad (3.31)$$

where $\tilde{\alpha}_{il} = \alpha_l + q(C_i = l) + \sum_w n_{iw} \eta_{iwl}$. α_l is the parameter of the prior Dirichlet distribution of λ .

Update each factor $q(\phi_l)$ We have

$$\begin{aligned} \ln q(\phi_l) &\propto \ln p(\phi_l) + \sum_{i,k} \mathbb{E}_q \ln p(w_{ik} | \phi_l, z_{ik}) \\ &= \ln p(\phi_l) + \sum_{i,k} q(z_{ik} = l) \ln \phi_{lw_{ik}} \end{aligned} \quad (3.32)$$

It is evident that ϕ_l has a Dirichlet posterior distribution with parameter:

$$\tilde{\Upsilon}_{lw} = \Upsilon + \sum_i n_{iw} \eta_{iwl}$$

Update each factor $q(z_{iw})$ We have

$$\begin{aligned} \ln \eta_{iwl} &\propto \mathbb{E}_q \ln p(z_{iw} = l | \lambda_i) + \mathbb{E}_q \ln \phi_{lw} \\ &= \mathbb{E}_q \ln \lambda_{il} + \mathbb{E}_q \ln \phi_{lw} \end{aligned} \quad (3.33)$$

where $\mathbb{E}_q \ln \phi_{lw} = \psi(\tilde{\Upsilon}_{lw}) - \psi(\sum_{w'} \tilde{\Upsilon}_{lw'})$. Then we need to normalize $\eta_{iwl}, l = 1, \dots, D$ to form valid probabilities.

3.1.9 Summary

In this problem, we propose a probabilistic model (MDC) that captures multi-domain characteristics of crowdsourcing questions and multi-dimensional trust of workers' knowledge. To show that our model MDC is very flexible and extensible to incorporate additional metadata associated with questions, we propose an extended model MDFC that incorporates continuously-valued features of questions and MDTC that also combines topic discovery. MDTC has the advantage that the domains are interpretable. We show that our proposed models have superior performance compared to state-of-the-art on two real datasets and can effectively recover the trust vectors of workers. This can be very useful in task assignment adaptive to workers' trust values in different dimensions in the future. We assume answers from workers are collected first and are then fed to models for inference. For future work, we will investigate the problem of choosing which question to be labeled next by which worker based on the trust vectors of workers.

The results in this chapter can be applied for fusion of information from multiple unreliable data sources instead of just workers in the open crowd. Examples of data sources are sensors, human input, and inference results given by another system backed by a different set of machine learning algorithms. Each of the data sources can be treated as a "worker" in this chapter and we can thereafter use models to estimate the multi-domain trust values of the data sources and true labels of questions.

3.2 Trust-aware crowdsourcing with domain knowledge

3.2.1 Motivation

In a typical crowdsourcing setting, multiple workers are solicited to provide answers for each of the questions. For example, Facebook users volunteer to perform various annotation tasks on Facebook edit page², or workers on *Amazon Mechanical Turk*³ get paid for solving various tasks uploaded by task requesters. An example task is to determine whether a given plaintext headline expresses one or more of the emotions anger, disgust, fear, joy, sadness, and surprise. So there are six questions associated with a single headline. Our observation is that these questions are not independent. If the system is more confident that a headline exhibits anger emotion, then the headline is not likely to express joy. In addition, workers that provide answers for these questions tend to give similar answers if they share same attributes. These observations motivate us to utilize these logical constraints, which we call *domain knowledge*, to more accurately estimate true labels of questions as well as trust values of workers.

In this problem, we propose a trust-aware crowdsourcing with domain knowledge framework (TCDK) [43]. It is a two-layered probabilistic graphical model, where the top layer encodes the logical relationships using first-order logic rules and the bottom layer encodes the probabilistic dependencies between random variables in traditional crowdsourcing graphical models. We show that the dependency between

²<https://www.facebook.com/editor>

³<https://www.mturk.com/mturk/welcome>

of the top and bottom layers is equivalent to a special rule, called *cost-function rule* with a fixed weight of 1.0. This two-layered framework can be seen as a generalized probabilistic soft logic framework that contains both logical and probabilistic relations while the probabilistic soft logic in [44] only contains logical relations. TCDK allows users to integrate high level domain knowledge easily into traditional crowdsourcing graphical models without having to derive a whole new model from scratch. More importantly, the leverage of domain knowledge can help the system better estimate true labels of questions and at the same time more accurately estimate the trust values of workers. To jointly infer the true labels of questions and trust values of workers, we develop an inference algorithm based on the alternating direction method of multipliers. More specifically, the algorithm alternates between optimizing variables in the lower layer while fixing variables in the upper layer and optimizing variables in the upper layer while fixing variables in the bottom layer.

Our contributions are the following:

1. We formulate a novel trust-aware crowdsourcing with domain knowledge framework that combines domain knowledge with a traditional crowdsourcing graphical model. Users can express high level domain knowledge without having to re-define the model and the framework can be used to integrate multiple data sources.
2. We develop a scalable joint inference algorithm for estimating true label variables and trust values of workers based on alternating consensus optimization. The inference algorithm can be easily scaled to multiple machines.

3.2.2 Related work

To address the issue of noisy and malicious workers in crowdsourcing systems, many models are developed to jointly estimate true labels of questions and trust of workers [33–35,45]. All these works are based on the assumption that questions’ true label variables are independent and the trusts of different workers are independent too. However, the assumption is shown to be invalid in the annotation of headline emotion example in Section 3.2.1.

[46] did consider dependency between workers by revealing the latent group structure among dependent workers and aggregated information at the group level rather than from individual workers. Still, their model did not capture the logical dependencies among questions as in our work. In the natural language processing literature, a framework called Fold·All [47] was proposed to integrate domain knowledge into Latent Dirichlet Allocation (LDA). Their framework can be seen as an extension of the Markov Logic Network while the top layer of our framework can be seen as the generalized probabilistic soft logic [44].

3.2.3 Graphical model framework for trust-aware crowdsourcing with domain knowledge

We consider a crowdsourcing task with N questions and M workers available in total. Each question is answered by a subset of M workers. Each worker j is modeled by a random variable $\beta_j \in [0, 1]$ that has a Dirichlet prior with parameter

θ . Higher value of β_j indicates that the worker is more trustworthy. The variable $z_i \in \{0, 1\}$ is used to denote question i 's true label. The answer to question i given by worker j is denoted by $l_{ij} \in \{0, 1\}$.

We first review the graphical model used in [35]:

$$p(L, z, \beta|\theta) \propto \prod_{i=1}^N \prod_{j \in M_i} p(\beta_j|\theta) p(l_{ij}|z_i, \beta_j) \quad (3.34)$$

where M_i is the set of workers that give answers to question i . The task is to infer questions' true labels z_i 's and estimate workers' trust values β_j 's.

In the above model, the true labels z_i 's are assumed to be independent. In TCDK, we incorporate the logical relations between questions using first-order logic rule syntax. Example rules are:

$$\begin{aligned} \text{ContainsHappiness}(i) &\Rightarrow \text{ContainsAnger}(i) \\ \text{Trust}(j_1) \wedge \text{SimilarBackground}(j_1, j_2) &\Rightarrow \text{Trust}(j_2) \end{aligned} \quad (3.35)$$

The first rule states that if text clip i expresses emotion happiness, then it is unlikely that the text expresses anger and the second rule states that if the worker j_1 is trustworthy and he has similar background with another worker j_2 , the worker j_2 tends to be trustworthy too. For each first-order logic rule ℓ as defined in equation (3.35), we represent the weight of the rule as λ_ℓ and the set of groundings of rule r as R_ℓ . Higher value of λ_ℓ indicates that the rule ℓ is more important compared to other rules. For each grounded rule r , we associate a non-negative potential function $\phi_r(z, \beta)$. We will discuss the specific definition of $\phi_r(z, \beta)$ later.

Putting together the domain knowledge expressed using first-order logic rules as in equation (3.35) and the traditional crowdsourcing model in equation (3.34), our proposed model Crowdsourcing with Domain Knowledge (TCDK) defines a generative model expressed as follows:

$$\begin{aligned}
 p(L, z, \beta|\theta) \propto \exp \left[- \sum_{r=1}^R \lambda_r \phi_r(z, \beta) \right] \\
 \times \prod_{i=1}^N \prod_{j \in M_i} p(\beta_j|\theta) p(l_{ij}|z_i, \beta_j)
 \end{aligned} \tag{3.36}$$

where R is the number of grounded first-order logic rules. The graphical model in equation (3.36) consists of two terms with the first term encoding the logical relations among variables z, β and the second encoding probabilistic dependencies among observed answers L and hidden variables z and β . The logical dependency encoded in $\phi_r(z, \beta)$ is very general and is determined by the specific grounded rule r . For example, it can be defined over true label variables z_i 's or over trust variables β_j 's or over a mixture of both as in equation (3.35). Fig. 3.7 shows an example causal structure of TCDK when ϕ_r 's are defined over z only. In Fig. 3.7, the statistical layer corresponds to the first term in equation (3.36) and the logical layer corresponds to the second term. The red dotted lines represent logical dependencies among z indicated by $\phi_r(z)$.

