POLITECNICO DI TORINO

MASTER's Degree in COMPUTER ENGINEERING

MASTER's Degree Thesis

Variance-Aware ColME: Enhancing Decentralized Online Mean Estimation

Supervisors

Prof. Emilio LEONARDI

Dr. Franco GALANTE

Candidate

Nikola STANKOVIC

September 2025

Abstract

The rapid growth of personal digital devices and the Internet of Things (IoT) has intensified the demand for collaborative and decentralized learning frameworks. Since these devices continuously generate sensitive and high-dimensional data, centralized transmission is often impractical, while purely local learning suffers from slow convergence as data accumulate gradually. Collaborative approaches can alleviate these issues by allowing agents to use information from one another to improve estimation, yet the challenge lies in the heterogeneity of data distributions. Each agent typically faces a personalized learning problem, and collaboration is only beneficial among agents whose data are generated from the same distributions, meaning they belong to the same similarity class. This thesis studies the problem of personalized online mean estimation in such heterogeneous environments, where each agent observes data from its own sigma-sub-Gaussian distribution. To address the challenge of heterogeneity, collaborative algorithms are introduced that enable agents to identify similarity classes in real time and exploit information from agents belonging to the similarity class within the same class to accelerate convergence and improve accuracy. The work builds on existing approaches: the collaborative mean estimation (colME) framework, which refines estimates through agent interaction using confidence intervals, and its graph-based extensions (C-colME and B-colME), which improve scalability and robustness in distributed settings. These state-of-the-art methods form the basis for the theoretical analysis and contributions developed in this thesis. The goal of this thesis is to provide algorithms and methods for fine-tuning and efficient implementation of these state of art approaches. Since the variance estimation plays a crucial role in the above mentioned algorithms, a method for accurate, local and real-time estimation of variance is proposed. A mean-independent and online variance estimator is introduced, with extensions for both class-homogeneous and class-heterogeneous cases, as well as for multidimensional data. The approach is further enhanced through the use of a forgetting factor, which allows adaptation to time-varying data distributions and mitigates the effect of initially misclassified observations. Estimation of sample kurtosis is also incorporated without explicit reliance on the agents' means, enabling the design of refined confidence intervals that account for differences not only in means and variances but also in higher-order characteristics (distribution) of the data. We derive the estimators for the sample standard deviation and sample kurtosis. These results are combined with sample mean estimation methods to design a unified procedure for constructing confidence intervals based jointly on the sample mean, sample variance, and sample kurtosis. This framework enables collaborative estimation in challenging scenarios, such as when classes share similar means but differ in variances, or when both means and variances are alike while the underlying distributions diverge in higher-order characteristics. Several algorithmic refinements of the collaborative mean estimation framework are proposed. These include simplified row-stochastic and Laplacian-based C-colME variants that reduce computational cost, reconnection mechanisms that prevent graph fragmentation when confidence intervals shrink, and weighted graph approaches, using Gaussian kernels, that improve convergence by emphasizing more reliable collaborations. The integration of reconnection and weighted graph strategies is shown to further accelerate learning. The proposed methods are validated through simulations and practical applications. Collaborative mean and variance estimation are also applied to image denoising, demonstrating improvements. All algorithms are implemented in Python from scratch, with efficient sparse matrix routines designed to handle data with small number of nonzero entries.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my thesis advisors, Prof. Emilio Leonardi and Dr. Franco Galante for introducing me to this interesting and challenging topic and for their valuable guidance throughout my research. I am also very grateful for the unwavering support and understanding of my family. I would like to thank my girlfriend Ivana for her constant encouragement and support during my studies and research.

Table of Contents

Introduction

The advances and widespread applications of personal digital devices, coupled with advancements in ubiquitous computing and the Internet of Things (IoT), have amplified the need for decentralized and collaborative computational paradigms. These devices are inherently data-centric, generating information that is often sensitive or voluminous, rendering centralized transmission impractical or undesirable. Although isolated local processing remains an option, learning in such a solitary manner can suffer from slow convergence, particularly when the data accumulates gradually.

To overcome this limitation, collaborative strategies, broadly termed federated learning (FL), [1], offer the potential to improve statistical power and accelerate the learning process by leveraging collective information between devices.

A key challenge in such collaborative settings arises from the inherent heterogeneity of data distributions across devices. The data collected by each agent reflect its unique usage patterns, production processes, and objectives, which requires the solution of personalized tasks. Despite this personalization, collaboration among agents with similar objectives or underlying data distributions can significantly reduce the time complexity and accelerate learning. Identifying these groups of similar agents becomes a crucial building block for designing effective collaborative algorithms, a task particularly challenging in online scenarios where data arrives sequentially over time.

This thesis addresses this challenging topic within the context of a recently introduced approach to online personalized mean estimation [2, 3]. A scenario where each agent continuously receives data from its own σ-sub-Gaussian distribution and aims to rapidly construct an accurate estimate of its individual mean is considered. To facilitate collaboration, the existence of an underlying class structure is assumed, where agents belonging to the same class share the same (or similar) mean value. Such assumptions are natural in various real-world applications, such as environmental monitoring where sensors in similar locations track related parameters [4], or collaborative filtering where user preferences are estimated by grouping users with comparable tastes [5, 6]. Critically, the number of these classes and their sizes are unknown to the agents and must be discovered in an online fashion.

In this thesis, we review the collaborative algorithms that enable agents to identify their class membership online and subsequently leverage the estimates of other agents within the same class to improve the speed and accuracy of their own mean estimation using the full-scale problem domain or scalable graph-based analysis domains. The presented approaches are rooted in Probably Approximately Correct (PAC) theory, allowing agents to iteratively discard agents from different classes with high confidence.

Basic theoretical analysis of the considered algorithms is presented, with the expected time required for an agent to correctly identify its class (separation time) and the time needed to achieve a desired accuracy in its mean estimate. The analysis underscores the importance of confidence intervals and variance estimation in the process of means and similarity classes. The results are compared with local ones (without collaboration) and the oracle results with prior knowledge of the class structure.

Special attention has been paid to the online collaborative and autonomous variance estimation, as a pre-request for the confidence intervals application. A simple algorithm for the variance estimation is proposed and used in simulation setups. Application of the proposed algorithm on complex case, with different variances for classes of agents in considered. A procedure with forgetting factor in the variance estimation is proposed and used in the case of time-varying means as well. The problem of overestimation and underestimation of variance, and consequently confidence intervals, is discussed. This discussion has lead to way to improve the collaborative algorithm convergence, in combination with a proposed simple local and online reconnection scheme. All results are illustrated and confirmed by statistical examples.

In this thesis, we have implemented the estimation of variance in a local and online way. We have also estimated the variance when it is different for each similarity class of agents, using the forgetting factor. The estimation of kurtosis is done. In addition to the standard colME, B-colME, and C-colME, we have implemented a simplified row-stochastic C-colME and a Laplacian-Based C-colME. We used the forgetting factor in the sums of the B-colME to deal with the time-varying distribution of the data. A reconnection approach with reduced confidence interval widths is described to improve convergence. Weighted graph forms of C-colME and B-colME to improve convergence are also considered. The combination of the weighted graphs and the reconnection approach is used to improve the convergence. Estimation of variance for multidimensional data with different variances for similarity classes is done. The application of collaborative mean estimation and variance estimation to image denoising is implemented. A Python code for the implementation of all previous methods is written using efficient sparse matrix calculations, both for one-dimensional data and for images.

Finally, the thesis is concluded with discussion and potential avenues for future research.

Related Works

First of all, this thesis reviews the method of collaborative mean estimation as introduced in [2, 3].

Over the past few years, the field of Federated Learning (FL) has witnessed extensive research into collaborative estimation and learning problems involving multiple agents with local datasets [1, 7]. Although traditional FL approaches focus on learning a single global model applicable to all participating agents, the inherent statistical heterogeneity across client data has spurred significant interest in personalized FL techniques. These aim to develop models tailored to the specific data distributions of individual clients [8, 9, 10, 11, 12, 10, 13, 14].

A common strategy in personalized FL involves grouping clients into clusters and subsequently training a specific model for each cluster [8, 10, 14]. The ideal clustering would group clients possessing similar local optimal models. However, as these optimal models are typically unknown a priori, the processes of model learning and cluster identification become intrinsically linked. Several studies have proposed using empirical measures of similarity, such as the Euclidean distance between local models or the cosine similarity of their updates, as a practical workaround, [8, 10]. Others assume some form of prior knowledge about the distances between clients' data distributions, [14, 15]. Nevertheless, accurately estimating these distances, particularly within the FL framework where clients may have limited local data, remains a significant challenge, as highlighted in [15]. This underscores the difficulty of identifying similar clients for effective collaborative model learning, a problem that is still largely unresolved.

In the online learning setting, collaborative learning has been predominantly explored in the context of multi-armed bandits (MAB). However, most existing approaches consider a single MAB instance solved collaboratively by multiple agents. Collaboration is often achieved through broadcast messages [16, 17], via a central server [18], or through local message exchanges over a network graph [19, 20, 21, 22, 23]. Some approaches even consider collision models where simultaneous arm pulls by multiple agents result in no reward [24, 21]. In all these cases, the agents share a common learning objective.

More recently, research has begun to address collaborative MAB settings where the arm means vary across agents. Extending their earlier work [24, 24] examine the scenario where arm means differ among players under a collision model, reducing the problem to a one-to-one assignment of agents to arms. In [25] the authors consider a model where local arm means are IID realizations of fixed global means, aiming to solve the global MAB using only local observations, drawing inspiration from traditional FL. Similarly, [26] extend the work of [17] to accommodate different local arm means, with the goal of identifying the arm with the largest aggregated mean. [25] introduces a limited form of personalization with optimization of a mixture of global and local MAB objectives. [27] further consider a known weighted combination of local MAB objectives, focusing on the pure exploration setting (best arm identification) rather than regret minimization. A key distinction from our work is that these MAB approaches do not require the discovery of relationships between local distributions to solve the respective problems.

Another related area involves identifying graph structures over the arms in MAB. In [28, 29] authors construct a similarity graph while solving the best arm identification problem, but within a single-agent setting. In contrast, our work focuses on a multi-agent setting with personalized estimation tasks, where the relationships between agents (based on their underlying data distributions) are unknown and must be learned online to facilitate collaboration.

The problem of mean estimation in a federated setting has been recognized as a fundamental building block [30, 31, 32] with practical significance in domains like smart agriculture, grid management, and healthcare [6].

In this context, the basic papers that we review here is the Collaborative Mean Estimation (ColME) algorithm proposed [2]. ColME considers an online, decentralized setting where clients exchange information with their peers to improve their mean estimates. However, ColME suffers from scalability issues due to its per-agent space and time complexities being linear in the number of clients. Furthermore, the original ColME algorithm is limited to scalar mean estimation with sub-Gaussian data distributions. Our work builds upon the principles of collaborative mean estimation but aims to address these limitations by developing scalable algorithms for multidimensional data with broader distribution classes, where agents self-organize based on perceived similarity to enhance collaboration. The scalability problem is solved by scalable approaches proposed in [3] with graphs as the analysis domains, in the form of message exchange approach B-colME and consensus-based approach C-colME. In this paper the authors also generalize the approach to multivariate data and relax conditions for the confidence interval calculations to the data with limited forth moments. This thesis follows the presentation, methods and notation presented in [3].

Chapter 1

Collaborative Mean Estimation

The problem of estimating means within a multi-agent setting is considered. Each agent aims to estimate its own mean of its underlying distribution. It is assumed that every distribution is σ-sub-Gaussian. The framework is both online and collaborative and is characterized by the following features: (i) Each agent receives data sequentially from its respective distribution. (ii) Each agent can interact with one other agent to exchange information.

These agents operate similarly to multiple user devices on a network, functioning concurrently, each collecting new data samples and communicating with other devices at every time step.

1.1 Basic Definitions

Consider $A = N$ agents that receive data and can perform some processing, along with communication with other agents. The key task considered here is to identify the set of agents with similar or the same data distribution. After this set is detected, collaborative processing, by exchanging the data, can significantly improve convergence and accuracy.

The basic approach uses a communication within the whole set of agents for collaborative processing [6]. However, this approach faces computational and memory complexity of order A, which is not practical for large systems. In order to reduce the system complexity, a graph can be assigned to the given set of agents as nodes. Then we start with a random graph with a defined starting degree r of each node (that is, establishing r links for communication with neighbors for each agent).

We assume that each agent $a \in \mathcal{A} = \{1, 2, , \dots, A\}$, receives a random sample $x_a(t) \in \mathbb{R}^1$, drawn from a distribution D_a with expected mean $\mu_a = \mathbb{E}\{x_a(t)\}$. The vector form of the signal is:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_A(t) \end{bmatrix}. \tag{1.1}$$

Similarity Classes: Agents that receive random samples $x_a(t) \in \mathbb{R}^1$, drawn from the same distribution D_a, with the same expected mean μ_a form a similarity class. The similarity class of an agent a is denoted by $\mathcal{C}_a$.

We will also assume that among all agents there are C similarity classes. Two agents a and a' belong to the same similarity class if $\Delta_{aa'} = |\mu_a - \mu_{a'}| = 0$ holds, that is, if the agents share the same mean value.

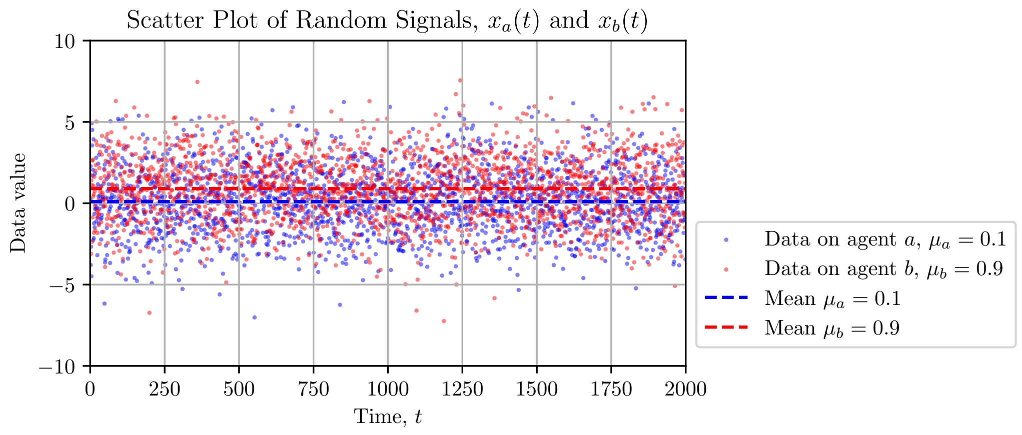


Figure 1.1: Illustration of data, $x_a(t)$, $x_b(t)$, of two agents, a and b, from different similarity classes for $t \in [0, 2000]$ with $\mu_a = 0.1$, $\mu_b = 0.9$, $\Delta_{ab} = 0.8$, and $\sigma = 2$.

The goal of each agent $a \in \mathcal{A}$ is to estimate its mean as fast as possible, using its own data and the data of some collaborative agents belonging to the same similarity class $\mathcal{C}_a$.

Local Means: The mean of one agent, after t samples are available (the sampling step is equal to 1, that is, $t = 1, 2, 3, \ldots, T$), is called *local mean*. It is defined as:

$$\bar{x}_{a,a}(t) = \frac{1}{t}\sum_{\tau=1}^{t} x_a(\tau) = \frac{1}{t}x_a(t) + \frac{t-1}{t}\bar{x}_{a,a}(t-1). \tag{1.2}$$

This empirical local mean slowly approaches the true mean μ_a. Convergence can be significantly improved if we can determine the similarity class, $\mathcal{C}_a$, of the agent a, or at least a subset of agents from this class. Then we can use more data to improve the mean convergence, by averaging the data not only over the time but over the set of data with the same mean (collaborative set).

The mean values for all agents, $a = 1, 2, \ldots, A = N$, will be written in vector form as:

$$\mathbf{X}(t) = \begin{bmatrix} \bar{x}_{1,1}(t) \\ \bar{x}_{2,2}(t) \\ \vdots \\ \bar{x}_{A,A}(t) \end{bmatrix} \quad \text{with } \mathbf{X}(t) = \frac{1}{t}\mathbf{x}(t) + \frac{t-1}{t}\mathbf{X}(t-1). \tag{1.3}$$

Illustration of many realizations of local means for data with two similarity classes, $\mu_a = 0.1$ and $\mu_b = 0.9$, are shown in Fig. 1.2. The local means for the agents with the mean value $\mu_a = 0.1$ are shown in red lines, while the local means corresponding to the agents with $\mu_a = 0.9$ are given in red. We can see that the realizations of local means can overlap for small t and then very slowly converge to their true mean values as time t progresses.

Local Sums: We will also use the local sums of samples (as they are used in one of the

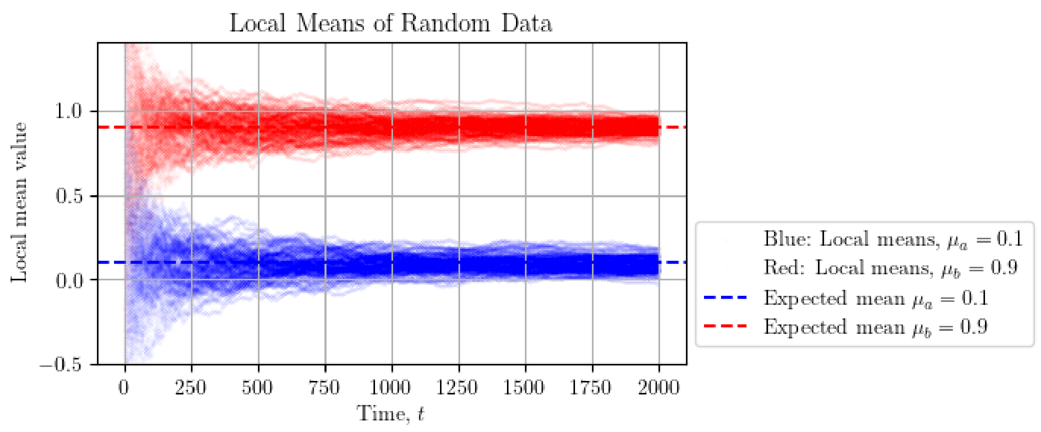


Figure 1.2: Illustration of local means, $\bar{x}_{aa}(t)$, of data $x_a(t)$ with 2 similarity classes, with $\mu_a = 0.1$, $\mu_b = 0.9$, $\Delta_{ab} = |\mu_a - \mu_b| = 0.8$, for $t \in [0, 2000]$. Data $x_a(t)$ are Gaussian distributed with the standard deviation $\sigma = 2$.

methods, B-colME, that will be described later) calculated as:

$$m_{a,a}(t) = \sum_{\tau=1}^{t} x_a(\tau) \tag{1.4}$$

or in vector form:

$$\mathbf{M}(t) = \begin{bmatrix} m_{1,1}(t) \\ m_{2,2}(t) \\ \vdots \\ m_{A,A}(t) \end{bmatrix} \quad \text{with } \mathbf{M}(t) = \mathbf{M}(t-1) + \mathbf{x}(t). \tag{1.5}$$

If the local sums are calculated (and stored), then the local means can be easily obtained as:

$$\mathbf{X}(t) = \frac{1}{t}\mathbf{M}(t). \tag{1.6}$$

The topic of this thesis is the variety of approaches for efficient and collaborative data mean estimations. After a review of the initial approach, called colME, we will describe the Message-Passing Algorithm, called B-colME and consensus-based approach C-colME. However, before any approach is used, we analyze the estimation of the similarity classes using the local means and the confidence intervals of each agent/node.

1.2 Confidence Intervals and Variance Estimation

Estimation of similarity classes is crucial for an efficient collaborative approach. Confidence intervals are used for this purpose. This is the reason why we first review the definitions and calculation of the confidence intervals. Confidence intervals are closely related to the variance of the data and possibly to the higher-order statistics, like kurtosis. Here we will propose and present a simple method for data statistics estimation.

1.2.1 Confidence Intervals

For the estimation of a set of agents that belong to the same similarity class, as the considered agent a, the confidence intervals will be used.

At each instant, t, after the agents receive new samples $x_a(t)$, that is a new vector $\mathbf{x}(t)$ is formed, they update their local means of local sums according to the previous formulae. For example, for local sums they perform $\mathbf{M}(t) = \mathbf{M}(t-1) + \mathbf{x}(t)$ and calculate local means by dividing this sum by the number of instants $\mathbf{X}(t) = \mathbf{M}(t)/t$.

Then, each agent compares its local mean with the local means of its neighbors (the definition of neighborhood will be specified by an algorithm) in order to try to conclude if the considered neighbor is within the same similarity class.

Confidence Intervals: In general, confidence intervals of a random variable (in our case local mean, $\bar{x}_{a,a}(t) = \frac{1}{t}\sum_{\tau=1}^{t} x_a(\tau)$, is the considered random variable) are defined as

$$\mathbb{I}_a(t) = [\bar{x}_{a,a}(t) - \beta_\delta(t), \bar{x}_{a,a}(t) + \beta_\delta(t)], \tag{1.7}$$

where $\beta_\delta(t)$ specifies the confidence interval width that will be discussed next.

Consider a random variable $\bar{x}_{a,a}(t)$ with a symmetric probability density function with respect to its mean μ_a. The confidence intervals are defined based on the probability that a value of the random variable $\bar{x}_{a,a}(t)$ differs from its mean value μ_a is lower than a given small probability δ.

Based on

$$\mathbb{P}\Big(|\bar{x}_{a,a}(t) - \mu_a| > \sigma B_\delta(t)\}\Big) < \delta, \tag{1.8}$$

where σ is the standard deviation of $x_a(t)$, we can state that the random variable $\bar{x}_{a,a}(t)$ satisfies the inequality

$$\mu_a - \sigma B_\delta(t) \ \leq \bar{x}_{a,a}(t) \leq \mu_a + \sigma B_\delta(t), \tag{1.9}$$

with probability greater than $1-\delta$.

For the case where

$$\lim_{t\to\infty} B_\delta(t) = 0 \tag{1.10}$$

we can state that the mean, μ_a, estimation based on sample mean, $\bar{x}_{a,a}(t) = \frac{1}{t}\sum_{\tau=1}^{t} x_a(\tau)$, is a Probably Approximately Correct (PAC) problem: We can make the estimation error smaller than an arbitrary small ε with probability higher than $1-\delta$ for any small δ, that is for any small (ε, δ) there exits τ_a such that for any $t > \tau_a$ we have

$$\mathbb{P}\Big(|\bar{x}_{a,a}(t) - \mu_a| < \varepsilon\Big) > 1 - \delta. \tag{1.11}$$

This fact will be used later to define two crucial properties of the confidence intervals that will be used in collaborative mean estimation.

Confidence intervals for special cases are as follows:

1. *Gaussian distribution:* If the local mean $\bar{x}_{a,a}(t)$ can be considered as a Gaussian distributed random variable, then it is distributed as $\bar{x}_{a,a}(t) \sim \mathcal{N}(\mu_a, \sigma\frac{1}{\sqrt{t}})$. For this

distribution we can easily calculate the confidence intervals for a given δ as

$$\sigma B_\delta(t) = \beta_\delta(t) = z_{1-\delta/2}\frac{\sigma}{\sqrt(t)}, \tag{1.12}$$

where the value $z_{1-\delta/2}$ for several commonly used values of δ is given next:

$1-\delta$	0.95	0.975	0.99	0.999
$z_{1-\delta/2}$	1.96	2.24	2.58	3.29

2. *Student's t-distribution:* If the number of samples is small and the standard deviation is also estimated by $\hat{\sigma}$, then we should use the Student's t-distribution, with the confidence bounds defined by:

$$\beta_\delta(t) = \sigma B_\delta(t) = t_{1-\delta/2,\nu}\frac{\hat{\sigma}}{\sqrt{t}}, \tag{1.13}$$

where the value $t_{1-\delta/2,\nu}$ depends on δ and the degree of freedom ν.

3. *σ-sub-Gaussian distribution:* The σ-sub-Gaussian distribution is a class of probability distributions characterized by tails that are no heavier than those of a Gaussian distribution with variance σ. If $x_a(t)$ is a random variable distributed as σ-sub-Gaussian, then:

$$\mathbb{E}[e^{\lambda(\mathrm{x}-\mu_a)}] \leq e^{\frac{\sigma^2\lambda^2}{2}}, \quad \text{for all } \lambda \in \mathbb{R}, \tag{1.14}$$

where the random variable is indicated by x. For this distribution holds:

$$\mathbb{P}(|\mathrm{x}-\mu_a| \geq t) \leq 2e^{-\frac{t^2}{2\sigma^2}}, \quad \text{for all } t \geq 0. \tag{1.15}$$

These bounds show the rapid decay of the probability of large deviations from the mean. For variance σ-sub-Gaussian distributed random varaible we have $\mathrm{Var}(\mathrm{x}) \leq \sigma^2$.

Examples of σ-sub-Gaussian distributions are:

- Gaussian distribution with $\mathrm{x} \sim \mathcal{N}(\mu_a, \sigma_1)$, for $\sigma_1 \leq \sigma$;
- Rademacher distribution with a random variable x taking values +1 and -1 with equal probability is 1-sub-Gaussian;
- Uniform distribution within $[-a, a]$ is $\frac{a}{\sqrt{3}}$-sub-Gaussian distributions;
- Symmetric Bernoulli distribution with a random variable taking values $+a$ and $-a$ with equal probability is a-sub-Gaussian.
- Linear combinations of independent σ_i sub-Gaussian distributions are $\sqrt{a_i\sigma_i^2}$ sub-Gaussian distribution.
- Vectors and matrices with independent σ-sub-Gaussian distributed elements are σ-sub-Gaussian distributed.

Their distributions are shown in Fig. 1.3.

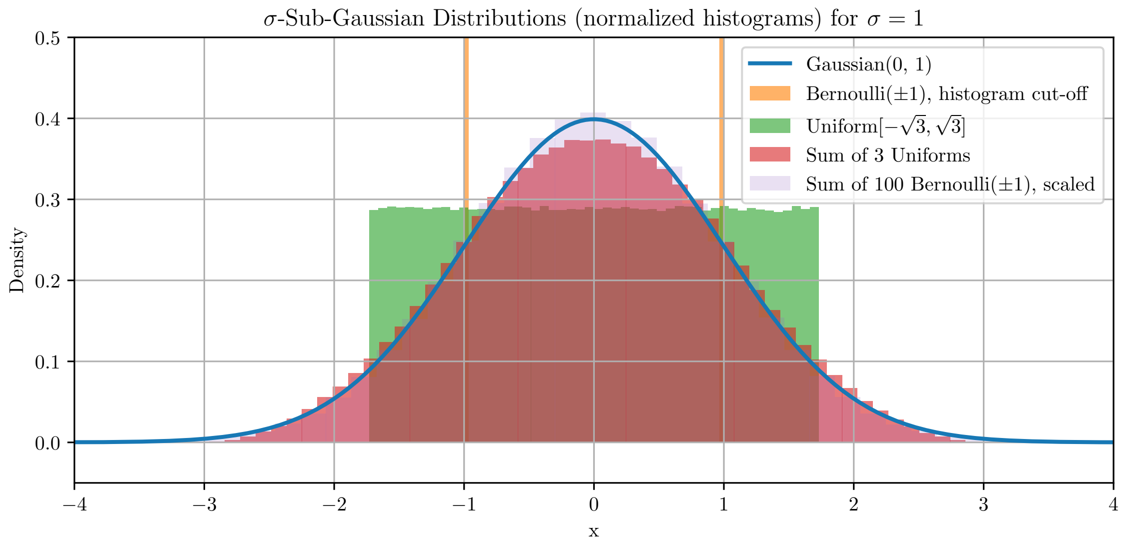


Figure 1.3: Illustration of several σ-sub Gaussian distributions with $\sigma = 1$.

The confidence intervals (by Laplace method, a direct consequence of Maillard Lemma) are defined using [33, 2, 3]

$$\mathbb{P}\Big(\bar{x}_{a,a}(t) - \mu_a\Big) > \sigma\sqrt{\frac{2}{t}\Big(1+\frac{1}{t}\Big)\ln\Big(\frac{\sqrt{t+1}}{\delta/2}\Big)\Big)} \tag{1.16}$$

as the Laplace bound

$$\beta_\delta(t) = \sigma B_\delta(t) = \sigma\sqrt{\frac{2}{t}\Big(1+\frac{1}{t}\Big)\ln\Big(\frac{\sqrt{t+1}}{\delta/2}\Big)}. \tag{1.17}$$

4. *Bounded Fourth-Moment Distributions:* Consider a distribution with bounded fourth-moment, [3]

$$\mathbb{E}\{(x_a(t) - \mu_a)^4\} \leq \mu_4. \tag{1.18}$$

The true mean, μ_a, with probability greater than $1-\delta$, belongs to the confidence intervals centered in $\bar{x}_{a,a}(t)$ and bounded with

$$\beta_\delta(t) = \sigma\sqrt[4]{\frac{2(\kappa+3)}{\delta/2}\frac{(1+\ln^2(t))}{t}}. \tag{1.19}$$

When all the variables are identically distributed, κ corresponds to the kurtosis

$$\kappa = \frac{\mathbb{E}\{(\mathrm{x}-\mu_a)^4\}}{\sigma^4}. \tag{1.20}$$

This result produces tighter confidence intervals than the one obtained using Laplace bounds. For Gaussian distribute $\bar{x}_{a,a}(t)$ we have $\kappa = 3$. For Gaussian distribution and $\delta = 0.01$ we can write $\beta_\delta(t) = 8.32\sigma\sqrt[4]{(1+\ln^2(t))/t}$.

Since confidence intervals will play a crucial role in the convergence of the presented approaches, we will discuss their properties and behavior in more detail.

The confidence interval properties: The confidence intervals contain the expected true value μ_a of the data $\bar{x}_{a,a}(t)$ with probability higher than $1-\delta$. They also tend to 0 as $t \to \infty$. Therefore, for sufficiently large t:

- Two confidence intervals of agents a and a' of the same similarity class will intersect with high probability, since they contain at least one common point, the true expected value, μ_a.
- Two confidence intervals of agents a and a' of different similarity classes will not intersect with a high probability, since they are centered around different true expected values, $\mu_a \neq \mu_{a'}$ and the widths of the confidence intervals tends to 0.

So, the *intersection of confidence intervals* will work as an indicator if the agents considered a' should be considered to belong to the similarity class of a or not. This criterion is used for similarity class determination.

Initially, all neighbors are assumed to belong to the similarity class of a. After each instant t, all agents first check the confidence intervals with each neighbor.

- If the confidence intervals intersect, that neighbor is kept in its neighborhood.
- If the confidence intervals do not intersect, then the edge connecting a and a' is disconnected and that agent is no longer considered as a neighbor (value of 1 in the adjacency matrix for that connection is set to 0 and we have a new adjacency matrix). This is why the adjacency matrix, defining the agent neighborhoods, is time dependent $\mathbf{A}(t)$.

Comments on the Confidence Intervals: The local sums, for large number of terms, according to the Central-Limit-Theorem approach to Gaussian distribution if the random variables $x_a(t)$ are independent-identically distributed (i.i.d.) with finite mean and variance. For statistical analysis, that means if $x_a(t)$ are not heavy-tailed (sub-Gaussian distributed are not), then after about $t = 30$ we may use Gaussian confidence intervals, with a small error. Even if $x_a(t)$ were heavy-tailed with finite mean and variance, then after about a hundred samples t, the Gaussian values would be good estimates for the confidence intervals.

To this aim, we shall plot the Laplace bound, the Gaussian bound and the student t-bound Fig. 1.4.

We can conclude that the Laplace bound overestimates the value of the confidence interval width by about 80% compared to the Gaussian bound. The widths of the overestimated confidence intervals slow the separation of the agents belonging to different classes of similarity, making the whole algorithm slower in convergence. This effect will be considered in the next section. An illustration of three local mean realizations, together with confidence intervals determined by the Laplace bounds, is given in Fig. 1.5.

1.2.2 Expected Separation Time

Consider two agents a and b from two different similarity classes with mean values μ_a and μ_b. Without loss of generality assume that $\mu_a > \mu_b$, that is $\Delta = \mu_a - \mu_b > 0$. Their local means

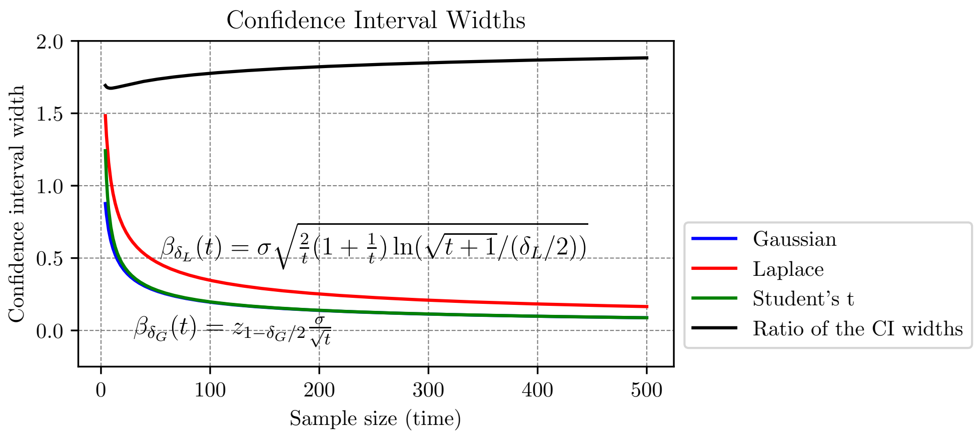


Figure 1.4: Ratio of confidence interval widths as a function of the number of samples t for: Gaussian distribution of local means, σ-sub Gaussian distribution (Laplace width), and the Student's t distribution, along with the ratio of the Gaussian and Laplacian widths.

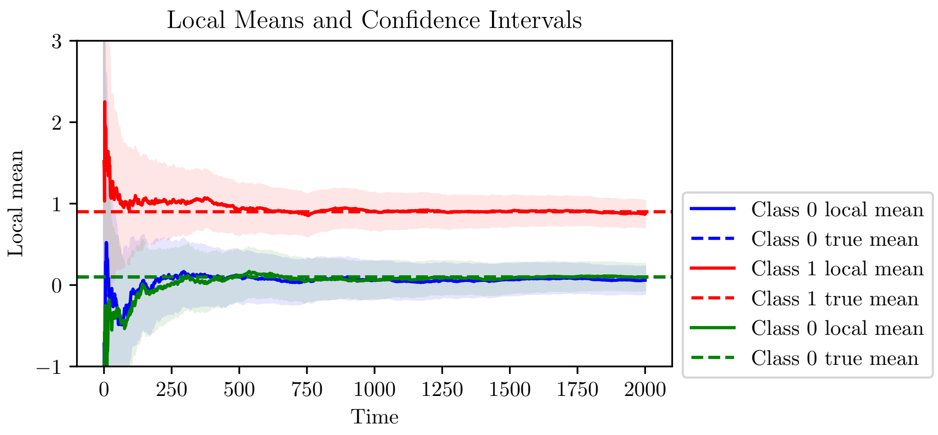


Figure 1.5: Local means and the corresponding confidence intervals for 3 agents belonging to two different classes, as a function of the number of samples t.

at time t are denoted by $\bar{x}_{aa}(t)$ and $\bar{x}_{bb}(t)$. The critical situation for the confidence intervals is when

$$\beta_\delta(t) + \beta_\delta(t-1) = \bar{x}_{aa}(t) - \bar{x}_{bb}(t). \tag{1.21}$$

Assuming that $\beta_\delta(t) \approx \beta_\delta(t-1)$ and taking the expected values of the previous equation, we get:

$$2\beta_\delta(t) = \mu_a - \mu_b. \tag{1.22}$$

By solving this equation for t, with a given $\mu_a - \mu_b$, we get the expected separation time T_{sep}. In general, if there are more than two classes, then the expected separation time is obtained by solving $2\beta_\delta(t) = \min|\mu_a - \mu_b| = \min\{\Delta_{ab}\}$ for all μ_a and μ_b.

The expected separation time is calculated for different values of Δ, using $\sigma = 2$, and is shown in Fig. 1.6.

The result is statistically checked for an example that will be described in detail later. The time when two confidence intervals do not intersect is recorded as the separation time T_{sep}. It corresponds to the separation of the agents into different classes. The histogram of the separation time for $\Delta = 0.8$ and $\sigma = 2$ is shown in Fig. 1.7. This numerical result fully corresponds to the expected separation time from Fig. 1.6. We can see that in this particular numerical realization, the separation time occurred as early as about $t = 50$ and as late as $t = 800$, with the mean separation time value $t = T_{sep} = 293$.

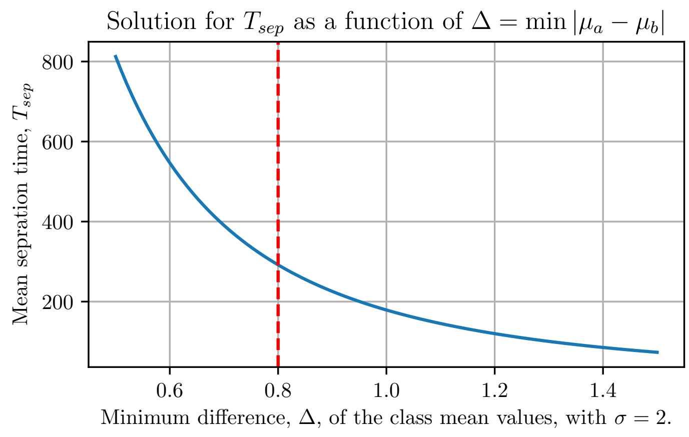


Figure 1.6: Expected separation time as a function of Δ For $\Delta = 0.8$ its value is $T_{sep} = 293$.

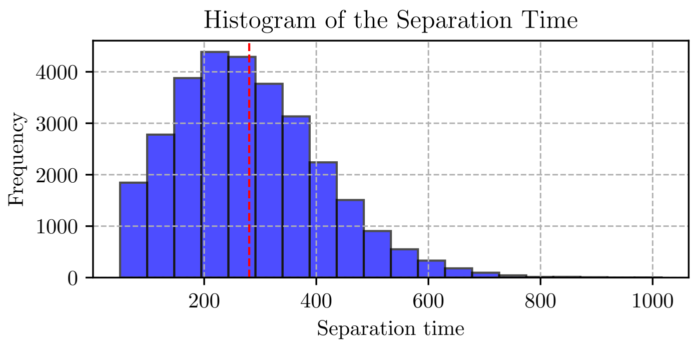


Figure 1.7: Histogram of separation time for $\Delta = 0.8$ and $\sigma = 2$ in the confidence interval experiment, with the mean separation time value indicated by a red vertical line.

1.2.3 Variance Estimation

In order to calculate the confidence intervals, we have to know the common standard deviation σ of the data $x_a(t)$ or, in reality, estimate it.

To estimate the variance, we will define a new random variable

$$d_a(t) = x_a(t) - x_a(t-1). \tag{1.23}$$

Note that the new random variable $d_a(t)$ has zero mean for all $a \in \mathcal{A}$, $\mathbb{E}\{d_a(t)\} = 0$, since we assumed that the exact expected value does not change over time for a given agent, $\mathbb{E}\{x_a(t)\} = \mu_a$. *It means that for $d_a(t)$ all agents belong to the same similarity class, with a known zero mean.*

We can efficiently estimate the variance from just several initial data samples (in the case of the same variance, all agents can collaborate in the variance estimation, since the estimation is mean-invariant).
The variance estimation is based on the fact that

$$\mathbb{E}\{d_a^2(t)\} = \mathbb{E}\{\Big(x_a(t) - x_a(t-1)\Big)^2\} \tag{1.24}$$

$$= \mathbb{E}\{\Big(x_a(t) - \mu_a - (x_a(t-1) - \mu_a)\Big)^2\} \tag{1.25}$$

$$= \mathbb{E}\{(x_a(t) - \mu_a)^2\} + \mathbb{E}\{(x_a(t-1) - \mu_a)^2\} = 2\sigma^2, \tag{1.26}$$

since $\mathbb{E}\{x_a(t) - \mu_a\}\mathbb{E}\{x_a(t-1) - \mu_a\} = 0$.

In practical implementation, we will consider two methods:
(i) One is based on using local data, without collaboration with neighborhood. This method will be used later in combination with local means. Then the estimated variance (and standard deviation as its squared root) for an agent a, at an instant t, is

$$\hat{\sigma}_a^2(t) = \frac{1}{2t}\sum_{\tau=1}^{t}\big(x_a(\tau) - x_a(\tau-1)\big)^2. \tag{1.27}$$

Since this method does not use collaboration, it can (and will) be used for estimation when each similarity class has different variance and for defining the confidence intervals for sample standard deviation.
(ii) The other, which assumes common inter-agent communication within the defined neighborhood only. It is of practical importance, as it does not require any additional communication links.

Agents can communicate only within their defined neighborhood (denoted by $\mathcal{N}_a$) over the established, initially r links. The agents exchange (depending on the collaboration method used) the local means or the local sums. In both cases, we can recover the most recent data values $x_a(t)$. Assuming that the local means are exchanged, at $t = 0$, the data values are obtained as

$$m_{aa}(0) = x_a(0), \text{ and } m_{aa'}(0) = x_{a'}(0), \text{ for } a' \in \mathcal{N}_a. \tag{1.28}$$

At $t = 1$ we have

$$m_{aa}(1) = x_a(0) + x_a(1), \; m_{aa'}(1) = x_{a'}(0) + x_{a'}(1), \; a' \in \mathcal{N}_a. \tag{1.29}$$

Then we can form r differences (for all r neighboring nodes) using $m_{aa'}(1) - 2m_{aa'}(0)$ as

$$x_a(1) - x_a(0), \text{ and } x_{a'}(1) - x_{a'}(0), \text{ for } a' \in \mathcal{N}_a. \tag{1.30}$$

We can estimate the variance using these r values. However, the number of differences may not be sufficient for an accurate estimate, as r is usually small. Then, this procedure should be repeated for several initial instants t in order to accumulate enough values for a satisfactory estimation. For example, for each $t = 2, 3 \dots, 10 = T_s$ we form r differences:

$$\begin{gathered} x_a(2) - x_a(1), \text{ and } x_{a'}(2) - x_{a'}(1), \text{ for } a' \in \mathcal{N}_a, \\ x_a(3) - x_a(2), \text{ and } x_{a'}(3) - x_{a'}(2), \text{ for } a' \in \mathcal{N}_a, \\ \dots \\ x_a(10) - x_a(9), \text{ and } x_{a'}(10) - x_{a'}(9), \text{ for } a' \in \mathcal{N}_a, \end{gathered}$$

and estimate

$$\hat{\sigma}_a^2(T_s) = \frac{1}{2T_s(|\mathcal{N}_a(T_s)| + 1)} \sum_{\substack{a' \in \mathcal{N}_a(0) \cup \{a\} \\ t=1,\dots,T_s}} \left(x_{a'}(t) - x_{a'}(t-1)\right)^2. \tag{1.31}$$

Note again that in the variance estimation process, all A agents in the system can collaborate in the variance estimation if *the variance of all agents is the same and classes (the data distributions) D_a differ only in the mean value.* So, all these values are squared and their mean is found. If we use, for example, 10 instants and $r = 12$ neighbors a', then we get 120 + 12 differences (including the central node a).

During the initial instants, until we estimate variance, we keep it very large so that all confidence intervals intersect (as they usually do initially), $\sigma_a = 10$.
Local Sums Case: If the agents exchange sums,then

$$x_{a'}(t) = m_{a,a'}(t) - m_{a,a'}(t-1) = \sum_{\tau=1}^{t} x_{a'}(\tau) - \sum_{\tau=1}^{t-1} x_{a'}(\tau). \tag{1.32}$$

Local Means Case: Agents may exchange collaborative local means $\bar{x}_{aa}(t)$. Then, for $t \le T_s$, we calculate the data values as

$$x_{a'}(t) = t\bar{x}_{a,a'}(t) - (t-1)\bar{x}_{a,a'}(t-1) \tag{1.33}$$

with $\bar{x}_{a,a'}(t) = \mu_a(t)$ for $t \le T_s$.

If the mean estimation algorithm was defined in such a way that the collaborative means were only exchanged, then to avoid the influence of neighboring agents on the exchanged means, during variance estimation, and to ensure that only local means are exchanged, in consensus-based approaches we can use $\alpha(t) = 0$ for $t \le T_s$ and the given value $\alpha(t)$ afterworlds. This will also avoid agent collaboration in mean calculation during the initial phase, when the neighbors selection is highly unreliable.

This procedure is summarized within the Algorithm 1.