Note that β_j 's are continuously-valued variables and z_i 's are discrete variables. If $\phi_r(z, \beta)$'s depend on β_j 's only, the first part in equation (3.36) is equivalent to a continuous Markov random field [44]. If $\phi_r(z, \beta)$'s also depend on z_i 's, the first part combined with the second part in equation (3.36) can be viewed as a Hybrid

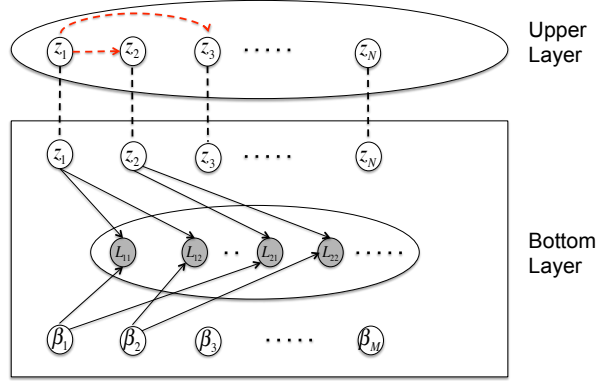


Figure 3.7: Graphical Model of Trust-aware Crowdsourcing with Domain Knowledge (TCDK). z_i 's are true label variables, β_j 's are workers' trust variables, and l_{ij} 's are worker-provided answers. The black-dotted lines in the bottom layer encode probabilistic dependencies between variables and the red-dotted lines in the upper layer encode logical dependencies.

Markov Logic Network (HMLN) [48]. However, HMLN relies solely on first-order logic to express causal structure among variables, therefore it falls short of expressing general dependencies as in the second term in equation (3.36).

3.2.4 Scalable inference algorithm based on ADMM

We are interested in the maximum a posteriori probability (MAP) estimates of true label variables z_i 's and trust variables β_j 's given answers L in TCDK. The MAP estimates are the solution to the following optimization problem:

$$\arg \min_{z, \beta} \sum_{r=1}^R \lambda_r \phi_r(z, \beta) - \sum_{i=1}^N \sum_{j \in M_i} \log p(l_{ij} | z_i, \beta_j) - \sum_{j=1}^M \log p(\beta_j | \theta) \quad (3.37)$$

It is challenging to solve the above optimization problem due to the large data size and possibly exponential groundings of first-order logic rules. One can relax the

discrete variables to take continuous values and resort to Alternating Optimization with Mirror Descent to avoid fully grounding first-order logic rules [47]. However, their algorithm still can not scale because a single machine is processing all the sampled groundings and the algorithm is not easily scaled to multiple machines. [49] proposed a scalable solution to constrained continuous Markov random fields based on the consensus optimization framework. However, it can not be directly applied to our problem because their optimization objective is based on hinge loss only.

In what follows, we propose a scalable inference algorithm based on the alternating direction method of multipliers (ADMM). First, we relax true label variables z_i 's to the interval $[0, 1]$ so that the potential functions $\phi_r(z, \beta)$'s are defined over continuous variables taking values from interval $[0, 1]$. The algorithm can be seen as *generalized probabilistic soft logic* (GPSL) because it contains special cost-function rules besides first-order logic rules. We briefly review the basics of probabilistic soft logic (PSL) below.

3.2.4.1 Definitions in probabilistic soft logic

Probabilistic soft logic declares first-order logic rules:

$$\lambda : A(i, j) \wedge B(j, k) \Rightarrow C(i, k) \tag{3.38}$$

where A , B and C are predicates and i, j, k are variables. Each ground predicate is an instantiation of predicates with instantiated values for i, j, k and takes a soft-truth value from $[0, 1]$. The logical connectives (AND, OR, NOT) are relaxed using

Lukasiewicz *t*-norm and its corresponding *co*-norm:

$$\begin{aligned}
 p \wedge q &= \max(0, p + q - 1), \\
 p \vee q &= \min(1, p + q), \\
 \neg p &= 1 - p
 \end{aligned}
 \tag{3.39}$$

Any grounded first-order logic rule has the form $r_{body} \rightarrow r_{head}$. An interpretation I is defined as an assignment of soft truth values to a set of ground predicates. PSL calculates a potential function for any grounded rule r under interpretation I through the following:

$$\phi_r(I) = \max\{0, I(r_{body}) - I(r_{head})\}
 \tag{3.40}$$

3.2.4.2 Scalable ADMM-based inference

ADMM is utilized to optimize objectives by iteratively solving local subproblems and finding consensus to the global objective [50]. We observe that z_i 's and β_j 's are coupled through the term $\log p(l_{ij}|z_i, \beta_j)$. Therefore we can iteratively optimize equation (3.37) while fixing z_i 's and vice versa. When β_j 's are fixed, equation (3.37) becomes:

$$\arg \min_z \sum_{r=1}^R \lambda_r \phi_r(z, \beta) - \sum_{i=1}^N \sum_{j \in M_i} \log p(l_{ij}|z_i, \beta_j)
 \tag{3.41}$$

The first term in equation (3.41) corresponds to weighted summation of potential functions of grounded first-order logic rules while the second term is the summation of logarithms of conditional probabilities. We show next that we can put the two

parts into a unified framework GPSL.

$$\begin{aligned}
\varphi(z_i, \beta_j) &= -\log p(l_{ij}|z_i, \beta_j) \\
&= -\mathbb{1}\{l_{ij} = z_i\} \log \beta_j - \mathbb{1}\{l_{ij} \neq z_i\} \log(1 - \beta_j) \\
&= -(z_i l_{ij} + (1 - z_i)(1 - l_{ij})) \log \beta_j \\
&\quad - (1 - z_i l_{ij} - (1 - z_i)(1 - l_{ij})) \log(1 - \beta_j)
\end{aligned} \tag{3.42}$$

Substituting equation (3.42) into equation (3.41), we have:

$$\arg \min_z \sum_{r=1}^R \lambda_r \phi_r(z, \beta) + \sum_{i=1}^N \sum_{j \in M_i} \varphi(z_i, \beta_j) \tag{3.43}$$

The second term in equation (3.43) is equivalent to the summation of potential functions introduced by N grounded special cost-function rules with weight 1.0. They encode the dependency between upper and bottom layer shown in Fig. 3.7.

Next we present how to optimize β_j 's while fixing z_i 's. equation (3.37) becomes:

$$\arg \min_{\beta} \sum_{r=1}^R \lambda_r \phi_r(z, \beta) + \sum_{j=1}^M \left(\sum_{i \in N_j} \varphi(z_i, \beta_j) - \log p(\beta_j | \theta) \right) \tag{3.44}$$

where the first term corresponds to the summation of potential functions for grounded first-order logic rules that involve β while the second term can be viewed as the summation of potential functions introduced by M grounded special cost-function rules with weight 1.0.

Let $z_r, r = 1, \dots, R$ be a local copy of the variables in Z that are used in potential function $\phi_r(z, \beta)$ and z_{i+R} be a local copy of the variables in Z that are

used in potential function $\sum_{j \in M_i} \varphi(z_i, \beta_j)$. Let $Z_i, i = 1, \dots, R + N$ be the global version of the local copies in z_i . Similarly, we define $\mathbf{b}_r, r = 1, \dots, R$ as a local copy of the global variables in \mathbf{B} that are used in $\phi_r(z, \beta)$ and $\mathbf{b}_{j+R}, j = 1, \dots, M$ as a local copy of the variables in \mathbf{B} used in $\sum_{i \in N_j} \varphi(z_i, \beta_j) - \log p(\beta_j | \theta), j = 1, \dots, M$. The ADMM-based inference algorithm is shown in Algorithm (6). It is scalable in nature because each grounded rule is a subproblem and can be run in parallel over multiple machines.

3.2.5 Case studies and experiments on real datasets

In order to evaluate the performance of our proposed TCDK framework, we performed experiments on two real datasets. In what follows, we describe each of the datasets, define first-order logic rules and special cost-function rules, and present experimental results. For each of the two datasets, we consider the following models for comparison:

1. TCDK: our proposed trust-aware crowdsourcing with domain knowledge.
2. TC (trust-aware crowdsourcing without domain knowledge): same as TCDK except that we omit domain knowledge by setting zero weights to first-order logic rules and special cost-function rules defined for each dataset.
3. MV (majority vote): a true value variable is estimated to be 1 if more than half workers answer 1 and is estimated to be 0 if less than half workers answer 0. Ties are broken randomly.

Algorithm 6: Consensus optimization for z and β

Input: $\phi, \lambda, L, \mathbf{Z}, \mathbf{B}, \theta, \varphi, \rho > 0$

Output: MAP estimates for z_i 's and β_j 's

while not converged do

 /* Optimize z_i 's while fixing β_j 's */

 Initialize \mathbf{z}_i as a copy of the variables in \mathbf{Z} that appear in $\phi_r, r = 1, \dots, R$

 Initialize \mathbf{z}_{i+R} as a copy of the variables in \mathbf{Z} that appear in

$\sum_{j \in M_i} \varphi(z_i, \beta_j), i = 1, \dots, N$

 Initialize dual variable $\mathbf{y}_k = 0, k = 1, \dots, |R| + N$

while not converged do

for $k = 1, 2, \dots, R, R + 1, \dots, R + N$ **do**

$\mathbf{y}_k = \mathbf{y}_k + \rho(\mathbf{z}_k - \mathbf{Z}_k)$

end

for $r = 1, 2, \dots, R$ **do**

$\mathbf{z}_r \leftarrow \arg \min_{\mathbf{z}_r \in [0,1]^{n_r}} \lambda_r \phi_r(z, \beta) + \frac{\rho}{2} \left\| \mathbf{z}_r - \mathbf{Z}_r + \frac{1}{\rho} \mathbf{y}_r \right\|_2^2$

end

for $i = 1, 2, \dots, N$ **do**

$\mathbf{z}_{i+R} \leftarrow \arg \min_{\mathbf{z}_{i+R} \in [0,1]^{n_{i+R}}} \sum_{j \in M_i} \varphi(z_i, \beta_j) + \frac{\rho}{2} \left\| \mathbf{z}_{i+R} - \mathbf{Z}_{i+R} + \frac{1}{\rho} \mathbf{y}_{i+R} \right\|_2^2$

end

 Set each entry z_i in \mathbf{Z} to the average of the all the local copies.

end

 /* Optimize β_j 's while fixing z_i 's */

 Initialize \mathbf{b}_r as a copy of the variables in \mathbf{B} that appear in $\phi_r, r = 1, \dots, R$

 Initialize \mathbf{b}_{j+R} as a copy of the variables in \mathbf{B} that appear in

$\sum_{i \in N_j} \varphi(z_i, \beta_j) - \log p(\beta_j | \theta), j = 1, \dots, M$

 Initialize dual variables $\mathbf{v}_k = 0, k = 1, \dots, M, M + 1, R + M$

while not converged do

for $k = 1, \dots, R, R + 1, R + M$ **do**

$\mathbf{v}_k = \mathbf{v}_k + \rho(\mathbf{b}_k - \mathbf{B}_k)$

end

for $r = 1, 2, \dots, R$ **do**

$\mathbf{b}_r \leftarrow \arg \min_{\mathbf{b}_r \in [0,1]^{n_r}} \lambda_r \phi_r(z, \beta) + \frac{\rho}{2} \left\| \mathbf{b}_r - \mathbf{B}_r + \frac{1}{\rho} \mathbf{v}_r \right\|_2^2$

end

for $j = 1, 2, \dots, M$ **do**

$\mathbf{b}_{j+R} \leftarrow \arg \min_{\mathbf{b}_{j+R} \in [0,1]^{n_{j+R}}} \sum_{i \in N_j} \varphi(z_i, \beta_j) - \log p(\beta_j | \theta) +$

$\frac{\rho}{2} \left\| \mathbf{b}_{j+R} - \mathbf{B}_{j+R} + \frac{1}{\rho} \mathbf{v}_{j+R} \right\|_2^2$

end

 Set each entry b_j in \mathbf{B} to the average of the all the local copies.

end

end

3.2.5.1 Affective Text Evaluation

This dataset was produced by crowdsourcing task [51] where each worker was given a headline and asked to give a rating (ranging from 0 to 100) about the degree of emotions that the headline expresses. Six emotions were considered: anger, disgust, fear, joy, sadness and surprise. We use the dataset provided by [32], where 100 pieces of headlines were selected and 10 answers were solicited for each of the six emotions from workers on Amazon Mechanical Turk. Note that each headline-emotion pair might be answered by a different group of workers.

We represent a headline Q expressing emotion X as predicate $tl(Q, X)$, where $Q = 1, \dots, N$ and $X \in \{Anger, Disgust, Fear, Joy, Sadness, Surprise\}$. The grounded predicate $tl(Q, X)$ takes value from $[0, 1]$. Our domain knowledge tells us that among those six emotions, there exists two types of relations between emotions X and Y , one is similar relation which we define as predicate $simRel(X, Y)$ and the other is opposite relation which we define as predicate $oppRel(X, Y)$. Y takes values from the six emotions as X does. We define the following first-order logic rules to represent our domain knowledge:

$$\begin{aligned}
 tl(Q, X) \wedge oppRel(X, Y) &\rightarrow \neg tl(Q, Y), & w : 5.0 \\
 tl(Q, X) \wedge simRel(X, Y) &\rightarrow tl(Q, Y), & w : 1.0
 \end{aligned}
 \tag{3.45}$$

The first rule states that if a headline expresses emotion X and the two emotions X and Y are opposite, it is unlikely that the headline expresses emotion Y whereas the second rule states that if X and Y are similar, there is a chance that a headline

Table 3.5: Emotions relations

Relations	Emotion pairs
Opposite	(Anger, Joy), (Anger, Fear), (Anger, Sadness), (Anger, Surprise) (Disgust, Joy), (Disgust, Sadness), (Fear, Joy), (Sadness, Joy), (Surprise, Joy), (Surprise, Sadness)
Similar	(Fear, Sadness)

expresses emotion Y if it expresses emotion X . The weights for the two first-order logic rules are assumed to be known and set to 5.0 and 1.0 respectively. Higher weight of the first rule indicates it is a more important rule than the second. The values of grounded predicates $oppRel(X, Y)$ and $simRel(X, Y)$ are assumed to be part of our domain knowledge. The details of these two grounded predicates are shown in Table 3.5. For example, we believe that a headline can not express *Anger* and *Joy* at the same time. In addition to the first-order logic rules, we define the cost-function rules:

$$LinearLoss(\beta, tl(Q, X)), w : 1.0 \quad (3.46)$$

The rule corresponds to the second term in equation (3.43). The predicate is called *LinearLoss* because the potential function associated with this rule is linear in $tl(Q, X)$ as can be observed from equation (3.42) and equation (3.43).

We conducted coarse-grained experiments on Affective Text dataset, i.e. each rating is mapped to 0 if the original value is smaller than 50 and 1 if larger or equal to 50. We calculate the precision, recall, F-measure and accuracy of all emotions for the TCDK model. The results are reported in Table 3.6. The highest scores in all the

Table 3.6: Performance of algorithms on affective text

Model	precision	recall	F1	accuracy
TCDK	31.91	75.00	44.48	93.83%
TC	34.04	51.61	41.03	92.33%
MV	34.04	47.06	39.51	91.83%

measures are in bold format. We observe that our model TCDK obtained best results with respect to recall, F1 score, and accuracy. This demonstrates the advantage of taking into consideration domain knowledge compared to TC that ignores it.

3.2.5.2 Fashion Social Dataset Evaluation

The dataset [52] contains 4711 images crawled from Flickr and along with each image, metadata are available such as the fashion topic used to query the image, title of the image, tags and comments made by Flickr users, etc. In this annotation task, a worker is presented with two questions for each image: Is the image fashion related? Is the image showing a specialty clothing item? Therefore we have in total 9422 questions. For each image, a number of workers from Amazon Mechanical Turk (AMT) provide their answers. Each answer takes values from $\{Yes, No, NotSure\}$. If a worker answers *NotSure*, we treat it as if the worker does not provide an answer for this question. We filter out questions that receive less than three answers and we are left with $N = 8538$ questions, each of which receives equal to or more than three answers from workers. We have in total $M = 201$ workers available for this annotation task. To generate ground truth, three trusted experts were recruited to give high-quality annotations. We take the majority vote from the three trusted experts as the ground truth and use it for evaluation of our models.

The question "Is the image related to fashion?" for image Q is denoted by predicate $fashion(Q)$, where $Q \in \{1, \dots, N\}$ and the question "Is the image related to cloth?" for image Q by predicate $cloth(Q)$. One piece of the domain knowledge we have is that if an image is related to cloth, the image is more likely to be related to fashion. This is illustrated in Fig. 3.8. Knowing that the probability of the image being cloth-related is conducive to estimating whether the image is fashion-related. This observation is captured in the following rule:

$$cloth(Q) \rightarrow fashion(Q), w : 5.0 \quad (3.47)$$

Another observation, as shown in Fig. 3.9, is that if two questions are similar in terms of the metadata, then the true labels of the two questions are likely to be the same. The following rules capture the observation:

$$\begin{aligned} sim(Q1, Q2) \wedge fashion(Q1) &\rightarrow fashion(Q2), \quad w : 1.0 \\ sim(Q1, Q2) \wedge cloth(Q1) &\rightarrow cloth(Q2), \quad w : 1.0 \end{aligned} \quad (3.48)$$

where Q denotes an image and the predicate $sim(Q1, Q2)$ represents a question-question similarity metric. Each specific similarity metric creates an instance of the two rules in equation (3.48).

We propose to use a context-based similarity metric. An image context refers to a group photo pool or a photoset. One of the example contexts is Artistic Photography. An image can be associated with one or more contexts. The intuition is that if two pictures are more likely to be in the same context, then they tend to

Table 3.7: Performance of algorithms on Fashion Dataset

Model	precision	recall	F1	accuracy
TCDK	87.83	89.84	88.82	89.27%
TC	86.79	83.6	85.20	85.39%
MV	86.55	83.73	85.11	85.32%

have the same label as well. We denote C_1 as the context set for Q_1 and C_2 as the context set for Q_2 . The context-based similarity score $sim(Q_1, Q_2)$ is defined as:

$$sim(Q_1, Q_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \quad (3.49)$$

To avoid quadratic groundings of $sim(Q_1, Q_2)$, for each question Q_1 we only keep pairs (Q_1, Q_2) 's whose similarity scores in equation (3.49) rank at the top 10 as in [53]. Similar to the model for the Affective Text dataset, the special cost-function rules that bridge the gap between the top and the bottom layers are:

$$\begin{aligned} &LinearLoss(\beta, fashion(Q)), \quad w : 1.0 \\ &LinearLoss(\beta, cloth(Q)), \quad w : 1.0 \end{aligned} \quad (3.50)$$

We perform ten-fold cross validation with each fold leaving out 10% of data. We estimate values of $fashion(Q)$ and $cloth(Q)$ on the held-out fold and then map them to 0 or 1 using threshold 0.5. The results are shown in Table 3.7. Again, results show that TCDK achieves better performance in all criteria with integrated domain knowledge than TC alone (without the leverage of domain knowledge).

The weights of first-order logic rules defined for both datasets in Section 3.2.5.1 and Section 3.2.5.2 are assumed to be known and given as part of domain knowledge.

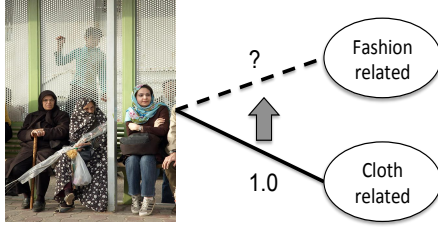


Figure 3.8: Estimated true labels for "cloth related" questions can be used for prediction of "fashion related" questions.