Algorithm 1 Local Collaborative Estimation of the Standard Deviation, σ, using Agent Neighborhood

```
1: for each $t = 0$ to $T$ do
2:     for each agent $a \in \mathcal{A}$ do
3:         $s \leftarrow 0, \quad \hat{\sigma}_a \leftarrow 10$ (a large value so that CI intersect)
4:         if $t \leq T_s$ then (such that $T_s|\mathcal{N}_a| \sim 100$)
5:             Update $s$:
                   $$s \leftarrow s + \sum_{a' \in \mathcal{N}_a \cup \{a\}} |x_{a'}(t) - x_{a'}(t-1)|^2$$
6:         else if $t = T_s$ then
7:             Compute $\hat{\sigma}_a$:
                   $$\hat{\sigma}_a \leftarrow \sqrt{\frac{s}{2T_s(|\mathcal{N}_a(0)| + 1)}}$$
8:         end if
9:     end for
10:    Compute Confidence Intervals and Collaborative Mean
11: end for
```

The results of the standard deviation estimation in a network of agents with $\mathcal{N}_a(0) = r = 12$ neighbors, using $T_s = 20$ initial instants, are shown as a histogram in Fig. 1.8. We can see that the true variance in the data was estimated by agents within the interval $\hat{\sigma}_a \in [1.6, 2.4]$, that is, with about 20% precision.

The same simulation is repeated, but using $T_s = 50$ initial instants. The results are given in the form of a histogram in Fig. 1.8. Here, the variance was estimated within the interval $\hat{\sigma}_a \in [1.8, 2.15]$, that is, with about 10% precision. With respect to standard deviation estimation, this accuracy could be considered as high, as will be seen next.

If the data are Gaussian, then, for a given number of samples of random variable $d_a(t)$, we can find the confidence intervals for the variance estimation. In that case, the variance follows a chi-squared (χ^2) distribution:

$$\frac{(n-1)\hat{\sigma_a}^2}{\sigma^2} \sim \chi^2_{n-1}, \tag{1.34}$$

with $n = T_s(|\mathcal{N}_a(0)| + 1)$, χ^2_{n-1} represents the chi-square distribution with $n-1$ degrees of freedom.

Confidence interval for the variance with, $\chi^2_{\alpha/2,n-1}$ and $\chi^2_{1-\alpha/2,n-1}$ being the $\alpha/2$ and $1-\alpha/2$ quantiles of the χ^2_{n-1} distribution, respectively, follow from

$$P\left(\chi^2_{1-\alpha/2,(n-1)} \leq \frac{(n-1)\hat{\sigma_a}^2}{\sigma^2} \leq \chi^2_{\alpha/2,n-1}\right) = 1 - \alpha \tag{1.35}$$

and

$$P\left(\frac{(n-1)\hat{\sigma_a}^2}{\chi^2_{\alpha/2,(n-1)}} \leq \sigma^2 \leq \frac{(n-1)\hat{\sigma_a}^2}{\chi^2_{1-\alpha/2,n-1}}\right) = 1 - \alpha. \tag{1.36}$$

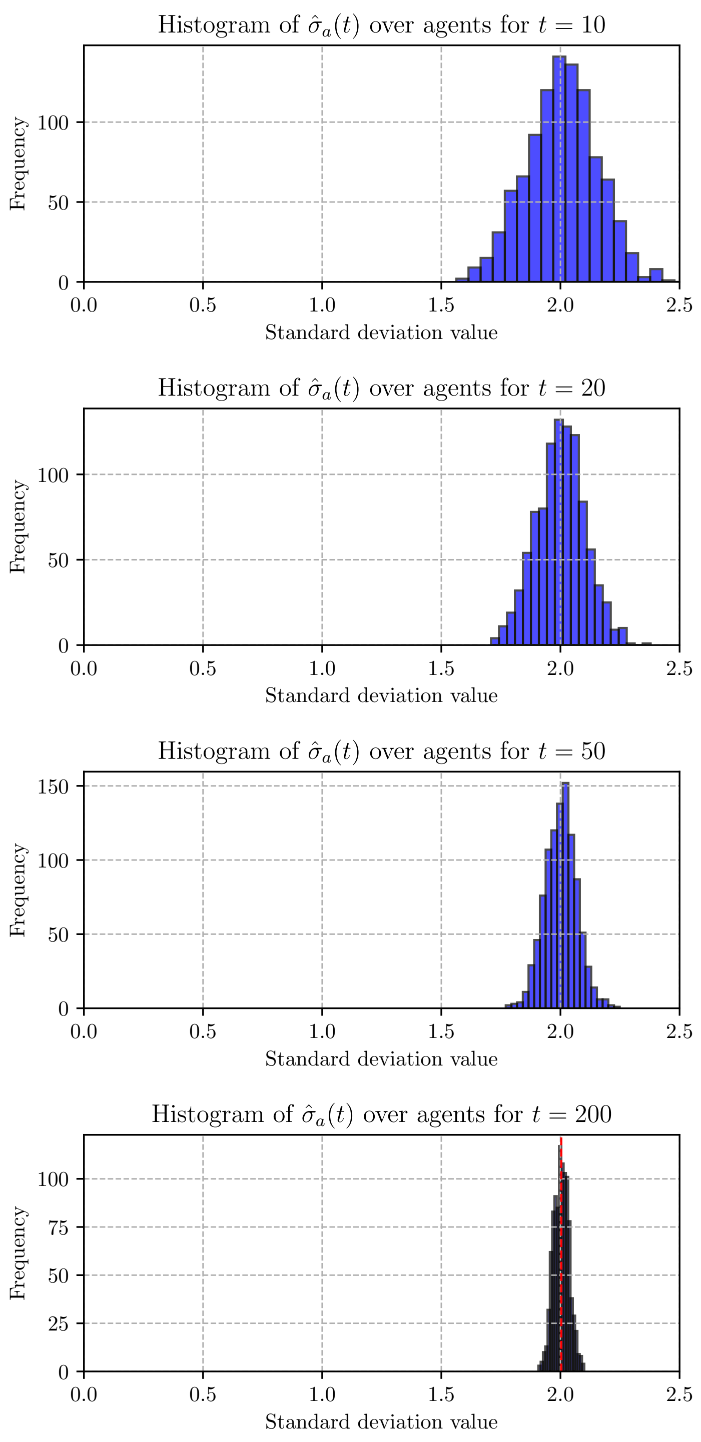


Figure 1.8: Histogram of the estimated standard deviations $\hat{\sigma}_a(t)$ over agents within $T_s = 10, 20, 50, 200$ time instants.

Taking square root for the standard deviation, we arrive at the bounds

$$\left[\sqrt{\frac{(n-1)\sigma^2}{\chi^2_{\alpha/2,n-1}}}, \sqrt{\frac{(n-1)\sigma^2}{\chi^2_{1-\alpha/2,n-1}}}\right]. \tag{1.37}$$

The confidence intervals are calculated for the standard deviation and given in Table 1.1. These values correspond to the values that can be read from the histogram, Fig. 1.8. Since we compared the confidence intervals with the statistical results (in the form of a histogram), we have used $\alpha = 0.001$ to correspond to the probability of $1/N$ with $N = 1000$ agents. Smaller values of α would produce more conservative bounds, while for larger α there would be many cases (out of 1000) outside the bounds.

Number of Samples	Standard Deviation Confidence Intervals
$n = 10(12+1) = 130$	(1.65, 2.50)
$n = 20(12+1) = 260$	(1.74, 2.33)
$n = 50(12+1) = 650$	(1.83, 2.20)
$n = 200(12+1) = 2600$	(1.91, 2.10)

Table 1.1: Sample Sizes and Corresponding Confidence Intervals for Standard deviation, σ_a, Estimation for $\alpha = 0.001$.

In order to illustrate the influence of errors on the standard deviation estimation, we will present the parameter $\beta_\delta(t)$ for the Laplace method with the maximum and minimum estimated standard deviation by an agent. The value of $\beta_\delta(t)$ for the agents with the estimated maximum and minimum standard deviations $\min_a(\hat{\sigma}[a])$ and $\max_a(\hat{\sigma}[a])$ is shown in Fig. 1.9 with a dashed red line, while the interval between these lines is shaded. We can see that the influence of the error in the standard estimation deviation is well below the influence of the method used for the confidence interval estimation (Gaussian or Laplace for sub-Gaussian).

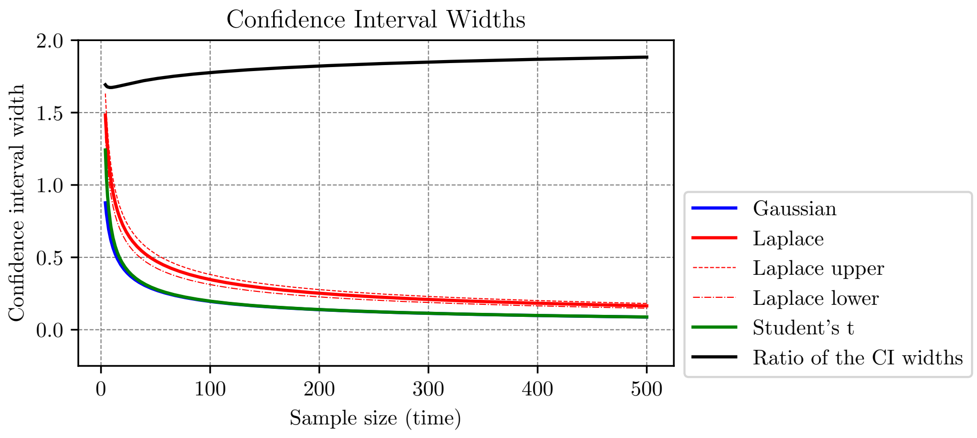


Figure 1.9: Ratio of confidence interval widths as a function of the number of samples t for the agents with maximum and minimum estimated standard deviation, from Fig. 1.8 with $T_s = 50$.

Finally, we will analyze the influence of the overestimated and underestimated standard deviation.

- **Overestimated Confidence Interval Width (or Standard Deviation):** If the standard deviation is overestimated then the confidence intervals $\mathbb{I}_a(t) = [\bar{x}_{a,a}(t) - \beta_\delta(t), \bar{x}_{a,a}(t) + \beta_\delta(t)]$ and $\mathbb{I}_{a'}(t) = [\bar{x}_{a',a'}(t) - \beta_\delta(t), \bar{x}_{a',a'}(t) + \beta_\delta(t)]$ for two agents will: (1) Certainly intersect with a higher probability than required by δ if the agents a and a' are from the same similarity class; (2) Increase the expected instant when they do not intersect with a higher probability than required by δ if the agents a and a' are not from the same similarity class. In general, we can conclude that this will only slow down the convergence of the approach toward the estimated values of the oracle, also reducing the probability of error by reducing the probability of isolated agents. Note that if the algorithm does not make mistakes in the pruning of the graph, the probability of having isolated nodes depends only on the initial random graph (on $\mathbf{A}$ and r) and on how the classes are assigned to the agents (distribution and the number of classes; more details will be given later). Mistakes, such as misclassifying the agents from the same class as agents from different classes , can increase the number of isolated terms. This kind of error appears when the confidence intervals of two agents from the same class mistakenly do not intersect, what can happen if we assume δ is relatively large, that is confidence intervals are narrow.

- **Underestimated Confidence Interval Width (or Standard Deviation):** If the standard deviation is underestimated then the confidence intervals $\mathbb{I}_a(t) = [\bar{x}_{a,a}(t) - \beta_\delta(t), \bar{x}_{a,a}(t) + \beta_\delta(t)]$ and $\mathbb{I}_{a'}(t) = [\bar{x}_{a',a'}(t) - \beta_\delta(t), \bar{x}_{a',a'}(t) + \beta_\delta(t)]$ for two agents will: (1) Intersect with a lower probability than required by δ if the agents a and a' are from the same similarity class, resulting in increased probability of non-decreasing the error (since collaborative agents can be excluded); (2) Decrease the expected instant when they do not intersect with a higher probability than required by δ if the agents a and a' are not from the same similarity class. In general, we can conclude that *this will improve the convergence of the approach, but with an increase in error due to additional isolated agents.* This case can even be very beneficial for the whole system if the confidence intervals widths are, by theory, underestimated (since they are calculated as upper bounds). We have illustrated three cases of confidence intervals shown in Fig. 1.10. In the first case, we calculated confidence intervals based on underestimated standard deviation. The true standard deviation was 2 and we used 1 in this case. In the second graph, we used the true standard deviation, while in the third graph, we used the standard deviation overestimated by 25%. As we can see, overestimation is reflected in the delayed separation of the different similarity classes. In summary, we can conclude that using confidence intervals as tight as possible would be very beneficial for the convergence. That would also mean that if we can estimate the distribution of the random variable and of the local means, it would be better to use the exact confidence interval calculation for the specific distribution than to use the general ones, commonly very overestimated.

1.2.4 Similarity Classes Differ in Both Means and Variances

Assume now a very complex case where the similarity classes of agents have different means and different variances. Since the problem is complex, here we will present a two-class system of agents with distributions (μ_1, σ_1) and (μ_2, σ_2). Systems with three and five classes, with different means and variances, will be simulated in the next sections, using C-colME and B-colME algorithms.

If we apply the accumulations as in the described algorithm, we would get:

$$s \leftarrow s + \sum_{a' \in \mathcal{N}_a \cup \{a\}} |x_{a'}(t) - x_{a'}(t-1)|^2 \tag{1.38}$$

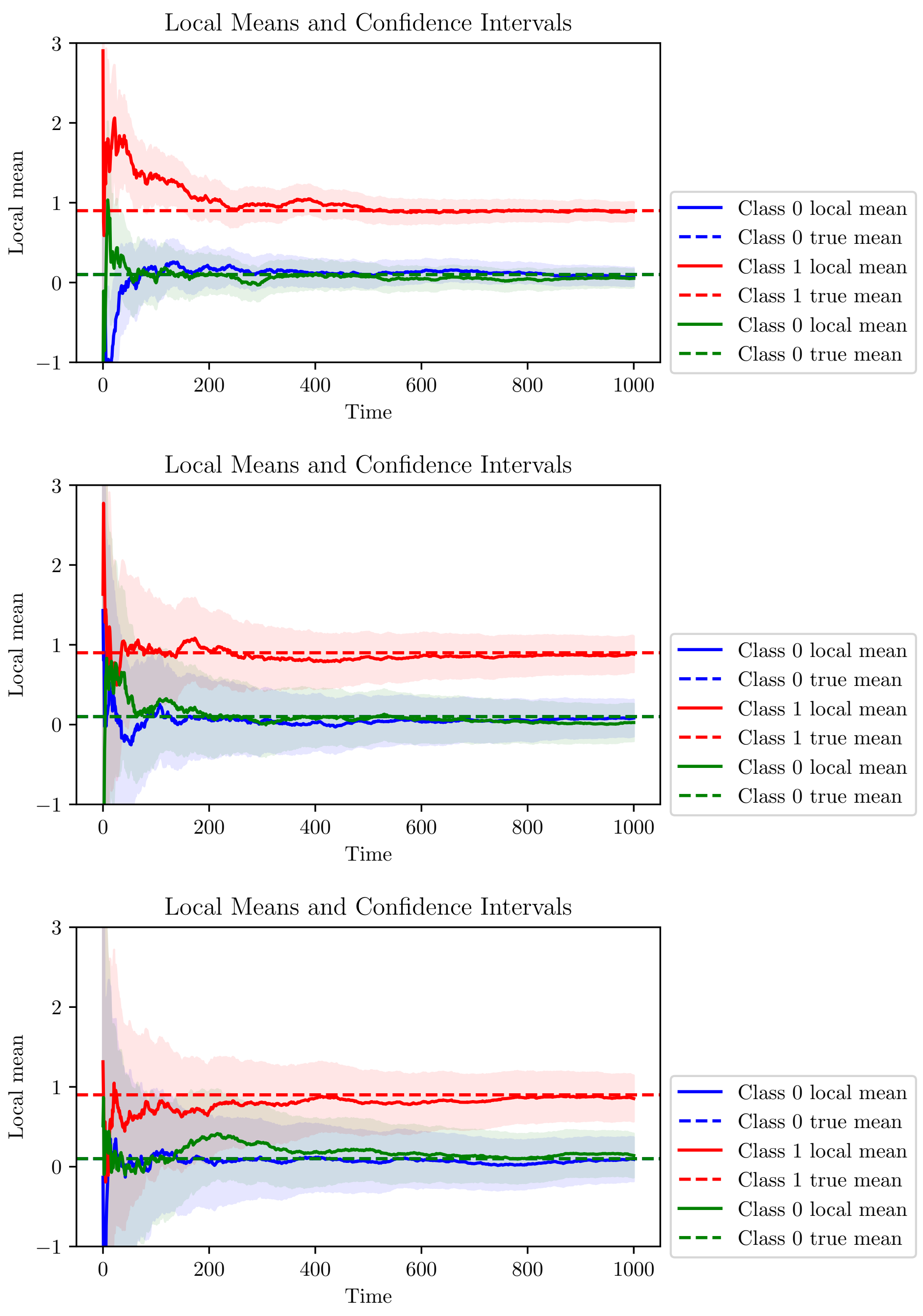


Figure 1.10: Local means and the corresponding confidence intervals for 3 agents belonging to two different classes, as a function of the number of samples t. (top): Underestimated variance for 50% with respect to the Laplace bound; (middle): Exactly estimated variance using Laplace bounds; (bottom): Overestimated variance for 25%.

The number of elements should also be accumulated as

$$N_t \leftarrow N_t + (|N_a(t)| + 1) \tag{1.39}$$

The value of such estimated variance would be

$$\hat{\sigma}_a(t) = \sqrt{\frac{s}{2N_t}} \tag{1.40}$$

This estimation would initially combine estimations of σ_1 and σ_2 assuming that they are equally probable, in the way

$$\hat{\sigma}_a^2(t) = \frac{1}{2}\Big(\hat{\sigma}_1^2(t) + \hat{\sigma}_2^2(t)\Big) \tag{1.41}$$

Assume that we can approximately write for the confidence interval bounds:

$$\sigma_i\sqrt{\frac{2}{t}\Big(1+\frac{1}{t}\Big)\ln(\sqrt{t+1}/(\delta/2))} + \sigma_j\sqrt{\frac{2}{t-1}\Big(1+\frac{1}{t-1}\Big)\ln(\sqrt{t}/(\delta/2))} \tag{1.42}$$

$$\approx (\sigma_i+\sigma_j)\sqrt{\frac{2}{t}\Big(1+\frac{1}{t}\Big)\ln(\sqrt{t+1}/(\delta/2))} \tag{1.43}$$

with $i, j \in \{1, 2\}$.

Possible cases for $(\sigma_i + \sigma_j)$ are $2\sigma_1$, $(\sigma_1 + \sigma_2)$ and $2\sigma_2$. However, if the probabilities of the agent classes are the same, we will initially use

$$2\hat{\sigma}_a(t) = \sqrt{2}\sqrt{\hat{\sigma}_1^2(t) + \hat{\sigma}_2^2(t)}$$

in the estimates of the confidence intervals for the three previous cases. Without loss of generality, assume that $\sigma_1 \le \sigma_2$. From the norms inequality we can write

$$2\sigma_1 \le (\sigma_1+\sigma_2) \le \sqrt{2}\sqrt{\hat{\sigma}_1^2(t) + \hat{\sigma}_2^2(t)} = 2\hat{\sigma}_a(t) \le 2\sigma_2 \tag{1.44}$$

This means that, for the class with lower variance and two agents from different classes, the confidence interval width bounds will initially be overestimated, since the used standard deviation will be greater than the true ones for these cases. Overestimated standard deviations will slow the convergence, as the expected separation time will be longer. At the same time, the confidence intervals will be underestimated for the class with higher variance, meaning that the probability of wrong disconnections will increase. This requires that *reconnection should be used* to avoid an increase in the number of isolated agents in this class.

Although the direct approach of previous accumulation in s and N_t, would later lead to the separation of classes (with the overestimation and underestimation effects described), it would also be significantly biased. That is, in initial estimations all classes overlap and the estimation of $\hat{\sigma}_a^2(t)$ for each class will be heavily loaded with other class components in the initial stages, and the process of forgetting would be extremely slow. To this end, we will estimate the sum s and the number of terms N_t with a **forgetting factor**, that is, as

$$s \leftarrow 0.975s + \sum_{a'\in\mathcal{N}_a\cup\{a\}} |x_{a'}(t) - x_{a'}(t-1)|^2 \tag{1.45}$$

$$N_t \leftarrow 0.975N_t + (|N_a(t)|+1) \tag{1.46}$$

It is important that both forgetting factors (here $\lambda = 0.975$) are the same, so there is no need to scale the variance estimation, that is, may still use $\hat{\sigma}_a(t) = \sqrt{\frac{s}{2N_t}}$.

Note: Application of the forgetting factor would be of crucial importance even in the case of equal variances if we can expect that the distribution of data (their means or variance) may change during the time horizon.

This process will be illustrated on an example with two classes $(0.1, 1)$ and $(0.9, 2)$. The histograms of the estimated standard deviations, for various instants t, are shown in Fig. 1.11

```
Tsigma=2000 # Instants until when the variance is estimated
Tstart=50 #Initial instants for estimated varinace used
if t<=Tstart: sigma_EST=10*np.ones((N, 1))  #  Initially large
if 1 < t <= Tsigma :
    # Calculate (x_a(t)-x_a(t-1)**2 from sums M
    sss = (M[:, t].reshape(-1, 1) -  M[:, t-1].reshape(-1, 1)-(M[:, t-1].reshape(-1,
    1) - M[:, t-2].reshape(-1, 1))) ** 2
    sk = 0.975 * sk + (A_current + I) @ sss # Incude neighbors and itself
    tk = 0.975 * tk + (A_current+I).sum(axis=1).astype(float).reshape(-1, 1)
if Tstart <= t <= Tsigma:
    sigma_EST = np.sqrt(sk/ 2 / tk )    #r neighboars and central node
```

Listing 1.1: Variance Estimation with Forgetting

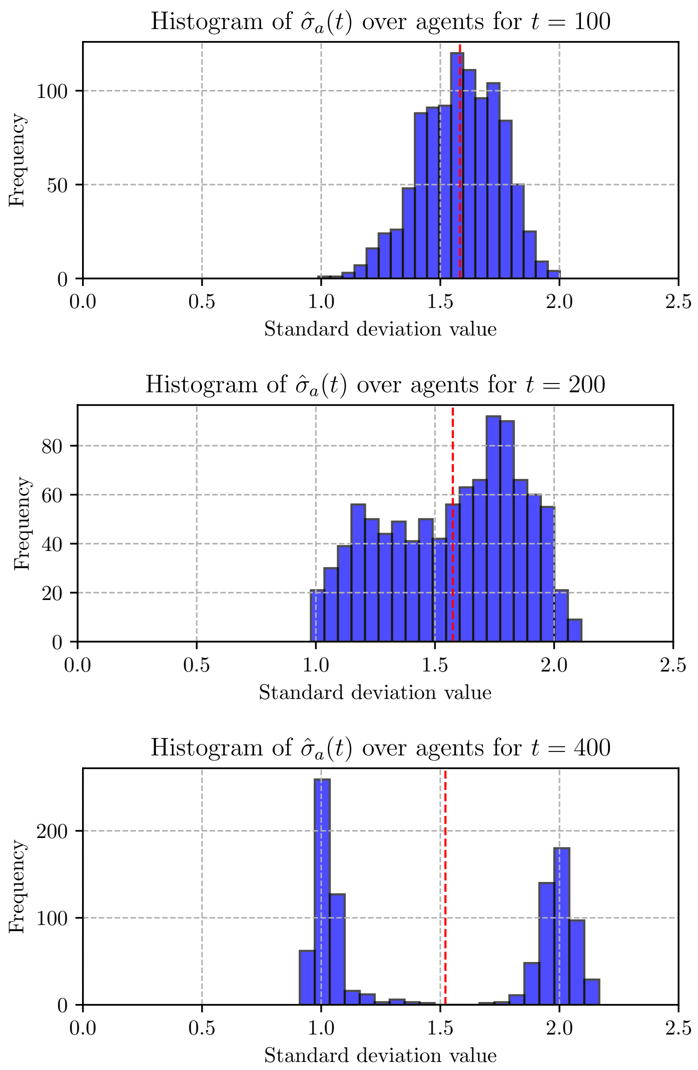


Figure 1.11: Histogram of the estimated standard deviations $\hat{\sigma}_a(t)$ over two-class agents with different means and standard deviations, $(0.1, 1)$ and $(0.9, 2)$, at $t = 100, 200, 400$ time instants.

1.2.5 Central Fourth Moment and Kurtosis Estimation

For confidence intervals based on the fourth-order moment we have to estimate the central fourth-order moment

$$\mu_4 = \mathbb{E}\{(\mathrm{x} - \mu_a)^4\} \tag{1.47}$$

Again, it will be done using the property that

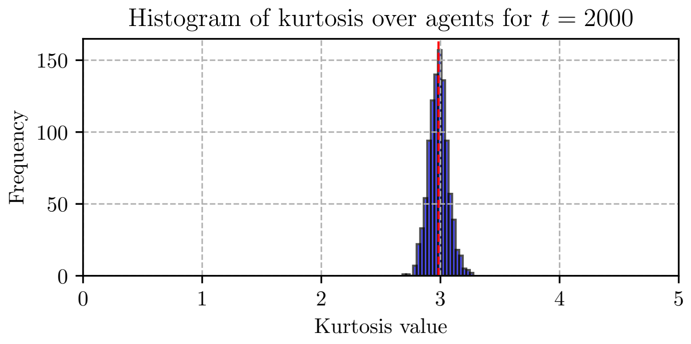
$$\begin{gathered}\mathbb{E}\{(x_a(t) - (x_a(t-1))^4\} = \mathbb{E}\{(x_a(t) - \mu_a - (x_a(t-1) - \mu_a))^4\} = \\ \mathbb{E}\{(x_a(t) - \mu_a)^4\} + 6\mathbb{E}\{(x_a(t) - \mu_a)^2\}\mathbb{E}\{(x_a(t-1) - \mu_a)^4\} + \mathbb{E}\{(x_a(t-1) - \mu_a)^4\} \\ = \mu_4 + 6\sigma^2\sigma^2 + \mu_4\end{gathered}$$

for symmetric distribution of data with the mean value not changing over time.

Therefore, the estimation can be done using:

$$\begin{gathered}\mu_4 = \frac{1}{2}\mathbb{E}\{(x_a(t) - (x_a(t-1))^4\} - 3\sigma^4 \\ \hat{\mu}_{4a} = \frac{1}{2T_s(|\mathcal{N}(t)| + 1} \sum_{\substack{a' \in \mathcal{N}_a(0) \cup \{a\} \\ \tau = 1, \dots, t}} \left(x_{a'}(\tau) - x_{a'}(\tau - 1)\right)^4 - 3\hat{\sigma}_a^4 \\ \hat{\kappa}_a = \frac{\hat{\mu}_{4a}}{\hat{\sigma}_a^4} = \frac{1}{2T_s(|\mathcal{N}(t)| + 1)\hat{\sigma}_a^4} \sum_{\substack{a' \in \mathcal{N}_a(0) \cup \{a\} \\ \tau = 1, \dots, t}} \left(x_{a'}(\tau) - x_{a'}(\tau - 1)\right)^4 - 3.\end{gathered}$$

For any Gaussian distribution, kurtosis is constant, $\kappa = 3$. The kurtosis in statistics is also used as a measure of the deviation of the distribution from the Gaussian distribution, using excess kurtosis $\kappa - 3$, which is always zero if the distribution is Gaussian. Distributions with high kurtosis have heavy tails. Kurtosis of a distribution smaller than 3 means lighter tails compared to a normal distribution.



If the considered data are Gaussian, there is no need to estimate the kurtosis. In that particular case, we may use the kurtosis estimation as a check of the approach used or the Gaussianity assumption about the data, Fig. 1.12. Kurtosis estimation for other distributions will be done later.

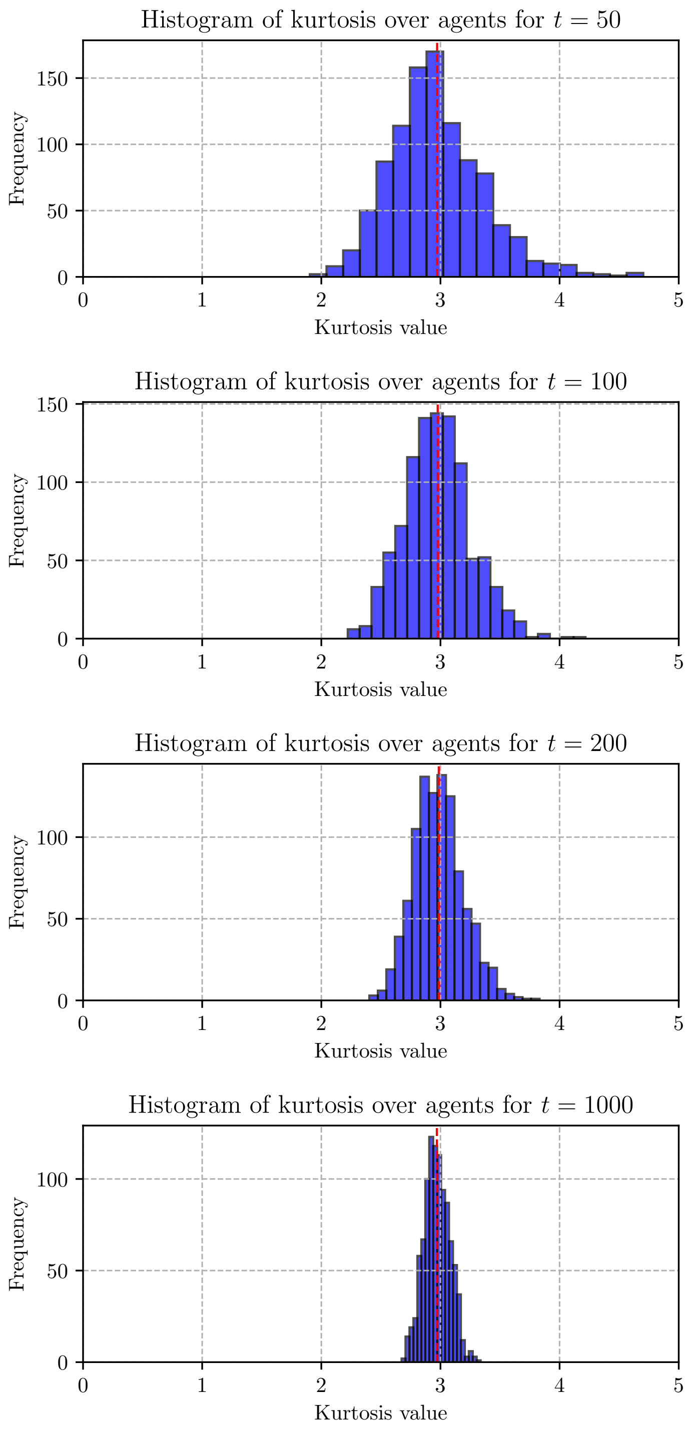


Figure 1.12: Histogram of kurtosis for the Gaussian distributed data.

1.3 colME Approach

The problem of estimating the mean within a scenario involving multiple agents is considered in [2]. Each agent in the group ($\mathcal{A} = \{1, 2, \dots, A\}$) is tasked with determining the mean μ_a of its distribution. It has been assumed that each distribution is σ-sub-Gaussian. The framework is both online and collaborative:

- Each agent receives data in sequence from its distribution.
- Each agent can interact with one agent to exchange data.

These agents function similarly to various user devices in a network that operate simultaneously, allowing each to collect a new sample and communicate with another agent at each time instance.

In a more formal sense, we assume synchronized timings between agents, where at each discrete time, $t = 0, 1, 2, \dots, T$, an agent a:

- Obtains a new sample $x_a(t)$ from its distribution with mean μ_a, and updates its local mean estimate, given by $\bar{x}_{aa}(t)$.
- The agent then selects another agent a' for querying. In return for its query, agent a receives the local mean $\bar{x}^t_{a'a'}$ from agent a', representing the aggregated information of a' of its independent samples, and records it in its own dataset as $\bar{x}_{a,a'}(t)$, in conjunction with the sample count $n_{a,a'}(t) = t$. The agent a' can also send the aggregated sum of its local samples $m^t_{a'a'}$ stored as $m^t_{aa'}$, together with $n_{a,a'}(t) = t$, with $\bar{x}_{a,a'}(t) = m_{a,a'}(t)/n_{a,a'}(t)$ or $m_{a,a'}(t) = n_{a,a'}(t)\bar{x}_{a,a'}(t)$.
- Each agent keeps a log of all recent local averages,
$$[\bar{x}_{a,1}(t), n_{a,1}(t), \dots, \bar{x}_{a,A}(t), n_{a,A}(t)] \tag{1.48}$$
obtained from other agents, including its own local average. This information is used to update the mean estimate of each agent at time t,
$$\mu_a(t) = \frac{\sum_{a'} m_{a,a'}(t)}{\sum_{a'} n_{a,a'}(t)} = \frac{n_{a,a'}(t) \sum_{a'} \bar{x}_{a,a'}(t)}{\sum_{a'} n_{a,a'}(t)} \tag{1.49}$$

When a query occurs, instead of acquiring just one sample, as in multiarmed bandit problems, the querying agent gains comprehensive insight into accumulated observations up to the current time. This method yields more extensive data than conventional bandit settings.

In practice, each agent a assumes that its initial neighborhood $\mathcal{N}a(0)$ is the whole set of other agents

$$\mathcal{N}_0(0) = [1, 2, 3, \dots, A], \;\; \mathcal{N}_1(0) = [2, 3, 4, \dots, A, 0], \;\; \dots, \;\; \mathcal{N}_A(0) = [0, 1, 2, \dots, A-1]$$

At instant $t = 0$ agent a visits agent $a + 1$ (the first element in the list), exchanges data, and decides if the visited agent is within the same similarity class or not (checking the confidence interval). If it is, then the neighborhood list is just roll-rotated left keeping all agents in the list, and the list

$$[\bar{x}_{a,1}(t), n_{a,1}(t), \ldots, (\bar{x}_{a,A}(t), n_{a,A}(t)]) \tag{1.50}$$

is updated with a new mean and number of terms. If it is not, then the first agent in the neighborhood list is deleted, and the remaining list is kept as it is (*Restricted-Round-Robin* approach) with the corresponding data being deleted (or set to zero) in the data list.

Algorithm 2 Mean Estimation using colME

```
1: Input: $\delta$, $\epsilon$, Distribution $D_a$,
2: Initialize ordered neighborhood $\mathcal{N}_a$ for all nodes $a$
3: Initialize exchanged local means and times list $(\bar{x}_{aa'}(0), n_{aa'}(0))$ for all $a$ and $a'$
4: for $t = 1$ to $T$ do
5:     for all nodes $a \in \mathcal{A}$ in parallel do
6:         Draw $x_a(t)$ from the data distribution
7:         Calculate local mean $\bar{x}_{aa}(t)$
8:         Calculate confidence intervals for $a$ and the first agent $a' = \mathcal{N}_a[0]$ using $\bar{x}_{aa}(t)$ and $\bar{x}_{a'a'}(t-1)$
9:         if confidence intervals for $a$ and $a'$ do not intersect then
10:            $\mathcal{N}_a[0] = [\ ]$, $\bar{x}_{aa'}(t) = 0$, and $n_{aa'}(t) = 0$
11:        else
12:            Roll-rotate left $\mathcal{N}_a$
13:            Update list of local means and times in $a$ by $\bar{x}_{aa'}(t)$ and $n_{aa'}(t) = t$.
14:        end if
15:        Output: Collaborated mean for agent $a$, calculated as a sum over all $a'$ of $n_{aa'}(t)\bar{x}_{aa'}(t)$ divided by the sum of $n_{aa'}(t)$.
16:    end for
17: end for
```

1.3.1 Numerical Example for colME

We simulated in Python the entire system with $A = N = 1000$ agents using the colME approach. A system with two similarity classes is assumed, $C = 2$, with means $\mu_1 = 0.1$ and $\mu_2 = 0.9$, and standard deviation $\sigma = 2$. The value of $\delta = 0.1$ is used. We compared the estimated agent means with the exact mean values, using $\varepsilon = 0.1$ as the error bound.

The variance is estimated in the initial step $t = 0$ and $t = 1$ (broadcasting approach) and used in the calculations of the confidence intervals. The estimate of σ is 2.031.

The colME algorithm is used to estimate the agent mean values. Based on the intersections of the confidence intervals, similarity classes are found over time, as illustrated in Fig. 1.13, for some time instants t, and used for collaborative mean value estimation in the described way. We can see that at the beginning the agents have a connection with all other agents, Fig. 1.13(Top). Over time, the agents whose intervals do not intersect are disconnected, as can be seen on Fig. 1.13(Middle panels), finally resulting in two separated components over the classes, Fig. 1.13(Bottom).

The estimated mean values are used to calculate the error for each time instant and agent. The errors over all agents are averaged for each time instant t and shown in Fig. 1.14 by the blue line. We used the estimate values to find the confidence intervals by the bootstrap method and shown this interval along with the mean squared error line.

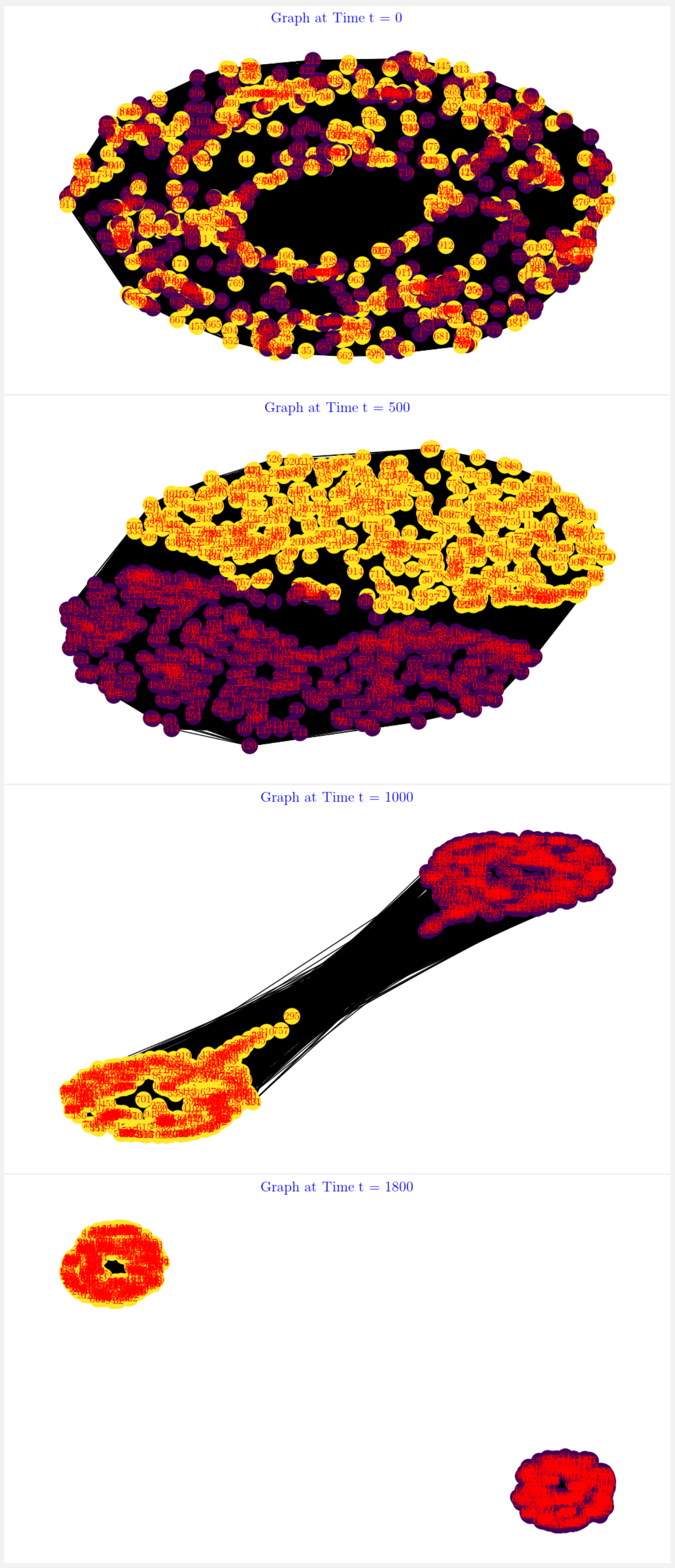


Figure 1.13: Connectivity graph for colME with the estimated agent classes, at $t = 0$, $t = 500$, $t = 1000$, and $t = 1800$, respectively.

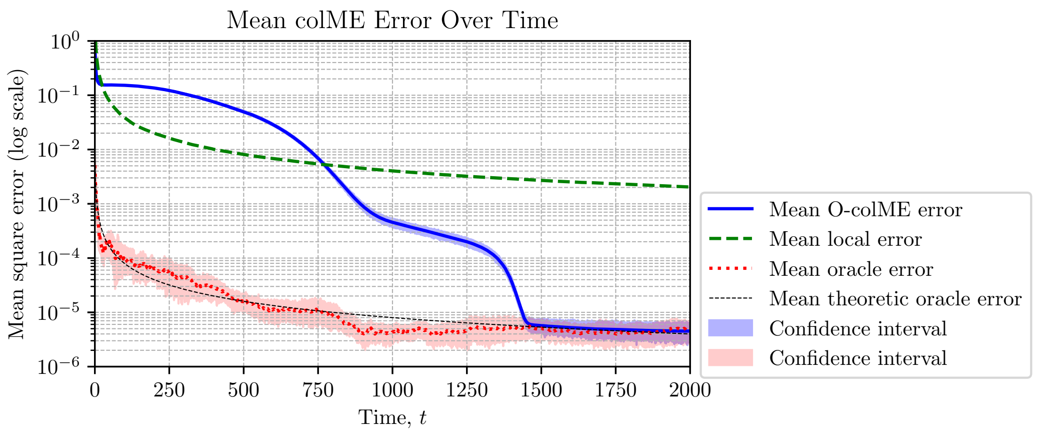


Figure 1.14: colME with $N = 1,000$: Total MSE of the estimation, averaged over all agents at time instants t. The results are averaged over 10 realizations. Confidence intervals for the MSE are calculated using Bootstrap with 0.95 confidence. **Calculation time** on MacBook Air with 8-core M3 CPU and 16 GB RAM is 217 seconds per realization, in average.

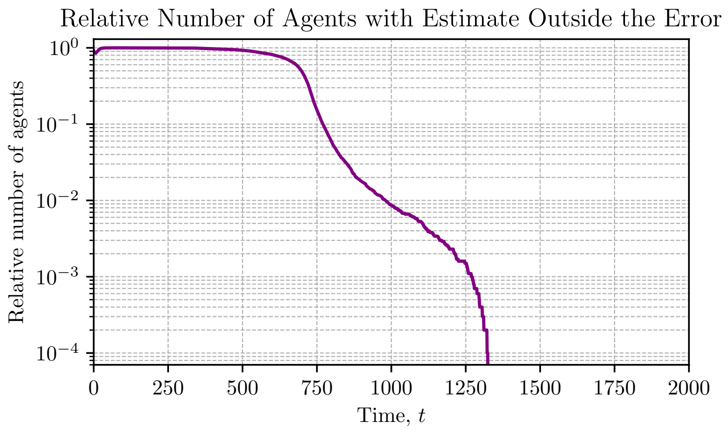


Figure 1.15: colME with $N = 1,000$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 10 realizations.

The results of estimated means by using colME are compared with the *local mean values*, when no collaboration is performed (green dashed line in Fig. 1.14. The *oracle estimated values*, if the similarity classes were known in advance, are also calculated. In the oracle case, it has been assumed that the agents communicate within the whole system, without any restriction, with the known oracle adjacency matrix. The oracle results are shown by the dotted red line along with their confidence intervals, Figs. 1.14.

We can see that the estimation results by the colME initially, up to around $t = 750$ are worse than the local results without any collaboration. This is the result of using wrong collaborative agents. As the collaborative set estimation improves, this result also is improving and finally at about $t = 1500$, it is very close to the oracle solution since the similarity sets are by then completely separated.

We have also calculated the relative number of agents whose estimated absolute error is above the given $\varepsilon = 0.1$. The results are shown in Fig 1.15. We can see that initially all agents(relative number 1) are with error above the threshold. The number of agents with the error falling outside the given threshold reduces starting from $t = 750$ and their number is very low after $t = 1300$. This result coincides with the result in the previous figure describing the mean squared error.

Comment on colME: We can conclude that the colMe approach uses means of agents until it is again revisited, after N time steps. If N is large and there are some wrong agents in the estimated similarity class, then these agents will be re-checked for similarity class (by the intersection of confidence intervals) only after log time, equal to N time steps. This fact leads to slower convergence of the collaborative mean estimation, in addition to a very high memory demand, since all N local means and time instants should be kept at each agent.

Chapter 2

Scalable Systems with Graphs

In large-scale systems, having agents query all other agents or maintain a memory that scales linearly with the number of agents A can be infeasible. Practically, each agent can instead focus on a limited subset of agents that is manageable in size, which will be done using a graph as the domain next.