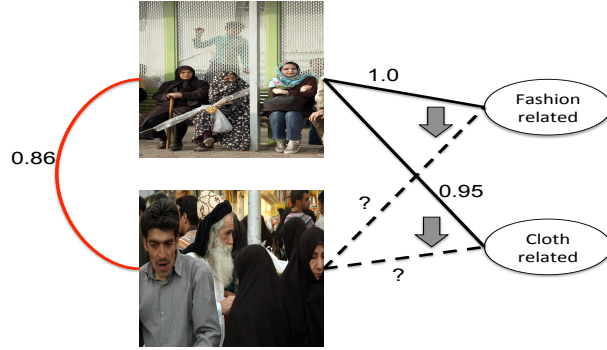


Figure 3.9: Estimated true labels for questions can be used for prediction of other questions using image similarity.

Though weights can be auto-tuned using maximum-likelihood estimation [44], we leave this problem for future work and aim to demonstrate the power of our model with fixed yet not fine-tuned values set by users of our model.

3.2.6 Summary

We presented trust-aware crowdsourcing with domain knowledge (TCDK), a unifying framework that combines the power of domain knowledge and traditional crowdsourcing graphical model. It allows users to express domain knowledge using first-order logic rules without redefining the model. To estimate questions' true labels and workers' trust values, we develop a scalable inference algorithm based on alternating consensus optimization. We demonstrate that our model is superior to

the state-of-the-art by testing it on two real datasets.

Chapter 4: Trust-Aware Optimal Crowdsourcing With Budget Constraint

4.1 Motivation

Crowdsourcing provides a convenient and efficient way for data collection without having to acquire costly labels from domain experts. In a typical crowdsourcing task, a requester distributes small jobs to non-expert workers and provides a small amount of payment upon job completion. Such a small job can be translating a sentence [54], annotating an image [55], classifying search queries [56], etc. Answers (or labels) obtained from workers are usually noisy due to workers' lack of expertise, carelessness, or malicious labeling. To mitigate the noise, one question is redundantly distributed to multiple workers and the answers are aggregated to produce a single answer, expected to be more accurate. Many crowdsourcing platforms are available, for example, Amazon Mechanical Turk, ESP game and reCaptcha.

One typical goal in crowdsourcing tasks is to infer the ground truth from collected answers. Much work [32, 36, 37] in crowdsourcing has been devoted to making aggregated decisions to predict true labels given noisy and even malicious input from workers. However, these algorithms do not consider the cost incurred

from obtaining a label from a worker; while in practice, the number of answers we can get is restricted by the budget coming with requesters. Under this constraint, a natural question to ask is how to allocate tasks to workers adaptively with limited budget.

Past approaches to crowdsourcing with budget constraint have assumed that all questions and workers are homogeneous – questions do not differ in difficulty level, and all workers are as capable as each other and get the same payment for answering any question. This can be an over-simplified setting for real problems. In practice, the cost depends on both the question and the worker. For example, fine category classification of different kinds of birds requires more domain knowledge than simply telling if there is a bird in an image; summarizing a paragraph needs more work than deciding if a tweet is positive or negative. Requesters generally pay more to workers for difficult tasks. On the other hand, skillful workers ask for higher payment than ordinary workers, and have a larger chance of providing the ground truth. For example, for the same task, consulting a domain expert is more costly than asking a random worker on Mechanical Turk; however, on average more Turks are required to infer the correct answer. Thus there is a trade-off between cost and answer quality. A more cost-efficient way to task distribution than blind random assignment would be to assign easy tasks to cheap workers and hard tasks to workers with more expertise. The answers given by workers are then combined with estimated trustworthiness of workers. We consider the trustworthiness of a worker as equivalent to the worker’s reliability. Therefore use trust and reliability interchangeably in this chapter. Specifically, expert level crowd has higher trust value

while common non-expert crowd has lower trust value.

In this chapter, we address the problem of trust-aware task allocation by considering cost and expertise variation among workers. We propose an easy-to-implement allocation algorithm in the setting of weighted majority vote with theoretical guarantee. We formulate the assignment problem as a nonlinear integer programming problem with budget constraint, and relax it to a convex optimization problem that has an analytical solution. We also give a theoretical error bound for the performance of our algorithm [57]

Our contributions are as follows:

- We formulate the problem of trust-aware task allocation in crowdsourcing and provide a principled way to solve it.
- Our formulation models the workers' trustworthiness and the costs depend on both the question and the worker group. Our method is ready to be extended to more complicated aggregation method other than the weighted majority vote as well.
- The trust-aware task allocation scheme we propose can achieve total error probabilities bounded by $\frac{N}{2} - \mathcal{O}(\sqrt{B})$, where N is the number of tasks and B is the total budget. Different from [4], the exact performance bound of error probability also incorporates both trustworthiness of crowds and cost. More trustworthy crowds and less costly jobs result in lower guaranteed bound.

4.2 Related work

Most previous works focus on aggregating labels from multiple workers. None of them address a practical issue: the job requester has a budget constraint and he wants to make the best use of the requester’s budget. A closely related work along this line is Crowdscreen [58] that developed algorithms for minimizing expected cost regarding number of questions asked and the estimation accuracy. However, the cost of assigning different questions is assumed to be uniform and the heuristics-based algorithms have no theoretical guarantee of performance. This guarantee is given in [59] where all questions are homogeneous and the upper bound they derived is valid only when the number of questions assigned approaches infinity, rendering it impractical. Works that further investigate the problem of task assignment for heterogeneous tasks include [4,60]. The former is focused on minimizing cost subject to a quality constraint when workers arrive online while the latter is in the direction of minimizing estimation error under budget constraint and the cost associated with questions varies w.r.t difficulty. In particular, in [4], cost is determined by only the difficulty of questions and they can not choose explicitly which experts to choose for the completion of the task.

4.3 Problem setting

We consider classification tasks where the wisdom of crowds is utilized to estimate the ground truth of each instance. We assume that there are N tasks and

the difficulty of task i can be mapped to a real number d_i . We consider binary classification and denote the unknown true label of task i by $r_i \in \{-1, 1\}$. However, our algorithm can be applied to general classification tasks as well. We further assume that there are M crowds available for the job requesters. As can be expected, in real life, some crowds behave professionally and provide reliable answers, while other crowds are not as trustworthy, either because they have lower expertise level or because they want to get the payment without investing enough effort. We denote the answer given by a worker k from crowd j for task i as $\ell_{jki} \in \{-1, 1\}$. The job requester comes to the crowdsourcing market with a fixed budget B and he/she expects to get the highest performance out of the given budget. The crowdsourcing platform has a scheduler that distributes tasks to its pool of workers. Each assignment of task i to a worker from crowd j is associated with a cost c_{ij} .

We adopt a 1-coin model to describe the worker’s stochastic behavior when answering a specific question. The 1-coin model assumes the probability of labeling a question with 1 given $r_i = 0$ equals the probability of labeling it with 0 given $r_i = 1$. We denote the probability of getting a correct answer of task i given by worker k from crowd j by u_{ijk} . A higher value of u_{ijk} indicates higher trust value. Extension of our work to a 2-coin model (a worker is modeled by two parameters when the truth label is binary, i.e. the probability of giving correct label when truth label is 0 and the probability of giving correct label when truth label is 1) is straightforward. Given the symmetry present in the definition of the 1-coin model, without loss of generality, we assume that the true label r_i of task i is 1. Thus the answer given by worker k from crowd j follows the Bernoulli distribution: $\ell_{ijk} \sim \text{Bin}(1, u_{ijk})$. For

each task i , a user from crowd j is sampled according to some unknown distribution and we denote the expected trust value of crowd j toward task i $\mathbb{E}[u_{ij_k}]$ as u_{ij} . Note that the random variables u_{ij} and u_{ij_k} are unknown.

We assume that there is a separate trust evaluation component that assesses each worker’s trustworthiness and outputs estimates of a crowd’s trust value, denoted by $w_j \in [0, 1]$, which represents the trust evaluation component’s belief about the probability that workers from crowd j ’s answer a question correctly. We choose to estimate the trust value of a whole crowd instead of individual workers. In reality, companies like CrowdFlower¹ provides hierarchies of workers ranging from domain experts to average open crowd, thus it is more reasonable to keep track of the performance of each crowd than that of individuals.

A common approach to ground truth inference in crowdsourcing is weighted majority vote:

$$\hat{r}_i = \text{sign} \left(\sum_{j=1}^M \sum_{k=1}^{n_{ij}} w_j \ell_{jki} \right) \quad (4.1)$$

where w_j is the estimated trust value of crowd j , ℓ_{jki} is the answer to question i provided by worker j who belongs to crowd j , and n_{ij} is the number of workers from crowd j allocated for question i . The above estimation is a very basic algorithm in crowdsourcing and is usually used as the baseline or a preprocessing step for more sophisticated methods. Therefore, we use the error probability based on the weighted majority vote as an upper bound of the error probability we can achieve. Given the fixed budget provided by the job requester, the scheduler has two options.

¹<http://www.crowdfLOWER.com/>

It either assigns a set of budget constraints B_i for each task i since we don't want to allocate all the budget to a single question or the scheduler just has a budget constraint on the total expense for completing all the tasks. For each task i , multiple workers are assigned to provide answers for it. The number of workers from crowd j assigned to task i is denoted by n_{ij} and the set of workers assigned to task i can be compactly expressed as $n_i = \{n_{ij}\}_{j=1}^M$. In the setting of fixed total budget across all tasks, the optimal crowdsourcing problem becomes:

$$\begin{aligned}
& \underset{n_{ij}}{\text{minimize}} && \sum_{i=1}^N \Pr(\hat{r}_i(\{n_{ij}\}_{j=1}^M, w) \neq r_i) \\
& \text{subject to} && \sum_{i,j} c_{ij} n_{ij} \leq B \\
& && n_{ij} \in \mathbb{N}
\end{aligned} \tag{4.2}$$

which is generally a non-deterministic nonlinear integer programming problem. When we substitute question i 's true label r_i with the estimated label \hat{r}_i using the weighted majority vote equation (4.1), equation (4.2) is relaxed.

There is a trust evaluation component that gives estimation of crowds' trustworthiness w_j . Note that sometimes we might need trustworthiness of a crowd with respect to different types of questions, which is questions of varying difficulty in our case. For simplicity, in Algorithm (7) and Algorithm (8) that follow in Section 4.4, just a scalar parameter w_j is assumed for each crowd. Extension to trustworthiness with respect to each type of questions is straightforward. The design of the trust evaluation algorithm is beyond the scope of this dissertation. Interested readers are referred to [24, 61] for basics on trust models. We assume we can get access to the

estimation of trustworthiness given by this component and our allocation scheme goes from there. Our allocation scheme works with a general trust estimation component. Note that trust estimation is usually not given and incurs further cost. However, practical crowdsourcing platforms use a pipeline model, where separate components are dedicated to trust estimation, task allocation and answer inference. We intend to keep our job (task allocation) as independent from others as possible, yet flexible enough to join with any algorithm of other components. The output of our allocation scheme is a set of assignments n_{ij} . Note that we are considering task assignment before tasks are deployed in the crowdsourcing market, i.e., trust values are static in this case. This is justified by the observation that most crowdsourcing marketplaces like Amazon Mechanical Turk require preset numbers of workers to questions before deployment. That said, given time-varying trust estimates, our method can be easily made online – do partial assignment, wait for answers, update trust estimates and do another batch of assignment.

4.4 Trust-aware task allocation

Our proposed budget allocation strategy is trust-aware in the sense that it utilizes the estimated trustworthiness of crowds given by trust evaluation component and allocation decision is partially influenced by the estimation. The process works as follows. We present the optimal budget allocation scheme with total budget constraint. The job allocator selectively assigns multiple workers from each crowd j to each task i given the estimated trustworthiness w_j and cost c_{ij} .

4.4.1 Assumptions

For question i , we assume that the user k from crowd j samples his/her answer $\ell_{jk i}$ from a Bernoulli distribution, i.e., $\ell_{jk i} \sim \text{Bin}(1, u_{ijk})$. The expected answer $\mathbb{E}_{\text{Bin}(1, u_{ijk})}[\ell_{jk i}]$ is μ_{ijk} . We assume that a user k is picked from a crowd j uniformly and $\mathbb{E}_{k \sim U_j}[\mu_{ijk}] = \mu_{ij}$, where μ_{ij} denotes expected trust value of crowd j . For an allocation $\{n_{ij}\}, i = 1, \dots, N, j = 1, \dots, M$, we define the expected answer for question i averaged over workers from crowd j as

$$\mu_i = \frac{\sum_{j=1}^M n_{ij} w_j \mu_{ij}}{\sum_{j=1}^M n_{ij}} = \sum_{j=1}^M \rho_{ij} w_j \mu_{ij} \quad (4.3)$$

where $\rho_{ij} = \frac{n_{ij}}{\sum_{j=1}^M n_{ij}}$ and is fully determined by the allocation $\{n_{ij}\}$. We assume that the weighted majority voting aggregating scheme yields a somewhat reasonable performance for the given task i under uniform allocation, i.e. $\rho_{ij} = \rho_i, \forall j$:

$$\begin{cases} \mu_i \geq 0 & \text{if } r_i = 1 \\ \mu_i < 0 & \text{if } r_i = -1 \end{cases} \quad (4.4)$$

This means that if our assignment for question i is at least as good as uniformly random assignment, the expected answer for question i in equation (4.3) has the same sign as the ground truth.

4.4.2 Optimization Problem

Let $Y_i = \sum_{j=1}^M \sum_{k=1}^{n_{ij}} w_j \ell_{jki}$. The error probability of task i in equation (4.2)

can be relaxed by using the Hoeffding concentration bound:

$$\Pr(\hat{r}_i \neq r_i) \leq \exp\left(-\frac{\left(\sum_{j=1}^M n_{ij} w_j (2u_{ij} - 1)\right)^2}{2 \sum_{j=1}^M n_{ij} w_j^2}\right) \quad (4.5)$$

where u_{ij} denotes the expected trust value of crowd j and w_j denotes the estimated trust value for crowd j . equation (4.5) makes the problem in equation (4.2) a deterministic optimization problem. However, this is not convex in general.

Next we discuss how to relax the deterministic objective function on the right hand side of equation (4.5) by probably approximately correct learning framework (PAC) [62]. We consider the situation where the actual obtained answer deviates from the expected answer by ϵ_i . Using the Hoeffding Inequality, we can get

$$\Pr\left(\left|\frac{1}{\sum_{j=1}^M n_{ij}} \sum_{j=1}^M \sum_{k=1}^{n_{ij}} w_j \ell_{jki} - \mu_i\right| \geq \epsilon_i\right) \leq 2 \exp\left\{-\frac{\epsilon_i^2 (\sum_{j=1}^M n_{ij})^2}{2 \sum_{j=1}^M n_{ij} w_j^2}\right\}$$

Now let $2 \exp\left\{-\frac{\epsilon_i^2 (\sum_{j=1}^M n_{ij})^2}{2 \sum_{j=1}^M n_{ij} w_j^2}\right\} = \beta$, where β is a chosen real number from 0 to

1. This means that with probability at least $(1 - \beta)^N$, the following holds: $\hat{r}_i \in [\mu_i - \epsilon_i, \mu_i + \epsilon_i] \forall i$. We express ϵ_i as:

$$\epsilon_i = \sqrt{\frac{-2 \ln \frac{\beta}{2} \sum_{j=1}^M n_{ij} w_j^2}{\left(\sum_{j=1}^M n_{ij}\right)^2}} \quad (4.6)$$

In practice, the value of β depends on the required confidence level. Usually β is small, thus the above interval for \hat{r}_i is of high probability. In the following argument we will only consider the case where \hat{r}_i lies in the interval $[\mu_i - \epsilon_i, \mu_i + \epsilon_i]$. If $\mu_i - \epsilon_i \geq 0$, then $\hat{r}_i = \text{sign}\left(\sum_{j=1}^M \sum_{k=1}^{n_{ij}} w_j \ell_{jki}\right) \geq 0$, the answer will always be correct. If $\mu_i - \epsilon_i < 0$, then we will get the wrong answer with probability $\frac{\epsilon_i - \mu_i}{2\epsilon_i}$. Therefore, in the interval we are considering, we have

$$\Pr(\hat{r}_i \neq r_i) = \max\left\{0, \frac{\epsilon_i - \mu_i}{\epsilon_i}\right\} \leq \frac{1}{2} - \mu_{\min} \frac{1}{2\epsilon_i} \quad (4.7)$$

where $\mu_{\min} = \min \mu_{ij}$.

We would like to minimize the error probability summing over the N tasks, and our optimization objective is

$$\underset{n_{ij}}{\text{minimize}} \quad - \sum_{i=1}^N \sqrt{\frac{\left(\sum_{j=1}^M n_{ij}\right)^2}{\sum_{j=1}^M n_{ij} w_j^2}}$$

This is not necessarily a convex function, however, notice that $\sum_{j=1}^M n_{ij} \geq \sum_{j=1}^M n_{ij} w_j^2$ since $w_j \in [0, 1]$, and we can relax $-\sum_{i=1}^N \sqrt{\frac{\left(\sum_{j=1}^M n_{ij}\right)^2}{\sum_{j=1}^M n_{ij} w_j^2}}$ to its upperbound

$$- \sum_{i=1}^N \sqrt{\frac{\left(\sum_{j=1}^M n_{ij} w_j^2\right)^2}{\sum_{j=1}^M n_{ij} w_j^2}}$$

. This relaxation results in a convex optimization problem given by:

$$\begin{aligned}
& \underset{n_{ij}}{\text{minimize}} && - \sum_{i=1}^N \sqrt{\sum_{j=1}^M n_{ij} w_j^2} \\
& \text{subject to} && \sum_{ij} c_{ij} n_{ij} \leq B \\
& && n_{ij} \geq 0, \quad i = 1, \dots, N, j = 1, \dots, M
\end{aligned} \tag{4.8}$$

The optimal solution for the above problem can be expressed as:

$$n_{ij} = \begin{cases} \frac{B}{\frac{c_{ij}^2}{w_j^2} \sum_{l=1}^N \frac{w_l^2}{c_{lj}^2}} & \text{if } j = j_i^* \\ 0 & \text{if } j \neq j_i^* \end{cases}, \tag{4.9}$$

where $j_i^* = \arg \max_j \frac{w_j^2}{c_{ij}}$ and $i = 1, 2, \dots, N$.

From the optimal allocation scheme we can see that our model prefers the most cost-efficient crowd in terms of the ratio of its level of trust over cost. Since n_{ij} might be fractional, we set it to be $\lfloor n_{ij} \rfloor$. The full algorithm is shown in Algorithm 7 and we call it TAA for short.