The random graph (agents with assumed links) is denoted as $\mathcal{G}(\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the set of nodes/agents and $\mathcal{E}$ are the edges, indicating the communication links between the nodes. In particular, the initial graph will be a random sample of a random graph $\mathcal{G}(A, r)$ with exactly r edges from each node at $t = 0$. The selection of r is very important and has been discussed in detail in [3], Fig. 2.1.

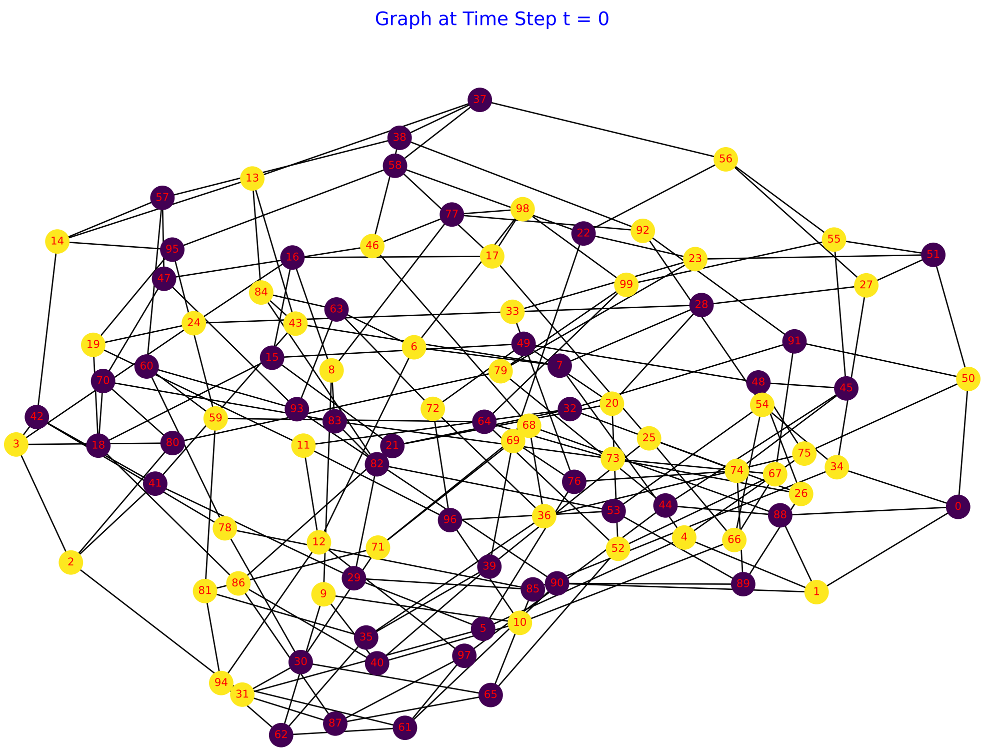


Figure 2.1: Random regular graph with $A = 100$ agents/nodes and classes, arbitrary connected, at $t = 0$, with $r = 4$ links using a random regular graph $\mathcal{G}(100, 4)$.

Since we do not now know anything about the class, many of the connections are wrong connections, meaning that agents from different classes established communication links. The task is as follows.

- To eliminate wrong links over time and to leave only the links among the same class agents.
- Define a good formula that will use the means and other data from neighbors to improve the estimation.

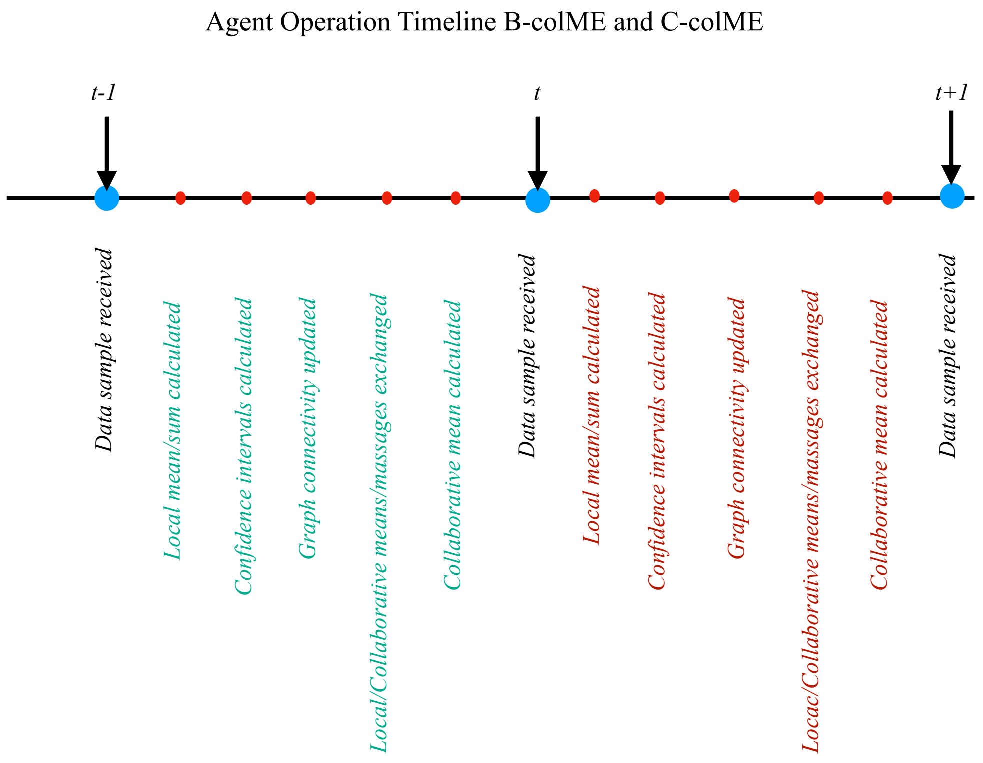

Agent Operation Timeline B-colME and C-colME
t-1
t
t+1
Data sample received
Local mean/sum calculated
Confidence intervals calculated
Graph connectivity updated
Local/Collaborative means/massages exchanged
Collaborative mean calculated
Data sample received
Local mean/sum calculated
Confidence intervals calculated
Graph connectivity updated
Locac/Collaborative means/massages exchanged
Collaborative mean calculated
Data sample received

Figure 2.2: Timeline for the graph-based collaborative mean calculation.

2.1 Consensus-based Approach C-colME

The consensus-based approach (C-colME) uses both the local means $\bar{x}_{a,a}(t) = \frac{1}{t}\sum_{\tau=1}^{t} x_a(\tau)$, or in vector form, for all agents $\mathbf{X}(t)$, and collaborative means calculated using the estimated similarity class, within the random neighborhood $\boldsymbol{\mu}(t)$. Note that if the local sums are calculated in the code, then the local mean vector is $\mathbf{X}(t) = \mathbf{M}(t)/t$.

We start with $\boldsymbol{\mu}(0) = \mathbf{X}(0)$ and calculate all next consensus values combining local means $\mathbf{X}(t)$, with weight $(1-\alpha(t))$ and previous consensus mean $\boldsymbol{\mu}(t-1)$ with weight $\alpha(t)$, that is

$$\boldsymbol{\mu}(t+1) = (1-\alpha(t))\mathbf{X}(t) + \alpha(t)\mathbf{W}(t)\boldsymbol{\mu}(t) \tag{2.1}$$

where $\mathbf{W}(t)$ is the weight matrix that will be discussed next.

- In order to obtain an unbiased result, we must obviously have the sum of the rows of the matrix $\mathbf{W}_t$ to be one. If we assume that all means are equal to 1, then $\mathbf{1} = (1-\alpha(t))\mathbf{1} + \alpha(t)\mathbf{W}(t)\mathbf{1}$, should hold, which means that all rows of $\mathbf{W}$ should sum to 1. Since this matrix is symmetric, this holds for columns as well, and such a matrix is called a double stochastic matrix.

- At the same time, the matrix $\mathbf{W}(t)$ should play the role of the adjacency matrix $\mathbf{A}(t)$ (weighted adjacency matrix) since it should combine the values of $\boldsymbol{\mu}(t-1)$ within the neighborhood of a given node only, hoping that this neighborhood would be a part of the similarity class (after some time steps).

- We will assume that the values of matrix $\mathbf{W}(t)$ are constant for each node, and take their value as

$$W_{i,j}(t) = W_{j,i}(t) = \frac{1}{\max\{D_i, D_j\}+1} \text{ for } A_{i,j}(t) = 1 \tag{2.2}$$

  and $W_{i,j}(t) = 0$ for $A_{i,j} = 0$. The values $D(i)$ are degrees of matrix $\mathbf{A}(t)$, that is $D(i) = \sum_j A_{i,j}(t)$

- In order to achiever that $\sum_j W_{i,j}(t) = 1$ we have

$$W_{i,i}(t) = 1 - \sum_j W_{i,j}(t) \tag{2.3}$$

Algorithm 3 Mean Estimation using C-colME with Variance Estimation

```
1: **Input:** Graph $G(\mathcal{A}, \mathcal{E})$, with matrix $\mathbf{A}(0)$, $\delta$, $\epsilon$, Distribution $D_a$,
2: Initialize neighborhood $\mathcal{N}_a(0)$ for all nodes $a \in \mathcal{A}$, $s = 0$, initial $\hat{\sigma}(a) = 10$
3: **for** $t = 1$ to $T$ **do**
4:     **for all** nodes $a \in \mathcal{A}$ **in parallel do**
5:         Draw $x_a(t)$ from the data distribution
6:         Calculate local mean $\bar{x}_{aa}(t)$
7:         Calculate confidence intervals for all $a' \in \mathcal{N}_a(t)$ using $\bar{x}_{aa}(t)$ and $\bar{x}_{a'a'}(t-1)$
8:         **if** $t \leq T_s$ **then**
9:             Update $s \leftarrow s + (x_{a'}(t) - x_{a'}(t-1))^2$ for $a' \in \mathcal{N}_a(t) \cup \{a\}$
10:        **else if** $t == T_s$ **then**
11:            Standard deviation $\hat{\sigma}(a) = \sqrt{\frac{s}{T_s(\mathcal{N}_a+1)}}$
12:        **end if**
13:        **if** confidence intervals for $a$ and $a'$ do not intersect **then**
14:            $\mathcal{N}_a(t) = \mathcal{N}_a(t) \setminus \{a'\}$, that is,
15:            Update the graph adjacency matrix, $A_{aa'}(t) = A_{a'a}(t) = 0$
16:            Calculate double stochastic matrix, $\mathbf{W}(t)$, $W(a, a') = W(a', a)$.
17:        **end if**
18:        Exchange local sums/collaborative means/messages with neighbors
19:        **Output:** Collaborated mean for agent $a$ according to
20:        $\boldsymbol{\mu}(t+1) = (1 - \alpha(t))\mathbf{X}(t) + \alpha(t)\mathbf{W}(t)\boldsymbol{\mu}(t)$
21:    **end for**
22: **end for**
```

Finally, let us define and discuss α_t. In consensus-based methods, it is common to take

$$\alpha(t) = t/(1+t) \tag{2.4}$$

This value starts from 0 and promotes the local mean only at $t = 0$, (since noting to combine). It should remain low for small values of t, as the estimated similarity set is highly unreliable. Later, it should promote collaborative means, being the second part of this adaptation process.

The first several values of $\alpha(t)$ are 0, 1/2, 2/3, 3/4, 4/5, 5/6,... and they converge to 1 very fast, within a few samples, which is in our opinion not good. In early instants, we should keep the local mean much longer, as a better estimate.

To this end, we have tried

$$\alpha(t) = \frac{t}{15+t} \tag{2.5}$$

This value significantly improved the convergence of the C-colME algorithm.

2.1.1 Numerical Example for C-colME

We simulated in Python the entire system with $A = N = 1000$ agents using the C-colME approach. A two-class system is assumed, $C = 2$ with means $\mu_1 = 0.1$ and $\mu_2 = 0.9$, and variance $\sigma = 2$. A random regular graph is formed with $r = 12$. The value of $\delta = 0.1$ is used. We compared the results obtained with the exact one using the error bound of $\varepsilon = 0.1$.

The variance is estimated for each agent individually using Algorithm 1 as in Fig. 1.8

The C-colME algorithm is used, along with the local estimation, when no collaboration is done, and with the oracle approach, when similarity classes are known in advance and the agents from the beginning communicate within the whole system, without any restriction, known oracle adjacency matrix.

The results are shown in Figs. 2.3, 2.4, 2.5, and 2.6.

We repeated all simulations for $\alpha(t) = 0$ for $t \leq T_s$ (to estimate variance) and then $\alpha(t) = t/(15+t)$. The results are shown in Figs. 2.7, 2.8, and 2.9. They are almost the same as in the previous case.

Finally, the bootstrap is used to estimate the confidence intervals of the MSE. This is a computationally intensive method for statistical inference that *avoids strong assumptions about the MSE distribution.* It generates many resampled datasets (in our case MSE values) by sampling with replacement of the original MSE data. These resampled datasets allow for the estimation of the MSE in many bootstrap simulated scenarios and to estimate the confidence intervals for the MSE.

Fig. 2.3illustrates the evolution of the C-colME model with $N = 1{,}000$ agents. At the initial stage ($t = 0$), the network is generated as a random connected graph based on the estimated agent classes. As time progresses, the connections are iteratively updated according to the confidence intervals that govern interaction dynamics. The snapshots presented at $t = 300$, $t = 500$, and $t = 900$ (from top to bottom) show how the structure of the network gradually adapts, reflecting the influence of these intervals on the formation and reorganization of links among agents.

Figures 2.4–2.9 summarize the performance of the C-colME model with $N = 1{,}000$ agents. The results are averaged over 10 realizations and, where relevant, confidence intervals are provided.

Figure 2.4 presents the total mean squared error (MSE) of the estimation across time, together with expected theoretical values, illustrating the accuracy of the method. Figure 2.5 shows the evolution of the total number of wrong links, i.e., connections between agents of different classes. Figure 2.6 reports the number of agents whose estimation error exceeds a tolerance threshold $\varepsilon = 0.1$.

The effect of setting $\alpha(t) = 0$ for $t \leq T_s = 50$ is highlighted in Figures 2.7–2.9. Here, the same performance metrics are displayed (MSE, number of wrong links, and number of agents with error greater than ε), allowing a comparison with the baseline case. Together, these figures demonstrate how the estimation accuracy and link dynamics evolve over time, and how the choice of $\alpha(t)$ impacts the overall behavior of the system.

2.1.2 Number of Agents and Calculation Time in C-colME

Next, we will simulate a C-colME system with different number of agents: $N = 5,000$ and $N = 10,000$.

The results are shown in Figs. 2.10, 2.11, and 2.12 for $N = 5,000$.

For $N = 10,000$, the results are shown in Figs. 2.13, 2.14, and 2.15.

The average calculation time will be given in the captions for a MacBook Air with 8-core M3 CPU and 16 GB RAM.

A table with all calculation times is given at the end of the simulations.

The resulting MSE for all three cases considered $N = 1,000$, $N = 5,000$, and $N = 10,000$ are summarized in Fig. 2.16. We can conclude that up to $t = 1000$ all three cases behave almost the same. As expected, the smallest MSE is achieved by $N = 10,000$ as the collaboration set is the largest. However, all results also depend on local means that are stopped updating in the C-colME after α is close to 1, while in the oracle they are updated all the time. Also, in C-colME there is the possibility of isolated agents that can degrade the overall MSE of the system.

Comment on C-colME:

The convergence behavior of the algorithm remains consistent for large values of time t, but a significant reduction in the steady-state Mean Squared Error (MSE) is observed. This lower MSE value converges towards the theoretical oracle case, indicating a more accurate and stable estimation.

The reduction in the final MSE, when $N = 1000$, is a direct consequence of performing the averaging over a larger number of agents. As expected, increasing the number of agents by a factor of five, results in a proportional decrease in the variance of the estimation error, reducing it by approximately a factor of five.

The results for the initial phase, up to $t = 50$, show similar behavior for both the fully collaborative case ($\alpha = 0$) and the local-only processing case. Setting $\alpha = 0$ means that only local data is used, with no contribution from neighboring agents. As seen in Figure 2.7, the results for this time period fully coincides with the mean local error (green dashed line). This does not degrade the algorithm's performance, as the local mean provides a more accurate estimate than the collaborative one during the early stages ($t < 50$).

To compare the results across different configurations, the data was normalized and plotted on the same scale. This comparison revealed that the overall transient behavior of the algorithm is similar, with only the final stationary value of the MSE being lowered. The observed performance demonstrates that the algorithm exhibits linear scalability, which is a crucial advantage for handling larger datasets.

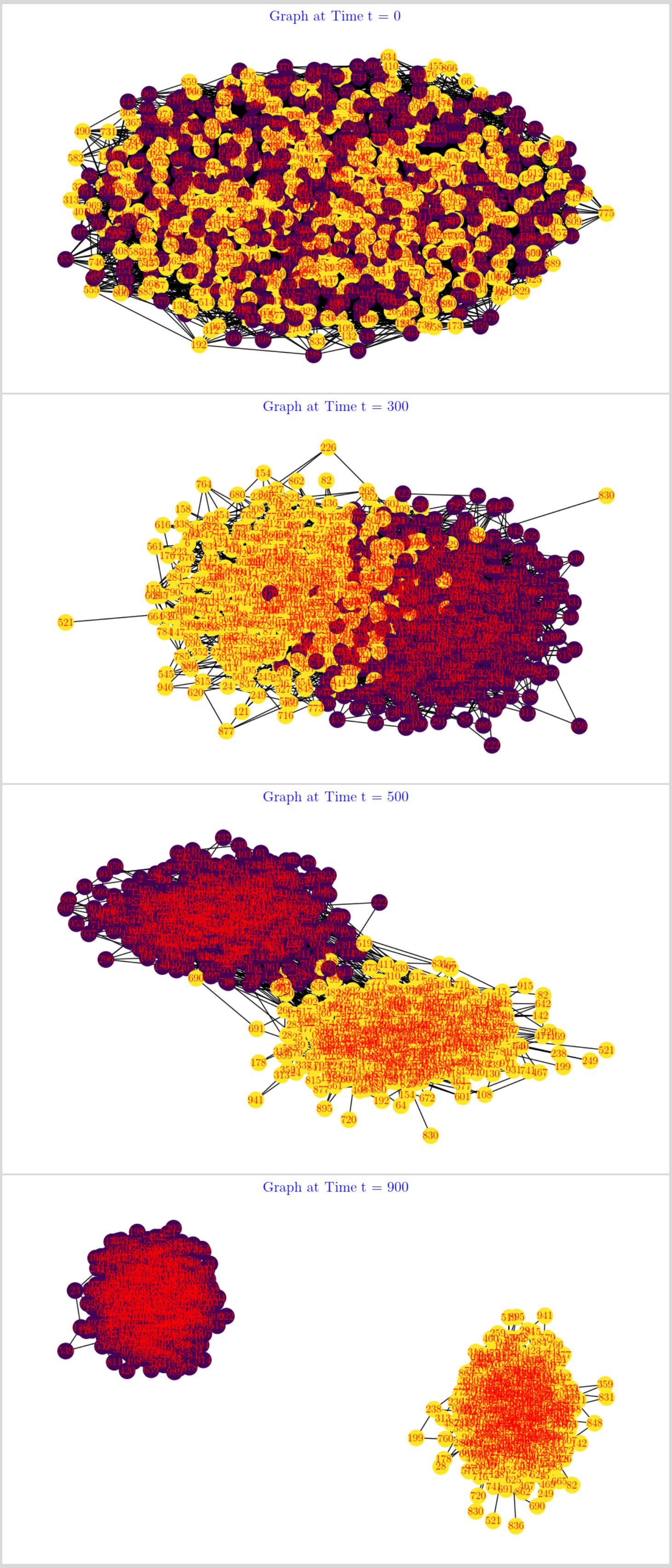


Figure 2.3: C-colME with $N = 1,000$: Random graph with the estimated agent classes, arbitrary connected at $t = 0$, and then the connections adjusted using the confidence intervals over time. Graphs at $t = 300$, $t = 500$, and $t = 900$, respectively, are shown from top to bottom.

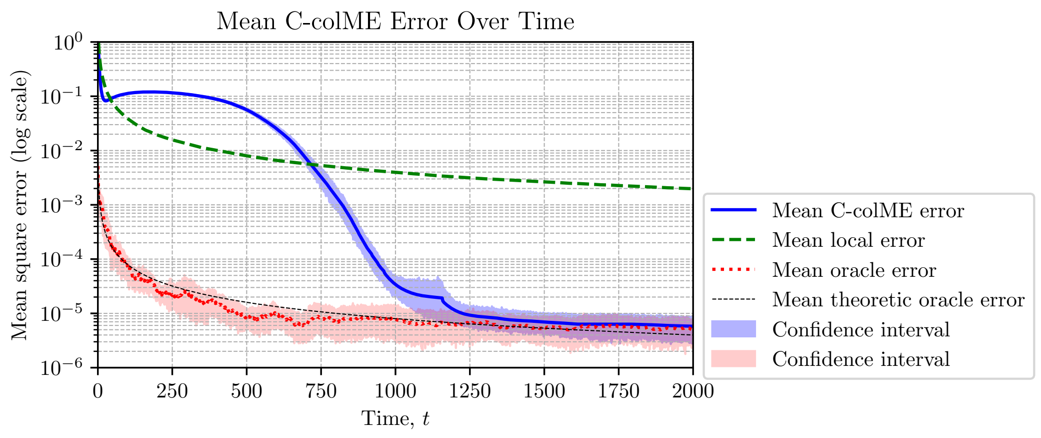


Figure 2.4: C-colME with $N = 1,000$: Total MSE of the estimation (estimated values minus exact values, squared) averaged over all agents at time instants t. The results are averaged over 10 realizations. Confidence intervals for the MSE are calculated using Bootstrap with 0.95 confidence. *Expected theoretic values are given as well.* **Calculation time** is 143 seconds per realization, on average.

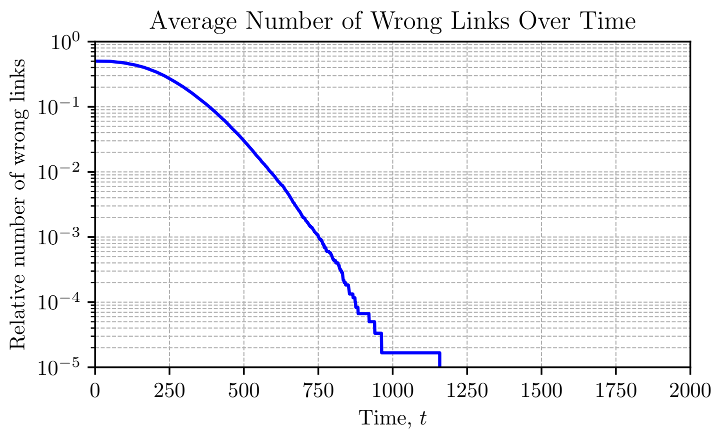


Figure 2.5: C-colME with $N = 1,000$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 10 realizations.

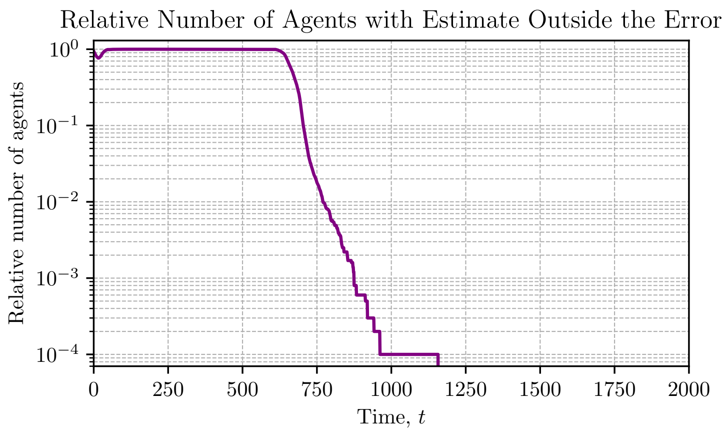


Figure 2.6: C-colME with $N = 1,000$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time (averaged over 10 realizations).

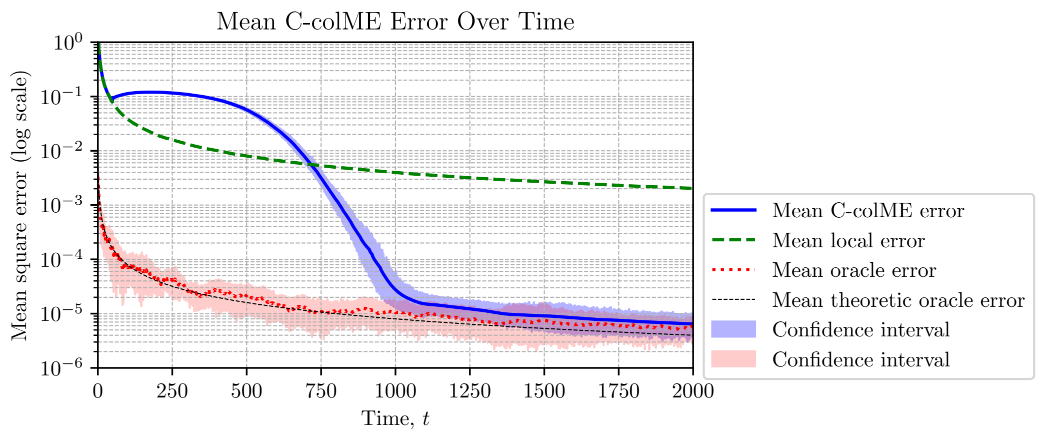


Figure 2.7: C-colME with $N = 1,000$, $\alpha(t) = 0$ for $t \leq T_s = 50$: Total MSE of the estimation averaged over all agents at time instants t. The results are averaged over 10 realizations, $\alpha(t) = 0$ for $t \leq T_s = 50$.

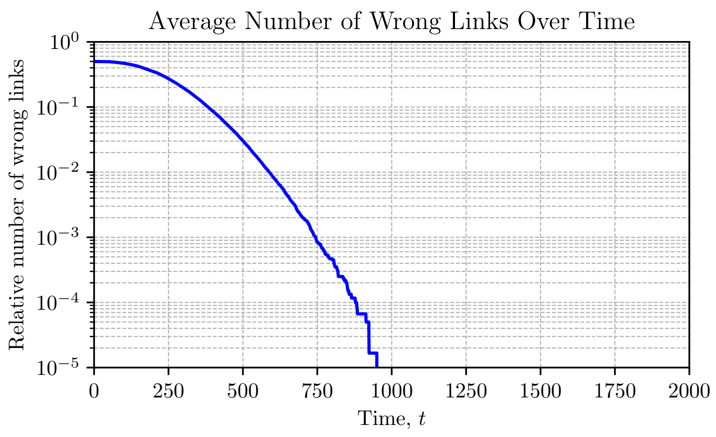


Figure 2.8: C-colME with $N = 1,000$, $\alpha(t) = 0$ for $t \leq T_s = 50$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 10 realizations, $\alpha(t) = 0$ for $t \leq T_s$.

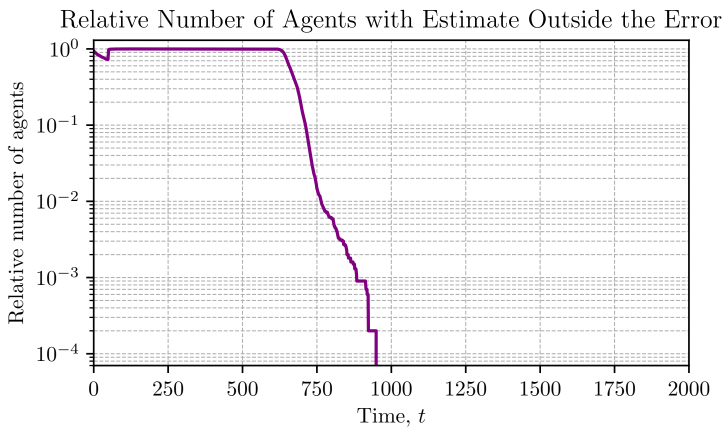


Figure 2.9: C-colME with $N = 1,000$, $\alpha(t) = 0$ for $t \leq T_s = 50$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time (averaged over 10 realizations), $\alpha(t) = 0$ for $t \leq T_s$.

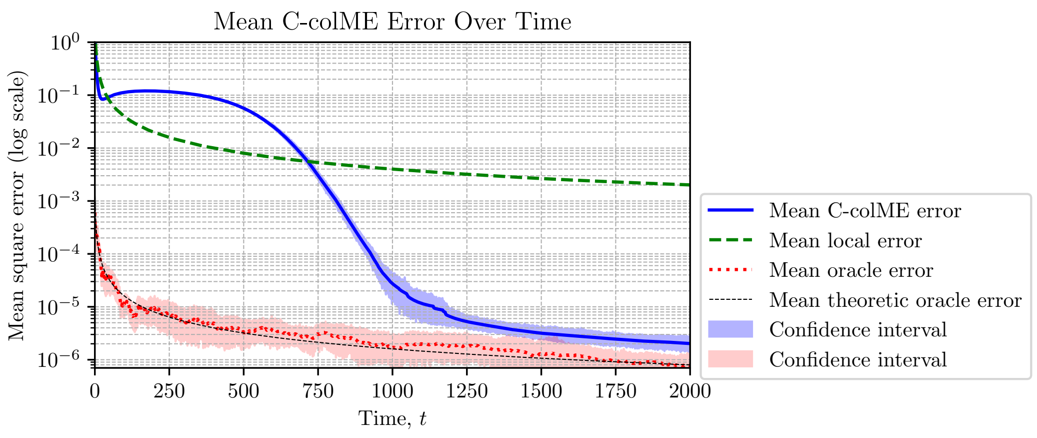


Figure 2.10: C-colME with $N = 5,000$: Total MSE of the estimation (estimated values minus exact values, squared) averaged over all agents at time instants t. The results are averaged over 10 realizations. *Expected theoretic values are given as well.* **Calculation time** is 846 seconds per realization, on average.

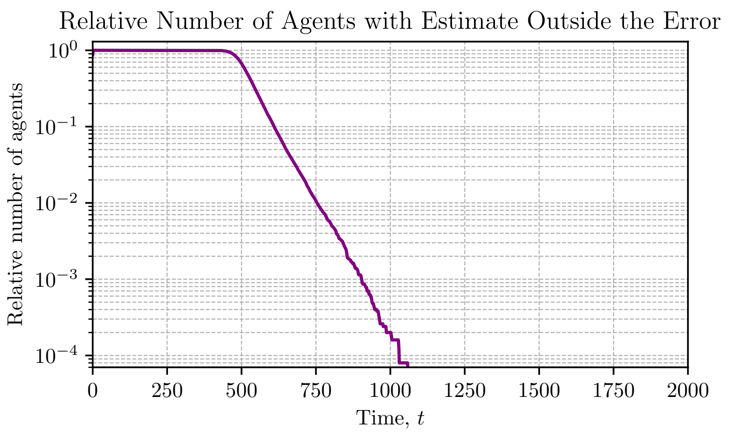


Figure 2.11: C-colME, $N = 5,000$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 10 realizations.

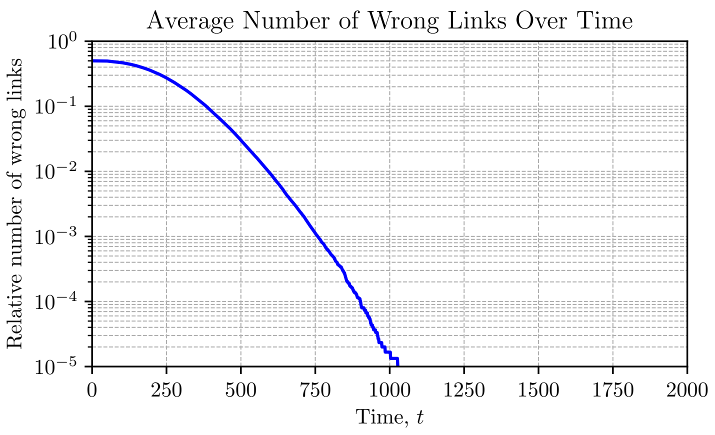


Figure 2.12: C-colME, $N = 5,000$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 10 realizations.

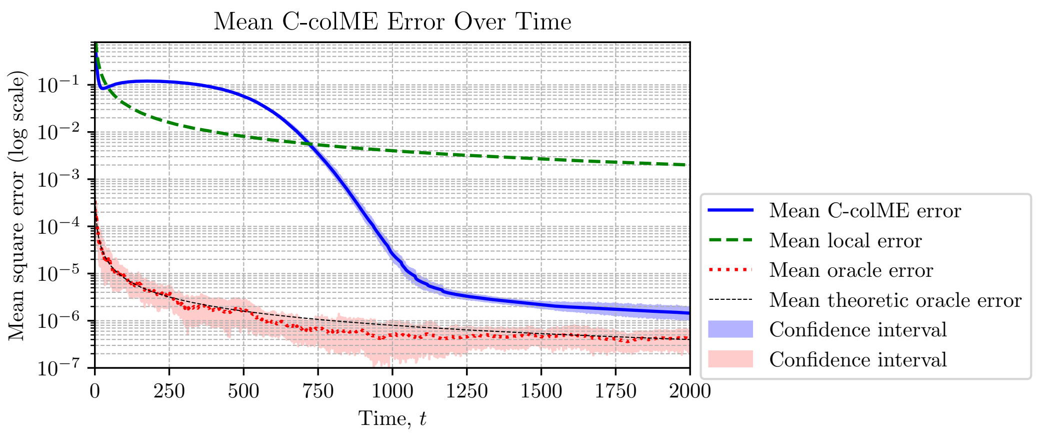


Figure 2.13: C-colME with $N = 10,000$: Total MSE of the estimation (estimated values minus exact values, squared) averaged over all agents at time instants t. The results are averaged over 5 realizations. *Expected theoretic values are given as well.* **Calculation time** is 2117 seconds per realization, on average.

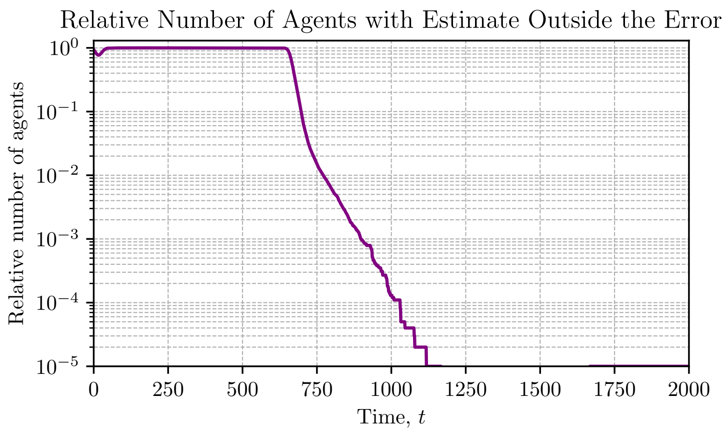


Figure 2.14: C-colME, $N = 10,000$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 3 realizations.

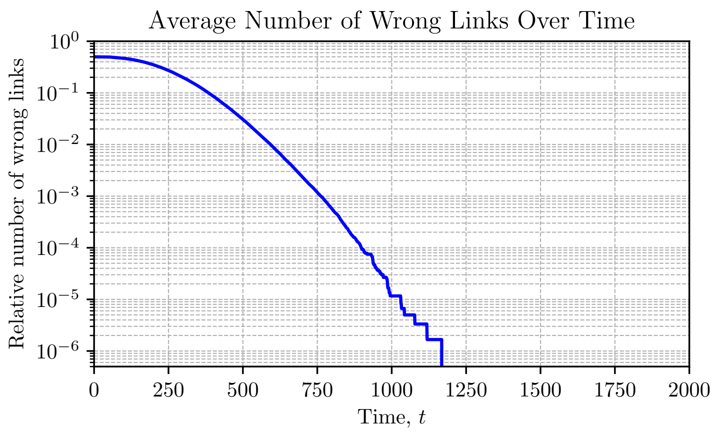


Figure 2.15: C-colME, $N = 10,000$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 3 realizations.

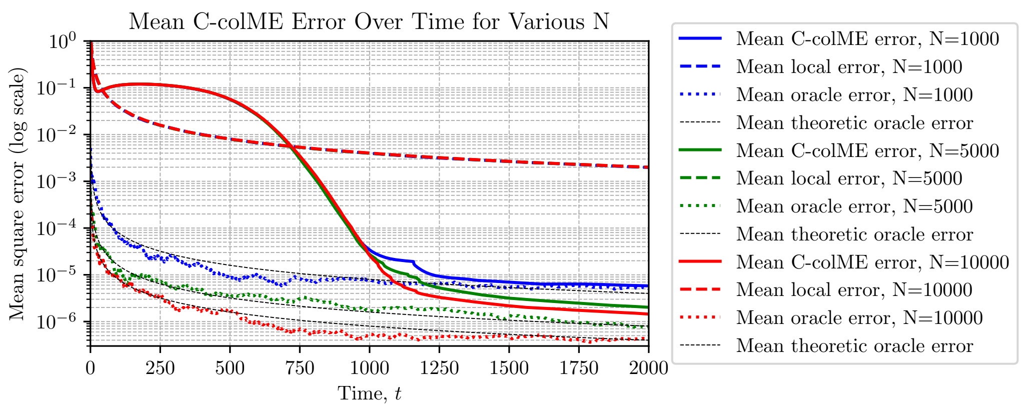


Figure 2.16: C-colME with various N: Overview of the total MSE of the estimation averaged over 10 realizations for various $N = 1000$, 5000, and 10000 (5 realizations). *Expected theoretic values are given as well.*

2.1.3 Simplified C-colME with Row Stochastic (Random-Walk) Matrix

We have implemented the C-colME with a row stochastic matrix $\mathbf{W}$ whose rows sum to 1. In this case, the collaborative mean calculation is simplified to

$$\mu_a(t+1) = (1-\alpha(t))x_{aa}(t) + \alpha(t)\text{mean}_{\mathcal{N}_a(t)\cup a}\{\mu_a(t)\} \tag{2.6}$$

The results we obtained are almost the same as in the C-colME case with double stochastic $\mathbf{W}$ that we have included just the MSE plot for $N = 5000$ in the text, Fig. 2.17.

The calculation time is shorter for about 15 % than in the case of a C-colME with double stochastic $\mathbf{W}$. For the case of a system with $N = 5000$ agents, it is about 733 seconds, for one realization. A detailed comparison of the calculation time for different methods is given in Table 2.1.

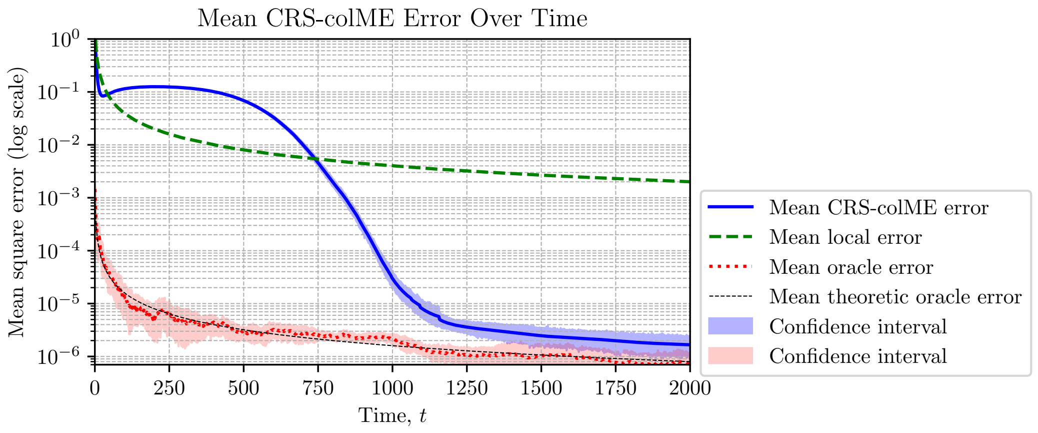


Figure 2.17: Simplified Right Stochastic C-colME with $N = 5,000$: Total MSE of the estimation The results are averaged over 10 realizations. **Calculation time** is 733 seconds per realization, on average.

Figure 2.17 shows the total mean squared error (MSE) of the estimation for the simplified C-colME model with a row-stochastic (random-walk) matrix and $N = 5,000$ agents, averaged over 10 realizations. The results confirm that the estimation accuracy closely matches the case with a doubly stochastic matrix, while the computational time is reduced by approximately. This can be seen by looking at the blue line on the graph which behaves very similar to the standard C-colME case, but does intersect with the green mean local error dotted line at 733 seconds.

This indicates that the row-stochastic formulation provides a more efficient alternative without compromising performance.

2.1.4 A Five Class C-colME System with Different Variances

Here we will present results of simulation with C-colME and a system with five classes, having different mean values and different variances:

$$(\mu_a, \sigma_a) \in \{(-0.5, 2), (0.2, 1.2), (0.9, 0.6), (1.5, 0.9), (2.4, 1.5)\}. \tag{2.7}$$

All five classes are equally probable. In order to have enough neighboring agents we assumed $r = 20$. We considered a system with $N = 1000$ agents, with 10 realizations.

The results for the graph evaluation, histogram of the estimated standard deviations, local means, and the MSE are given in Fig. 2.18, Fig. 2.19, Fig. 2.20, and Fig. 2.21.

At the time instant $t = 100$, all five classes are connected. Later, at $t = 300$, class 1 is fully disconnected from the rest, while classes 2 and 3 stayed fully connected with a few links to classes 4 and 5. In the next time instant $t = 500$, class 4 is also fully separated, classes 2 and 3 slowly start growing apart and there is still a link between class 2 and class 5. At $t = 900$, all five classes are separated and do not have any links between each other making each class clearly distinguishable.

This process has also been examined using histograms which coincide with the above explained behavior. At $t = 100$, standard deviation value is from 0.7 to 1.8, without any distinct peaks. It can be noticed that two peaks have formed at $\sigma = 0.5$ and $\sigma = 1.0$ while the rest are still mixed together, at $t = 300$. As we saw before, at instant $t = 500$, three out of the five classes are separated (0.5, 1.0, 2.0), while classes 2 and 3 with standard deviations 1.2 and 1.5 respectively are still not disconnected. Finally, in the last histogram, we can see all 5 classes peaking at five different standard deviation values without intersecting between themselves.

As we can see in Figure 2.21, the mean square error follows the class separation process, exhibiting step-like behavior that reflects the sequential disentangling of the classes over time. The stationary error is higher in this case because it is averaged over fewer agents per class($N/C = 1000$) compared to the case when $N = 5000$ and $C = 2$, where each class has 2500 agents. Since the mean squared error is inversely proportional to the number of agents, we observe that the final MSE is approximately 2.5 times higher than in the $C = 2$ case. This is to be expected since increasing the number of classes while keeping the total number of agents fixed reduces the number of agents per class, assuming uniform distribution of agents per class, leading to a higher average error. For C_1 and C_2 being different values of classes, the ratio between the mean squared errors would be C_2/C_1.

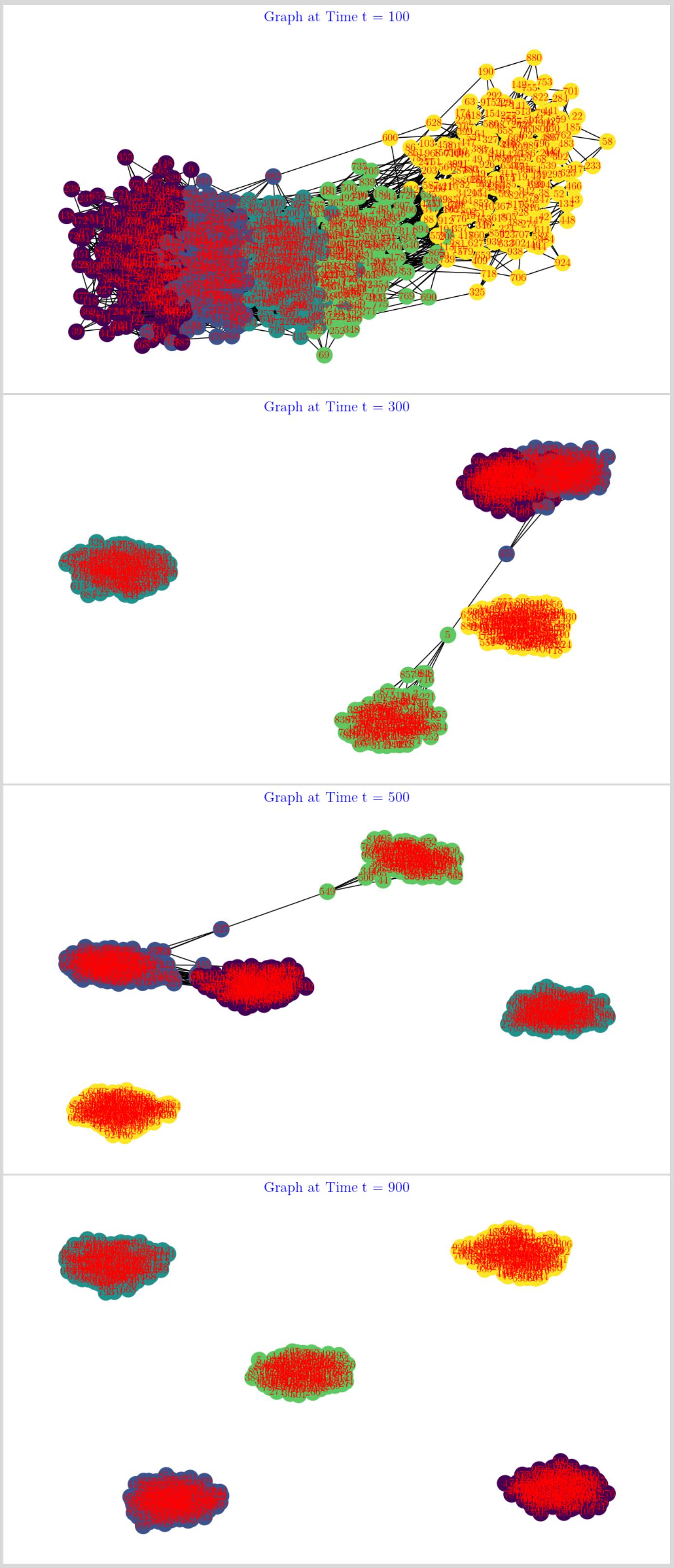


Figure 2.18: C-colME with $N = 1,000$ and $C = 5$ classes: Random graph with agent classes at $t = 100$, with the connections adjusted using the confidence intervals over time. Graphs at $t = 300$, $t = 500$, and $t = 900$, respectively, are shown from top to bottom.

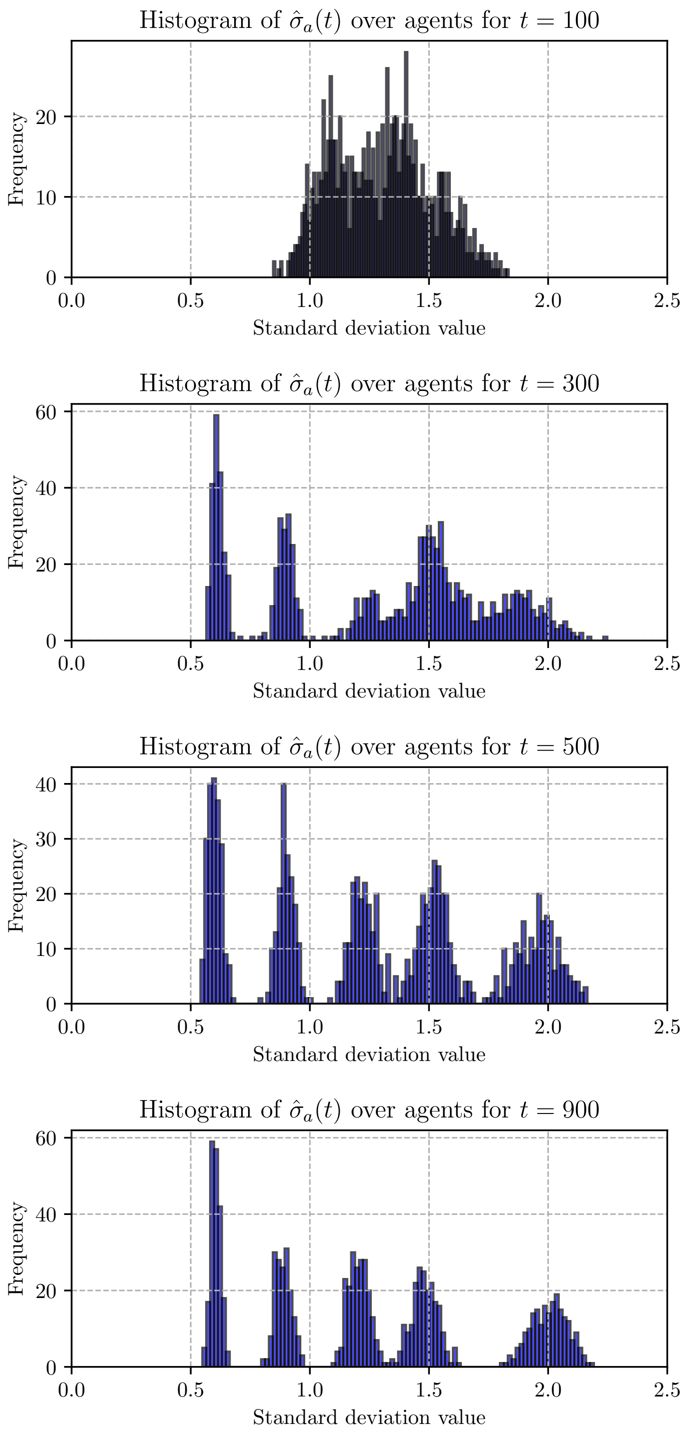


Figure 2.19: Histogram of the estimated standard deviations $\hat{\sigma}_a(t)$ over five-class agents with different means and standard deviations, at $t = 100, 300, 500, 900$ time instants.

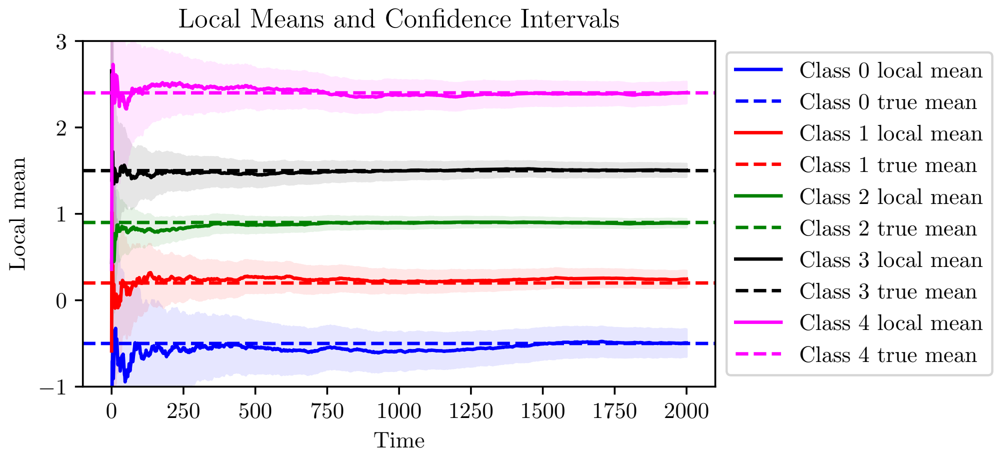


Figure 2.20: Local means and the corresponding confidence intervals for 5 agents belonging to five different classes, as a function of the number of samples t. Standard deviations are estimated by the proposed algorithm.

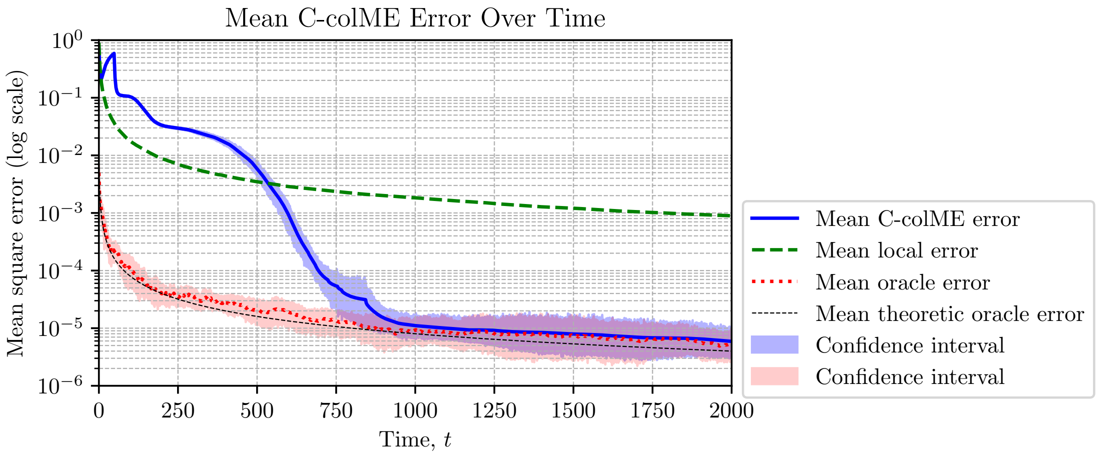


Figure 2.21: C-colME with $N = 5,000$ and $C = 5$ classes: Total MSE of the estimation The results are averaged over 10 realizations. Standard deviations are estimated by the proposed algorithm.

2.1.5 Convergence of C-colME to the Oracle Solution

Consider first a general form of a double stochastic matrix, $\mathbf{W}$. This matrix is a symmetric real-valued function, therefore it is diagonalizable. We can write

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T, \tag{2.8}$$

where $\mathbf{Q}$ is a matrix whose columns are eigenvectors $\mathbf{u}_i$ of $\mathbf{W}$ and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues λ_i of $\mathbf{W}$ as diagonal elements. By definition $\mathbf{W}\mathbf{u}_i = \lambda_i\mathbf{u}_i$.

The rows of the double stochastic matrix (and the row stochastic matrix as well) $\mathbf{W}$ sum to one, therefore, it satisfies the condition

$$\mathbf{W}\mathbf{1} = \mathbf{1} \tag{2.9}$$

where $\mathbf{1}$ is a column vector with all elements equal to 1. It means that for the double stochastic matrix: (i) $\lambda = 1$ is an eigenvalue; (ii) The corresponding eigenvector is $\mathbf{1}$ or commonly a constant vector, normalized so that its energy is 1. The elements of the eigenvector are $u_1(n) = 1/\sqrt{N}$ or in vector form $\mathbf{u}_1 = \mathbf{1}/\sqrt{N}$.

It can be shown that $|\lambda_i| < 1$ for any $i \neq 1$ for a connected graph, with $\lim_{t\to\infty} |\lambda_i|^t = 0$ for $i \neq 1$.

Now consider the iterative relation

$$\boldsymbol{\mu}(t) = \mathbf{W}\boldsymbol{\mu}(t-1) \tag{2.10}$$

Then $\boldsymbol{\mu}(t) = \mathbf{W}^t\boldsymbol{\mu}(0)$. Using $\mathbf{W}^t = \mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^T$, we can write

$$\lim_{t\to\infty} \boldsymbol{\mu}(t) = \lim_{t\to\infty} \mathbf{W}^t\boldsymbol{\mu}(0) = \lim_{t\to\infty} \mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^T\boldsymbol{\mu}(0) = \boldsymbol{\mu}_{oracle} \tag{2.11}$$

where $\boldsymbol{\mu}_{oracle}$ is a vector with the elements equal to the mean of $\boldsymbol{\mu}(0)$. Initially we assume only one similarity class and the oracle is average over all agents.

Next we will prove the previous relation. Consider

$$\lim_{t\to\infty} \mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^T = \lim_{t\to\infty} \begin{bmatrix} \frac{\mathbf{1}}{\sqrt{N}} & \mathbf{u}_2 & \dots & \mathbf{u}_N \end{bmatrix} \begin{bmatrix} 1^t & 0 & 0 & \dots & 0 \\ 0 & \lambda_2^t & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \vdots & \lambda_N^t \end{bmatrix} \begin{bmatrix} \mathbf{1}^T/\sqrt{N} \\ \mathbf{u}_2^T \vdots \\ \mathbf{u}_N^T \end{bmatrix} \tag{2.12}$$

where $\mathbf{1}\sqrt{N}$, $\mathbf{u}_2$, ..., $\mathbf{u}_N$, are the eigenvectors (that form matrix $\mathbf{Q}$). Since $|\lambda_2| < 1$, $|\lambda_3| < 1$, ..., $|\lambda_N| < 1$ we can write

$$\lim_{t\to\infty} \mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^T = \begin{bmatrix} \frac{\mathbf{1}}{\sqrt{N}} & \mathbf{u}_2 & \dots & \mathbf{u}_N \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \vdots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{1}^T/\sqrt{N} \\ \mathbf{u}_2^T \vdots \\ \mathbf{u}_N^T \end{bmatrix} \tag{2.13}$$

producing

$$\lim_{t\to\infty} \mathbf{Q}\boldsymbol{\Lambda}^t\mathbf{Q}^T\boldsymbol{\mu}(0) = \frac{\mathbf{1}\mathbf{1}^T}{N}\boldsymbol{\mu}(0) = \begin{bmatrix} 1/N & 1/N & 1/N & \dots & 1/N \\ 1/N & 1/N & 1/N & \dots & 1/N \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1/N & 1/N & 1/N & \vdots & 1N \end{bmatrix} \boldsymbol{\mu}(0) = \boldsymbol{\mu}_{oracle}. \tag{2.14}$$

Convergence Rate: In order to examine the convergence rate we will rewrite the previous equation for large t, but keeping λ_2 in addition to the eigenvalue 1, that is

$$\mathbf{Q}\boldsymbol{\Lambda}^t\mathbf{Q}^T = \begin{bmatrix} \frac{\mathbf{1}}{\sqrt{N}} & \mathbf{u}_2 & \dots & \mathbf{u}_N \end{bmatrix} \begin{bmatrix} 1^t & 0 & 0 & \dots & 0 \\ 0 & \lambda_2^t & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \vdots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{1}^T/\sqrt{N} \\ \mathbf{u}_2^T \vdots \\ \mathbf{u}_N^T \end{bmatrix} \tag{2.15}$$

Now we get

$$\mathbf{W}^t = \mathbf{Q}\boldsymbol{\Lambda}^t\mathbf{Q}^T = \frac{\mathbf{1}\mathbf{1}^T}{N} + \lambda_2^t\mathbf{u}\mathbf{u}_2^T$$

$$\mathbf{W}^t\boldsymbol{\mu}(0) = \mathbf{Q}\boldsymbol{\Lambda}^t\mathbf{Q}^T\boldsymbol{\mu}(0) \tag{2.16}$$

$$= \frac{\mathbf{1}\mathbf{1}^T}{N}\boldsymbol{\mu}(0) + \lambda_2^t\mathbf{u}\mathbf{u}_2^T\boldsymbol{\mu}(0) = \boldsymbol{\mu}_{oracle} + \lambda_2^t\mathbf{b}, \tag{2.17}$$

resulting in

$$\mathbf{W}^t\boldsymbol{\mu}(0) - \boldsymbol{\mu}_{oracle} = \lambda_2^t\mathbf{b} \tag{2.18}$$

where $\mathbf{b} = \mathbf{u}\mathbf{u}_2^T\boldsymbol{\mu}(0)$.

Next we will show that the convergence rate (ratio of errors in two consecutive iterations t and $t-1$) is proportional to $|\lambda_2|$,

$$\lim_{t\to\infty} \frac{\|\mathbf{W}^t\boldsymbol{\mu}(0) - \boldsymbol{\mu}_{oracle}\|_2}{\|\mathbf{W}^{t-1}\boldsymbol{\mu}(0) - \boldsymbol{\mu}_{oracle}\|_2}$$

$$= \lim_{t\to\infty} \frac{\|\lambda_2^t\mathbf{b}\|_2}{\|\lambda_2^{t-1}\mathbf{b}\|_2} = |\lambda_2|. \tag{2.19}$$

Application to C-colME: Now we may consider the weight matrix $\mathbf{W}(t)$ in C-colME. For sufficiently large t the agents from different classes are fully disconnected. For C classes, after appropriate rearranging of agent indices, we have a matrix in a block diagonal form, with block corresponding to graph components. For $C = 2$ we have

$$\mathbf{W}(t) = \begin{bmatrix} \mathbf{W}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_b \end{bmatrix} \tag{2.20}$$

In this case we have $C = 2$ eigenvectors with eigenvalue 1, the first one having constant values, $\mathbf{1}/\sqrt{|\mathcal{N}_a|}$, for $\mathbf{W}_a$ indices and the second one having constant values, $\mathbf{1}/\sqrt{|\mathcal{N}_b|}$, for $\mathbf{W}_b$ indices.

Also assume a simplified form of $\alpha(t)$, as $\alpha(t) = 1$ for $0 \leq t \leq T_L$ and $\alpha(t) = 0$ for $T_L < t < \infty$. Denote the value $\mathbf{X}(T_L)$ applied to the initial instant of iteration with $\mathbf{W}$, T_L, by $\boldsymbol{\mu}(0)$. Then, with $\tau = t - T_L$, we have

$$\begin{aligned}\lim_{\tau\to\infty} \boldsymbol{\mu}(t+1) &= \lim_{\tau\to\infty} \mathbf{W}^{\tau}\boldsymbol{\mu}(0) \\ &= \lim_{t\to\infty} \mathbf{Q}\boldsymbol{\Lambda}^{\tau}\mathbf{Q}^{T}\boldsymbol{\mu}(0) = \begin{bmatrix}\boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b\end{bmatrix}\end{aligned} \tag{2.21}$$

where averaging is done not only over the neighbors but over all class $\mathcal{C}_a$ for agents from this class and over all agents in class $\mathcal{C}_b$ for the agents from the second class, corresponding to oracle averaging, if the graph is correctly separated into two components, independently of the local connections between nodes within components, since $\mathbf{W}^{\tau}$ will take all nodes within a component for large τ.

Final Remarks on C-colME:
(i) The C-colME is numerically simple and efficient.
(ii) It averages the collaborative mean over the whole class component as long as the component is not disconnected.
(iii) Convergence rate is determined by the second eigenvalue (first less than 1) of the matrix $\mathbf{W}(t)$.
(iv) The result depends on $\alpha(t)$ and the accumulated local means. If $\alpha(t) \sim 1$ after some t, then the local means do not accumulate any more after this instant and do not further improve the accuracy. If $\alpha(t)$ stays below 1 (to take into account local means for large t), then local means (with significant error) will always be included in the final collaborative result.

2.1.6 C-colME Example with Uniformly Distributed Data

Simulations are performed for uniformly distributed data with variance σ. The mean value μ_a of data is defined by the similarity class. This is a sub-Gaussian distribution, since obviously its tails are lighter than the Gaussian ones (its tails are zero after $\sqrt{3}\sigma$).

The results for the standard deviation estimation, the kurtosis estimation, and the mean square error are given in Fig. 2.22. Kurtosis $\kappa < 3$ indicates that the data distribution is not Gaussian, having lighter tails. This is a kind of the so-called *platykurtic distribution.*

Histograms for both standard deviation and kurtosis value peak at 2.0 and 1.8 respectively confirming the theoretical expectation. The result also confirms that kurtosis is $\kappa < 3$ therefore having lighter tails. The mean squared error gives similar results to the ones achieved using Gaussian distribution without giving any significant improvements

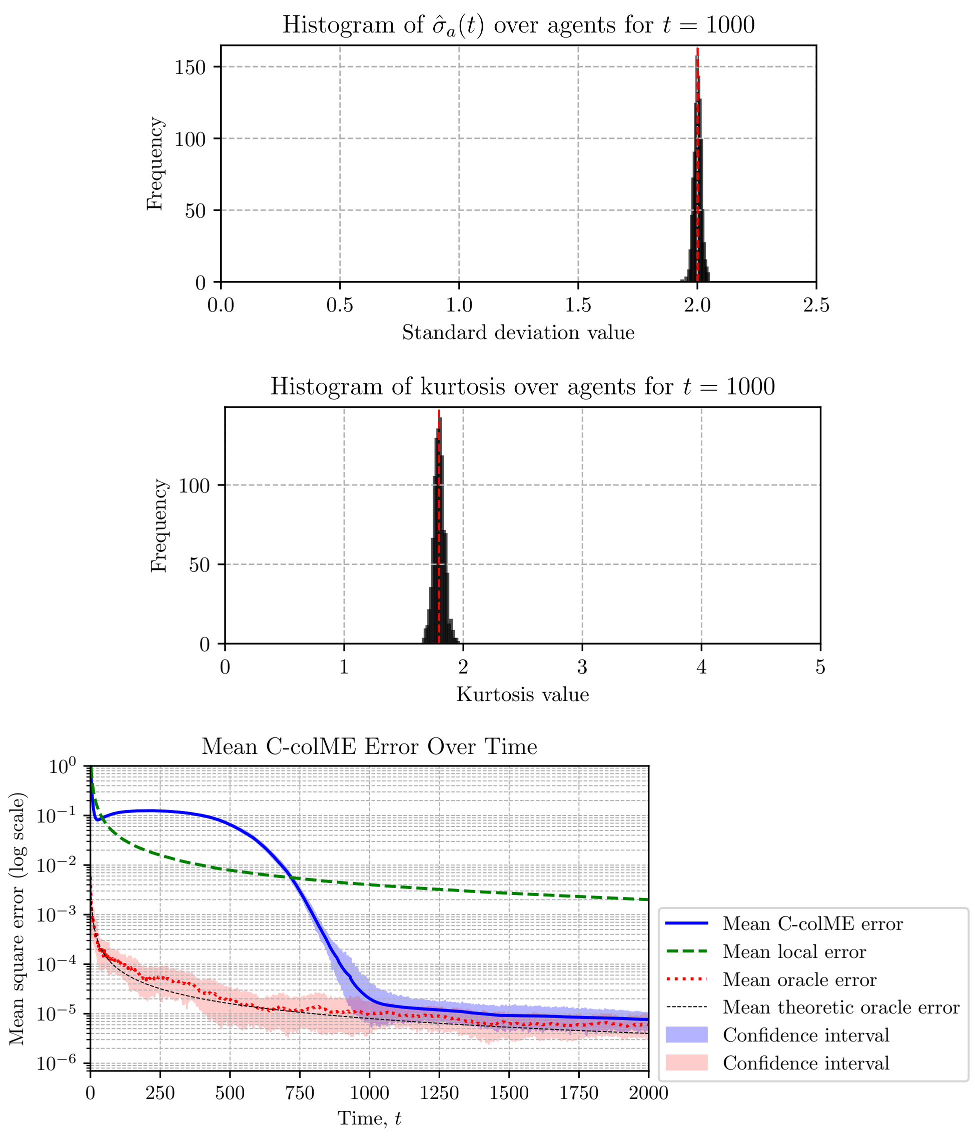


Figure 2.22: C-colME with $N = 1,000$ and uniformly distributed data: (Top) Histogram of standard deviation; (Middle) Histogram of kurtosis, and (Bottom) The total MSE of the estimation. The results are averaged over 10 realizations.

2.1.7 C-colME with Scaled Rademacher (Bernoulli) Distribution

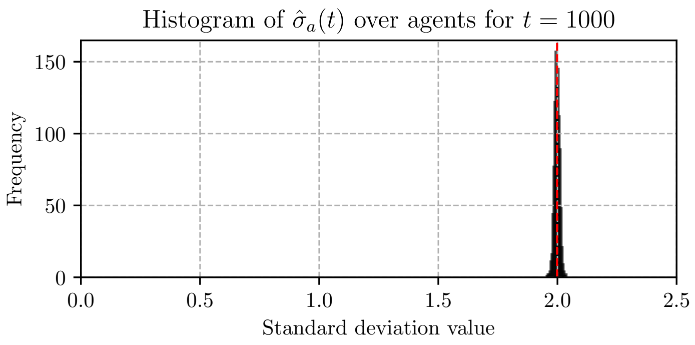
We repeated the previous simulations for data distributed according to Rademacher distribution scaled by σ to get variance σ^2. The data values are formed according to

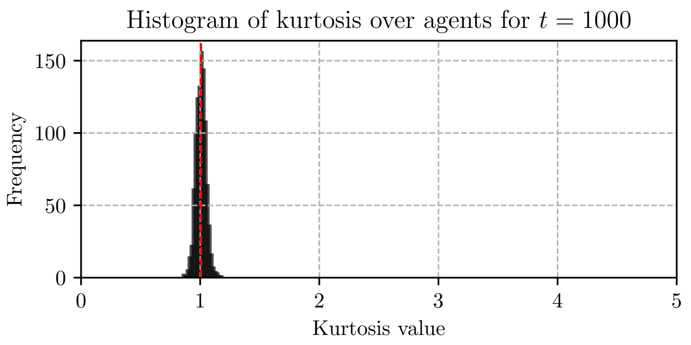
$$x_a(t) = \mu_a + \sigma w(t) \tag{2.22}$$

where μ_a is defined by the similarity class, $w(t)$ takes values $+1$ and -1 with equal probability. For such random variable $w(t)$ we say that it is Rademacher distributed (and $\sigma w(t)$ is Bernoulli distributed). This is a sub-Gaussian distribution, since obviously its tails are lighter than the Gaussian ones (its tails are zero after σ).

The results for the standard deviation estimation, kurtosis estimation, and means square error are given in Fig. 2.23. The kurtosis value is here $\kappa = 1$.

Histograms for both standard deviation and kurtosis value peak at 2.0 and 1.0 respectively





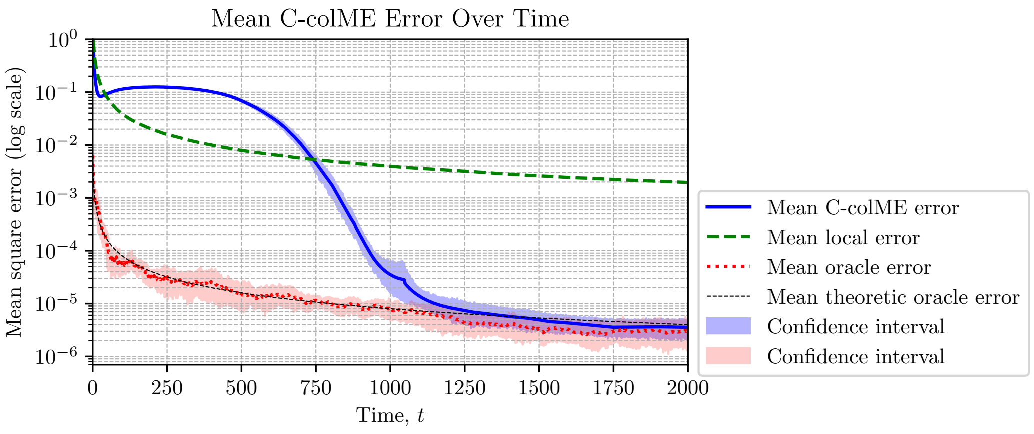


Figure 2.23: C-colME with $N = 1,000$ and Bernoulli distributed data: (Top) Histogram of standard deviation; (Middle) Histogram of kurtosis, and (Bottom) The total MSE of the estimation The results are averaged over 10 realizations.

confirming the theoretical expectation. The result also confirms that kurtosis is $\kappa < 3$ therefore having lighter tails. The mean squared error gives similar results to the ones achieved using Gaussian distribution without giving any significant improvements.

2.2 Message-Passing Algorithm B-colME

In this approach, the agents form an exchange message of dimension $d \times 2$, where d is the distance in the graph (a parameter). Each agent first calculates a sum of its own samples up to the actual time instant t

$$m^{a}_{1,1}(t) = \sum_{\tau=1}^{t} x_a(\tau). \tag{2.23}$$

These sums for all agents $a \in \mathcal{A}$ can be considered as elements of a column vector denoted by $\mathbf{M}(t)$. The sums are exchanged as part of the message (in particular, this is *the first row in the message table*, along with the number of samples in the sum t, as the second element of the first row in the message). Before continuing with the next rows, we will state that these messages are exchanged among neighboring vertices. A vertex a' from the neighborhood of a, denoted by $\mathcal{N}_a$, sends this information

$$m^{a'\to a}_{1,1}(t) = \sum_{\tau=1}^{t} x_{a'}(\tau). \tag{2.24}$$

Notice that the second element of the first row, $m^{a'\to a}_{1,2}(t) = t$, is the number of samples.

The node a accumulates all values from the messages from $\mathcal{N}_a$ as $\sum_{a'\in\mathcal{N}_a} m^{a'\to a}_{1,1}(t)$.

In matrix form sum of all sums over neighbors, for agents, can be written for the whole system of agents as

$$\mathbf{M_1}(t) = \mathbf{A}(t)\mathbf{M}(t) \tag{2.25}$$

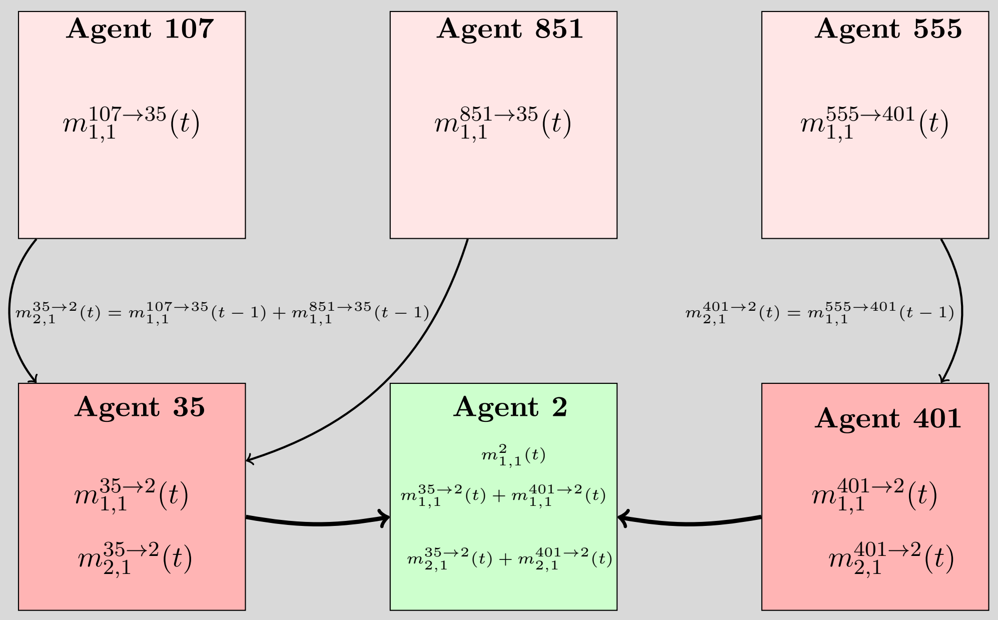
since the symmetric adjacency matrix of the graph $\mathcal{G}(\mathcal{A}, \mathcal{E})$, denoted at an instant t by $\mathbf{A}(t)$ contains 1s in the row corresponding to the adjacent agents to each node and by multiplying the matrix $\mathbf{A}(t)$ with vector $\mathbf{M}(t)$ we exactly sum the sums of all neighboring agents to each node, that is, we sum the first elements, $m^{a'\to a}_{1,1}(t)$ in the first row of the exchange matrix for all neighbors of each a.

The number of elements in the sum $\mathbf{M_1}(t)$ for each agent a is equal to the number of its neighbors multiplied by t. The number of neighbors for each agent is equal to the sum of matrix $\mathbf{A}(t)$ over its rows or columns (symmetric matrix), that is $\mathbf{D1}(t) = [\sum_j A_{i,j}(t), i = 1, 2, , \dots, n]$

Next we consider the *second row of the message table.* This row contains the sums of the first neighbors of neighbors of an agent a. The first element in the second row of the exchange messages is

$$m^{a'\to a}_{2,1}(t) = \sum_{a''\in\mathcal{N}_{a'}, a''\neq a} m^{a''\to a'}_{1,1}(t-1). \tag{2.26}$$

This element of the exchange message gives the sum of the sums of agents at distance 2 in the graph of the agent considered, a. These sums will be summed for all $a' \in \mathcal{N}_a$ and used along with the total number of terms (contained in the second element of the second row of the message table) to estimate the collaborative mean.



Principle of Sum Exchanges Toward One Selected Agent 2 in the B-colME: The first column of messages sent to and received by Agent 2 at an instant t in the $d = 2$ case. Agent 2 has 2 neighbors (Agent 35 and Agent 401). The neighbors of Agent 35 are Agent 107 and Agent 851, while the neighbor of Agent 401 is Agent 555 (excluding Agent 2). The local sums of agents are indicated by $m_{1,1}(t)$. The local sum of the considered Agent 2 is $m_{1,1}^{2}(t)$. The sum of all sums for Agent 2 (and all other agents) is obtained in matrix form as $\mathbf{M}(t) + \mathbf{A}(t)\mathbf{M}(t) + \mathbf{A2}(t-1)\mathbf{M}(t-1)$. The second column of messages contains the total number of agent sums included in the first column.

This summation can be written in matrix form for the entire system of agents, as

$$\mathbf{M_2}(t) = \mathbf{A2}(t-1)\mathbf{M}(t-1) \tag{2.27}$$

where $\mathbf{A2}(t)$ is the matrix defining all agents that are at a distance of exactly 2 from each of the agents. Its value is

$$\mathbf{A2}(t) = \mathbf{A}^2(t) - \mathbf{A}^2(t) \odot \mathbf{A} - \mathbf{A}^2(t) \odot \mathbf{I} \tag{2.28}$$

where Boolean algebra is used for matrix multiplication and $\mathbf{A}^2(t) \odot \mathbf{A}$, $\mathbf{A}^2(t) \odot \mathbf{I}$ remove possible agents reached with one or zero steps. The element product (Hadamard product) is denoted by $\odot$.

The number of elements in the sum $\mathbf{M_2}(t)$ for each agent a is equal to the number of its step-two neighbors multiplied by $(t-1)$. The number of step-two neighbors for each agent is equal to the sum of matrix $\mathbf{A}2(t-1)$ over its rows or columns (symmetric matrix), that is

$$\mathbf{D}2(t) = [\sum_j A2_{i,j}(t-1), i = 1, 2, , \ldots, A] \tag{2.29}$$

Similarly, the *third row of the message table* has the first element that can be defined for the whole system by

$$\mathbf{M_3}(t) = \mathbf{A3}(t-2)\mathbf{M}(t-2) \tag{2.30}$$

where $\mathbf{A3}(t)$ is the matrix defining all agents which are at a distance of exactly 3 from each of the agents. Its value is

$$\mathbf{A3}(t) = \mathbf{A}^3(t) - \mathbf{A}^3(t) \odot \mathbf{A}^2 - \mathbf{A}^3(t) \odot \mathbf{A} - \mathbf{A}^3(t) \odot \mathbf{I} \tag{2.31}$$

In general,

$$\mathbf{M_d}(t) = \mathbf{Ad}(t-d+1)\mathbf{M}(t-d+1) \tag{2.32}$$

where $\mathbf{Ad}(t)$ is the matrix that defines all agents that are at a distance of exactly d from each of the agents. Its value is

$$\mathbf{Ad}(t) = \mathbf{A}^d(t) - \sum_{i=0}^{d-1} \mathbf{A}^d(t) \odot \mathbf{A}^i \tag{2.33}$$

The agent a estimates its mean using data from the exchange table summing all sums and dividing the resulting sum by the total number of data samples, as

$$\bar{\mu}_a(t) = \frac{m^a_{1,1}(t) + \sum_{a' \in \mathcal{N}_a} \sum_{h=1}^{d} m^{a' \to a}_{h,1}(t)}{t + \sum_{a' \in \mathcal{N}_a} \sum_{h=1}^{d} m^{a' \to a}_{h,2}(t)} \tag{2.34}$$

In matrix form we can calculate this value for the whole system as

$$\boldsymbol{\mu}(t) = \Big(\mathbf{M}(t) + \mathbf{M_1}(t) + \mathbf{M_2}(t) + \ldots\Big)./(\mathbf{1}t + \mathbf{D}(t)t + \mathbf{D2}(t)(t-1) + ...) \tag{2.35}$$

where $\mathbf{D}(t)$ is the degree vector of the symmetric matrix $\mathbf{A}(t)$, $\mathbf{D2}(t)$ is the degree vector of symmetric matrix $\mathbf{A2}(t-1)$, and so on. The degree vectors are obtained by adding the values of the corresponding matrices over columns or rows. The element-wise division is denoted by $./$.

2.2.1 Simplified Convergence Analysis of B-colME

Here we will present a simplified analysis of the expected behavior of the B-colME system for large t and different d.

- For $d = 0$, the expected squared error for B-calME obviously coincides with the local means approach, since no collaboration is done with neighbors. The expected mean square error is equal to the variance of the estimated means. For any t, it is equal to

$$\varepsilon^2_{d=0} = \frac{\sigma^2}{t} \tag{2.36}$$

- For $d = 1$ the result of B-colME collaboration is the mean averaged over expected number of $|\mathcal{N}(0)|/C$ neighbors ($|\mathcal{N}(0)| = r$), after separation time, $t > T_{sep}$. The expected mean squared error, after assumed successful separation of the components, is

$$\varepsilon^2_{d=1} = \frac{\sigma^2}{t(1 + \frac{|\mathcal{N}(0)|}{C})} \tag{2.37}$$

- For $d = 2$ the collaboration in B-colME is done over expected number of $|\mathcal{N}(0)|/C$ neighbors and over the expected number of $(|\mathcal{N}(0)|/C)^2$ neighbors of neighbors. Assuming again that the separation is successful the expected error is

$$\varepsilon^2_{d=2} = \frac{\sigma^2}{t(1 + \frac{|\mathcal{N}(0)|}{C} + (t-1)(\frac{|\mathcal{N}(0)|}{C})^2)} \tag{2.38}$$

- For any d the collaboration in B-colME is done over expected number of $|\mathcal{N}(0)|/C$ neighbors and over the expected number of $(|\mathcal{N}(0)|/C)^2$ neighbors of neighbors and so on. Assuming again that the separation is successful the expected error is

$$\varepsilon^2_{d} = \frac{\sigma^2}{t(1 + \frac{|\mathcal{N}(0)|}{c}) + (t-1)(\frac{|\mathcal{N}(0)|}{C})^2 + \cdots + (t-d+1)(\frac{|\mathcal{N}(0)|}{C})^d} \tag{2.39}$$

  Of course the number of collaborative neighbors is limited by N/C, so in the denominator we have to take N/C if the value obtained is greater than N/C.

Comment on B-colME: It has been assumed that neighbors of neighbors of different neighboring agents, can not be the same, that is, we cannot reach one ending node over different paths, starting from the considered agent. Our simulations are done by excluding those cases. In reality for large d, for example $d = 4$, $C = 2$, $r = 12$ and $N = 1000$ we get the expected number of neighbors as follows, 6 for $d = 1$, $36 + 6$ for $d = 2$, $216 + 36 + 6$, for $d = 3$, $1296 + 216 + 36 + 6 = 1554$ for $d = 4$. Since there are only 500 neighbors per class, obviously there must be multiple nodes in neighbors. This will worsen the estimation error if not addressed. If addressed, it will reduce the number of collaborators expected in the previous formulas and produce slightly worse results than expected, for higher values of d.

2.2.2 Numerical Example for B-colME

We simulated in Python the entire system with $A = 1000$ agents using the B-colME approach. A two-class system is assumed, $C = 2$ with means $\mu_1 = 0.1$ and $\mu_2 = 0.9$, and variance $\sigma = 2$. A random regular graph is formed with steps $r = 12$ and $d = 4$. The value of $\delta = 0.1$ is used. We compared the obtained results with the exact ones using $\varepsilon = 0.1$.

The variance is estimated for each agent individually, as explained in Algorithm 1.

The B-colME algorithm is used, along with the local one, when no collaboration is done, and with the described oracle approach.

The results are shown in Figs. 2.24, 2.25, 2.26, and 2.27.

2.2.3 Number of Agents and Neighborhood in B-colME

Next, we will simulate a B-colME system with different numbers of agents: $N = 5,000$ and $N = 10,000$.

We will also consider various neighborhood distances d in all cases.

The average calculation time will be given on a MacBook Air with 8-core M3 CPU and 16 GB RAM.

The results, along with detailed explanations, are shown in the figures and their captions below.

Figures 2.24–2.35 present the performance and behavior of the B-colME model under different environment settings such as different values of N, and different depths. In general, we can notice that the mean squared error for the B-colME behaves similarly to the one in C-colME. The time it intersects the mean local error is slightly earlier than C-colME being 650. It also converges towards the oracle solution slightly above 1000 which is also slightly faster than C-colME.

Figure 2.24 shows the evolution of the network topology for $N = 1{,}000$ agents. Starting from a random connected graph at $t = 0$, the snapshots at $t = 300$, where all agents are mixed together, $t = 500$ where different classed agents slowly start separating from each other, and $t = 800$ when the agents from different classes are completely separated and easily differentiated, illustrate how the connections reorganize over time under the B-colME dynamics.

Figure 2.25 reports the mean squared error (MSE) of the estimation. The top panel corresponds to the case $N = 1{,}000$ with $d_{\max} = 4$, where the error decreases steadily and confidence intervals are also shown. The bottom panel compares the results for different neighborhood sizes d, with theoretical values given for reference. On average, the calculation time in this case is 214 seconds per realization.

Figure 2.26 presents the total number of wrong links, i.e., when an agent from one class is connected to an agent from the other class. The number of such links decreases over time, reflecting improved class separation.

Figure 2.27 shows the number of agents with estimation error greater than $\varepsilon = 0.1$. This number gradually falls, demonstrating convergence of individual estimates toward the true values.

Figures 2.28–2.31 extend the analysis to larger systems. For $N = 5{,}000$ agents, the MSE is again reported both for a fixed $d_{\max} = 4$ and for varying neighborhood sizes d (Figure 2.28). The corresponding number of agents exceeding the error threshold $\varepsilon = 0.1$ is given in Figure 2.29. Figure 2.30 shows that increasing $d_{\max}$ from 4 to 5 significantly raises the computation time (by nearly a factor of three), due to the exponential increase in the number of neighbors. Figure 2.31 illustrates the error-threshold statistics for this setting.

Figure 2.35 provides an overview of the estimation MSE across different system sizes ($N = 1{,}000$, $5{,}000$, and $10{,}000$).

In summary, it is noticed that the B-colME exhibits convergence behavior similar to the C-colME, with MSE decreasing toward the oracle solution and the number of wrong links diminishing over time. Larger systems and larger neighborhoods improve accuracy but significantly increase computational complexity.

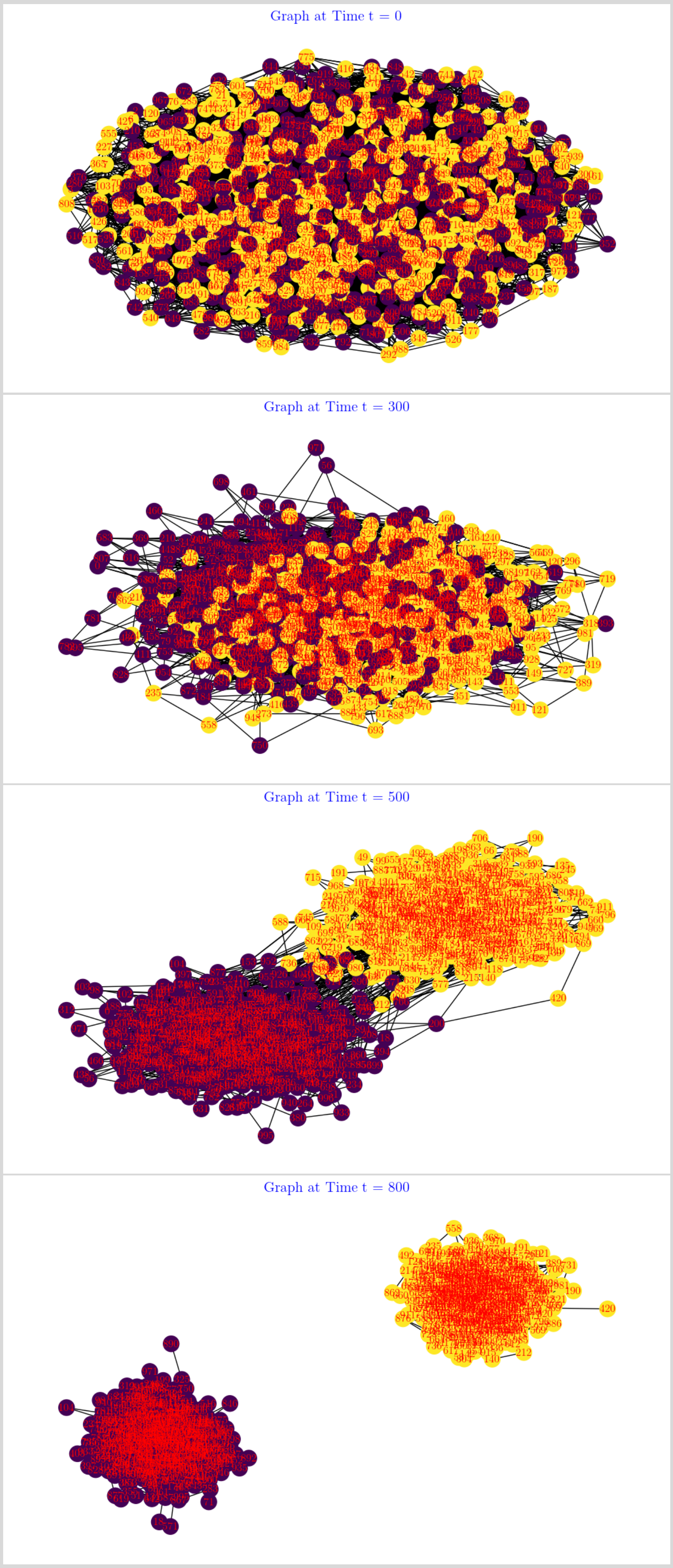


Figure 2.24: B-colME: Random graph with agent classes, arbitrary connected, at $t = 0$, $t = 300$, $t = 500$, and $t = 800$, respectively.