The solution in equation (4.9) exhibits sparsity features since: 1) for any question, budget is allocated to only one of the crowds; and 2) when taking the floor, difficult questions tend to get 0 budget while easy questions get the whole share of the budget. We propose to address this problem by introducing an extra regularization term which penalizes the sparse behavior of allocation in Algorithm 7. For the sake of convenience, we relax the objective function in equation (4.8) by $-\sum_{i=1}^N \sqrt{\sum_{j=1}^M n_{ij} w_j^2} \geq -\sum_{i=1}^N \sum_{j=1}^M n_{ij} w_j^2$. Therefore the optimization problem

Algorithm 7: Trust-Aware Assignment

Input: N tasks, budget B , worker cost $c_{ij}(i = 1, \dots, N, j = 1, \dots, M)$

Output: job allocations n_{ij} , predicted answer \hat{r}_i

$Br = B;$

for $i = 1 : N$ **do**

$j_i^* = \arg \max_j \frac{w_j^2}{c_{ij}}$

for $j = 1 : M$ **do**

if $j = j_i^*$ **then**

$$n_{ij} = \left\lfloor \frac{B}{\frac{c_{ij_i^*}^2}{w_{j_i^*}^2} \sum_{l=1}^N \frac{w_l^2}{c_{lj_i^*}}} \right\rfloor$$

$$Br \leftarrow Br - \sum_{j=1}^M n_{ij_i^*} c_{ij_i^*}$$

else

$n_{ij} = 0$

end

end

end

while $Br > 0$ and $i \leq N$ **do**

$n_{ij_i^*} = n_{ij_i^*} + 1, Br = Br - c_{ij_i^*}, i = i + 1$

end

Use weighted majority voting to estimate answers

becomes:

$$\begin{aligned} \underset{n_{ij}}{\text{minimize}} \quad & - \sum_{i=1}^N \sum_{j=1}^M n_{ij} w_j^2 + \frac{\xi}{2} \|n\|_2^2 \\ \text{subject to} \quad & \sum_{ij} c_{ij} n_{ij} \leq B \end{aligned} \tag{4.10}$$

$$n_{ij} \geq 0, \quad i = 1, \dots, N, j = 1, \dots, M$$

The optimal solution of this problem is

$$n_{ij} = \frac{1}{\xi} \left(w_j^2 - c_{ij} \frac{\sum_{kl} c_{kl} w_l^2}{\sum_{kl} c_{kl}^2} \right) + \frac{B c_{ij}}{\sum_{kl} c_{kl}^2}, \quad \forall i, j$$

where ξ should be chosen such that n_{ij} is positive. We can see from this solution structure that for each question, budget will be allocated to multiple crowds instead

of just one. The penalty term in equation (4.10) gives credits to allocations that are more spread out, which makes the bound closer to equation (4.8). The full algorithm is shown in Algorithm 8 and we call it TAAP for short.

Algorithm 8: Trust-Aware Assignment With Penalty

Input: N tasks, budget B , worker cost $c_{ij}(i = 1, \dots, N, j = 1, \dots, M)$

Output: job allocations n_{ij} , predicted answer \hat{r}_i

$Br = B;$

for $i = 1 : N$ **do**

for $j = 1 : M$ **do**

$$n_{ij} = \left\lfloor \frac{1}{\xi} \left(w_j^2 - c_{ij} \frac{\sum_{kl} c_{kl} w_l^2}{\sum_{kl} c_{kl}^2} \right) + \frac{Bc_{ij}}{\sum_{kl} c_{kl}^2} \right\rfloor$$

$$Br \leftarrow Br - \sum_{j=1}^M n_{ij} c_{ij}$$

end

end

while $Br > 0$ **do**

for $i = 1 : N$ **do**

if $Br > 0$ **then**

 Randomly choose j th crowd

$$n_{ij} = n_{ij} + 1, Br = Br - c_{ij}$$

else

 Break

end

end

end

Use weighted majority voting to estimate answers

4.5 Theoretical performance bound

In this section, we discuss the performance of the allocation solution given by our proposed trust-aware allocation by providing the guaranteed upper bound of the error probability of the original optimization problem of equation (4.2).

Theorem 4.5.1. *For any $\sum_{ij} c_{ij} n_{ij} \leq B$, the total error probability is less than or*

equal to $\sum_{i=1}^N \exp \left(- \frac{Bw_{j_i^}^2 (2u_{ij_i^*} - 1)^2}{2c_{ij_i^*}^2 \sum_{l=1}^N \frac{j_l^*}{c_{lj_i^*}}} \right)$, where $j_i^* = \arg \max_j \frac{w_j^2}{c_{ij}}$.*

Proof. We first derive the solution structure for equation (4.8). The Lagrangian function is

$$L(\{n_{ij}\}, \lambda, \nu) = - \sum_{i=1}^N \sqrt{\sum_{j=1}^M n_{ij} w_j^2} + \lambda \left(\sum_{i=1}^N \sum_{j=1}^M n_{ij} c_{ij} - B \right) - \sum_{i,j} \nu_{ij} n_{ij} \quad (4.11)$$

where $\{\lambda\}$ and $\{\nu_{ij}\}$ are the set of dual variables. According to the KKT conditions, suppose for task i , the number of workers we assign to crowd j is positive, i.e. $n_{ij} > 0$, then the following equation holds:

$$\sum_{j=1}^M n_{ij} w_j^2 = \frac{1}{4\lambda^2} \frac{w_j^4}{c_{ij}^2} \quad (4.12)$$

If there exists no two crowds j_1 and j_2 that are equally efficient, i.e. $\frac{w_{j_1}^2}{c_{ij_1}} \neq \frac{w_{j_2}^2}{c_{ij_2}}, \forall j_1 \neq j_2$, which is usually the case, the optimal budget allocation strategy for task i is to allocate partial budget to one and only one of the crowds j_i^* . Therefore the optimal allocation becomes $n_{ij} = \frac{1}{4\lambda^2} \frac{w_{j_i^*}^2}{c_{ij_i^*}^2}$. The dual variable λ is obtained by solving $\sum_{i=1}^N \sum_{j=1}^M n_{ij} c_{ij} - B = 0$. To see what value j_i^* takes, we plug n_{ij} back to equation (4.8) and the objective function becomes:

$$- \sqrt{B \sum_{i,j} \frac{w_{j_i^*}^2}{c_{ij_i^*}}},$$

which can be minimized by letting $j_i^* = \arg \max_j \frac{w_j^2}{c_{ij}}$. Therefore the optimal solution

is:

$$n_{ij} = \begin{cases} \frac{B}{\frac{c_{ij}^2}{w_j^2} \sum_{l=1}^N \frac{w_l^2}{c_{lj}^2}} & \text{if } j = j_i^* \\ 0 & \text{if } j \neq j_i^* \end{cases}, \quad \text{where } j_i^* = \arg \max_j \frac{w_j^2}{c_{ij}} \quad (4.13)$$

where $i = 1, 2, \dots, N$.

Note that there might be multiple crowds that maximize the reliability-cost ratio, in which case we can randomly choose from those crowds. However, the performance bound is the same. We can pluggin n_{ij} back to equation (4.8) and summing over tasks gets us the result in Theorem 4.5.1. \square

This result is intuitive in that the larger the budget we have, the lower the error probability bound we can obtain. The bound improves exponentially with respect to budget increase. In addition, lower cost of c_{ij} and higher trust value $u_{ij_i^*}$ lead to lower error bound.

We can actually obtain an improved upper bound that holds with high probability from the perspective of PAC, like the work in [4].

Theorem 4.5.2. *For any $\sum_{j=1} c_{ij} n_{ij} \leq B$, with probability $(1 - \beta)^N$, the total error probability satisfies:*

$$\sum_{i=1}^N \Pr(\hat{r}_i \neq r_i) \leq \max \left\{ 0, \frac{N}{2} - \sum_{i=1}^N \mu_{\min} \sqrt{\frac{1}{-8 \ln \frac{\beta}{2}} \frac{B w_{j_i^*}^4}{c_{ij_i^*}^2 \sum_{l=1}^N \frac{w_l^2}{c_{lj_i^*}^2}}} \right\} \quad (4.14)$$

, where $j_i^* = \arg \max_j \frac{w_j^2}{c_{ij}}$.

Proof. The error probability in equation (4.7) holds with probability $(1 - \beta)^N$. ϵ_i can be determined by the solution in equation (4.9) through its relation with ϵ_i in equation (4.6). Therefore, with probability $(1 - \beta)^N$, the following error probability upper bound holds:

$$\begin{aligned}
\Pr(\hat{r}_i \neq r_i) &\leq \frac{1}{2} - \mu_{\min} \frac{1}{2\epsilon_i^*} \\
&\leq \frac{1}{2} - \mu_{\min} \sqrt{\frac{1}{-2 \ln \frac{\beta}{2}} n_{ij_i^*} w_{j_i^*}^2} \\
&\leq \frac{1}{2} - \mu_{\min} \sqrt{\frac{1}{-8 \ln \frac{\beta}{2}} \frac{Bw_{j_i^*}^4}{c_{ij_i^*}^2 \sum_{l=1}^N \frac{w_{j_l^*}^2}{c_{lj_i^*}}} }
\end{aligned} \tag{4.15}$$

Summing over tasks i , we get the PAC upper bound for total error probability in Theorem 4.5.2. □

4.6 Experimental results

Besides the theoretical results given in Section 4.5, we also evaluate the performance of our proposed *trust-aware assignment* (TAA) and *trust-aware assignment with penalty* (TAAP) on a real dataset and compared them against benchmark algorithms such as *uniform assignment* (UA) and algorithms from [4] adjusted to our setting, i.e. *crowd-quality-seeking assignment* (CQSA) and *cheap assignment* (CA). We show that our algorithms outperform state-of-the-art.