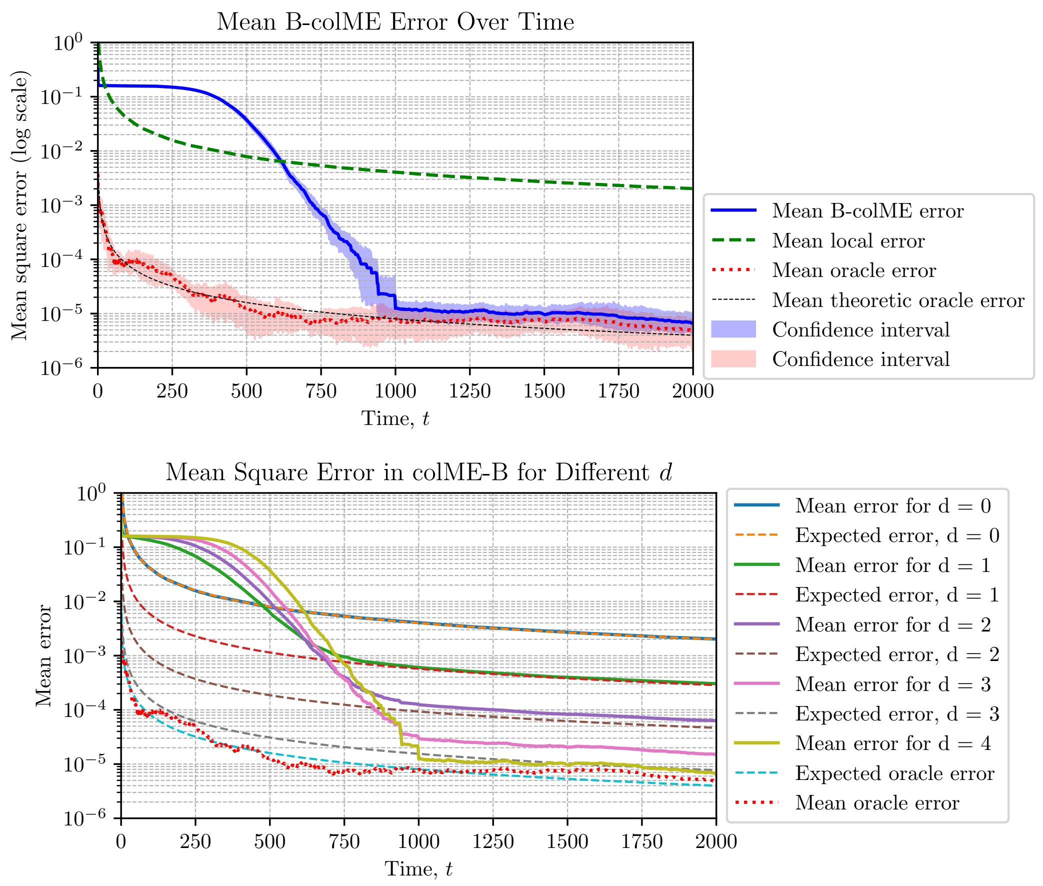


Figure 2.25:
B-colME: (Top) B-colME , $N = 1,000$ $d_{max} = 4$: Total MSE of the estimation averaged over all agents at time instants t. The results are averaged over 10 realizations. Confidence intervals for the MSE are calculated using Bootstrap with 0.95 confidence. (Bottom) B-colME with for various d: Total MSE of the estimation. The results are averaged over 10 realizations. *Expected theoretic values are given as well.* **Calculation time** is 214 seconds per realization, on average.

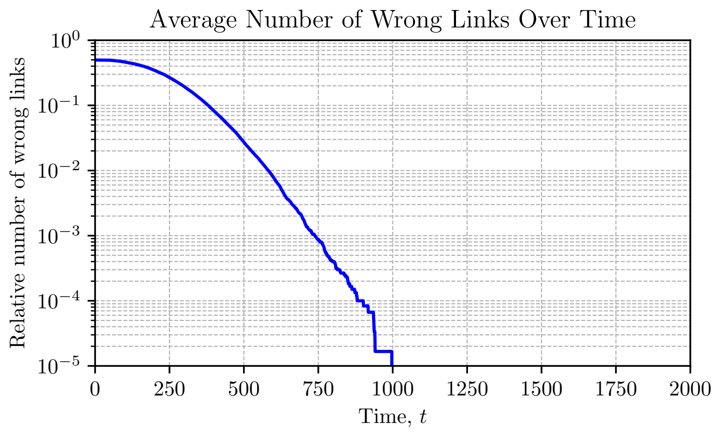


Figure 2.26: B-colME, $N = 1,000$ $d_{max} = 4$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 10 realizations.

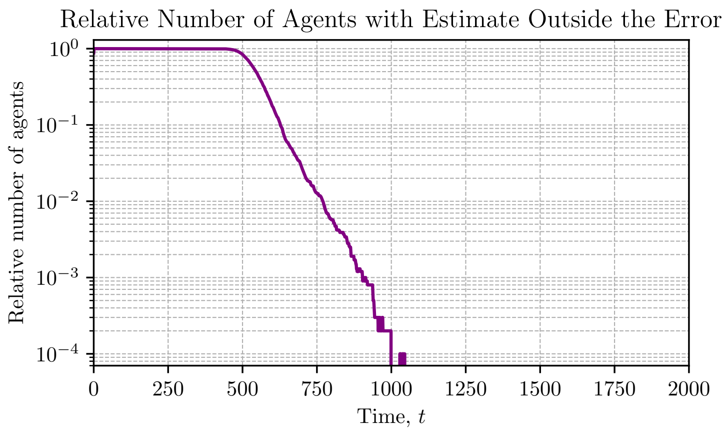


Figure 2.27: B-colME, $N = 1,000$ $d_{max} = 4$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 10 realizations.

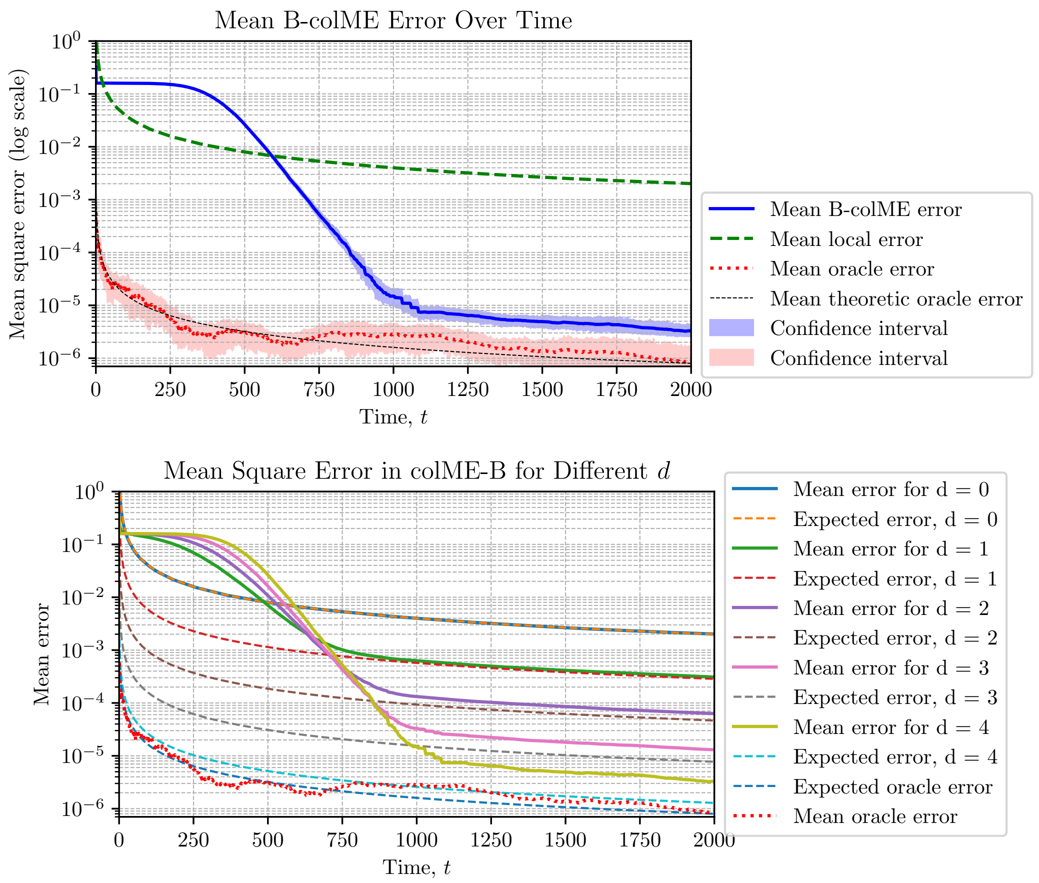


Figure 2.28: B-colME with $N = 5000$ for various d $d_{max} = 4$: Total MSE of the estimation (estimated values minus exact values, squared) averaged over all agents at time instants t. The results are averaged over 10 realizations. *Expected theoretic values are given as well.* **Calculation time** is 1542 seconds per realization, on average.

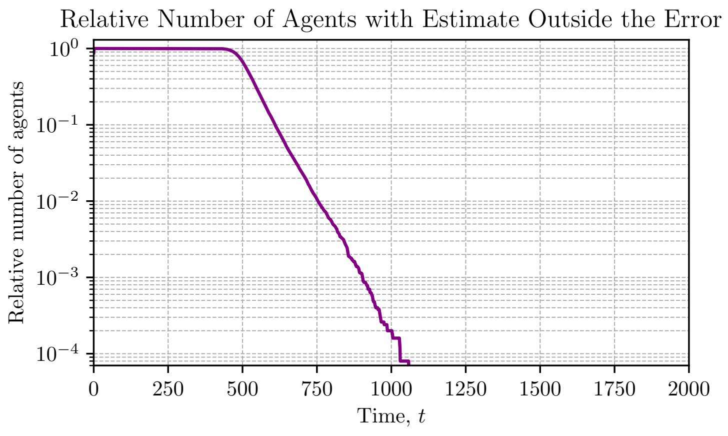


Figure 2.29: B-colME, $N = 5000$ $d_{max} = 4$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 10 realizations.

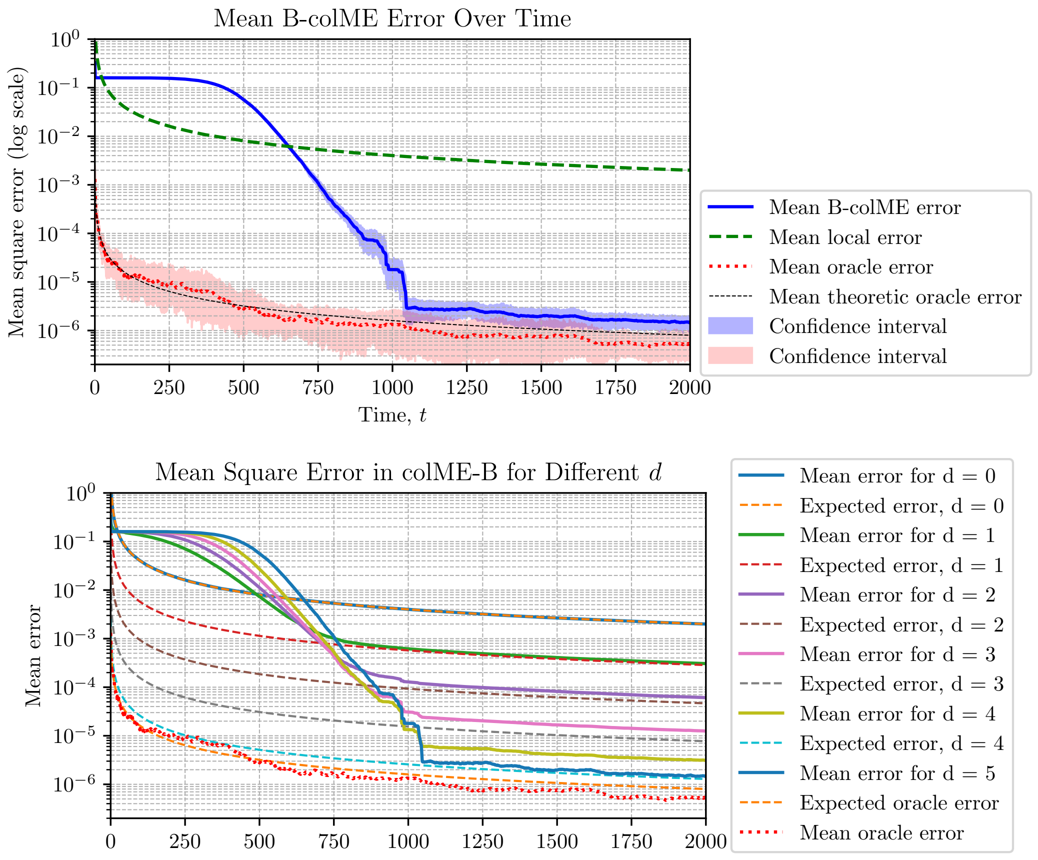


Figure 2.30: B-colME with $N = 5000$ for various d with $d_{max} = 5$: Total MSE of the estimation (estimated values minus exact values, squared) averaged over all agents at time instants t. The results are averaged over **5 realizations**. *Expected theoretic values are given as well.* **Calculation time** is 3627 seconds per realization, on average. Note that $d_{max} = 5$ increased computation time with respect to $d_{max} = 4$ almost 3 times since the number of neighbors is exponentially increased with d_{max}, up to $N/2$.

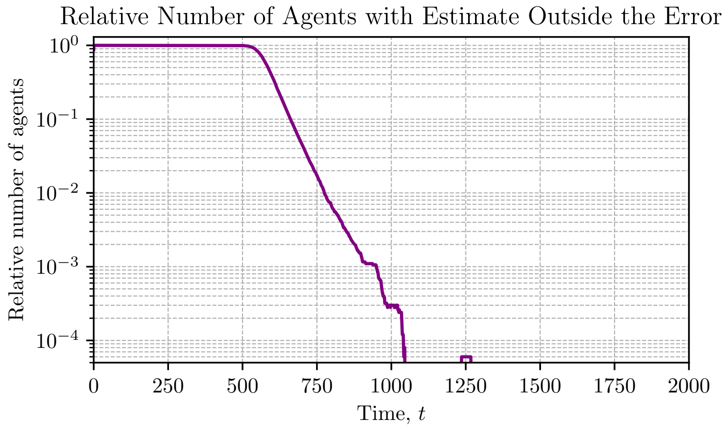
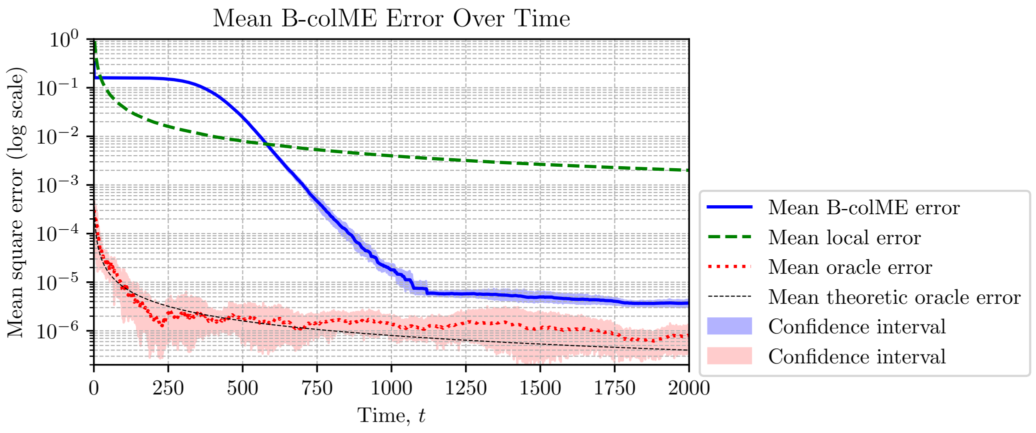


Figure 2.31: B-colME, $N = 5000$ $d_{max} = 5$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 10 realizations.



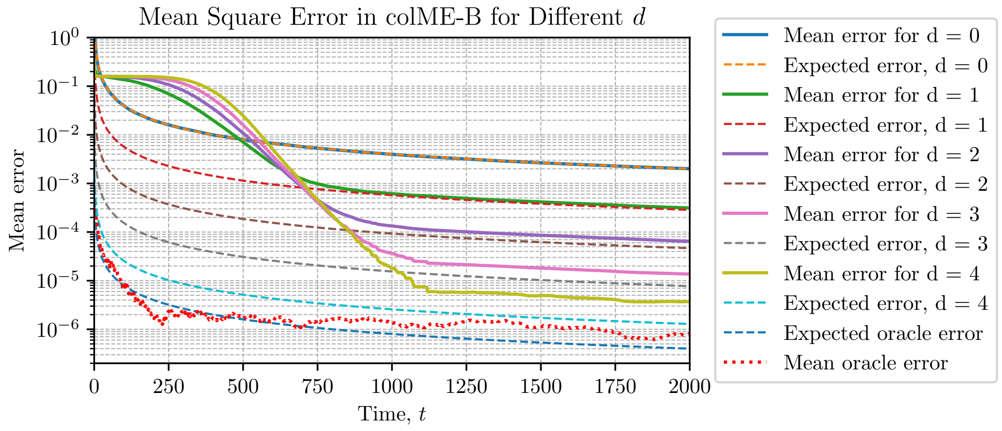


Figure 2.32: B-colME with $N = 10,000$ for various d $d_{max} = 4$: Total MSE of the estimation (estimated values minus exact values, squared) averaged over all agents at time instants t. The results are averaged over 5 realizations. *Expected theoretic values are given as well.* **Calculation time** is 4056 seconds per realization, on average.

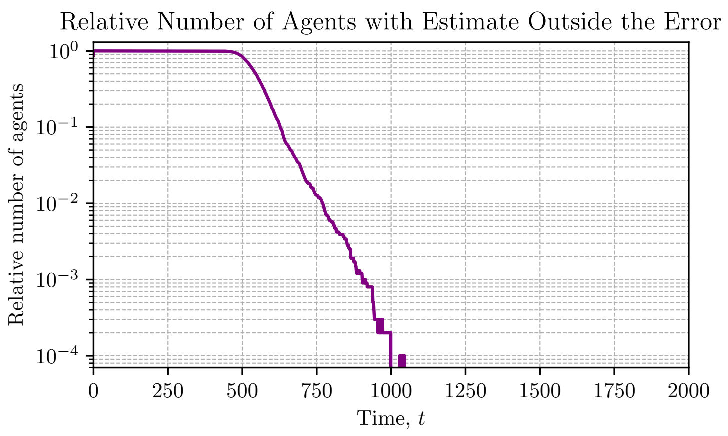


Figure 2.33: B-colME, $N = 10,000$ $d_{max} = 4$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 5 realizations.

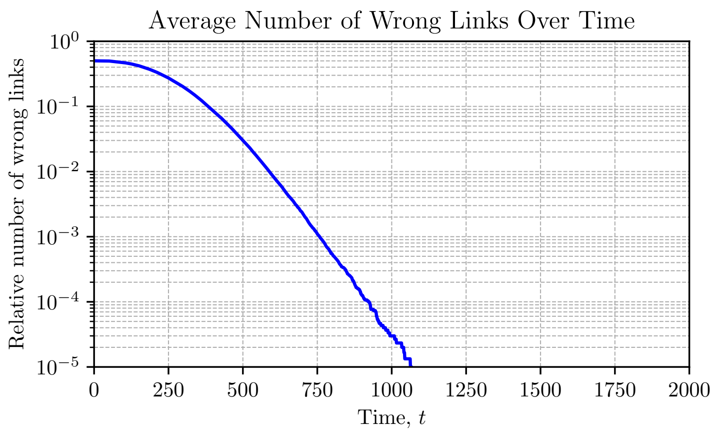


Figure 2.34: B-colME, $N = 10,000$ $d_{max} = 4$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 5 realizations.

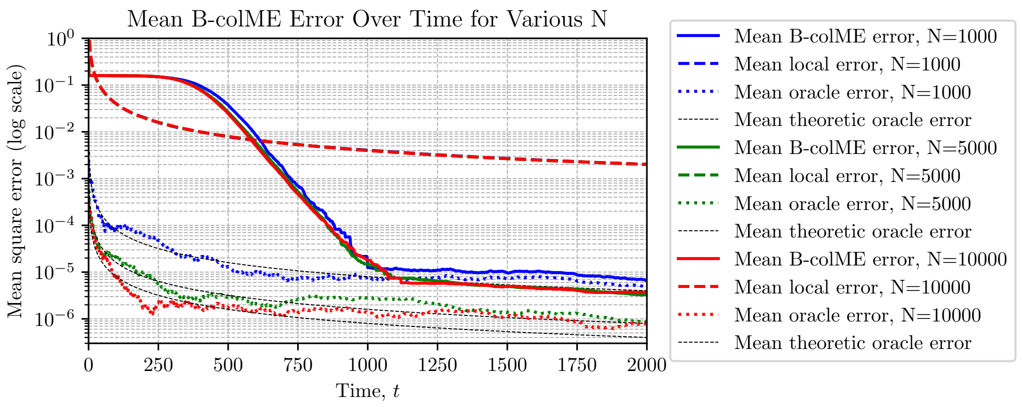


Figure 2.35: B-colME with various N and $d = 4$: Overview of the total MSE of the estimation averaged over 10 realizations for various $N = 1000$, 5000, and 10000 (5 realizations). *Expected theoretic values are given as well.*

2.2.4 Overview of Calculation Times

In the next table we shall present an overview of the calculation times for various methods and number of agents.

We can see that the fastest time is achieved with the row-stochastic (random walk) C-colME, since it uses the lowest calculation complexity. It is followed by C-colME, B-colME, and colME. The slowest is the weighted C-colME since it requires all floating point calculations related to the graph adjacency matrix.

The calculation time increases as $\mathcal{O}(N)$ with the number of agents. In B-colME the number d has a significant influence to the calculation time.

Table 2.1: Calculation times in seconds for C-colME, B-colME, and colME. W-C-colME is weighted C-colME and R-C-colME is C-colME with reduced confidence intervals and reconnection

	C-colME	B-colME	colME	W-C-colME	R-C-colME
$N = 1,000$	143	214	165	389	103
$N = 5,000$, $d_{max} = 4$	846	1542			
$N = 5,000$, $d_{max} = 5$		3627			
$N = 10,000$	3217	4056			

2.2.5 A Three Class B-colME System with Different Variances

Simulations with B-colME and a system with three classes, having different mean values and different variances:

$$(\mu_a, \sigma_a) \in \{(0.1, 2), (0.9, 1.5), (1.5, 1)\}.$$

are given. All three classes are equally probable.

The results for local means and confidence intervals, the histogram of the standard deviation evaluation, and the MSE are given in Figs. 2.38, 2.37, 2.39, respectively.

At the time instant $t = 100$, all three classes are connected. Later, at $t = 300$, the classes start to separate, but still maintain some links between them. At $t = 500$, three distinct clusters become visible, although they are still weakly connected to each other. By $t = 900$, all three classes are fully separated, with no links remaining across classes, making the clusters clearly distinguishable.

This process is further examined using histograms, which are consistent with the observed separation behavior. At $t = 100$, the estimated standard deviations lie in a mixed range without distinct peaks. By $t = 300$, the emergence of separate peaks begins, although there is still overlap between the classes. At $t = 500$, three peaks corresponding to the three classes become evident, but remain partially intersecting. Finally, at $t = 900$, the histogram clearly displays three well-separated peaks, each corresponding to one class, without overlap.

As shown in Figure 2.39, the mean squared error (MSE) follows the same separation process, exhibiting step-like behavior that mirrors the progressive disentangling of the classes. Also, the Figure shows the mean square error and its behavior based on different depth values.

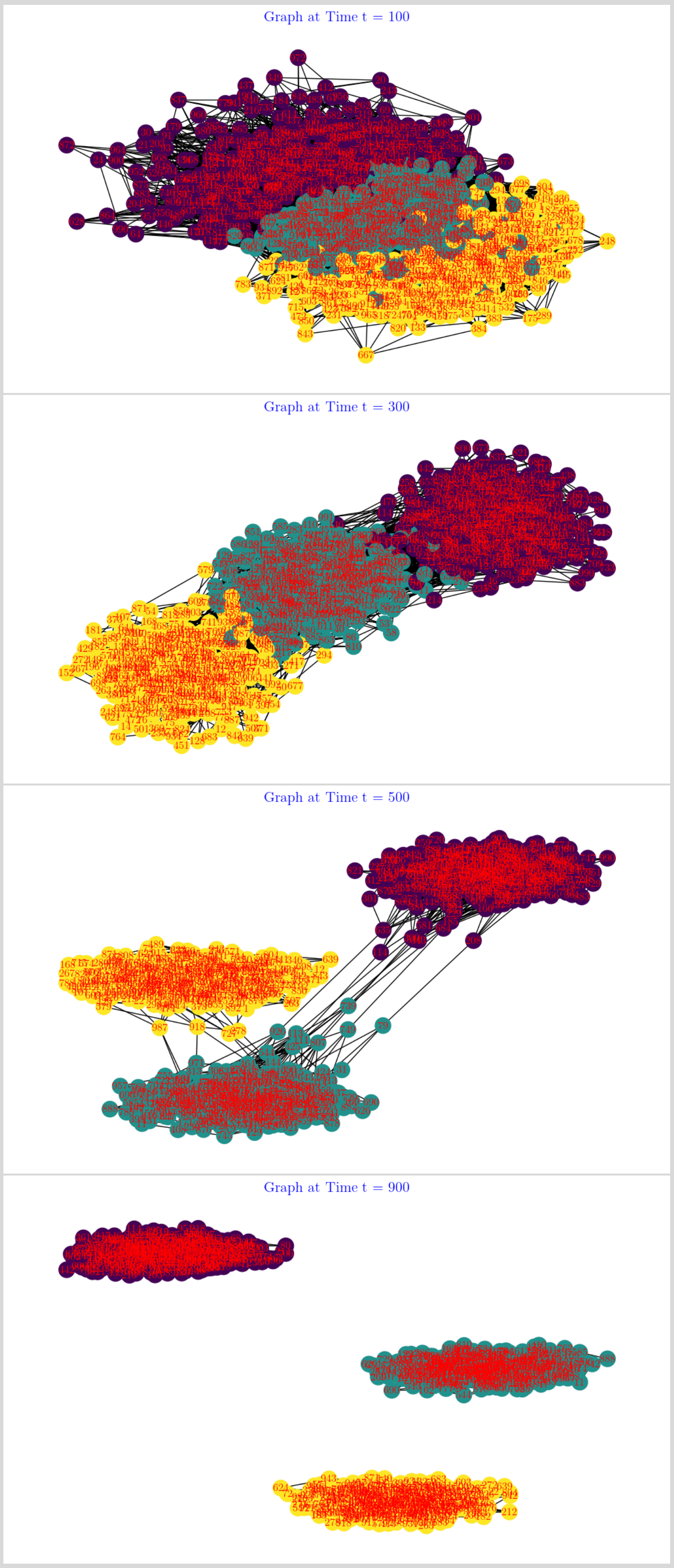


Figure 2.36: B-colME with $N = 1,000$ and $C = 3$ classes: Random graph with agent classes at $t = 100$, with the connections adjusted using the confidence intervals over time. Graphs at $t = 300$, $t = 500$, and $t = 900$, respectively, are shown from top to bottom.

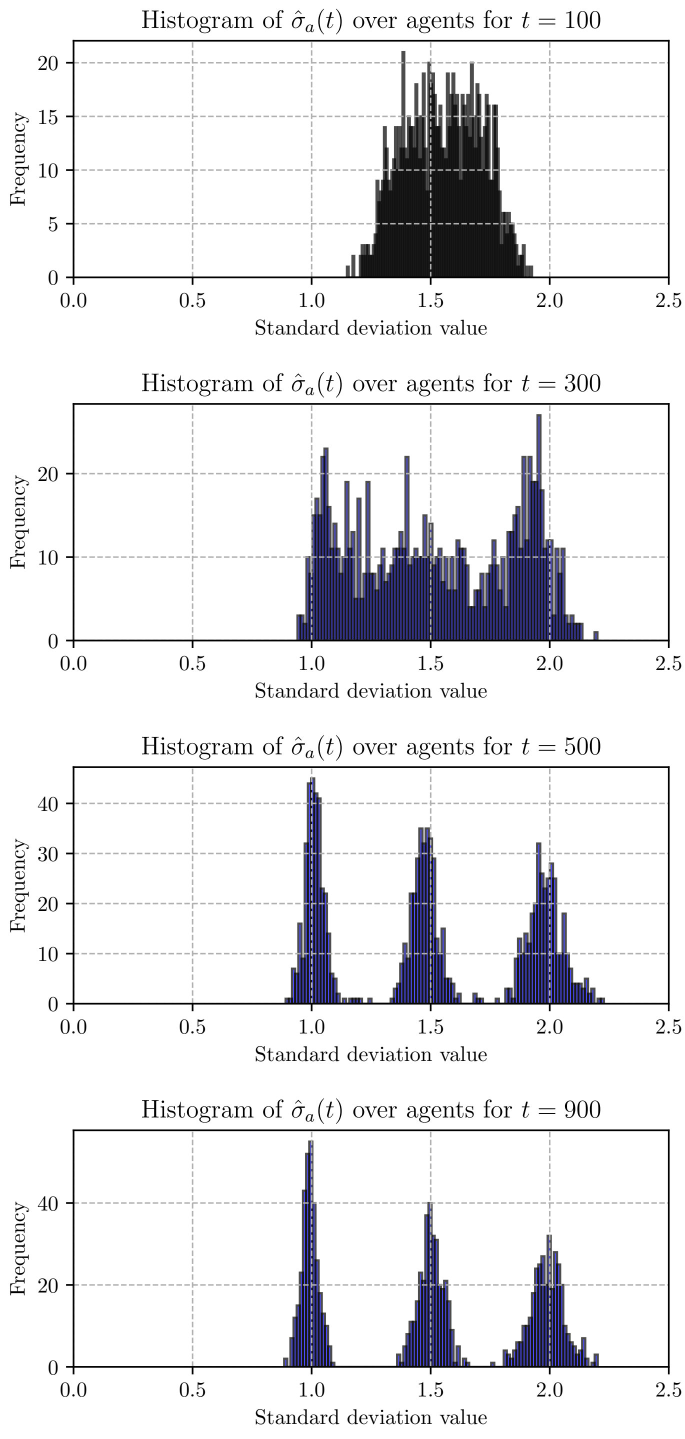


Figure 2.37: Histogram of the estimated standard deviations $\hat{\sigma}_a(t)$ over three-class agents with different means and standard deviations, at $t = 100, 300, 500, 900$ time instants.

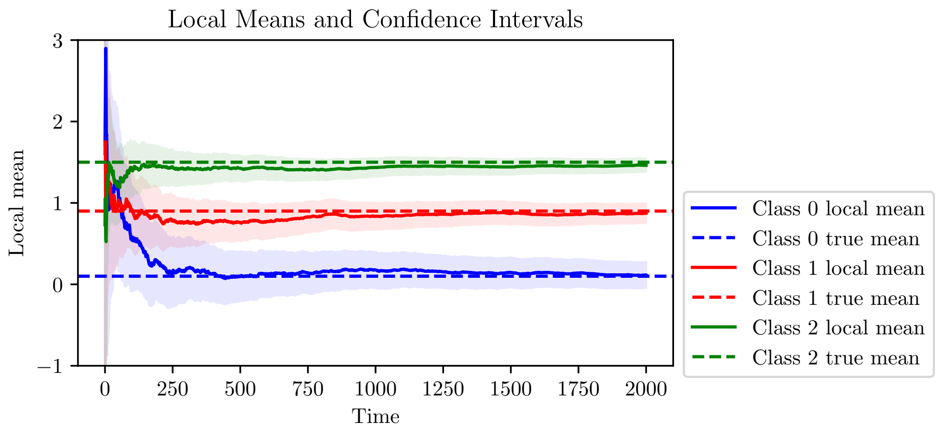


Figure 2.38: Local means and the corresponding confidence intervals for 3 agents belonging to three different classes, as a function of the number of samples t. Standard deviations are estimated by the proposed algorithm.

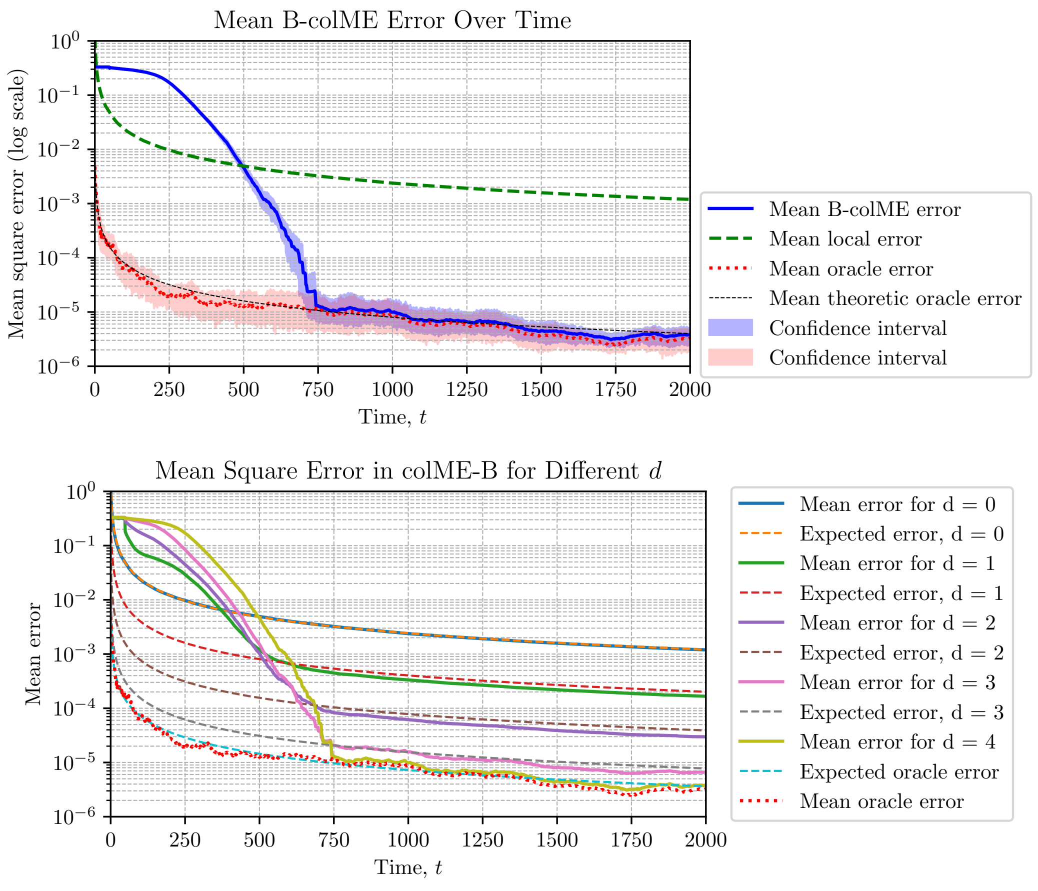


Figure 2.39: B-colME with $N = 5,000$, d up to 4, and $C = 3$ classes: Total MSE of the estimation The results are averaged over 10 realizations.

2.2.6 B-colME with Forgetting for Time-Varying Data

If the distribution of data changes in time, then once accumulated data in local sums will not be removed. They will cause extremely slow convergence if a change occurs, for example if the mean value $(0.1, 0.9)$ changes to $(0.9, 0.1)$ for classes at $t = 2000$.

In this case we have to introduce forgetting factor for local sums and time (for number of terms in sums) used in collaborative mean calculation only:

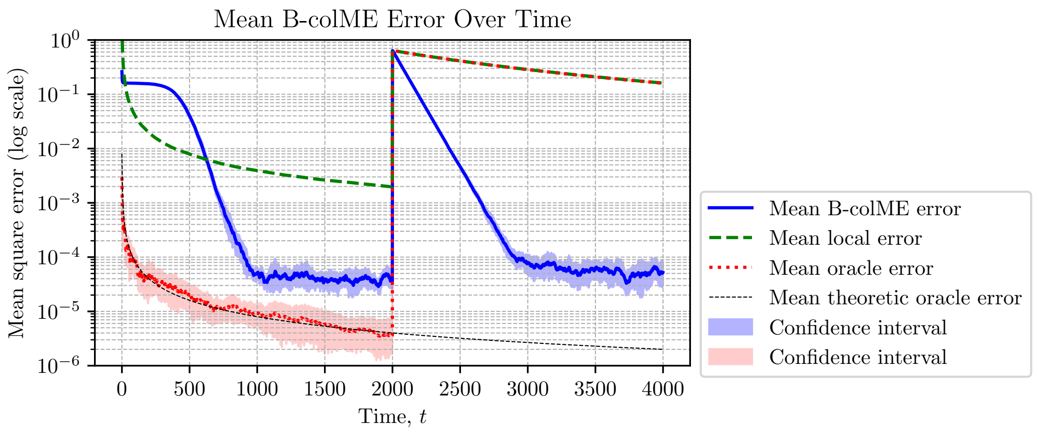
$$\mathbf{M}_{Forg}(t) = \lambda\ \mathbf{M}_{Forg}(t-1) + \mathbf{x}(t) \tag{2.40}$$

and

$$t_{Forg} = \lambda\ t_{Forg} + 1 \tag{2.41}$$

With other relations as in B-colME and for example, $\lambda = 0.995$, with this new cumulative time and sums, we will be able to track time changes with forgetting factor λ.

The results are shown in Figs. 2.40, 2.41, and 2.42.



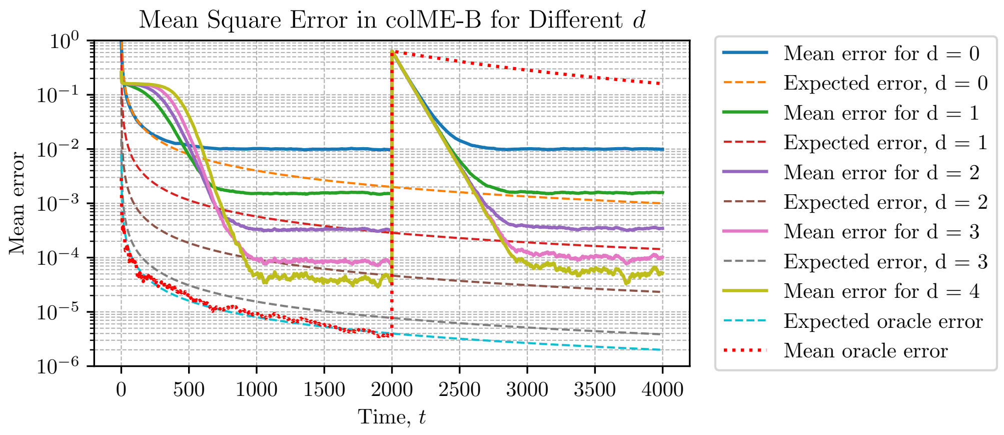


Figure 2.40: (Top) B-colME , $N = 1,000$ $d_{max} = 4$ with distribution change at $t = 2000$: Total MSE of the estimation averaged over all agents at time instants t. The results are averaged over 10 realizations. Confidence intervals for the MSE are calculated using Bootstrap with 0.95 confidence. (Bottom) B-colME with for various d: Total MSE of the estimation. The results are averaged over 10 realizations. *Expected theoretic values are given as well.*

Figures 2.40–2.42 present the performance of the B-colME model with $N = 1{,}000$ agents and different agents' depth value up to $d_{\max} = 4$ under a distribution change occurring at $t = 2000$.

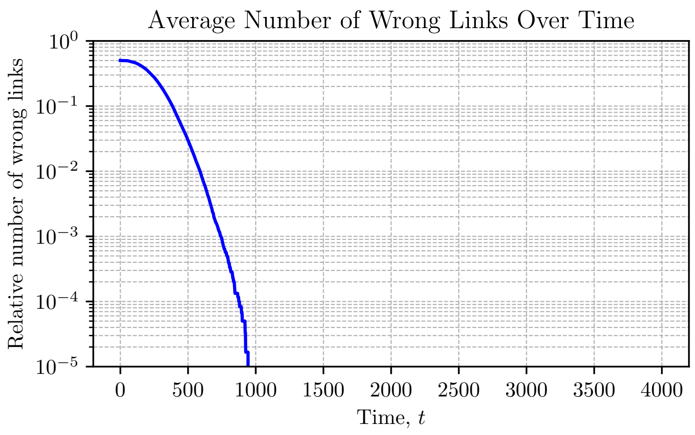


Figure 2.41: B-colME, $N = 1,000$ $d_{max} = 4$ with distribution change at $t = 2000$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time. The results are averaged over 10 realizations.

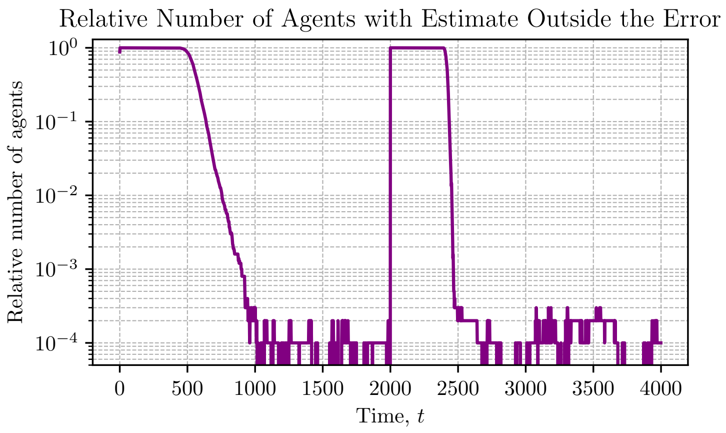


Figure 2.42: B-colME, $N = 1,000$ $d_{max} = 4$ with distribution change at $t = 2000$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time. The results are averaged over 10 realizations.

Figure 2.40 shows the evolution of the total mean squared error (MSE). In the top panel, the MSE is plotted for the baseline case, where we clearly observe an increase in error immediately after the distribution change, followed by convergence toward new steady-state values. The bottom panel reports the effect of different neighborhood sizes d, with theoretical values included for comparison.

Figure 2.41 depicts the average number of wrong links over time. The results show that the average number goes below $10^{-}5$ and goes far below it. It gets so small that even the distribution change is not visible on the graph since it only goes as low as $10^{-}5$

Finally, Figure 2.42 shows the number of agents whose estimation error exceeds the threshold $\varepsilon = 0.1$. After the distribution change, the number of such agents rises sharply but then steadily declines, reflecting the recovery of estimation accuracy as the system stabilizes.

2.3 Reconnection Algorithm

2.3.1 Isolated Agents

We can see that in graph domain settings, it can happen that all neighbors of some agents are from other similarity class(classes). For example, if we use neighbors of $r = 10$, then the expected number of agents with all neighbors of nonsimilarity classes is 1 in 1024 (with $C = 2$ classes). This agent will be *isolated* and will not communicate with any other agent, using any method. Its estimation mean will be equal to its local mean. So, if we evaluated our system by the performance of the worst agent, then it would be very often close to the local mean approach.

To avoid this effect, it is very important to define a simple localized, decentralized, and online approach for the *reconnection of agents* with a critically low number of remaining connections.

Here we will present such an approach. In addition to the neighborhood, $\mathcal{N}_a$ with cardinality $r = |\mathcal{N}_a|$, which is given to each agent, we give alternative nodes $\mathcal{M}_a$, with cardinality $k = |\mathcal{M}_a|$ for possible connections, in the case where the number of connections becomes critically low, for example less than 3.

Then the algorithm works in the following way:

- If the number of connections $|\mathcal{N}_a(t)|$ for agent a becomes less than 3, then this agent tries to connect one randomly chosen node a' from $\mathcal{M}_a$.
- If that agent a' is not with the maximum number of connections, $|\mathcal{N}_{a'}(t)| < r$ it will allow connection with a, otherwise it will refuse connection.
- If the connection is refused, our agent will try to connect to another node in $\mathcal{M}_a$ at the next time instant.
- When agent a connects successfully to node a' from $\mathcal{M}_a$, then that agent is removed from the reconnection list $\mathcal{M}_a$.

In this way, we significantly decrease the probability of getting an isolated agent. For example, if $k = r = 10$ the probability of an isolated agent reduces from 1 in 1,024 to 1 in 1,048,576, that is, from $\propto 10^{-3}$ to $\propto 10^{-6}$.

2.3.2 Improving convergence by Reducing Confidence Intervals

Reconnection will be used here along with the confidence intervals with reduced widths.
Confidence intervals overestimation and convergence: We have discussed that the overestimation of confidence intervals (or variance) may slowdown the convergence. The authors in [2] were aware of this fact and proposed **soft weighting** (and its variant aggressive weighting, when for some agents zero weights are used, but they are not excluded from $\mathcal{N}_a(t)$) to address the problem. Namely, in [2] they "introduced a heuristic weighting mechanism which leverages the intuition that the more the confidence intervals of two agents overlap, the more likely that they are in the same class. Moreover, the smaller the union of the agent means, the more confident we are that the agents are in the same class. In other words, we are

not equally confident about all the agents that are selected for estimation, and this weighting mechanism incorporates this information."

To better understand this problem and justify another approach to this problem, consider a simple example of mean estimation in a system with 1000 agents, divided into two classes, with means 0.1 and 0.9 and variance 2. Assume that the data (their local means) are Gaussian. The Gaussian confidence interval will be defined with $\delta = 0.025$ and $z_\alpha = 2.24$. Its deviation is $\pm 2.24\sigma/\sqrt{t}$. It means that at an instant t there will be 2.5 % of collaborative agents outside the confidence intervals. Now we consider the time instant $t = T_{sep}$, when the **expected confidence intervals** of two classes

$$\mathbb{E}\{\mathbb{I}_a(t)\} = \mathbb{E}\{[\bar{x}_{a,a}(t) - \beta_\delta(t), \bar{x}_{a,a}(t) + \beta_\delta(t)]\} = [\mu_a - \beta_\delta(t), \mu_a(t) + \beta_\delta(t)], \tag{2.42}$$

$$\mathbb{E}\{\mathbb{I}_{a'}(t)\} = \mathbb{E}\{[\bar{x}_{a',a'}(t) - \beta_\delta(t), \bar{x}_{a',a'}(t) + \beta_\delta(t)]\} = [\mu_{a'} - \beta_\delta(t), \mu_{a'}(t) + \beta_\delta(t)] \tag{2.43}$$

touches, that is when $|\mu_a - \mu_{a'}| = 2\beta_\delta(t) = 2|I_a|$, or

$$2|I_a| = 2 \times 2.24\frac{\sigma}{\sqrt{t}} = \frac{8.96}{\sqrt{t}} = 0.9 - 0.1 = 0.8 \tag{2.44}$$

with $t = T_{sep} = 125.44$. Recall that the expected separation time for the Laplacian bounds and the same means and variance was much later, $T_{sep} = 293$.

Note that the Laplace bounds in our numerical examples are calculated with $\delta = 0.1$, as in [3]. They are already very reduced with respect to the that would be obtained with the same δ as in the Gaussian case.

If we use Laplace bounds with $\delta = 0.1$, then the confidence interval bound at $t = 125.44$ is

$$\beta_\delta(t = 125.44) = \sigma\sqrt{\frac{2}{t}\Big(1 + \frac{1}{t}\Big)\ln(\sqrt{t+1}/(0.1/2))} = 0.59 \tag{2.45}$$

what is equal to $1.5|I_a|$. It overestimates the bound for about 50 % (if we used the same $\delta = 0.025$, then we wold get $1.65|I_a|$). This bound will completely avoid that 2.5 % of collaborative agents are excluded, but will wrongly consider as collaborative a huge percent of agents with $\mathbb{P}(\mu_a(t) > \frac{2.24}{2}\frac{\sigma}{\sqrt{t}}) = 0.1314$ or about 13 % of agents (or 23 % with $\delta = 0.025$). So, the ratio of wrong non-collaborative agents and collaborative agents will be slightly greater than 5:1 for $\delta = 0.1$ in the Laplacian bound (or almost 10:1 for the same δ as in Gaussian bound).

Proposed approach: Instead of keeping a big percentage of wrong agents, even with reduced influence with application of the soft weights, while keeping very small number of collaborative agents, we can:

- Reduce appropriately the Laplacian confidence intervals (for the case of $\delta = 0.1$, reduce the interval for about 33 %). In this way, we will remove almost all errors from wrong non-collaborative agents.
- Reconnect to solve the problem that some collaborative agents (in our example, one in five or one in ten) are disconnected.

This can be *more accurate and computationally more efficient* approach to this problem.

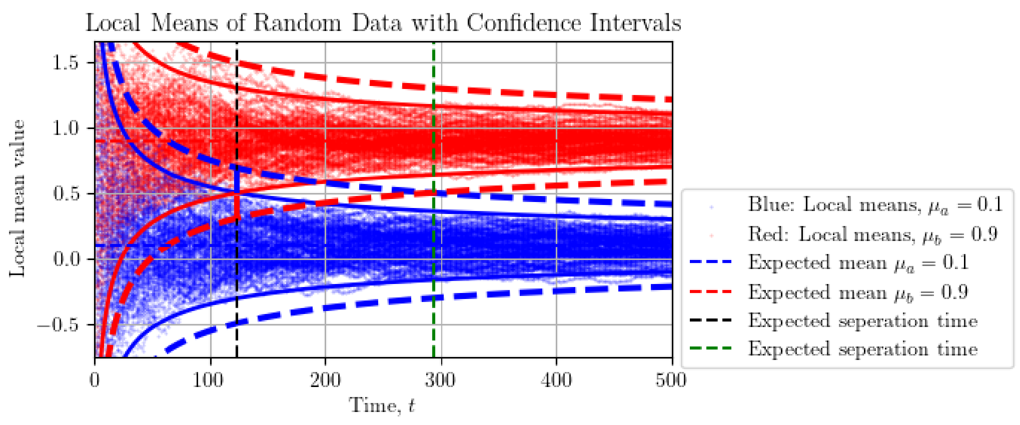


Figure 2.43: Illustration of the *expected confidence intervals*, $\mathbb{E}\{\mathbb{I}_a(t)\}$: Laplacian bounds with $\delta = 0.1$ shown by dashed line and reduced confidence intervals given by the full line. Expected separation times are given as well

Note: The same effect can be achieved by *underestimating the standard deviation* or by *increasing the parameter* δ in $\beta_\delta(t)$.

2.3.3 Results with Reduced Interval Widths and Reconnection

The results with C-colME with Gaussian confidence interval width and reconnection are shown in Figs. 2.48, 2.50 and 2.51. The convergence is improved with respect to Fig. 2.4.

In all simulations, an estimated (local) standard deviation (Algorithm 1) is used.

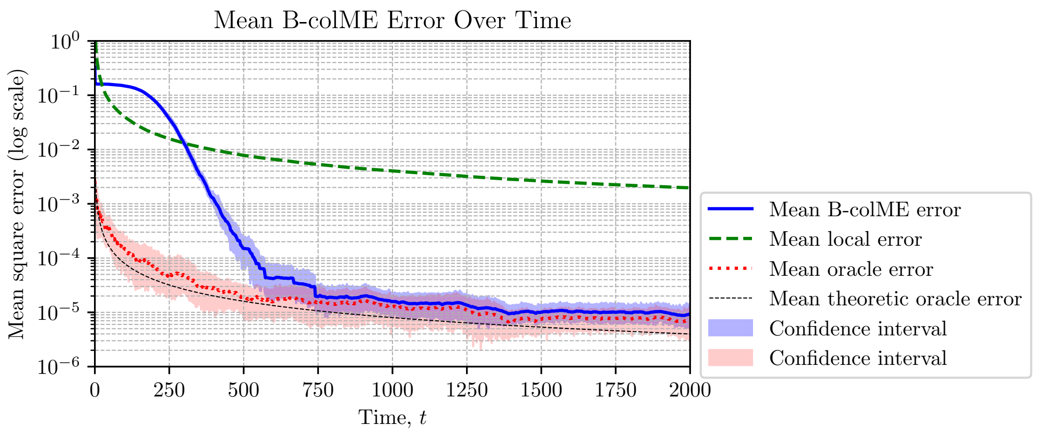
Figures 2.44–2.56 illustrate the effect of reconnection and reduced confidence interval widths on both B-colME and C-colME algorithms with $N = 1000$ agents.

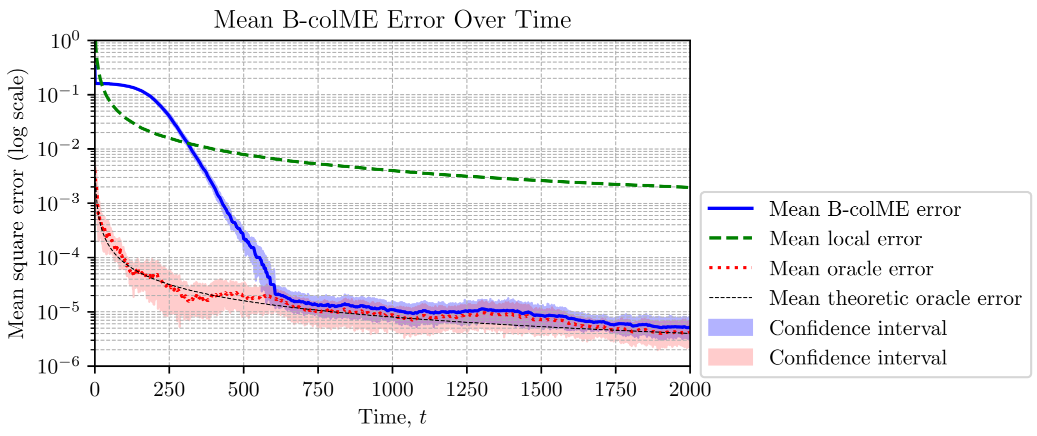
To improve algorithm convergence, the confidence interval widths are reduced using those defined for the Gaussian distribution with $\delta = 0.025$. In addition, isolated agents—either intrinsic or arising due to the higher probability of removing agents from the same similarity class—are eliminated using the presented reconnection algorithm.

For the B-colME case, Figure 2.44 shows the total MSE of the estimation under different reconnection thresholds. When agents reconnect if $|\mathcal{N}_a(t)| < 3$ or $|\mathcal{N}_a(t)| < 6$, convergence improves compared to the baseline case without reconnection. The reduced number of links also results in faster computation times. Figure 2.45 demonstrates that reconnection reduces the number of agents with large estimation errors. Figure 2.46 provides examples of confidence intervals for two selected agents, confirming improved stability. Figure 2.47 compares MSE performance across different neighborhood sizes d, again showing gains from reconnection.

The effects of intensive reconnection with a larger threshold ($|\mathcal{N}_a(t)| \leq 5$) and a wider Gaussian-based confidence interval ($\delta = 0.05$) are presented in Figures 2.52–2.56. With these settings, convergence rate is further improved, and both the number of wrong links and the number of agents with large estimation errors are reduced. These results confirm the benefits of combining reconnection with appropriately tuned confidence intervals.

For the C-colME case, Figures 2.48–2.51 report analogous results. The MSE plots in Figure 2.48 clearly show faster and more stable convergence with reconnection ($|\mathcal{N}_a(t)| < 3$ or $|\mathcal{N}_a(t)| < 6$), compared to the no-reconnection baseline. Figure 2.50 shows confidence intervals for two agents, while Figure 2.51 demonstrates that reconnection eliminates isolated-agent errors, significantly improving overall robustness.





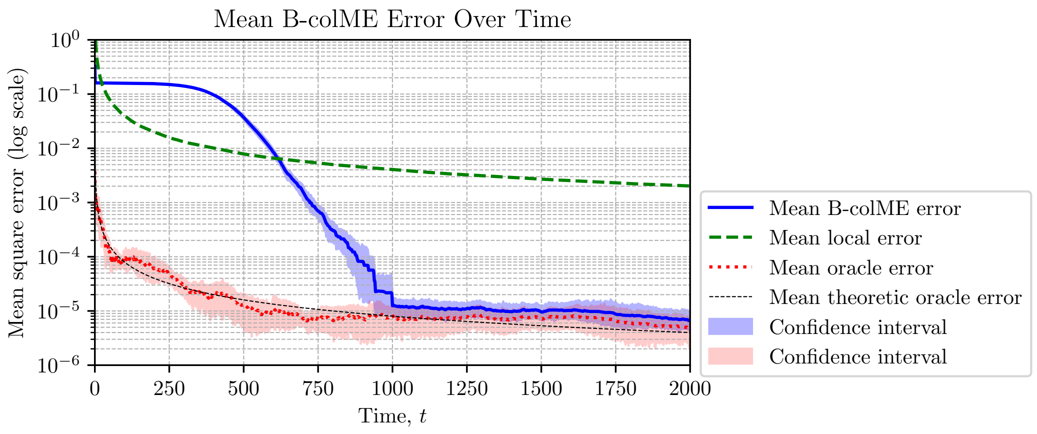


Figure 2.44: B-colME with reconnection, $N = 1000$: (Top) Total MSE of the estimation averaged over all agents at time instants t. Reconnection for $|\mathcal{N}_a(t)| < 3$. (Middle) Total MSE of the estimation averaged over all agents at time instants t. Reconnection for $|\mathcal{N}_a(t)| < 6$. The results are averaged over 10 realizations with reconnection and reduced confidence intervals calculated according to the Gaussian distribution. (Bottom) Repeated B-colME without reconnection. **Calculation time** for Reconnection with $|\mathcal{N}_a(t)| < 3$ is 178 seconds per realization, on average (due to overall reduced number of connections).

Similarly, for C-colME, reconnection and Gaussian confidence intervals improve convergence, reduce errors due to isolated agents, and provide more stable estimates. Overall, these results demonstrate that combining reconnection with reduced Gaussian confidence intervals significantly enhances the performance, stability, and robustness of both B-colME and C-colME algorithms.

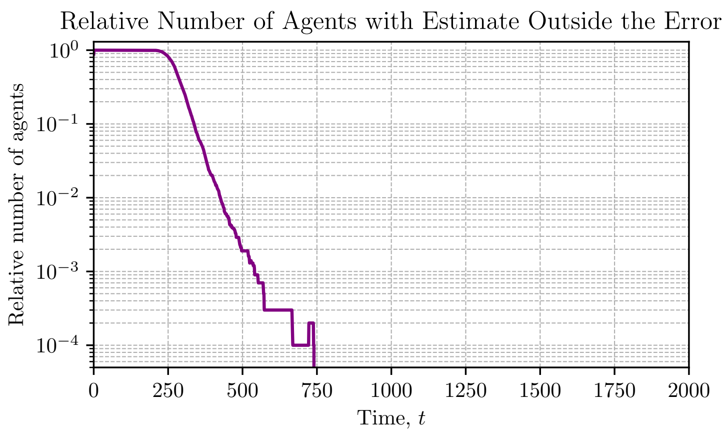


Figure 2.45: B-colME with Reconnection: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time (averaged over 10 realizations) with reconnection.

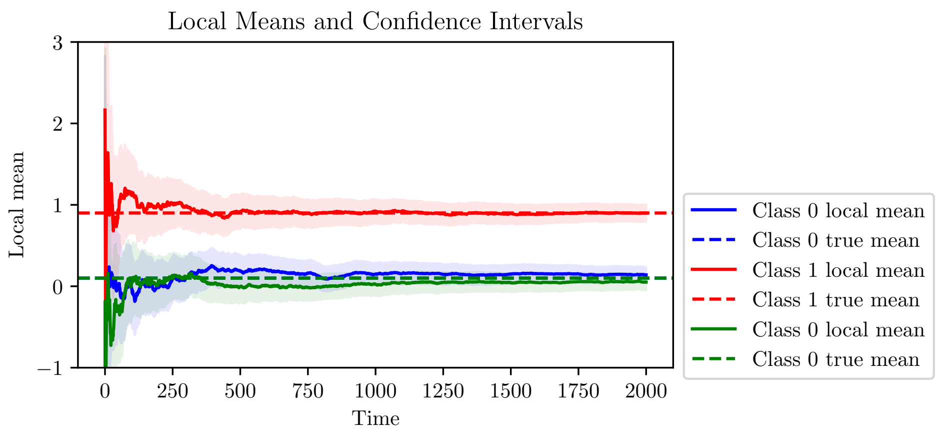


Figure 2.46: B-colME with Reconnection: Means and confidence intervals for two arbitrary chosen agents from two different classes, over time.

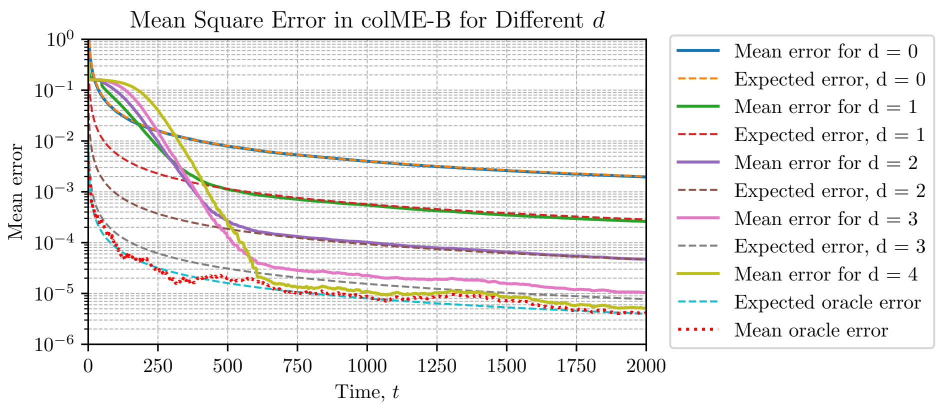


Figure 2.47: B-colME with reconnection, $N = 1000$: Total MSE of the estimation averaged over all agents at time instants t and various d. Reconnection for $|\mathcal{N}_a(t)| < 6$.

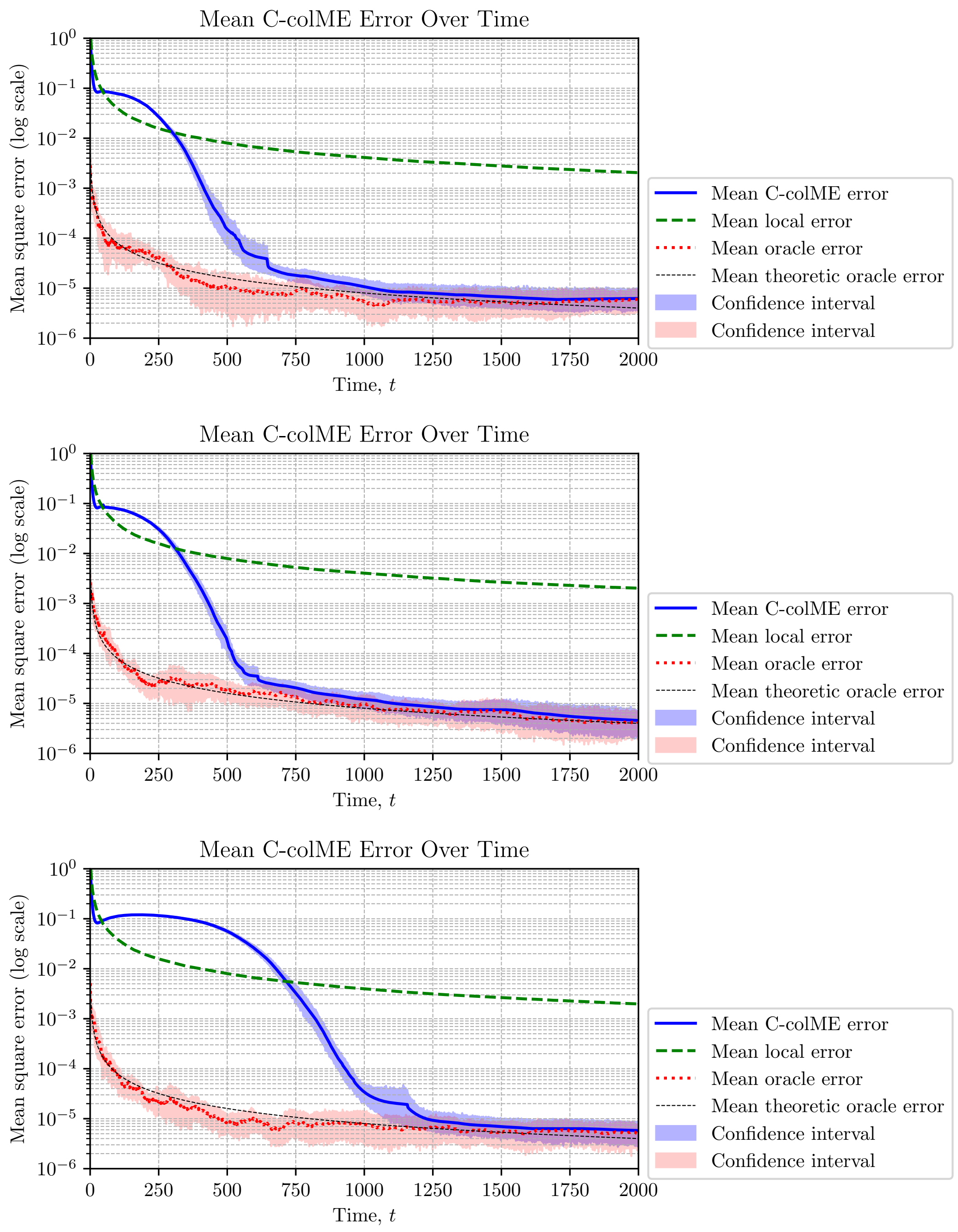


Figure 2.48: C-colME with reconnection, $N = 1000$: (Top) Total MSE of the estimation averaged over all agents at time instants t. Reconnection for $|\mathcal{N}_a(t)| < 3$. (Middle) Total MSE of the estimation averaged over all agents at time instants t. Reconnection for $|\mathcal{N}_a(t)| < 6$. The results are averaged over 10 realizations with reconnection and reduced confidence intervals calculated according to the Gaussian distribution. (Bottom) Repeated C-colME without reconnection from Fig.2.4. **Calculation time** is 103 seconds per realization, on average (due to overall reduced number of connections).

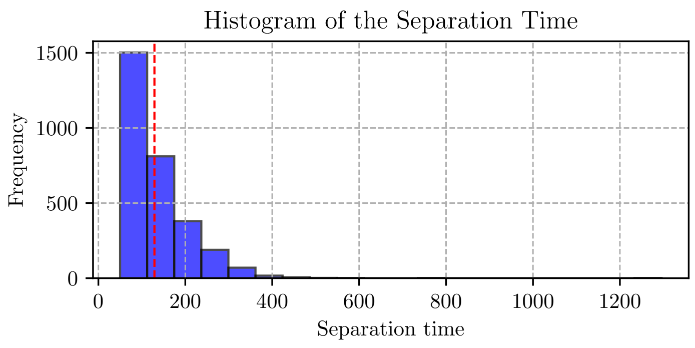


Figure 2.49: B-colME with reconnection: Histogram of the separation time.

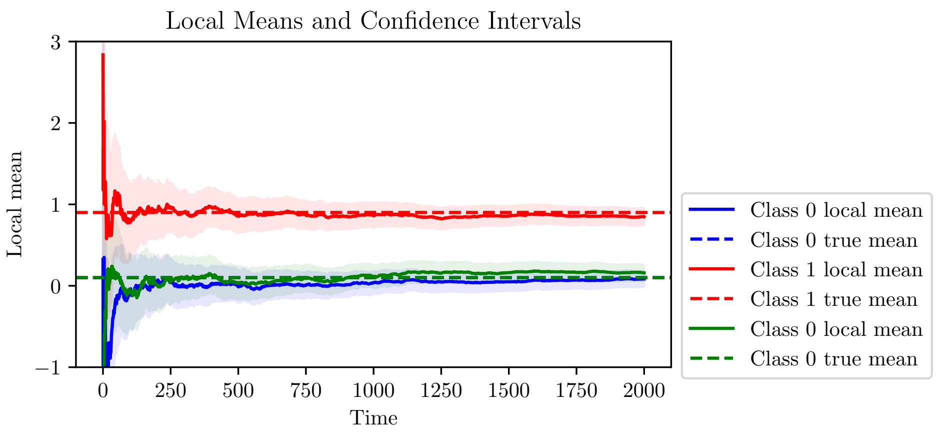


Figure 2.50: C-colME with reconnection: Means and confidence intervals for two arbitrary chosen agents from two different classes, over time.

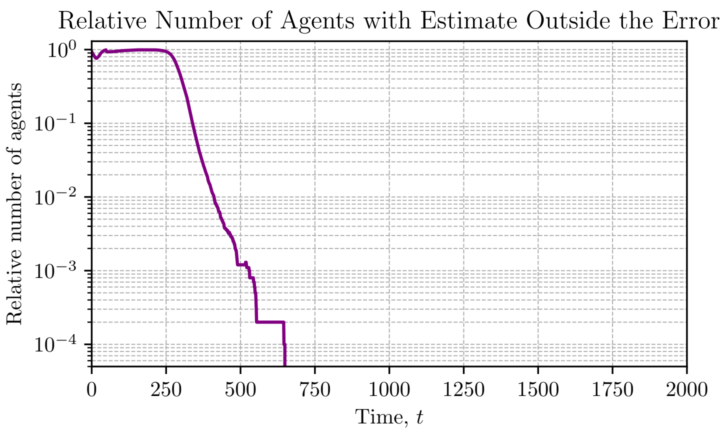
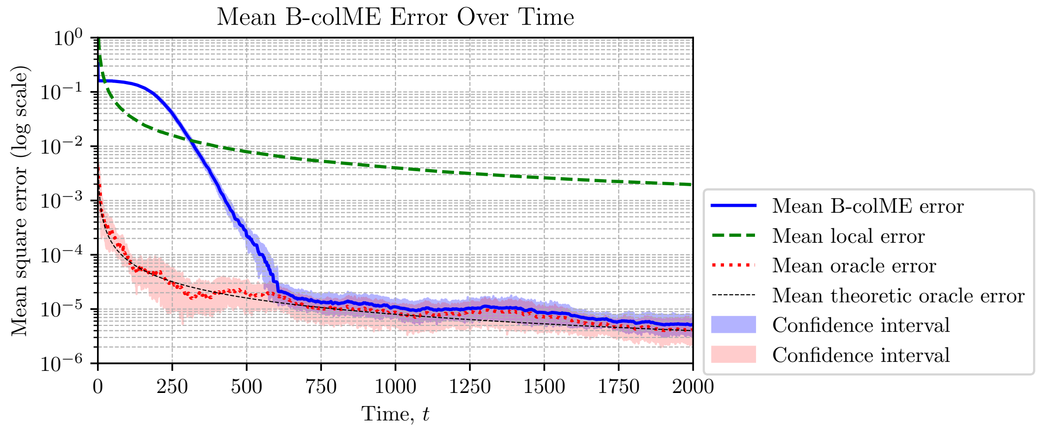


Figure 2.51: C-colME with reconnection: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time (averaged over 10 realizations) with reconnection. Obviously, there is no error due to isolated agents.



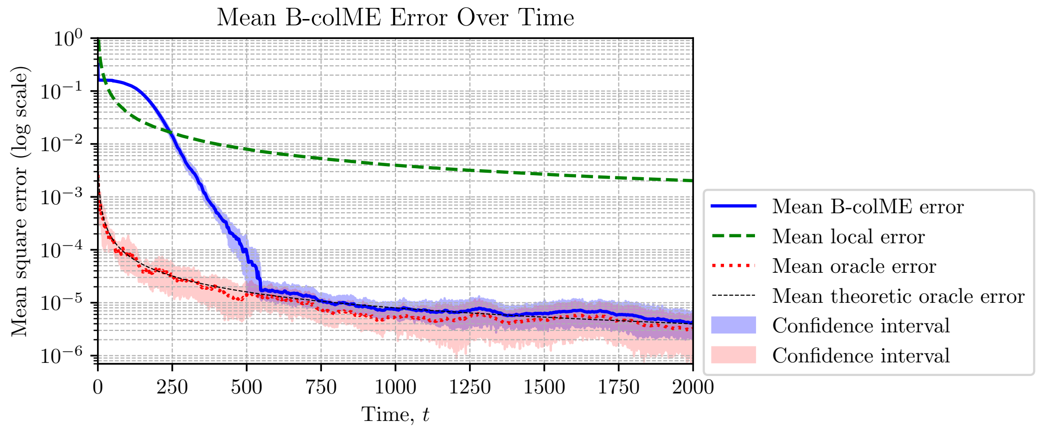


Figure 2.52: B-colME with reconnection, $N = 1000$: (Top) Total MSE of the estimation averaged over all agents at time instants t. Reconnection for $|\mathcal{N}_a(t)| < 6$ and $\delta = 0.025$. (Bottom) Total MSE of the estimation averaged over all agents at time instants t. Reconnection for $|\mathcal{N}_a(t)| < 6$ and $\delta = 0.05$. **Calculation time** for Reconnection with $|\mathcal{N}_a(t)| < 6$ is 218 seconds per realization, on average (due to overall reduced number of connections).

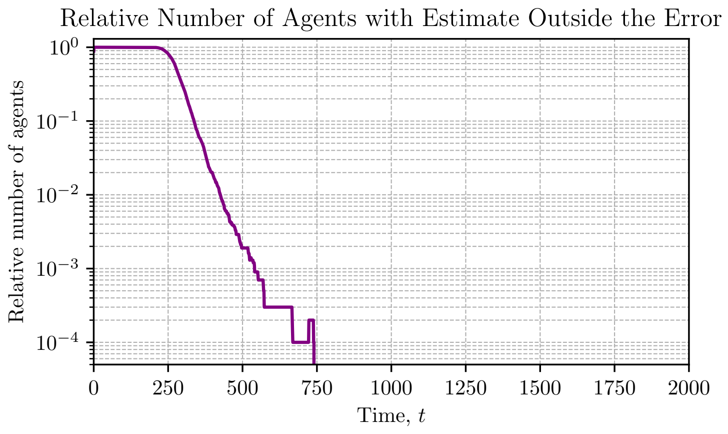


Figure 2.53: B-colME with Reconnection, $\delta = 0.05$: Number of agents with absolute error greater than $\varepsilon = 0.1$ over time (averaged over 10 realizations) with reconnection.

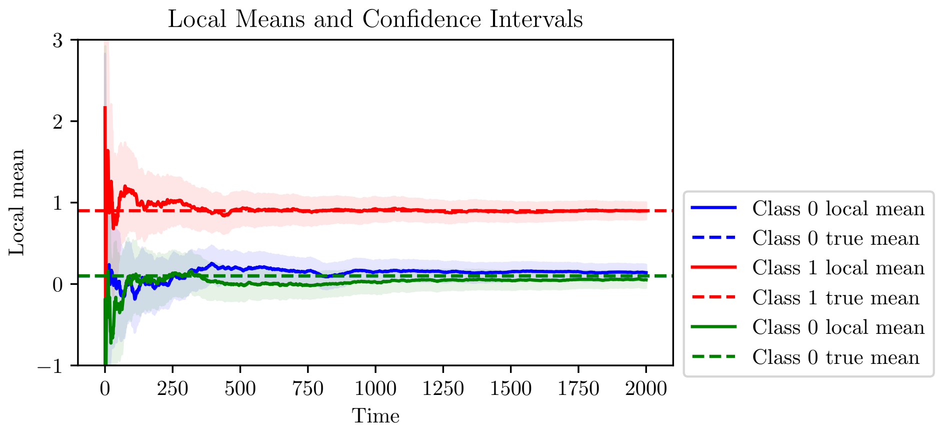


Figure 2.54: B-colME with Reconnection, $\delta = 0.05$: Means and confidence intervals for two arbitrary chosen agents from two different classes, over time.

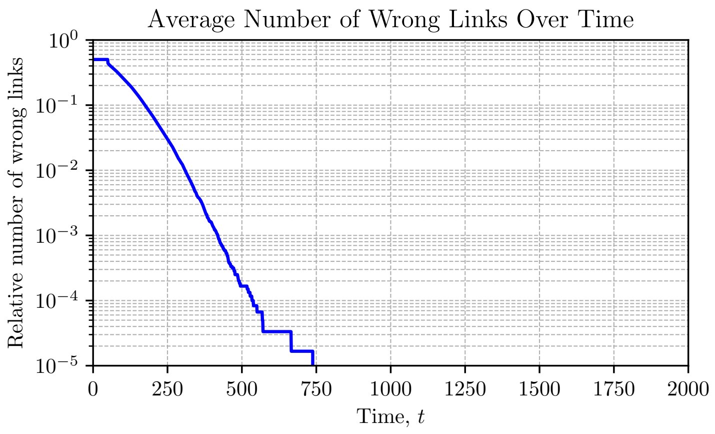


Figure 2.55: B-colME with Reconnection, $\delta = 0.05$: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time..

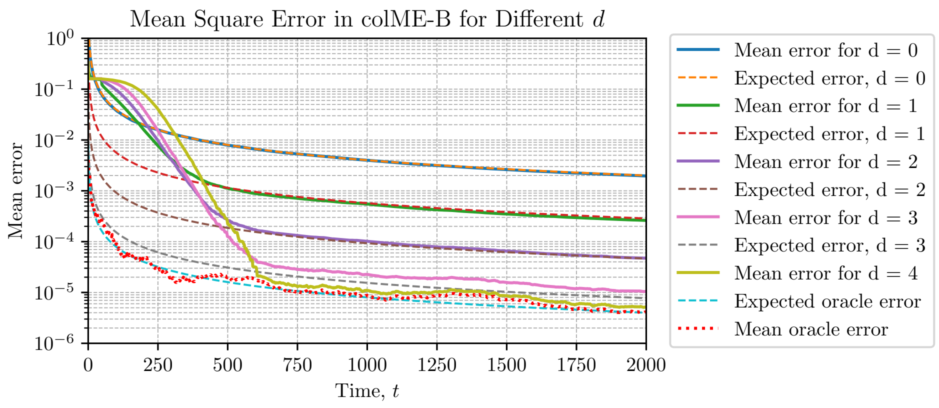


Figure 2.56: B-colME with reconnection, $N = 1000$, $\delta = 0.05$: Total MSE of the estimation averaged over all agents at time instants t and various d. Reconnection for $|\mathcal{N}_a(t)| < 6$.

2.4 Weighted Graphs in C-colME and B-colME

The idea of reducing the confidence intervals was implemented in the previous section by using narrower interval bounds and the reconnection approach. Here, we will implement the same idea, but instead of a direct reduction in confidence interval width, we will use a weighted graph to effectively reduce confidence intervals by lowering data weights. The graph weights are defined by using the distance between local means of considered agents. If the distance is greater, then the weight is lower and vice versa. This approach is inspired by soft weighting in [2].

If two confidence intervals

$$\mathbb{I}_a(t) = [\bar{x}_{a,a}(t) - \beta_\delta(t), \bar{x}_{a,a}(t) + \beta_\delta(t)] \tag{2.46}$$

$$\mathbb{I}_{a'}(t) = [\bar{x}_{a',a'}(t) - \beta_\delta(t), \bar{x}_{a',a'}(t) + \beta_\delta(t)] \tag{2.47}$$

fully overlap with $\bar{x}_{a,a}(t) = \bar{x}_{a',a'}(t)$ then

$$D_{a,a'}(t) = |\bar{x}_{a,a}(t) - \bar{x}_{a',a'}(t)| - 2\beta_\delta(t) = -2\beta_\delta(t) \tag{2.48}$$

The distance between local means, which is in this case equal to zero, is obtained as $|D_{a,a'}(t)| - 2\beta_\delta(t)$. In the case when two confidence intervals just touch each other $|\bar{x}_{a,a}(t) - \bar{x}_{a',a'}(t)| = 2\beta_\delta(t)$ then

$$D_{a,a'}(t) = |\bar{x}_{a,a}(t) - \bar{x}_{a',a'}(t)| - 2\beta_\delta(t) = 0 \tag{2.49}$$

Our aim is to favor the data in the first case ($\bar{x}_{a,a}(t)$ close to $\bar{x}_{a',a'}(t)$), with the weights almost equal to 1, and to reduce the weight towards 0 for the other case when $\bar{x}_{a,a}(t)$ and $\bar{x}_{a',a'}(t)$ significantly differ. We can do that in many ways. The most common way in graph theory is defined using an exponential function for the matrix **AW**, with elements

$$AW_{a,a'}(t) = e^{-\left(\eta \frac{\bar{x}_{a,a}(t) - \bar{x}_{a',a'}(t)}{2\beta_\delta(t)}\right)^m} \quad \text{when } A_{a,a'}(t) = 1 \tag{2.50}$$

and $AW_{a,a'}(t) = 0$ when $A_{a,a'}(t) = 0$. Note that

$$0 \leq AW_{a,a'}(t) \leq 1 \tag{2.51}$$

The graph with these weights and connectivity defined by $A_{a,a'}(t)$ is called a weighted graph. All calculations are performed in the same way, with $A_{a,a'}(t)$ replaced by $AW_{a,a'}(t)$.

We will use $\eta = 2$ and $m = 4$, which would roughly correspond to taking data from half of the confidence interval.

2.4.1 Simulation Results with a Weighted Graph and C-colME

The same simulation setup is used as in the previous examples. The number of agents is $N = 1000$.

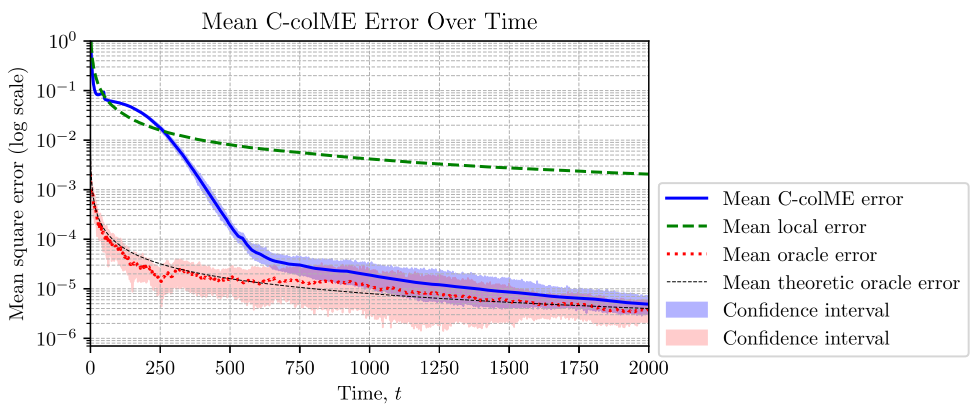


Figure 2.57: C-colME on weighted graph, $N = 1000$, $\delta = 0.1$: Total MSE of the estimation averaged over all agents at time instants t in 10 realizations. **Calculation time** is 389 seconds per realization, on average (increased due to float form of weighting matrix).

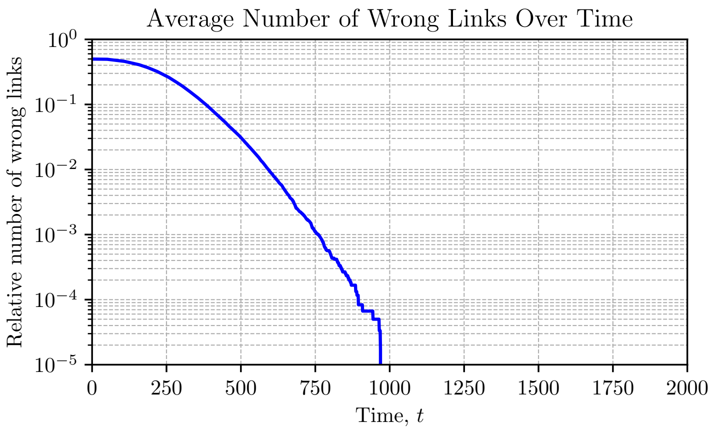


Figure 2.58: C-colME on weighted graph: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time.

2.4.2 Simulation Results with a Weighted Graph and B-colME

The simulation from the previous section is repeated with B-colME on the weighted graph. In B-colME, we keep the matrix $\mathbf{A}$ for all neighborhood determinations (neighbors, neighbors of neighbors, etc.), and the weighted matrix $\mathbf{AW}$ is used only for weighting of the sums and the number of terms in collaborative mean calculation.

2.4.3 Weighted Graph in B-colME and C-colME with Reconnection

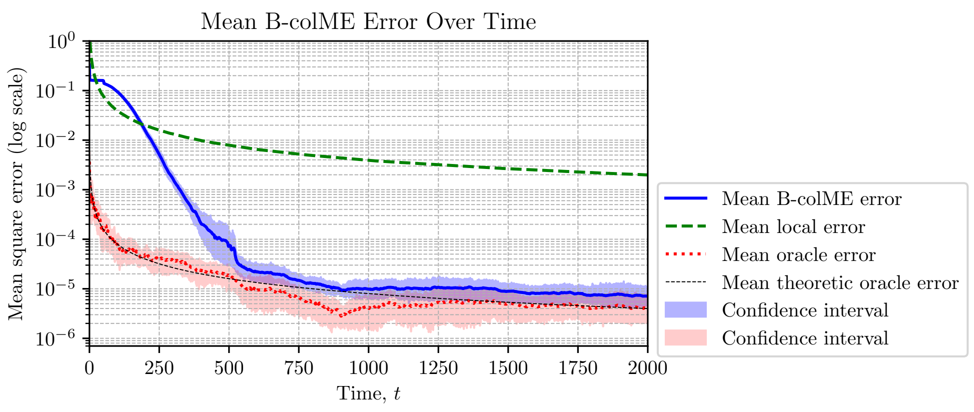
Now we combine the weighted graph with a reduced confidence interval width and reconnection and apply them to the B-colME and C-colME.

The results are shown in Fig. 2.61 and Fig. 2.62.

2.4.4 Simulation Results on Weighted Graphs

Figures 2.57–2.60 present the performance of C-colME and B-colME algorithms on weighted graphs with $N = 1000$ agents and $\delta = 0.1$ for Gaussian-based confidence intervals.

For C-colME, Figure 2.57 shows the total mean squared error (MSE) averaged over all agents. Compared to the unweighted graph case (Fig. 2.4), convergence is significantly faster



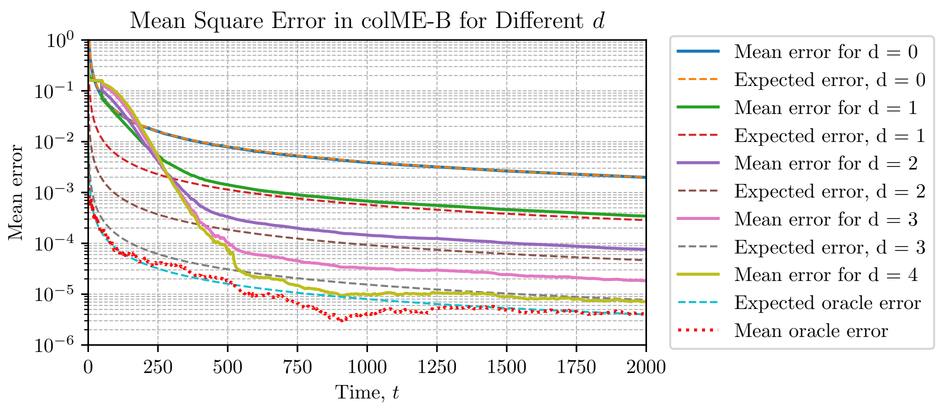


Figure 2.59: B-colME on weighted graph, $N = 1000$, $\delta = 0.1$: Total MSE of the estimation averaged over all agents at time instants t in 10 realizations. **Calculation time** is 599 seconds per realization, on average (increased due to float form of weighting matrix).

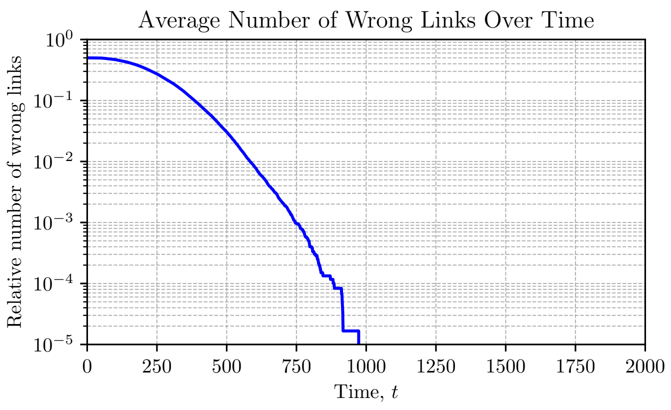
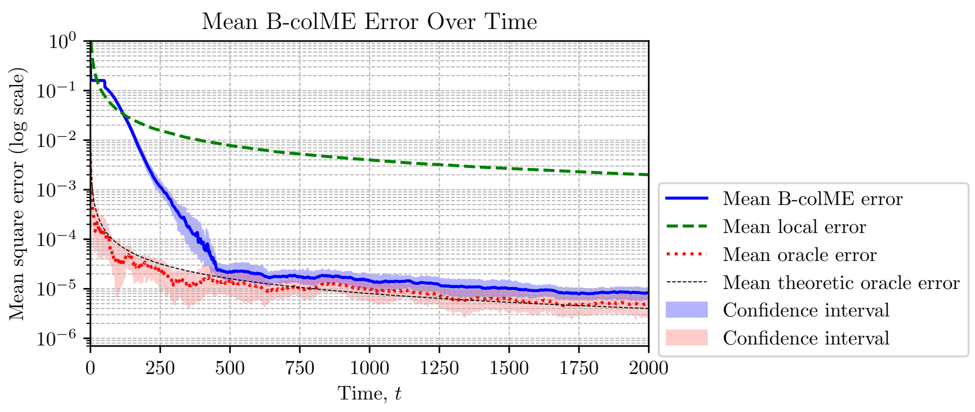


Figure 2.60: B-colME on weighted graph: Total number of wrong links.

because weighting effectively narrows the confidence intervals. Figure 2.58 shows the total number of wrong links over time, which behaves similarly to the unweighted case since the confidence intervals and graph pruning remain unchanged.