4.6.1 Benchmark Algorithms

The set of benchmark algorithms we use for comparison are:

1. UA: the algorithm tends to allocate the same number of people to answer a question from each available crowd. If the budget is not used up, for each question, it randomly chooses an expert from the set of crowds.
2. CQSA: for each question, the algorithm only chooses people from the most trustworthy crowd to assign to that question according to $n_{ij_i} = \left\lfloor \frac{B}{c_{ij_i}^2 \sum_{i=1}^N \frac{1}{c_{ij_i}}} \right\rfloor$, where $j_i = \arg \max_j w_j$. If budget is not consumed, it iterates the question set again and randomly chooses an expert from the set of crowds for each question.
3. CA: the algorithm only chooses the cheapest crowd (the least trustworthy crowd) for questions according to $n_{ij_i} = \left\lfloor \frac{B}{c_{ij_i}^2 \sum_{i=1}^N \frac{1}{c_{ij_i}}} \right\rfloor$, where $j_i = \arg \min_j w_j$. The same procedure is done as in crowd-quality-seeking assignment when budget is not used up.

After the assignment stage, weighted majority vote, as in equation (4.1), is applied to the algorithms.

4.6.2 Experiment Setup on Galaxy Zoo Dataset

The real dataset we use is Galaxy Zoo [63], a set of galaxy annotations contributed by a crowd of volunteers who are non-experts. The dataset contains statistics about votes of these volunteers for over 900,000 galaxies. The images of these galaxies are classified as elliptical (E), combined spiral (CS), or unknown by volunteers. The dataset from Galaxy Zoo used in this dissertation is SDSS image release 7. A subset of 700 galaxies that are classified as class elliptical or combined spiral is randomly chosen. These classified galaxies have more than 80% agreement and

the class agreed upon can be treated as truth label.

Classification of galaxy images from Galaxy Zoo does not have explicit difficulty levels and volunteers that participate in giving classifications do not have explicit level of trust either. However, we first divide the 700 galaxies into 2 groups based on the level of agreement. The first group is considered easy questions and the second group is considered difficult questions. The level of agreement in the first group is higher than that in the second. Then we simulate three kinds of crowds with increasing level of trustworthiness. Let α_t denote the difficulty parameter of type t question and β_j denote the trust parameter of type crowd j . Specifically α_t is scaled to $[5.0, 1.0]$ for easy and difficult questions respectively and the β scaled to $[0.65, 0.85, 0.98]$. Then we choose the trust value of crowd j toward question i as a sigmoid function of α_{t_i} and β_j : $u_{ij} = \frac{1}{1 + \exp(-\alpha_{t_i} \beta_j)}$, where t_i is the type of the i th question, which is easy or difficult in our case. Next we assume the input from the trust evaluation component is $w_{ij} = 2u_{ij} - 1$. In practice, this might not be the case. However, any good design of trust evaluation algorithm should output higher trust value for more reliable crowd and lower trust value for less reliable crowd and the assumption that $w_{ij} = 2u_{ij} - 1$ also exhibits such behavior.

With these models, we choose the cost function that maps the question difficulty and crowd's trust value to money in the following way: for easy questions, the cost of different crowds is $[0.1, 0.5, 0.9]$ and the cost for difficult questions is $[0.3, 0.6, 1.0]$. The cost function along with the trustworthiness values captures the following intuitive ideas: 1) for each question type, more trustworthy crowd incurs higher cost; and 2) for a particular crowd, answering difficult questions incurs higher

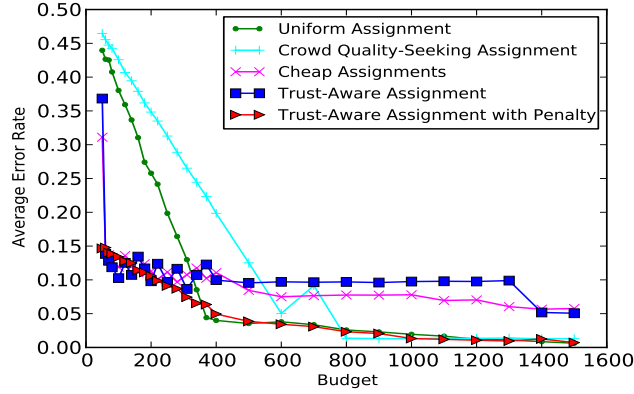


Figure 4.1: Total error probability of algorithms UA CQSA CA TAA TAAPon Galaxy Zoo dataset with budget ranging from 50 to 1500.

cost than answering easy ones.

4.6.3 Analysis

To test the performance of our proposed algorithm, we plot the total probability of error as the budget increases from 50 to 1500. The result is depicted in Fig. 4.1. It is easy to see that our proposed TAAP outperforms all other algorithms across the span of budget. In particular, when the budget is relative small, i.e. $B \leq 200$, both TAA and TAAP improve over CQSA and UA by up to 30%. This indicates that our algorithms excel in efficient allocation when budget is not abundant. Also, the cheap assignment algorithm does equally well when budget is small since there is not enough budget for answering difficult questions and people from a cheap crowd can answer easy questions equally well compared to an expensive crowd. When the budget is abundant, however, TAA behaves poorly compared to other algorithms except for CA. This is due to two reasons: 1) taking the floor in equation (4.9) makes many of the assignments 0, greatly deteriorating performance;

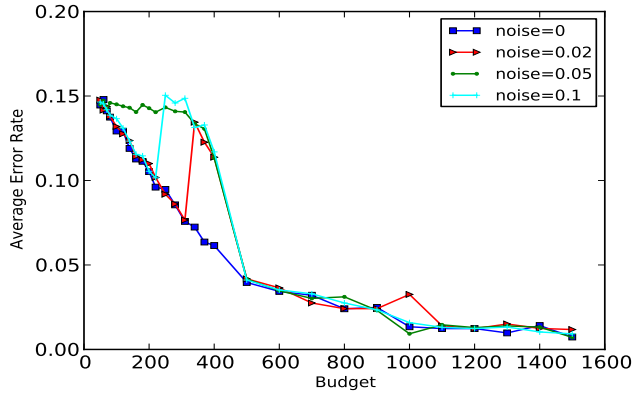


Figure 4.2: Total error probability of algorithm TAAP on Galaxy Zoo dataset under noise variance from 0 to 0.1 and the budget is from 50 to 1500.

and 2) the sparsity feature of equation (4.9), as mentioned earlier, did not switch to most trustworthy crowd even if budget is very high. TAAP addresses this problem and we can see that when budget is high, the algorithm does equally well compared to CQSA and UA.

The result in Fig. 4.1 assumes the trustworthiness can be perfectly estimated. Next we investigate the performance of TAAP when w_{ij} can not be perfectly estimated. By adding a Gaussian noise ϵ to u_{ij} , we have $w_{ij} = 2(u_{ij} + \epsilon) - 1$. We test TAAP with increasing variance of the noise ϵ ranging from 0 (perfectly estimated) to 0.1. Since u_{ij} takes value from $[0.5, 1]$ in our case, 0.1 is a significant noise variance. In Fig. 4.2, when budget is low, TAAP is to some extent affected by increasing noise variance. However, the error rate never increases by more than 5%, which is acceptable. When budget is sufficient, the algorithm is robust to varying noise variance levels and performs equally well compared to the case when trust can be perfectly estimated.

4.7 Summary

In this chapter, we considered the practical problem of budget allocation with trust estimation of different crowds. We would like to maximize the prediction accuracy within a given budget. In our setting, costs depend on both the question and the crowds grouped by level of expertise. We relaxed this accuracy-cost trade-off problem to a convex optimization problem by a PAC bound. We showed that there is a simple and intuitive closed-form solution to the convex problem. TAA always selects the most cost-efficient group for a given question and has at most $\frac{N}{2} - \mathcal{O}(\sqrt{B})$ prediction error. In addition, to address the problem of flooring and sparsity feature exhibited in TAA, we proposed TAAP and showed its outstanding performance through experiments on a real dataset across budget span.

Note that though we experimentally investigated the effect of trustworthiness estimation error, we did not theoretically explore the effect of it on the total error probability. We plan to further analyze this in the future. Additionally, the truth label is assumed to be discrete and binary. We also would like to investigate the continuous values.

Chapter 6: Conclusions

In this dissertation, we proposed a trust model with various decision rules based on local evidence in the setting of distributed consensus with adversaries. The global trust evaluation (trust propagation) can be used to obtain more accurate trust evaluation results if local evidences alone are not sufficient. The design of the trust-aware consensus algorithm is flexible in that it can incorporate more delicate decision rules and trust models. To evaluate the performance of the trust-aware consensus algorithm, we provided both novel theoretical analysis on the security performance in terms of miss detection rate and false alarm rate and empirical results through simulations. For the theoretical performance, we analysed miss detection rate and false alarm rate under regular trust graph assumption and more interestingly, we provided the upper bound of the miss detection rate under general trust graph assumption using probably approximately correct learning (PAC) techniques. The theoretical results show that the rate decreases exponentially with respect to the number of headers deployed and that smaller absorption probabilities on malicious nodes results in lower miss detection rate. These theoretical results serve as useful guideline for practical design of distributed computation network and the deployment of expensive headers in the network. For the empirical side, we ran sim-

ulations and we showed that our proposed trust-aware consensus algorithm could effectively detect various malicious strategies even in network with very low connectivity. The results can be applied to distributed collaborative sensor networks, sensor fusion and collaborative control.