For B-colME on weighted graphs, Figures 2.59 and 2.60 provide analogous results. The matrix **A** is used for neighborhood determination, while the weighted matrix **AW** is used for weighting the collaborative mean calculations. Figure 2.59 demonstrates that convergence is faster than in the unweighted case (Figs. 2.25), while Figure 2.60 shows that the number of wrong links is comparable to the unweighted case (Fig. 2.26), due to unchanged confidence intervals and graph pruning.



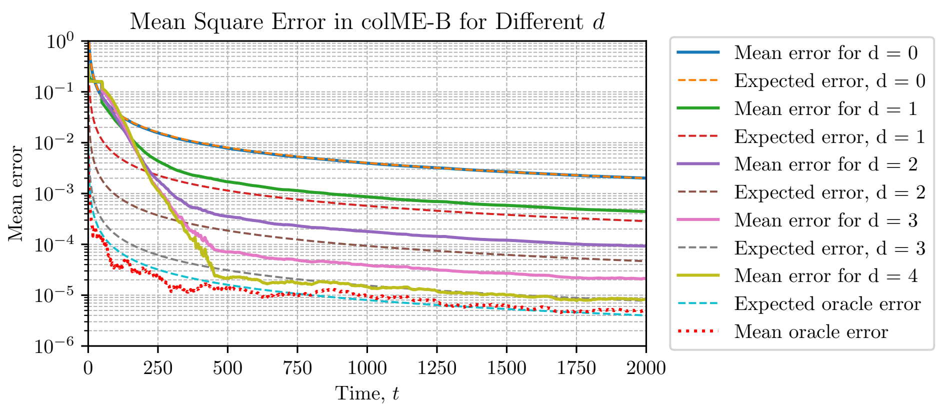


Figure 2.61: B-colME on weighted graph with reconnection, $N = 1000$, $\delta = 0.01$, Gaussian confidence interval bounds: Total MSE of the estimation averaged over all agents at time instants t in 10 realizations. **Calculation time** is 578 seconds per realization, on average (increased due to float form of weighting matrix, reduced for faster graph pruning).

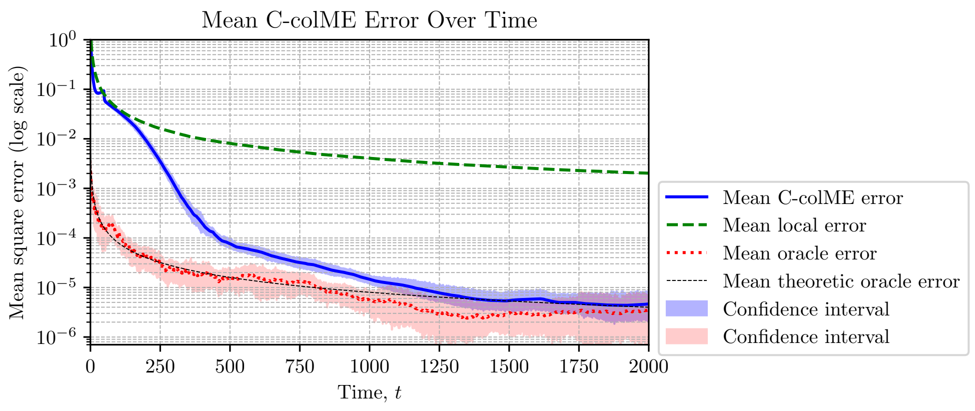


Figure 2.62: C-colME on weighted graph with reconnection, $N = 1000$, $\delta = 0.01$, Gaussian confidence interval bounds: Total MSE of the estimation averaged over all agents at time instants t in 10 realizations. **Calculation time** is 382 seconds per realization, on average (increased due to float form of weighting matrix, reduced for faster graph pruning).

2.4.5 Weighted Graphs with Reconnection and Reduced Confidence Intervals

Figures 2.61 and 2.62 combine weighted graphs with reconnection and reduced Gaussian confidence intervals ($\delta = 0.01$).

For B-colME, Figure 2.61 shows that convergence is further accelerated when reconnection is applied alongside the weighted graph. MSE is lower and stabilizes faster compared to the baseline weighted graph case. The results for various neighborhood distances d are also shown, confirming that the benefits of reconnection and weighting are consistent across different neighborhood sizes.

For C-colME, Figure 2.62 illustrates similar improvements. The combination of weighting and reconnection reduces the MSE significantly and speeds up convergence compared to the unweighted or no-reconnection cases. Calculation times are slightly increased due to the float representation of the weighted matrices, but reduced graph pruning compensates, keeping computation times reasonable.

Overall, these simulations demonstrate that weighting the graph accelerates convergence by effectively narrowing the confidence intervals, while combining weighting with reconnection and reduced confidence intervals further enhances convergence, stability, and robustness for both B-colME and C-colME algorithms.

2.5 Consensus Laplacian-Based Variant Cl-colME

Consider the data set $x_a(t)$ in the graph $\mathcal{G}(\mathcal{A})$ with adjacency matrix $\mathbf{A}$. The Laplacian on this graph is defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal matrix (called degree matrix) with diagonal elements equal to the number of neighbors of each agent (sum of the row elements in $\mathbf{A}$), that is

$$D_{aa} = |\mathcal{N}_a| = \sum_j A_{a,j} \tag{2.52}$$

Data $x_a(t)$ smoothness on the graph is defined by

$$\mathcal{L} = \sum_a \sum_j A_{a,j}(x_a(t) - x_j(t))^2 = \mathbf{x}^T\mathbf{L}\mathbf{x} \tag{2.53}$$

The data smoothness is a non-negative function, having minimum $\mathcal{L} = 0$ when $x_a(t) = x_j(t) =$ *constant* for each node a within a connected component of the graph.

To achieve a smooth version $\mu_a(t)$ of local means $X_a(t)$, we can use steepest descent approach, moving by small steps in the direction of negative gradient of smoothness of $\mu_a(t)$ on graph defined by $\boldsymbol{\mu}^T\mathbf{L}\boldsymbol{\mu}$,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}^T} = 2\mathbf{L}\boldsymbol{\mu}(t)) \tag{2.54}$$

or in the standard iterative steepest descent form

$$\boldsymbol{\mu}(t) = \boldsymbol{\mu}(t-1) - \beta\mathbf{L}\boldsymbol{\mu}(t-1) = (\mathbf{I} - \beta\mathbf{L})\boldsymbol{\mu}(t-1) \tag{2.55}$$

with $\boldsymbol{\mu}(0) = \mathbf{X}(0)$ being the initial vector of local means.

Since we want to include local means for initial period, until the graph is sufficiently separated and wrong connections are excluded, we add local means and use the iterative formula (in consensus form)

$$\boldsymbol{\mu}(t) = (1 - \alpha(t))\mathbf{X}(t) + \alpha(t)(\mathbf{I} - \beta\mathbf{L})\boldsymbol{\mu}(t-1) \tag{2.56}$$

or

$$\boldsymbol{\mu}(t) = (1 - \alpha(t))\mathbf{X}(t) + \alpha(t)(\mathbf{I} - \beta\mathbf{D} + \beta\mathbf{A})\boldsymbol{\mu}(t-1) \tag{2.57}$$

Obviously, this formula produces unbiased estimate since $\mathbf{L}$ multiplied by any constant vector is 0. Also, from $\boldsymbol{\mu}(t) = \boldsymbol{\mu}(t-1) - \beta\mathbf{L}\boldsymbol{\mu}(t-1) = (\mathbf{I} - \beta\mathbf{L})\boldsymbol{\mu}(t-1)$ follows that

$$\lim_{t\to\infty} \mu_i(t) = \text{mean}_i\{X_i(t)\} = \mu_a \tag{2.58}$$

within each graph component provided that $|\beta\lambda_{max}| < 1$, where λ_{max} is the maximum eigenvalue of the Laplacian, $\mathbf{L}$. This condition provides that $-1 < (1 - \beta\lambda) < 1$. Note that $\lambda_{min} = 0$ by the Laplacian definition, with multiplicity equal to the number of graph components (classes of agents, after wrong links are disconnected, assuming no isolated agents appear).

Convergence analysis is similar as in Section 2.1.5. For the Laplacian holds

$$\mathbf{L}\mathbf{1} = \mathbf{0} \tag{2.59}$$

It means than $\lambda_1 = 0$ is the eigenvalue for a constant eigenvector. All other eigenvalues λ_i are greater than 0 since the Laplacian is a real-valued symmetric matrix.

The eigenvalues of $\mathbf{I} - \beta\mathbf{L}$ are $1 - \beta\lambda$, where λ_i are the eigenvalues of the Laplacian. Therefore, for the recursion

$$\boldsymbol{\mu}(t) = (\mathbf{I} - \beta\mathbf{L})\boldsymbol{\mu}(t-1) \tag{2.60}$$

holds the same as in Section 2.1.5: The largest eigenvalue of $\mathbf{I} - \beta\mathbf{L}$ is 1, and all other values are less than 1 in absolute value provided that $|\beta\lambda_{max}| < 1$. Then the same conclusion holds that this recursion tends toward class oracle solutions.

Comment on Cl-colME: The convergence rate of C-colME and Cl-colME may be different since the second largest eigenvalue of $\mathbf{I} - \beta\mathbf{L}$ and $\mathbf{W}_t$ are not the same. It has been shown [3] that the double stochastic matrix $\mathbf{W}_t$ minimizes the error obtained by averaging the estimates of all agents in the component. The Laplacian form avoids divisions in calculation of double stochastic matrix, which is the most demanding operation performed for all agents at each time instant.

2.5.1 Simulation Results for Cl-colME and Cl-colME on Weighted Graph

Figure 2.63 compares the total MSE of C-colME, B-colME, and Cl-colME for $N = 5000$ agents. The results are averaged over 10 realizations with $\beta = 0.1$, and for B-colME a neighborhood size of $d = 4$ is used. It can be observed that Cl-colME achieves comparable convergence to the other methods, while maintaining reasonable calculation time. For L-colME, the average computation time per realization is 748 seconds.

The effect of a weighted graph on Cl-colME is illustrated in Figures 2.64 and 2.65. In Figure 2.64, the total MSE is shown for $N = 1000$ agents using Gaussian confidence intervals with $\delta = 0.1$. Compared to the unweighted graph, convergence is faster due to the weighting of the graph, which effectively narrows the confidence intervals. The calculation time is slightly higher than the unweighted $N = 1000$ case due to the float representation of the weighted matrix, but overall it remains lower than the $N = 5000$ simulation.

Figure 2.65 shows the total number of wrong links over time. Similar to the previous cases, the number of wrong links is largely unchanged compared to unweighted simulations, since the confidence intervals and pruning rules remain the same. Overall, these results demonstrate that weighting the graph accelerates convergence in Cl-colME without negatively affecting the number of incorrect links.

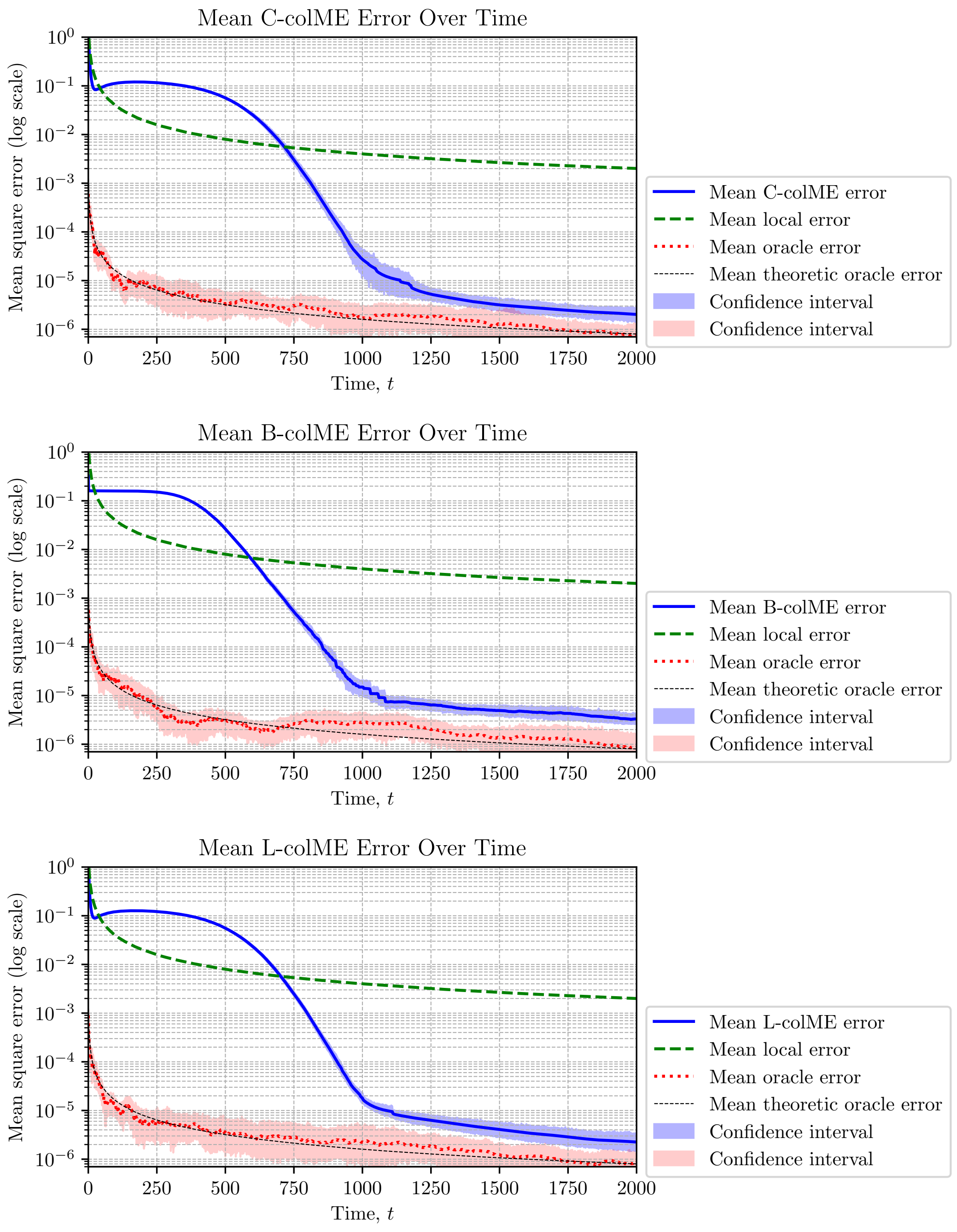


Figure 2.63: C-colME, B-colME, CL-colME (comparison for $N = 5000$): Total MSE of the estimation averaged over all agents at time instants t. The results are averaged over 10 realizations with $\beta = 0.1$, in B-colME we used $d = 4$. **Calculation time** for L-colME is 748 seconds per realization, on average.

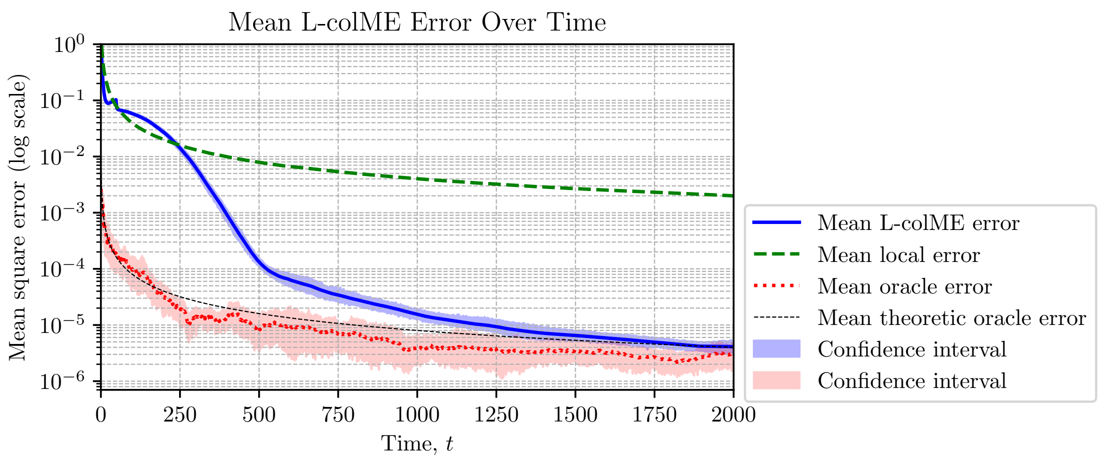


Figure 2.64: CL-colME on weighted graph, $N = 1000$, $\delta = 0.1$: Total MSE of the estimation averaged over all agents at time instants t in 10 realizations. **Calculation time** is 335 seconds per realization, on average (increased due to float form of weighting matrix).

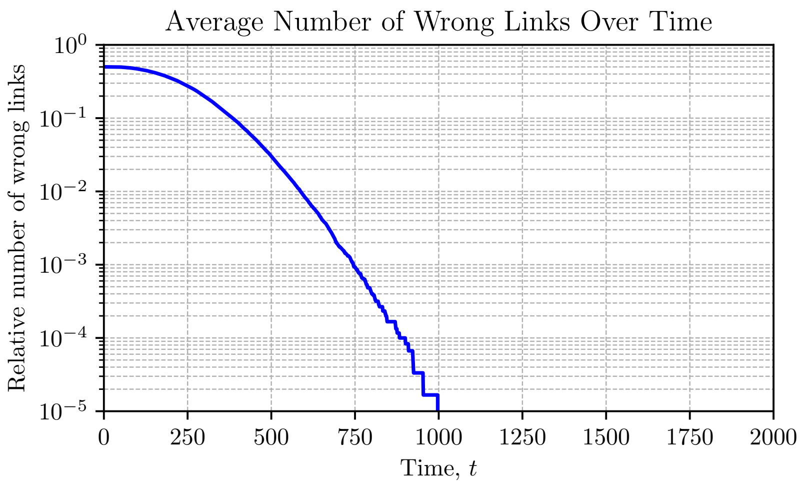


Figure 2.65: CL-colME on weighted graph: Total number of wrong links (when an agent from one class is connected and collaborates with an agent from the other class), over time.

Chapter 3

Multidimensional Data

The theory and experiment from the previous sections will be extended on multidimensional data. In the first part, we will consider the case of three-dimensional data, $\mathbf{x}_a(t) \in \mathbb{R}^3$. Then, in the second part we will consider images as high-dimensional data, since each pixel of an image can be considered as picture dimension. For an $M \times K$ monochrome image the data are $\mathbf{x}_a(t) \in \mathbb{R}^{MK}$, while for color images $\mathbf{x}_a(t) \in \mathbb{R}^{3MK}$.

3.1 Variance Estimation in Multidimensional C-colME

We assume that each agent $a \in \mathcal{A} = \{1, 2, , \dots, A\}$, receives a random sample $\mathbf{x}_a(t) \in \mathbb{R}^3$, drawn from a distribution $[D_{ax}, D_{ay}, D_{az}]$ with expected mean

$$[\mu_{ax}, \mu_{ay}, \mu_{az}] = [\mathbb{E}\{x_a(t)\}, \mathbb{E}\{y_a(t)\}, \mathbb{E}\{z_a(t)\}]. \tag{3.1}$$

The coordinates of $\mathbf{x}_a(t) = [x_a(t), y_a(t), z_a(t)]$ are called features of data vector $\mathbf{x}_a(t)$.

The random vector $\mathbf{x}_a(t) \in \mathbb{R}^3$ is said to be σ-sub-Gaussian distributed if it is centered and for any unity vector $\mathbf{u} \in \mathbb{R}^3$, $||\mathbf{u}||_2 = 1$, the random variable $\mathbf{u}^T\mathbf{x}_a(t)$ is σ-sub-Gaussian distributed.

As a special case, for canonic basis, this means that all random coordinates $x_a(t)$, $y_a(t)$, and $z_a(t)$ are also σ-sub-Gaussian.

Vector form of the signal is

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & y_1(t) & z_1(t) \\ x_2(t) & y_2(t) & z_2(t) \\ \vdots & & \\ x_A(t) & y_A(t) & z_A(t) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_A(t) \end{bmatrix}. \tag{3.2}$$

Agents that receive random samples $[x_a(t), y_a(t), z_a(t)] \in \mathbb{R}^3$, drawn from the same distribution $[D_{ax}, D_{ay}, D_{az}]$, with the same expected means, $[\mu_{ax}, \mu_{ay}, \mu_{az}]$, form a similarity class.

We will also assume that among all agents there are C similarity classes. For all agents, other agents a' that belong to the similarity class of agent a, holds

$$\Delta_{aa'}^{(p)} = ||[\mu_{ax}, \mu_{ay}, \mu_{az}] - [\mu_{a'x}, \mu_{a'y}, \mu_{a'z}]||_p = 0 \tag{3.3}$$

that is, since the agents share the same mean value vector.

Commonly chosen norms for distance of vectors are two norm:

$$||[\mu_{ax},\mu_{ay},\mu_{az}]-[\mu_{a'x},\mu_{a'y},\mu_{a'z}]||_2=\sqrt{(\mu_{ax}-\mu_{a'x})^2+(\mu_{ay}-\mu_{a'y})^2+(\mu_{az}-\mu_{a'z})^2} \quad (3.4)$$

and infinity norm

$$||[\mu_{ax},\mu_{ay},\mu_{az}]-[\mu_{a'x},\mu_{a'y},\mu_{a'z}]||_\infty=\max\{|\mu_{ax}-\mu_{a'x}|,\ |\mu_{ay}-\mu_{a'y}|,\ |\mu_{az}-\mu_{a'z}|\}. \quad (3.5)$$

If we want to use two norm and want to keep distance for various dimensions, for example, distance 1 and $K=1$ we can use (for any norm)

$$\mu_a=0 \text{ and } \mu_{a'}=1 \text{ with } \Delta_{aa'}=1. \quad (3.6)$$

For two norm, $p=2$ and $K=3$ we should use

$$[\mu_{ax},\mu_{ay},\mu_{az}]=(0,0,0) \text{ and } [\mu_{a'x},\mu_{a'y},\mu_{a'z}]=\frac{1}{\sqrt{3}}(1,1,1)=1 \text{ with } \Delta^{(2)}_{aa'}=1. \quad (3.7)$$

If variance of data remains the same for each feature, we may expect that increasing the dimensionality of the data, the convergence will slow-down, [3].

For infinity norm $p=\infty$

$$[\mu_{ax},\mu_{ay},\mu_{az}]=(0,0,0) \text{ and } [\mu_{a'x},\mu_{a'y},\mu_{a'z}]=(1,1,1)=1 \text{ with } \Delta^{(\infty)}_{aa'}=1. \quad (3.8)$$

The distance along at least one coordinate should be equal to one that is equal to one-dimensional case for the best coordinate. Note that in this case, the widest difference determines the similarity classes gap. If variance of data remains the same for each feature, we may expect that increasing the dimensionality of the data, the convergence will not degrade.

In simulations that follow we will use $\Delta^{(\infty)}_{aa'}$, meaning that the largest difference (including possible influence of variance to the confidence intervals) will determine the convergence. For the case of images with 10s of thousands order of dimensionality we will use a few percent of maximum differences for confidence intervals check, in order to achieve solution robustness.

The local mean values for all agents, $a=1,2,\dots,A=N$, will be written in a vector form as

$$\mathbf{X}(t)=\begin{bmatrix}\bar{x}_{11}(t) & \bar{y}_{11}(t) & \bar{z}_{11}(t)\\ \bar{x}_{22}(t) & \bar{y}_{22}(t) & \bar{z}_{22}(t)\\ \vdots & & \\ \bar{x}_{AA}(t) & \bar{y}_{AA}(t) & \bar{z}_{AA}(t)\end{bmatrix} \text{ with } \mathbf{X}(t)=\frac{1}{t}\mathbf{x}(t)+\frac{t-1}{t}\mathbf{X}(t-1). \quad (3.9)$$

Local Sums: We will also use the local sums of samples in vector form

$$\mathbf{M}(t)=\begin{bmatrix}mx_{11}(t) & my_{1a}(t) & mz_{11}(t)\\ mx_{22}(t) & my_{22}(t) & mz_{22}(t)\\ \vdots & & \\ mx_{AA}(t) & my_{AA}(t) & mz_{AA}(t)\end{bmatrix} \text{ with } \mathbf{M}(t)=\mathbf{M}(t-1)+\mathbf{x}(t). \quad (3.10)$$

If the local sums are calculated (and stored in a computer) then the local means can be always easily obtained as

$$\mathbf{X}(t) = \frac{1}{t}\mathbf{M}(t). \tag{3.11}$$

The square distance between two sets of data, using two norm, is

$$d^{(2)}(t) = ||\mathbf{x}_a(t) - \mathbf{x}_{a'}(t)||_2. \tag{3.12}$$

For the σ-sub-Gaussian distribution. For a and a' of the same similarity class the random variable $\mathbf{x}_a(t) - \mathbf{x}_{a'}(t)$ is centered with

$$\mathbb{E}\{||\mathbf{x}_a(t) - \mathbf{x}_{a'}(t)||_2^2\} \leq CK\sigma^2, \tag{3.13}$$

with $K = 3$ and C is a constant. For agents of different classes

$$\mathbb{E}\{||\mathbf{x}_a(t) - \mathbf{x}_{a'}(t)||_2^2\} = \mathbb{E}\{||\mathbf{x}_a(t) - \boldsymbol{\mu}_a - (\mathbf{x}_{a'}(t) - \boldsymbol{\mu}_{a'}) - (\boldsymbol{\mu}_{a'} - \boldsymbol{\mu}_a)||_2^2\} \tag{3.14}$$

$$= 2\mathbb{E}\{||\mathbf{x}_a(t) - \mathbf{x}_{a'}(t)||_2^2\} + ||\boldsymbol{\mu}_{a'} - \boldsymbol{\mu}_a||_2^2 \leq C(2K\sigma^2 + (\Delta_{aa'}^{(2)})^2). \tag{3.15}$$

For infinity norm, the maximum values are considered.

3.1.1 Multidimensional Confidence Intervals

In general, confidence boxes of a multidimensional random variable, are defined as

$$\mathbb{I}_{ax}(t) = [\bar{x}_{aa}(t) - \beta_{x\delta}(t), \bar{x}_{aa}(t) + \beta_{x\delta}(t)], \tag{3.16}$$

$$\mathbb{I}_{ay}(t) = [\bar{y}_{aa}(t) - \beta_{y\delta}(t), \bar{y}_{aa}(t) + \beta_{y\delta}(t)], \tag{3.17}$$

$$\mathbb{I}_{az}(t) = [\bar{z}_{aa}(t) - \beta_{z\delta}(t), \bar{z}_{aa}(t) + \beta_{z\delta}(t)], \tag{3.18}$$

where $\beta_{x\delta}(t)$, $\beta_{y\delta}(t)$, $\beta_{z\delta}(t)$, specify the widths of the confidence intervals for respective features that are σ-sub-Gaussian distributed.

Confidence Intervals Intersection: By using infiny norm approach, the three-dimensional confidence intervals (boxes) intersect only if all three intervals intersect. It means that the three-dimensional confidence boxes do not intersect if the confidence interval of only one feature do not intersect.

Convergence can be improved since we may consider that the agents belong to different classes if *at least one confidence interval does not intersect.* In this case, the most distant means along any dimension determine the convergence rate, if the variances for all features are the same. If the variances are not the same, the the most favorable distance-to-standard deviation ratio determines the convergence (expected separation time).

This will be of great importance for high-dimensional data. For example, in the case of images, we may always expect that the most distant features (pixels) will have difference of 255.

Standard Deviation Estimation: The standard deviations

$$[\sigma_x, \sigma_y, \sigma_z] = [Std\{x_a(t)\}, Std\{y_a(t)\}, Std\{z_a(t)\}] \tag{3.19}$$

are estimated using

$$[s_x, s_y, s_z] \leftarrow [s_x, s_y, s_z] + [\sum_{a' \in \mathcal{N}_a \cup \{a\}} |x_{a'}(t) - x_{a'}(t-1)|^2, \tag{3.20}$$

$$\sum_{a' \in \mathcal{N}_a \cup \{a\}} |y_{a'}(t) - y_{a'}(t-1)|^2, \sum_{a' \in \mathcal{N}_a \cup \{a\}} |z_{a'}(t) - z_{a'}(t-1)|^2] \tag{3.21}$$

The number of elements should also be accumulated as

$$N_t \leftarrow N_t + (|N_a(t)| + 1) \tag{3.22}$$

The value of such estimated variance would be

$$[\hat{\sigma}_{ax}(t), \hat{\sigma}_{ay}(t), \hat{\sigma}_{az}(t)] = \frac{[\sqrt{s_x}, \sqrt{s_y}, \sqrt{s_z}]}{\sqrt{2N_t}} \tag{3.23}$$

Now we can implement, for example C-colMe, with a row stochastic matrix as

$$[\mu_{ax}(t+1), \mu_{ay}(t+1), \mu_{az}(t+1)] = (1 - \alpha(t)[\bar{x}_{aa}(t), \bar{y}_{aa}(t), \bar{z}_{aa}(t)] + \tag{3.24}$$

$$+\alpha(t)\text{mean}_{\mathcal{N}_a \cup a}[\mu_{ax}(t), \mu_{ay}(t), \mu_{az}(t)] \tag{3.25}$$

The C-colME with double stochastic weighting matrix is

$$\boldsymbol{\mu}(t+1) = (1 - \alpha(t))\mathbf{X}(t) + \alpha(t)\mathbf{W}_t\boldsymbol{\mu}(t), \tag{3.26}$$

where $\mathbf{W}_t$ is of the same form as in one-dimensional case, while $\boldsymbol{\mu}(t)$ is an $N \times 3$ matrix for each t. Meaning that it is indeed an $N \times 3 \times T$ tensor, as it is $\mathbf{X}(t)$.

The system is simulated with $N = 1000$ agents within a time interval from 0 to $T = 2000$. The initial number of neighbors is $r = 12$. The mean values of the data are

$$[\mu_{ax}, \mu_{ay}, \mu_{az}] = [0.1, 0.2, 0] \tag{3.27}$$

and

$$[\mu_{a'x}, \mu_{a'y}, \mu_{a'z}] = [0.8, 0.7, 1] \tag{3.28}$$

with stabndard deviations

$$[\sigma_x, \sigma_y, \sigma_z] = [1, 1.5, 2]. \tag{3.29}$$

For confidence intervals $\delta = 0.1$ is used. The maximum feature difference is 1, meaning that based on Fig. 1.6, the expected separation time will at about $t = 180$.

The estimated standard deviations are shown in Fig. 3.1. The evolution of graph structure is given in Fig. 3.2, while the MSE is shown in Fig. 3.3, along with the local and collaborated means for two selected agents from two different classes.

Figure 3.1 shows histograms of the estimated standard deviations $\hat{\sigma}_a(t)$ at selected time instants $T_s = 50, 100, 2000$. Initially, the distributions are broad and overlapping. Over time, distinct peaks corresponding to separate classes emerge, illustrating the progressive separation of agent classes in three-dimensional feature space.

Figure 3.2 illustrates the evolution of the graph structure. At $t = 0$, all agents are

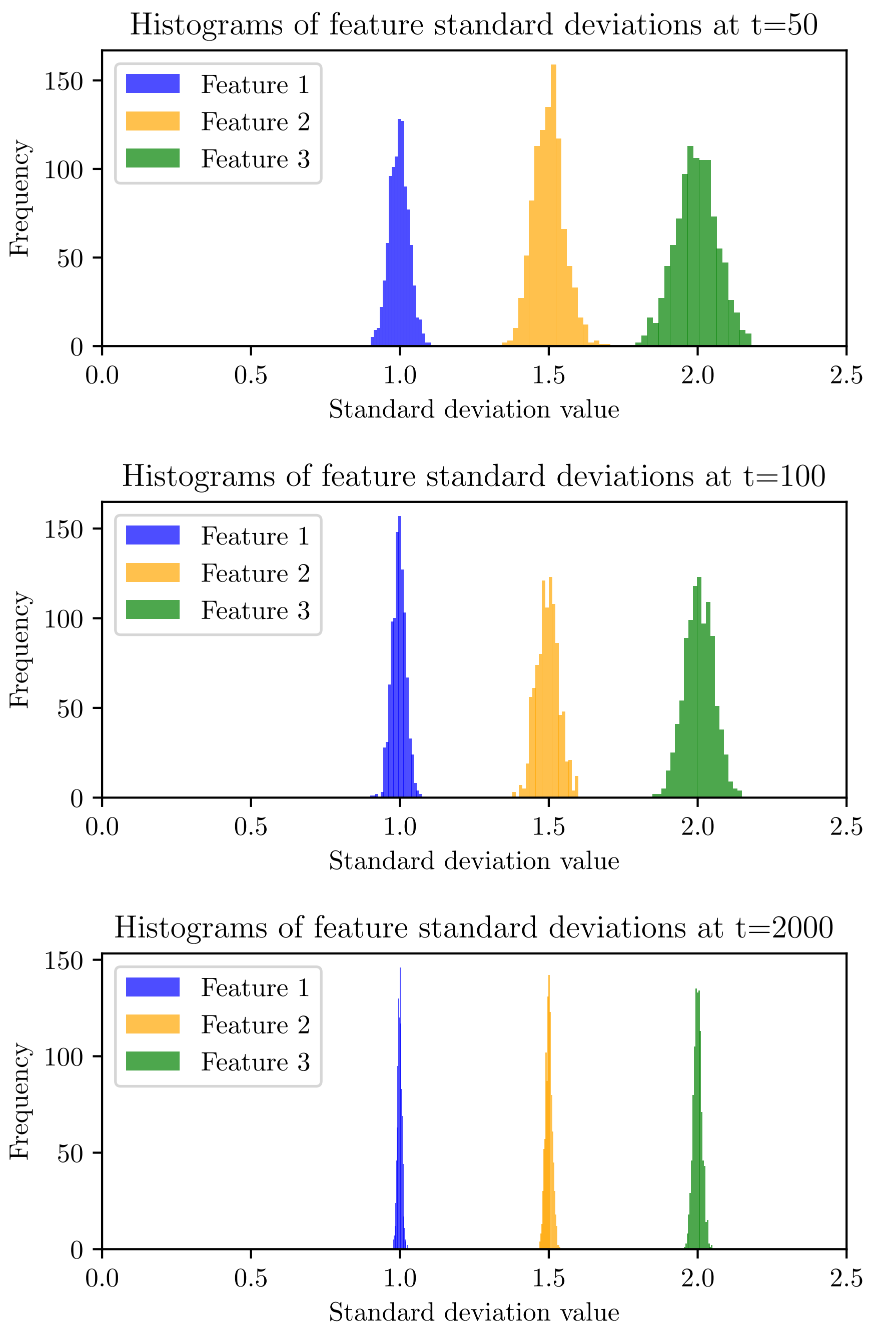


Figure 3.1: Histogram of the estimated standard deviations $\hat{\sigma}_a(t)$ over agents within $T_s = 50, 100, 2000$ time instants.

arbitrarily connected. By $t = 300$ and $t = 500$, the connections adjust according to the confidence intervals, showing gradual separation of the classes. At $t = 1000$, each class is more clearly distinguishable, with fewer inter-class links.

Figure 3.3 presents the total mean squared error (MSE) over time, averaged over 10 realizations. The top plot shows the overall MSE, while the bottom plot includes local and collaborated means for two selected agents from different classes. The MSE decreases as agents correctly separate into classes, reflecting improved estimation accuracy over time.

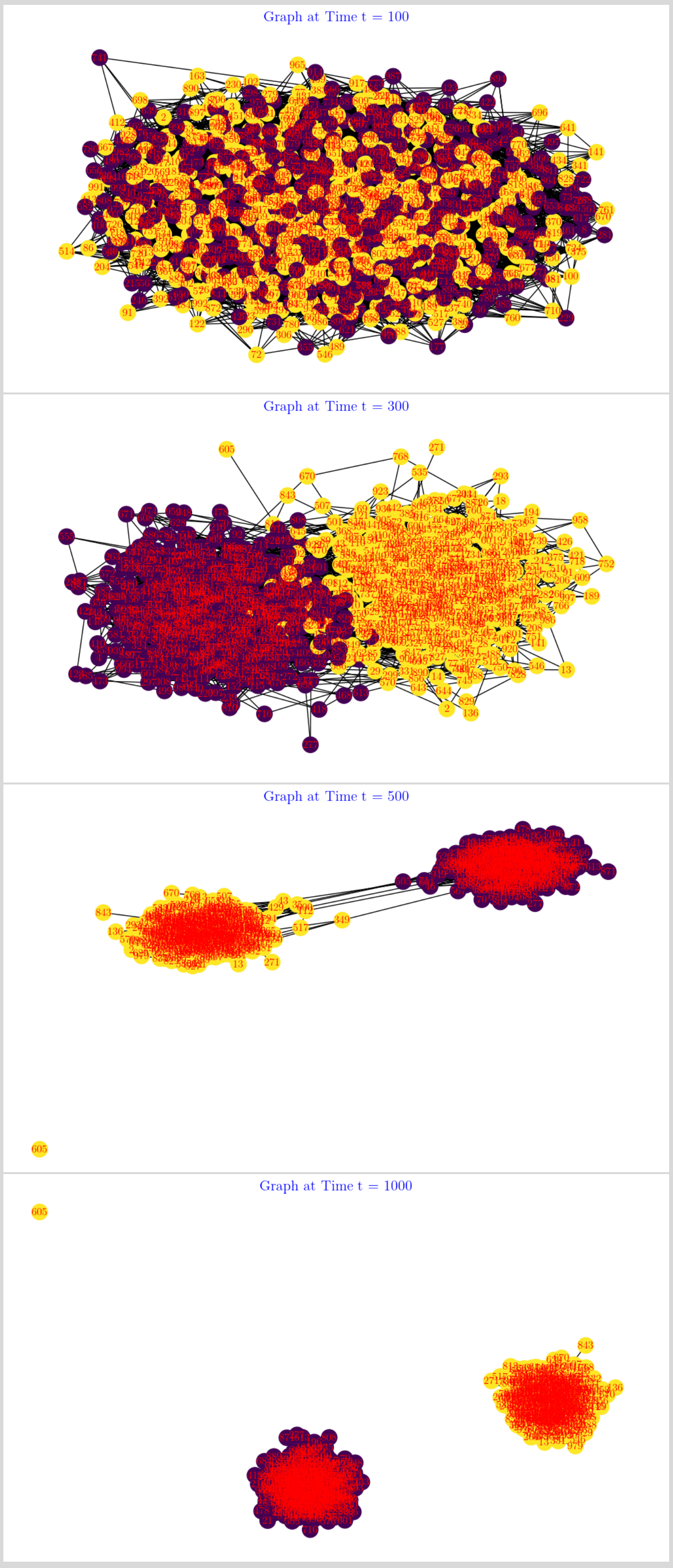
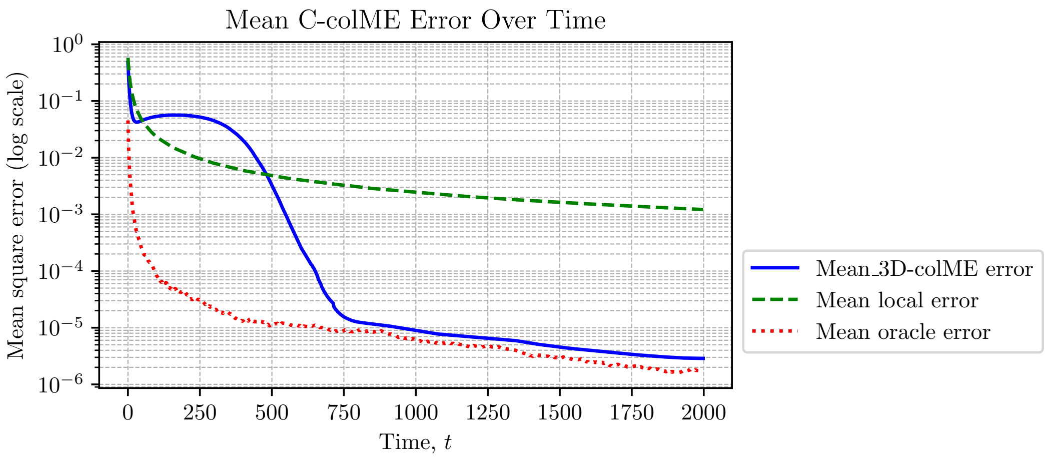


Figure 3.2: C-colME of three-dimensional data with $N = 1,000$: Random graph with agent classes, arbitrary connected, at $t = 0$ and then the connections adjusted using the confidence intervals over time. Graphs at $t = 300$, $t = 500$, and $t = 900$, respectively, are shown from top to bottom.



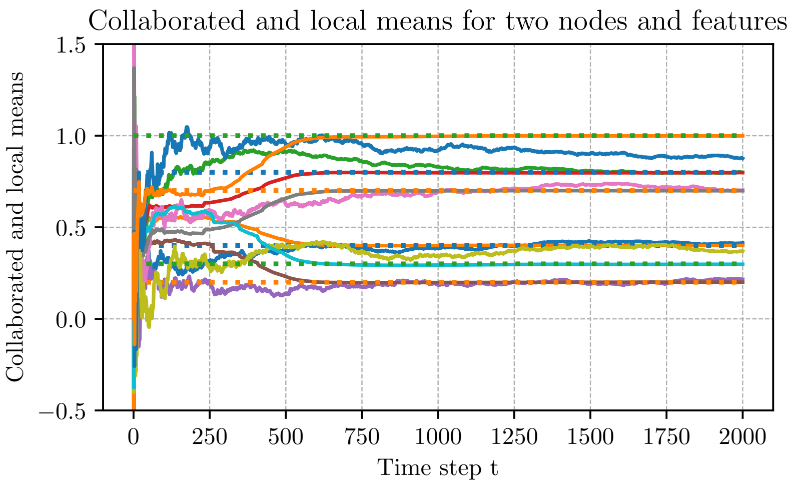


Figure 3.3: C-colME of three-dimensional data with $N = 1,000$: Total MSE of the estimation. The results are averaged over 10 realizations.

3.1.2 Variance Different for Each Class and Each Dimension

In this case we assumed that both mean values and standard deviations change for different classes of agents. We considered $N = 1000$ agents with the initial number of neighbors is $r = 12$. The mean values of the data are

$$[\mu_{ax}, \mu_{ay}, \mu_{az}] = [0.4, 0.2, 0.3] \tag{3.30}$$

and

$$[\mu_{a'x}, \mu_{a'y}, \mu_{a'z}] = [0.8, 0.7, 1] \tag{3.31}$$

with standard deviations

$$[\sigma_{ax}, \sigma_{ay}, \sigma_{az}] = [0.9, 1.25, 1.75] \tag{3.32}$$

and

$$[\sigma_{a'x}, \sigma_{a'y}, \sigma_{a'z}] = [1.1, 1.5, 2.1] \tag{3.33}$$

and

The standard deviations

$$[\sigma_x, \sigma_y, \sigma_z] = [\mathrm{Std}\{x_a(t)\}, \mathrm{Std}\{y_a(t)\}, Std\{z_a(t)\}] \tag{3.34}$$

are estimated using forgetting factor

$$[s_x, s_y, s_z] \leftarrow 0.99[s_x, s_y, s_z] + [\sum_{a' \in \mathcal{N}_a \cup \{a\}} |x_{a'}(t) - x_{a'}(t-1)|^2, \tag{3.35}$$

$$\sum_{a' \in \mathcal{N}_a \cup \{a\}} |y_{a'}(t) - y_{a'}(t-1)|^2, \sum_{a' \in \mathcal{N}_a \cup \{a\}} |z_{a'}(t) - z_{a'}(t-1)|^2] \tag{3.36}$$

The number of elements should also be accumulated as

$$N_t \leftarrow 0.99N_t + (|N_a(t)| + 1) \tag{3.37}$$

The value of such estimated variance would be

$$[\hat{\sigma}_{ax}(t), \hat{\sigma}_{ay}(t), \hat{\sigma}_{az}(t)] = \sqrt{\frac{[s_x, s_y, s_z]}{2N_t}} \tag{3.38}$$

For confidence intervals $\delta = 0.1$ is used. Figure 3.4 shows histograms of the estimated standard deviations $\hat{\sigma}_a(t)$ at different time instants. Initially, the distributions are broad and overlapping, reflecting variability within each class. Over time, distinct peaks emerge for each class, indicating progressive separation.

Figure 3.5 illustrates the network graph evolution. At $t = 0$, agents are randomly connected. By $t = 300$ and $t = 500$, connections adjust based on confidence intervals, and by $t = 900$, classes are clearly separated with few inter-class links remaining.