In the distributed computation with supervisors setting, we investigated the model of trust in crowdsourcing problems in face of adversaries. We proposed a probabilistic model (MDC) that captures multi-domain characteristics of crowdsourcing questions and multi-dimensional trust of workers' knowledge. To show that our model MDC is very flexible and extensible to incorporate additional metadata associated with questions, we proposed an extended model MDFC that incorporates continuously-valued features of questions and MDTC that also combines topic discovery. MDTC has the advantage that the domains are interpretable. We showed that our proposed models have superior performance compared to state-of-the-art on two real datasets and can effectively recover the trust vectors of workers. This can be very useful in task assignment adaptive to workers' trust values in different dimensions in the future. We assume answers from workers are collected first and are then fed to models for inference. The results in this problem can be applied for fusion of information from multiple unreliable data sources instead of just workers in the open crowd. Examples of data sources are sensors, human input, and inference results given by another system backed by a different set of machine learning algorithms. Each of the data sources can be treated as a "worker" in this dissertation and we can thereafter use models to estimate the multi-domain trust values of the data sources and true labels of questions. Logical constraints between true

label of questions and worker trusts can be utilized to enhance the performance of trust estimation and true label estimation. We presented trust-aware crowdsourcing with domain knowledge (TCDK), a unifying framework that combines the power of domain knowledge and traditional crowdsourcing graphical model. It allows users to express domain knowledge using first-order logic rules without redefining the model. To estimate questions’ true labels and workers’ trust values, we develop a scalable inference algorithm based on alternating consensus optimization. We demonstrate that our model is superior to the state-of-the-art by testing it on two real datasets.

We also explored the problem of task allocation in crowdsourcing with adversaries. We aimed maximize the prediction accuracy within a given budget. In our setting, costs depend on both the question and the crowds grouped by level of expertise. We relaxed this accuracy-cost trade-off problem to a convex optimization problem by a PAC bound. We showed that there is a simple and intuitive closed-form solution to the convex problem. TAA always selects the most cost-efficient group for a given question and has at most $\frac{N}{2} - \mathcal{O}(\sqrt{B})$ prediction error. In addition, to address the problem of flooring and sparsity feature exhibited in TAA, we proposed TAAP and showed its outstanding performance through experiments on a real dataset across budget span.

Bibliography

- [1] Ali Khanafer, Behrouz Touri, and Tamer Basar. Consensus in the presence of an adversary. In *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys)*, pages 276–281, 2012.
- [2] Juan A. Garay and Rafail Ostrovsky. Almost-everywhere secure computation. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology, EUROCRYPT’08*, pages 307–323, Berlin, Heidelberg, 2008. Springer-Verlag.
- [3] Fabio Pasqualetti, A. Bicchi, and F. Bullo. Consensus computation in unreliable networks: A system theoretic approach. *Automatic Control, IEEE Transactions on*, 57(1):90–104, 2012.
- [4] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013.
- [5] A. MOSTEFAOUI and M. RAYNAL. Leader-based consensus. *Parallel Processing Letters*, 11(01):95–107, 2001.
- [6] Achour Mostéfaoui, Sergio Rajsbaum, and Michel Raynal. A versatile and modular consensus protocol. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks, DSN ’02*, pages 364–373, Washington, DC, USA, 2002. IEEE Computer Society.
- [7] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, March 1996.
- [8] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing, PODC ’83*, pages 27–30, New York, NY, USA, 1983. ACM.

- [9] Roy Friedman, Achour Mostefaoui, Sergio Rajsbaum, and Michel Raynal. Asynchronous agreement and its relation with error-correcting codes. *IEEE Trans. Comput.*, 56(7):865–875, July 2007.
- [10] A. Jadbabaie, Jie Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988–1001, 2003.
- [11] Wei Ren, R.W. Beard, and E.M. Atkins. A survey of consensus problems in multi-agent coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1859–1864 vol. 3, 2005.
- [12] Wei Ren and R.W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *Automatic Control, IEEE Transactions on*, 50(5):655–661, 2005.
- [13] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, 2004.
- [14] Reza Olfati-saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. In *Proceedings of the IEEE*, page 2007, 2007.
- [15] S. Sundaram and C.N. Hadjicostis. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *Automatic Control, IEEE Transactions on*, 56(7):1495–1508, 2011.
- [16] Heath J. LeBlanc, Haotian Zhang, Shreyas Sundaram, and Xenofon Koutsoukos. Consensus of multi-agent networks in the presence of adversaries using only local information. In *Proceedings of the 1st international conference on High Confidence Networked Systems, HiCoNS '12*, pages 1–10, New York, NY, USA, 2012. ACM.
- [17] Heath J. LeBlanc and Xenofon D. Koutsoukos. Low complexity resilient consensus in networked multi-agent systems with adversaries. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control, HSCC '12*, pages 5–14, New York, NY, USA, 2012. ACM.
- [18] Kiran K Somasundaram and John S Baras. Performance improvements in distributed estimation and fusion induced by a trusted core. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 1942–1949. IEEE, 2009.
- [19] John S Baras, Tao Jiang, and Punyaslok Purkayastha. Constrained coalitional games and networks of autonomous agents. In *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pages 972–979. IEEE, 2008.

- [20] Ion Matei, John S Baras, and Tao Jiang. A composite trust model and its application to collaborative distributed information fusion. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 1950–1957. IEEE, 2009.
- [21] Baobing Wang and John S Baras. Integrated modeling and simulation framework for wireless sensor networks. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*, pages 268–273. IEEE, 2012.
- [22] Dariusz G Mikulski, Frank L Lewis, Edward Y Gu, and Greg R Hudas. Trust method for multi-agent consensus. In *Proc. of SPIE Vol*, volume 8387, pages 83870E–1, 2012.
- [23] Xiangyang Liu and John S Baras. Using trust in distributed consensus with adversaries in sensor and other networks. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–7. IEEE, 2014.
- [24] Audun Jsang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, pages 41–55, 2002.
- [25] Xiangyang Liu, Peixin Gao, and John S Baras. Trust aware consensus with adversaries. Submitted for publication, 2014.
- [26] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigen-trust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.
- [27] Tao Jiang and John S. Baras. Graph algebraic interpretation of trust establishment in autonomic networks. *Wiley Journal of Networks*, 2009.
- [28] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.
- [29] Xiangyang Liu, He He, and John S Baras. Crowdsourcing with multi-dimensional trust. In *Information Fusion (Fusion), 2015 18th International Conference on*, pages 574–581. IEEE, 2015.
- [30] Murat Sensoy, Geeth de Mel, Lance Kaplan, Tien Pham, and Timothy J Norman. Tribe: Trust revision for information based on evidence. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 914–921. IEEE, 2013.
- [31] M. Richardson and P. Domingos. Learning with knowledge from multiple experts. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pages 624–631, 2003.

- [32] R. Snow, B. O’Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Empirical Methods on Natural Language Processing (EMNLP)*, pages 254–263, 2008.
- [33] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, pages 889–896, 2009.
- [34] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.
- [35] Q. Liu, J. Peng, and A. Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 701–709, 2012.
- [36] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, pages 1207–1216, 2009.
- [37] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2424–2432, 2010.
- [38] Y. Bachrach, T. Minka, J. Guiver, and T. Graepel. How to grade a test without knowing the answers - a bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1183–1190, 2012.
- [39] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [40] David J Newman, Seth Hettich, Cason L Blake, and Christopher J Merz. {UCI} repository of machine learning databases. 1998.
- [41] Andrey Rzhetsky, Hagit Shatkay, and W John Wilbur. How to get the most out of your curation effort. *PLoS computational biology*, 5(5):e1000391, 2009.
- [42] Stephen J Wright and Jorge Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.
- [43] Xiangyang Liu and John S Baras. Trust-aware crowdsourcing with domain knowledge. In *54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015.

- [44] Angelika Kimmig, Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4, 2012.
- [45] Xiangyang Liu, He He, and John S Baras. Crowdsourcing with multi-dimensional trust. In *Information Fusion (FUSION), 2015 18th International Conference on*, pages 574–581. IEEE, 2015.
- [46] Guo-Jun Qi, Charu C Aggarwal, Jiawei Han, and Thomas Huang. Mining collective intelligence in diverse groups. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1041–1052. International World Wide Web Conferences Steering Committee, 2013.
- [47] David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1171, 2011.
- [48] Jue Wang and Pedro Domingos. Hybrid markov logic networks. In *AAAI*, volume 8, pages 1106–1111, 2008.
- [49] Stephen Bach, Matthias Broecheler, Lise Getoor, and Dianne O’leary. Scaling mpe inference for constrained continuous markov random fields with consensus optimization. In *Advances in Neural Information Processing Systems*, pages 2654–2662, 2012.
- [50] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
- [51] Carlo Strapparava and Rada Mihalcea. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics, 2007.
- [52] Babak Loni, Maria Menendez, Mihai Georgescu, Luca Galli, Claudio Mas-sari, Ismail Sengor Altingovde, Davide Martinenghi, Mark Melenhorst, Raynor Vliegendhart, and Martha Larson. Fashion-focused creative commons social dataset. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 72–77. ACM, 2013.
- [53] Shobeir Fakhraei, Bert Huang, Louiqa Raschid, and Lise Getoor. Network-based drug-target interaction prediction with probabilistic soft logic. 2014.
- [54] Omar F. Zaidan and Chris Callison-BurchRichard. Crowdsourcing translation: Professional quality from non-professionals. 2011.

- [55] Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. Multiclass recognition and part localization with humans in the loop. 2011.
- [56] Richard M. C. McCreddie, Craig Macdonald, and Iadh Ounis. 2010.
- [57] Xiangyang Liu, He He, and John S Baras. Trust-aware optimal crowdsourcing with budget constraint. In *Communications (ICC), 2015 IEEE International Conference on*, pages 1176–1181. IEEE, 2015.
- [58] A. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: Algorithms for iterating data with humans. In *International Conference on Management of Data (SIGMOD)*, 2012.
- [59] D. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. neural information processing systems. In *Neural Information Processing Systems (NIPS)*, 2011.
- [60] C. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *International Conference on Machine Learning (ICML)*, 2013.
- [61] G. Theodorakopoulos and J.S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):318–328, 2006.
- [62] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- [63] Chris Lintott, Kevin Schawinski, Steven Bamford, Anze Slosar, Kate Land, et al. Galaxy Zoo 1 : Data Release of Morphological Classifications for nearly 900,000 galaxies. 2010.