Figure 3.6 presents the total mean squared error (MSE) over time, averaged across 10 realizations. The top plot shows overall MSE, while the bottom plot includes local and collaborated means for selected agents. The MSE decreases as agents segregate correctly, reflecting improved estimation accuracy.

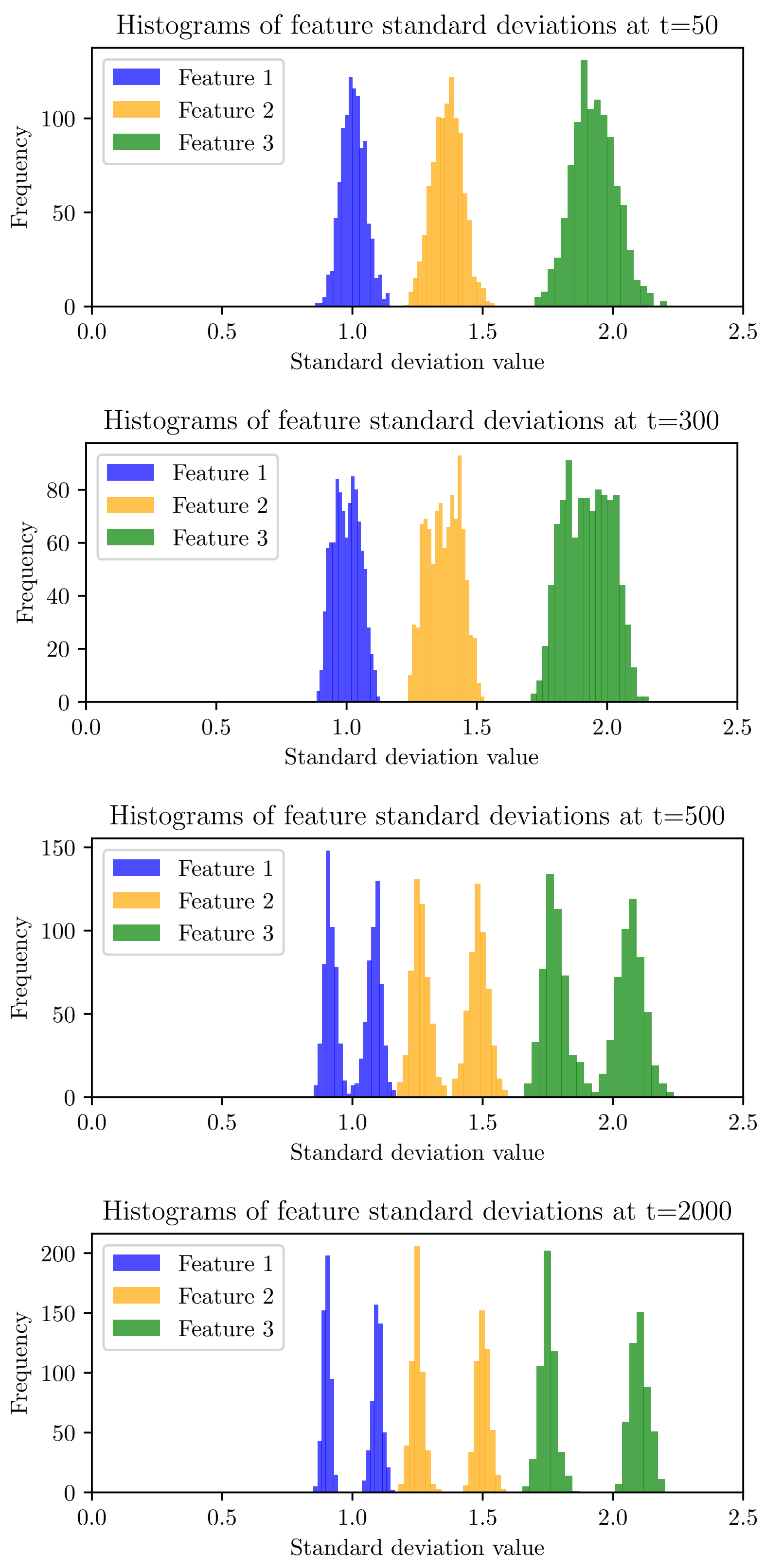


Figure 3.4: Histogram of the estimated standard deviations $\hat{\sigma}_a(t)$ over agents within $T_s = 50, 100, 2000$ time instants.

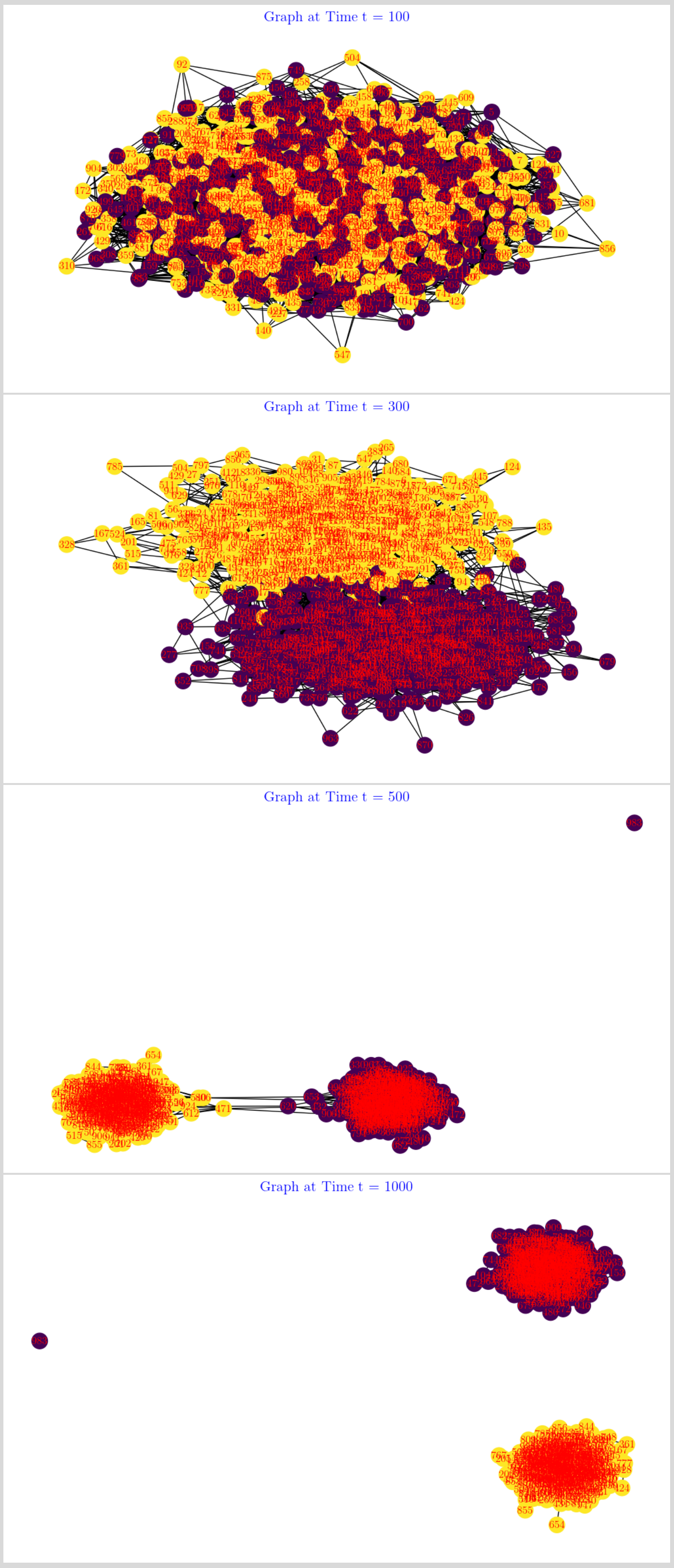
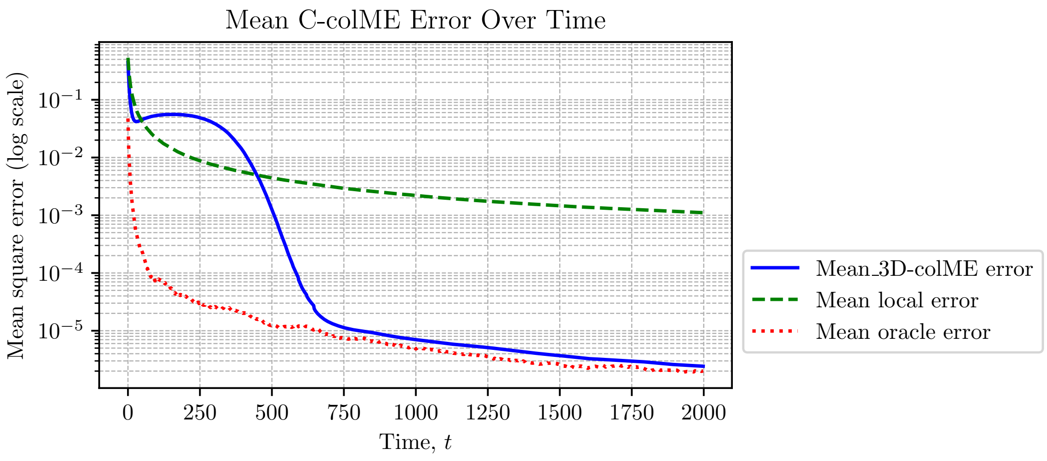


Figure 3.5: C-colME of three-dimensional data with $N = 1,000$ and different variance: Random graph with agent classes, arbitrary connected, at $t = 0$ and then the connections adjusted using the confidence intervals over time. Graphs at $t = 300$, $t = 500$, and $t = 900$, respectively, are shown from top to bottom.



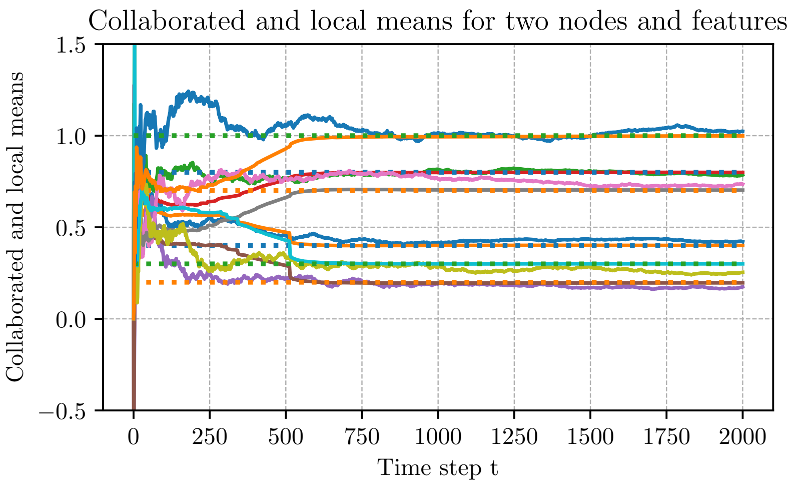


Figure 3.6: C-colME of three-dimensional data with $N = 1,000$ and different variance: Total MSE of the estimation. The results are averaged over 10 realizations.

3.2 Confidence Intervals with Variance and Mean

So far, we have used confidence intervals based solely on the local means. In the cases where **the true means are very close and the standard deviations differ more significantly**, we can use the standard deviation estimates and their confidence intervals. The estimated standard deviation at an instant t, for each agent, is already defined in Section 1.2.3. Its local value, using only the considered agent a, is

$$\hat{\sigma}_a(t) = \sqrt{\frac{1}{2t}\sum_{\tau=1}^{t}\big(x_a(\tau)-x_a(\tau-1)\big)^2} \tag{3.39}$$

$$\hat{\sigma}_a^2(t) = \frac{1}{2t}\sum_{\tau=1}^{t}\big(x_a(\tau)-x_a(\tau-1)\big)^2 = \frac{1}{2t}\sum_{\tau=1}^{t} d_a^2(\tau). \tag{3.40}$$

The statistics of the standard deviation for the sub-Gaussian data is quite complex. To estimate the variance of the sample variance estimator, recall that for a zero-mean σ-sub Gaussian random variable $\mathbf{x}$, holds $\mathbb{E}\{\mathbf{x}^4\} \le C\sigma^4$ for some constant C that depends on the specific distribution. For a Gaussian distribution, $C = 3$. For the distributions when the kurtosis exists we can write $\mathbb{E}\{\mathbf{x}^4\} = \kappa_x\sigma^4$.

First, consider the variance of $d_a^2(t) = (x_a(t)-x_a(t-1))^2$. Using $\mathrm{Var}(\mathbf{x}) = \mathbb{E}(\mathbf{x}^2)-(\mathbb{E}(\mathbf{x}))^2$, for $d_a^2(t)$ we can write

$$\mathrm{Var}(d_a^2(t)) = \mathbb{E}\{d_a^4(t)\} - (\mathbb{E}\{d_a^2(t)\})^2 = \kappa_d\sigma_d^4 - \sigma_d^4 = (\kappa_d-1)\sigma_d^4 = (\kappa_d-1)(2\sigma^2)^2. \tag{3.41}$$

Now, we estimate the variance of $\hat{\sigma}_a^2(t)$ as

$$\mathrm{Var}(\hat{\sigma}_a^2(t)) = \mathrm{Var}(\frac{1}{2t}\sum_{\tau=1}^{t}(x_a(\tau)-x_a(\tau-1))^2) = \frac{1}{4t^2}\mathrm{Var}(\sum_{\tau=1}^{t}(x_a(\tau)-x_a(\tau-1))^2) \tag{3.42}$$

$$= \frac{1}{4t^2}\sum_{\tau=1}^{t}(\kappa_d-1)(2\sigma^2)^2 = \frac{(\kappa_d-1)\sigma^4}{t}. \tag{3.43}$$

To ensure statistical independence of two values of $d_a(t) = x_a(t) - x_a(t-1)$ we can use $t = 1, 3, \dots$. However, using every t produced the same statistical results.

Estimating the variance of the sample standard deviation involves the Delta method, for $g(x) = \sqrt{x}$, with $g'(x) = \frac{1}{2\sqrt{x}}$. Using this method we get

$$\mathrm{Var}(g(\hat{\sigma}^2)) \approx [g'(\sigma^2)]^2\mathrm{Var}(\hat{\sigma}^2) = (\frac{1}{2\sqrt{\sigma^2}})^2\mathrm{Var}(\hat{\sigma}^2) = \frac{\mathrm{Var}(\hat{\sigma}^2)}{4\sigma^2} = \frac{1}{4\sigma^2}\frac{(\kappa_d-1)\sigma^4}{t} = \frac{(\kappa_d-1)\sigma^2}{4t}. \tag{3.44}$$

Therefore the standard error (standard deviation of the estimated standard deviation) is

$$SE_a(t) = \sqrt{\mathrm{Var}(g(\hat{\sigma}^2))} = \sqrt{\mathrm{Var}(\hat{\sigma})} = \frac{\sqrt{\kappa_d-1}}{2}\frac{\sigma}{\sqrt{t}} \propto \frac{\sigma}{\sqrt{t}}. \tag{3.45}$$

The value of κ_d for a few distributions that we will use in the examples is given next. For the scaled Rademacher (Bernoulli) distribution of $x_a(t)$ we have $\kappa = 1$, while $\kappa_d = 2$ since $d(t)$ is a linear sum of two distributions. For $x_a(t)$ being a sum of two uniform distributions

$\kappa = 2.4$ and $\kappa_d = 2.7$. For Gaussian distributions $\kappa = \kappa_d = 3$.

Since we used the first-order approximation in the Delta method, which tends to underestimate the variance, and $1/2 \leq \sqrt{\kappa_d - 1}/2 \leq 1/\sqrt{2}$, for the considered distributions, we decided to further simplify the calculations that follow by using the same bounds for the standard error as for the local means. This has been statistically checked comparing histograms with theoretically assumed distributions of standard deviations, Figs. 3.7 and 3.8.

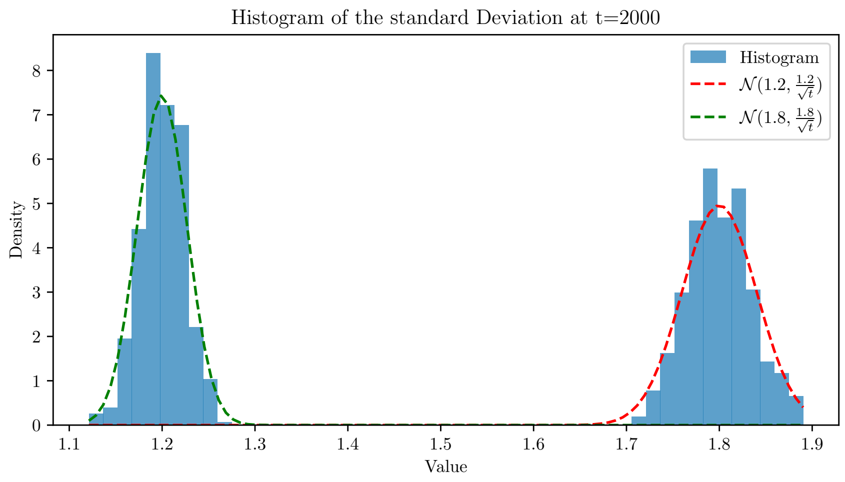


Figure 3.7: Histogram and theoretical distribution of the standard deviation.

Based on the previous assumptions and analysis, we can perform the confidence interval intersection check on both the local mean and the estimated standard deviation, defined by

$$\mathbb{I}_a(t) = [\bar{x}_{aa}(t) - \beta_\delta(t), \quad \bar{x}_{aa}(t) + \beta_\delta(t)] \tag{3.46}$$

$$\mathbb{I}_a^\sigma(t) = [\hat{\sigma}_a(t) - \beta_\delta(t), \quad \hat{\sigma}_a(t) + \beta_\delta(t)]. \tag{3.47}$$

Formally, we can consider this case as a two-dimensional data problem and use norm infinity, meaning that the graph edge between agents is disconnected if any of the confidence interval pairs $(\mathbb{I}_a(t), \quad \mathbb{I}_{a'}(t))$ or $(\mathbb{I}_a^\sigma(t), \quad \mathbb{I}_{a'}^\sigma(t))$ do not intersect. All other calculations (B-colMe, C-colME) are the same as in the rest of the text.

Example: Consider data with two similarity classes with very close means $(\mu_a, \mu_{a'}) = (0.9, 1.1)$ and different standard deviations $(\sigma_a, \sigma_{a'}) = (1.2, 1.8)$.

The expected separation time for the local mean-based confidence intervals, according to the results of Section 1.2.2, would occur very late, at

$$T_{sep} = 4,265. \tag{3.48}$$

The expected separation time for standard deviation-based confidence intervals is obtained from $2\beta_\delta(t) = \sigma_a - \sigma_{a'} = 1.8 - 1.2$ as

$$T_{sep} = 416. \tag{3.49}$$

Therefore the standard deviation based separation will dominate and drive the system.

For collaborative mean estimation, we used B-colME with $N = 1000$ agents. The results are presented next, in Fig. 3.8, Fig. 3.9, and Fig. 3.10.

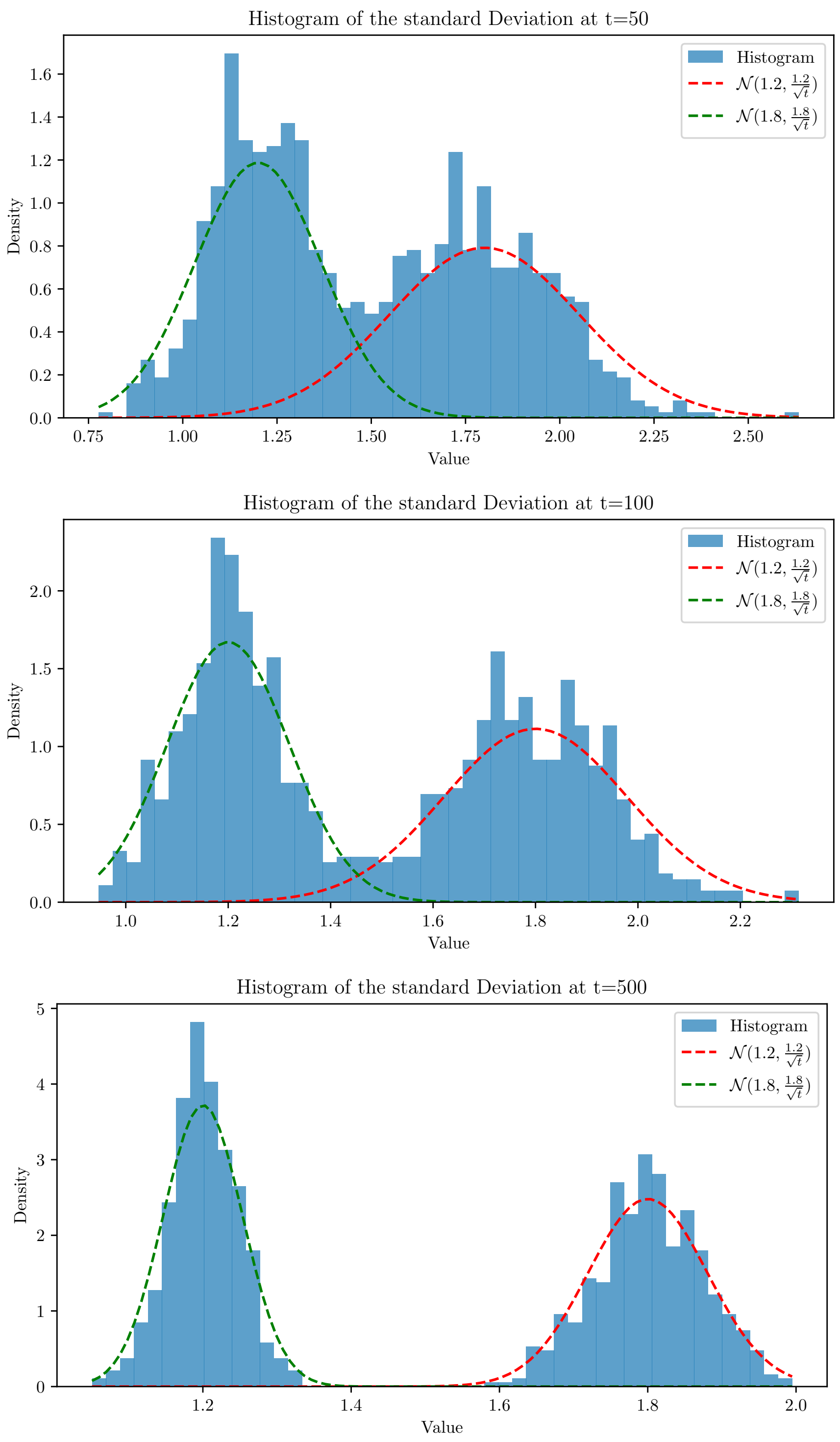
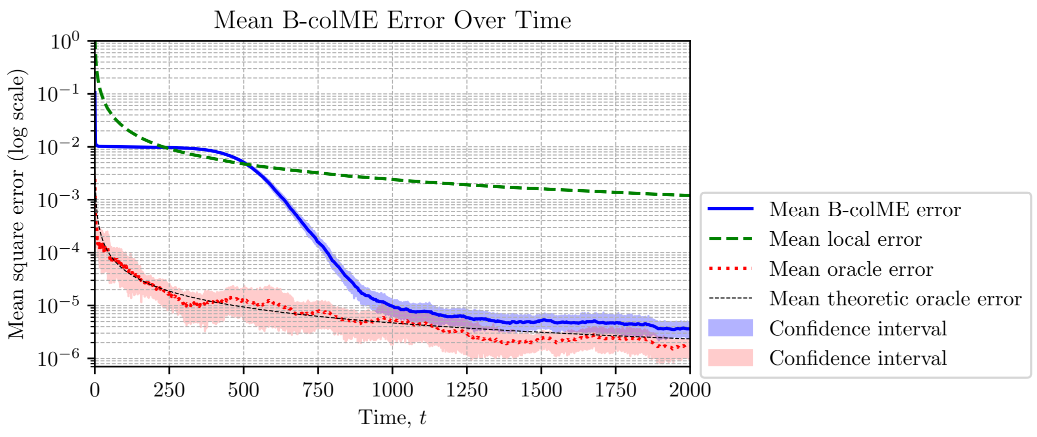


Figure 3.8: Histogram and theoretical distribution of the standard deviation.

Example: We also considered the case with very close means (0.9,1.1) and different standard deviations (1.2, 1.8) with C-colME and $N = 1000$ agents. The results are shown in Figs. 3.11 and 3.12.



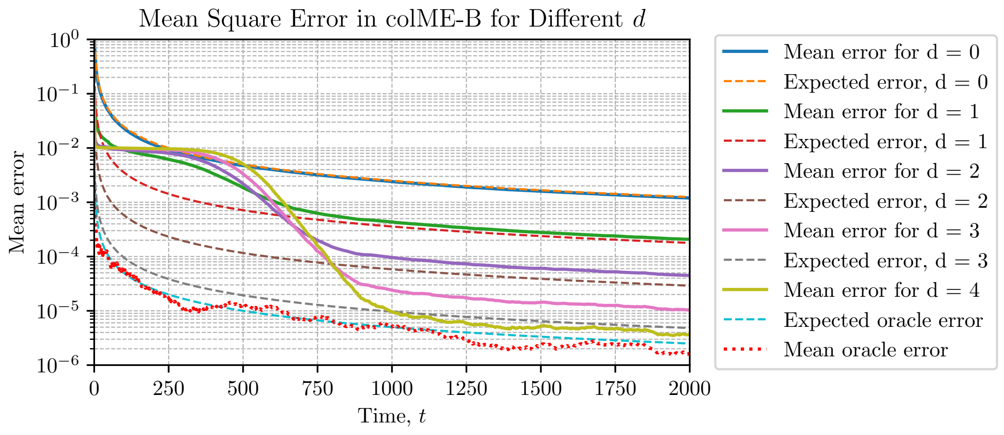


Figure 3.9: B-colME on data with the same mean, $N = 1000$, $\delta = 0.01$. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.

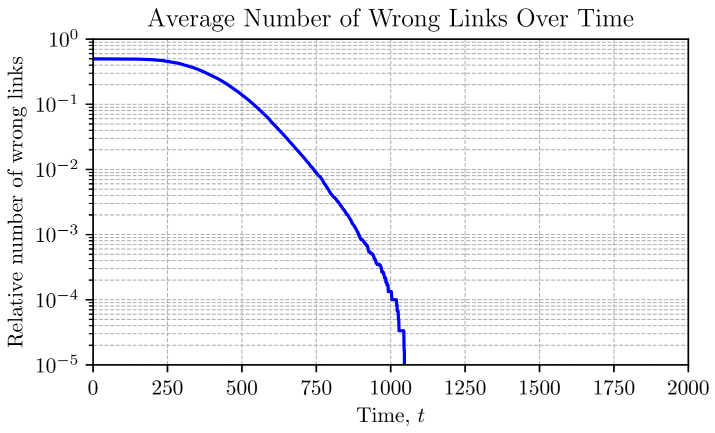


Figure 3.10: B-colME on data with the same mean: Total number of wrong links.

3.3 Confidence Intervals with Kurtosis, Variance, and Mean

Now consider the cases where **both the true means and true standard deviations are very close, but the similarity class differ in distribution type**. For example, one is Gaussian distributed (with kurtosis equal to 3) and the other is uniformly distributed (with kurtosis equal to 1.8). Here, we can use the confidence intervals based on the estimated

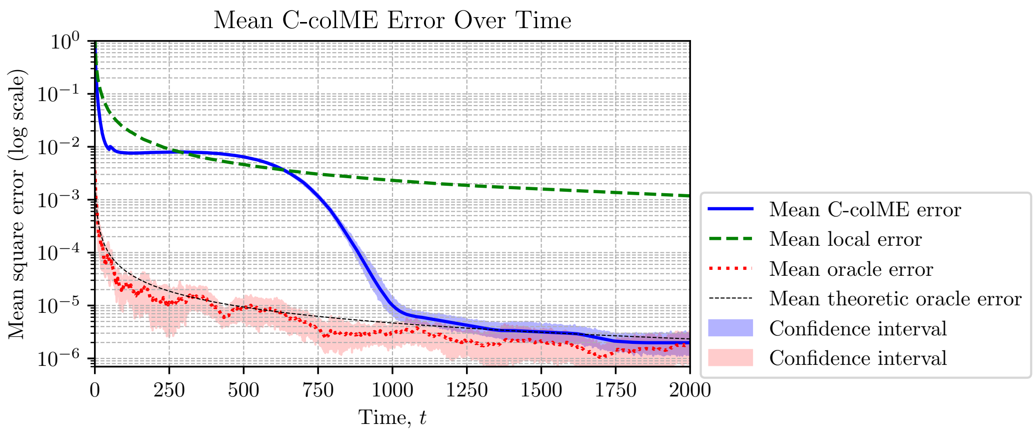


Figure 3.11: C-colME on data with similar mean, $N = 1000$, $\delta = 0.01$. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.

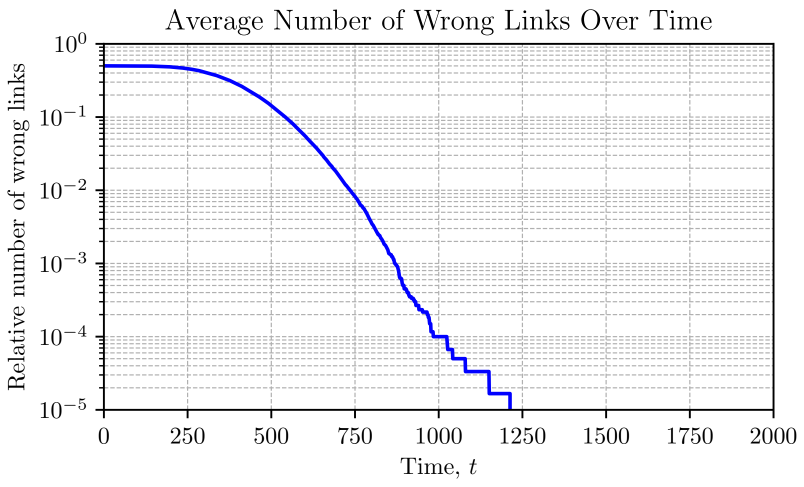


Figure 3.12: C-colME on data with similar mean: Total number of wrong links.

kurtosis,

$$\hat{\kappa}_a(t) = \frac{1}{2t\hat{\sigma}_a^4(t)} \sum_{\tau=1}^{t} \left(x_a(\tau) - x_a(\tau-1)\right)^4 - 3. \tag{3.50}$$

in addition to the confidence intervals based on the mean and the standard deviation.

The statistics of kurtosis for the sub-Gaussian data is quite complex. However, it can be shown when the number of the instants is sufficiently large (what is the case in our region of interest, when t includes hundreds of samples) for the variance of kurtosis we can use the approximation [34]

$$\sigma^2_{a,kurtosis}(t) = \frac{24t(t-1)^2}{(t-3)(t-2)(t+3)(t+5)} \approx \frac{24}{t}. \tag{3.51}$$

Then we can perform the confidence intervals check on all three parameters: the local mean, the estimated standard deviation, and the estimated kurtosis, using

$$\mathbb{I}_a(t) = [\bar{x}_{aa}(t) - \beta_\delta(t), \quad \bar{x}_{aa}(t) + \beta_\delta(t)] \tag{3.52}$$

$$\mathbb{I}_a^\sigma(t) = [\hat{\sigma}_a(t) - \beta_\delta(t), \quad \hat{\sigma}_a(t) + \beta_\delta(t)] \tag{3.53}$$

$$\mathbb{I}_a^\kappa(t) = [\hat{\kappa}_a(t) - z_\delta \frac{\sqrt{24}}{\sqrt{t}}, \quad \hat{\kappa}_a(t) + z_\delta \frac{\sqrt{24}}{\sqrt{t}}]. \tag{3.54}$$

For kurtosis we used confidence intervals defined under Gaussian assumption, but with very high probability δ, resulting in wider intervals, since this assumption is rough. The histogram of kurtosis is always plotted along with its assumed distribution.

As an example, consider the expected separation time of the confidence intervals based on kurtosis for the Bernoulli distribution (expected kurtosis equal to 1) and the sum of 4 uniform random data (expected kurtosis 2.7) with confidence probability $\delta = 0.001$ and $z_\delta = 3.89$. From the relation $k_{4U} - k_B = 2z_\delta\sqrt{24/t}$ in the form

$$2.7 - 1.0 = 2 \times 3.89\sqrt{24/t} \tag{3.55}$$

we get $T_{sep} = 502$.

The problem in which the local mean, the local standard deviation, and the local kurtosis are used can be considered as a three-dimensional data problem, with each agent associated with

$$\mathbf{x}_a(t) = \Big(\bar{x}_{aa}(t), \quad \hat{\sigma}_a(t), \quad \hat{\kappa}_a(t)\Big). \tag{3.56}$$

Within this framework, we will use the norm infinity, which means that the edge of the graph is disconnected if any pair of confidence intervals:

$$(\mathbb{I}_a(t),\ \mathbb{I}_{a'}(t)) \text{ or } (\mathbb{I}_a^\sigma(t),\ \mathbb{I}_{a'}^\sigma(t)) \text{ or } (\mathbb{I}_a^\kappa(t),\ \mathbb{I}_{a'}^\kappa(t)) \tag{3.57}$$

does not intersect.

Example: Consider the case with close means (0.9, 1.1), close standard deviations (1.9, 2.1), but with different distribution types. One class was Bernoulli distributed with kurtosis equal to 1 and the other class was a sum of 4 uniformly distributed random variables with kurtosis equal to 2.7 (close to Gaussian when kurtosis is 3).

In this case the confidence intervals for the mean and standard deviation are checked all the time from $t = 0$ to $t = 2000$. The expected separation time for both of them is obtained from $2\beta_\delta(t) = \sigma_a - \sigma_{a'} = \mu_a - \mu_{a'}$, as $T_{sep} = 7,825$. Since the kurtosis is calculated using the fourth power of data, we have waited and applied the kurtosis confidence interval checks after $t = 500$, with reconnection option being active. We have checked and concluded that all edge disconnections in this case were based on the kurtosis intervals, since the mean and the standard deviation are too close and their separation time would come much later. The histograms of kurtosis for various values of t are given next.

The collaboration results with B-colME are given in Fig. 3.14 and Fig. 3.15.

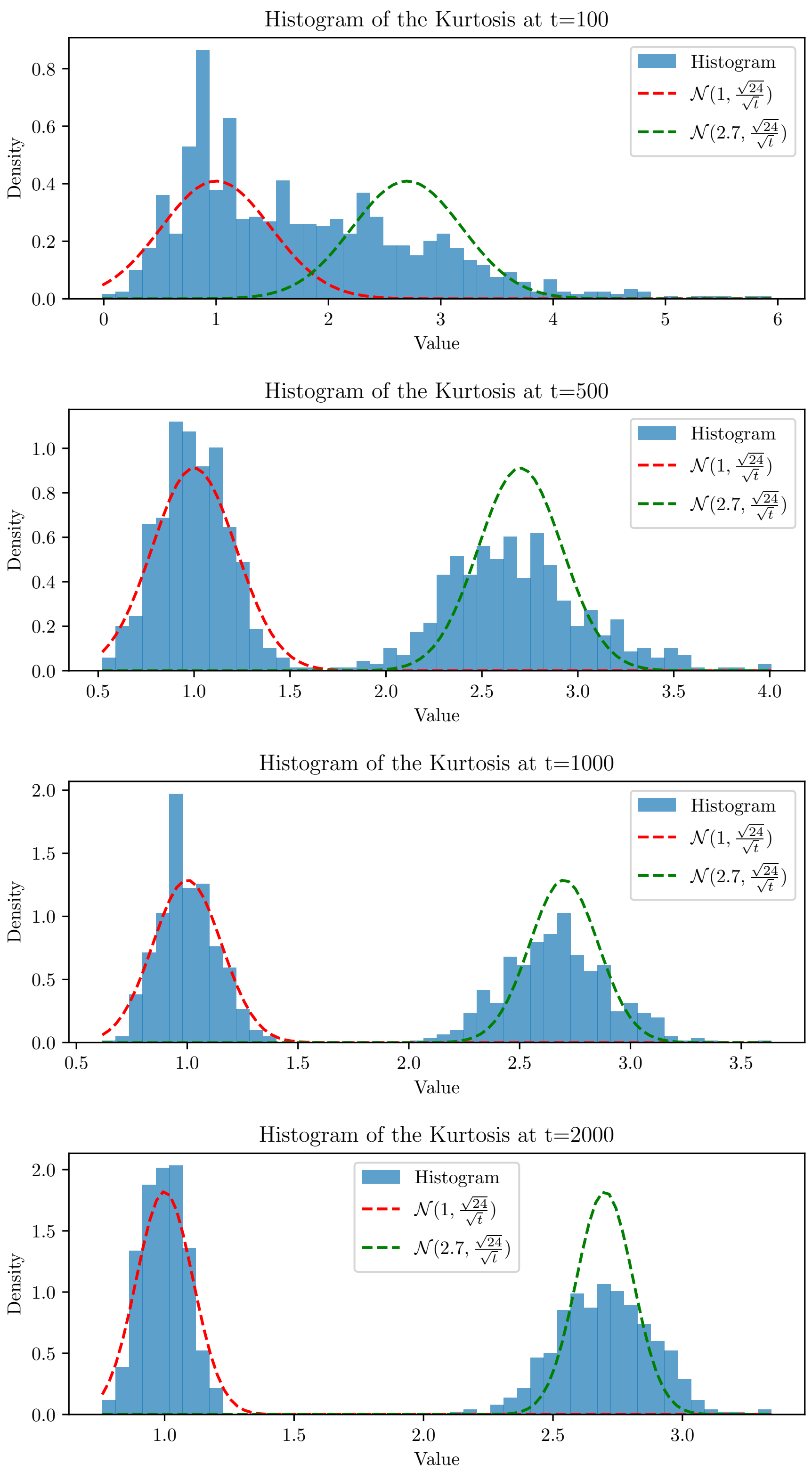
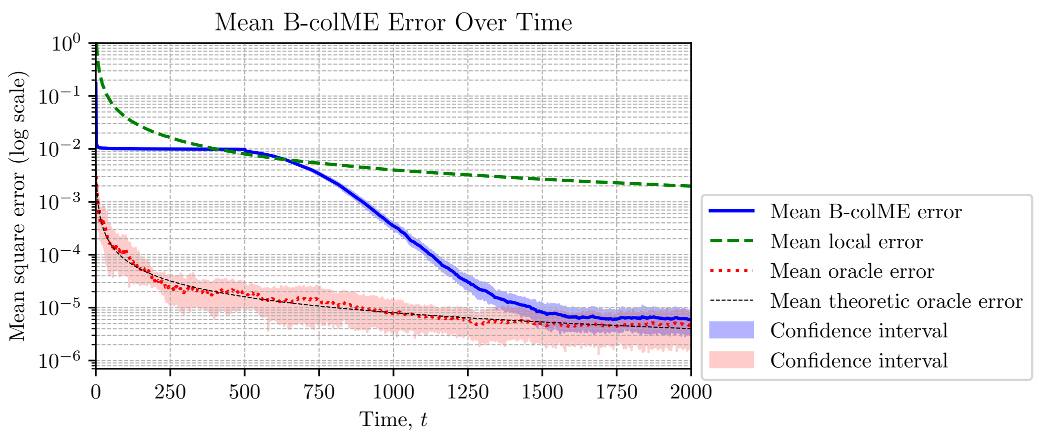


Figure 3.13: Histogram and theoretical distribution of the kurtosis.



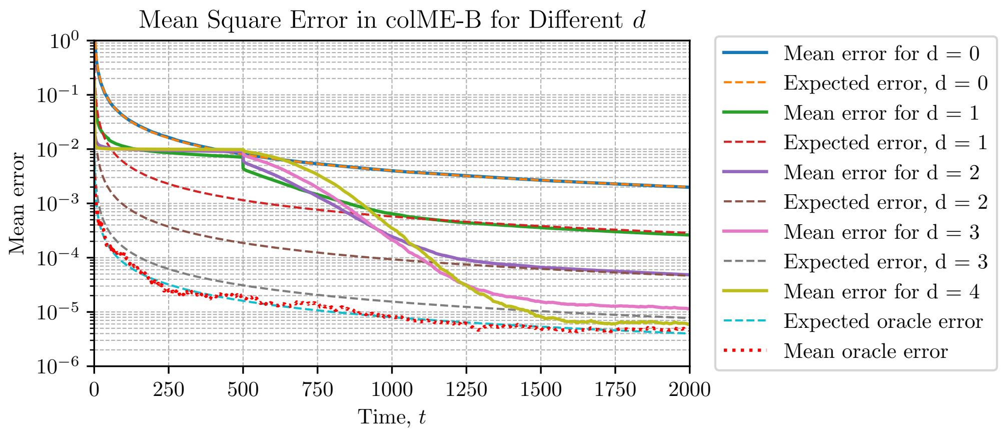


Figure 3.14: B-colME on data with similar mean and variance, different distribution of data, $N = 1000$, $\delta = 0.01$. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.

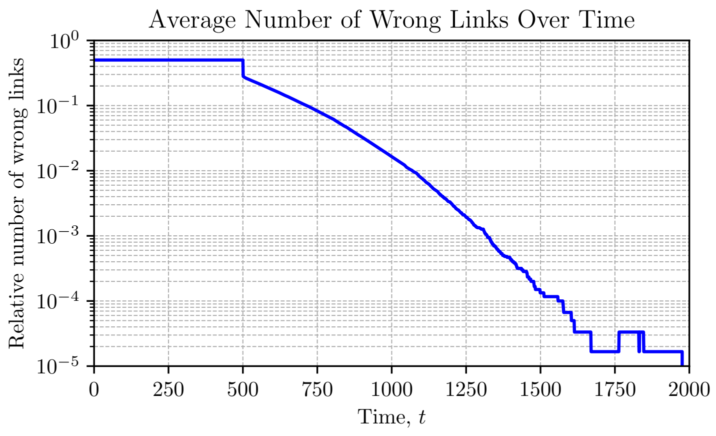


Figure 3.15: B-colME on data with similar mean and variance, different distribution of data: Total number of wrong links.

Figure 3.13 shows histograms of the estimated kurtosis for agents at different time instants, along with the corresponding theoretical distributions. Initially, the distributions are overlapping, but over time the histograms reflect the differences in underlying data distributions, highlighting class separation based on kurtosis.

Figure 3.14 presents the total mean squared error (MSE) over time for B-colME applied to data with similar mean and variance but different distributions. The top plot shows the MSE for a fixed neighborhood, while the bottom plot shows the MSE for various neighborhood sizes. The decreasing MSE indicates improved estimation as agents segregate according to distributional differences.

Figure 3.15 shows the total number of wrong links over time. Despite differences in distribution, the number of incorrect inter-class connections behaves consistently, reflecting the effect of confidence interval-based edge pruning on the network structure.

3.4 Multiclass Example with Kurtosis, Variance, and Mean

Consider now a case with, for example, four classes of agents and no single confidence interval approach can be used. For some pairs of classes, the means are so close that they cannot be separated using local means, but variances may be different, while for other pairs of classes the means are distant, but variances are close, while there are pairs of classes when only the distribution type significantly differs one class from another.

In specific, consider the following classes, with corresponding means, standard deviations, distribution types:

$$\textbf{Class 1}:\ (\mu_1,\ \sigma_1) = (0.9,\quad 1.8) \quad \text{sum of two uniform distributions} \tag{3.58}$$

$$\textbf{Class 2}:\ (\mu_2,\ \sigma_2) = (0.1,\quad 2.0) \quad \text{sum of two uniform distributions} \tag{3.59}$$

$$\textbf{Class 3}:\ (\mu_3,\ \sigma_3) = (1.1,\quad 1.2) \quad \text{sum of two uniform distributions} \tag{3.60}$$

$$\textbf{Class 4}:\ (\mu_4,\ \sigma_4) = (1.0,\quad 1.9) \quad \text{Bernoulli distribution} \tag{3.61}$$

Table 3.1: Table with colored symbols indicating what confidence intervals will be used for separation of agents: Intervals $(\mathbb{I}_a(t),\ \mathbb{I}_{a'}(t))$, $(\mathbb{I}^{\sigma}_a(t),\ \mathbb{I}^{\sigma}_{a'}(t))$, and $(\mathbb{I}^{\kappa}_a(t),\ \mathbb{I}^{\kappa}_{a'}(t))$ are given in this respective order. Red circle means that this form is not used for separation (expected separation time is very late), while green means that this interval can be used (reasonable separation time). Of course, in all cases all confidence intervals are checked.

Agent classes (means, standard deviations, kurtosis) intervals

	1	2	3	4
1		🟢🔴🔴 $(373, 7023, \infty)$	🔴🟢🔴 $(4264, 416, \infty)$	🔴🔴🟢 $(28552, 28552, 741)$
2	🟢🔴🔴		🟢🟢🔴 $(161, 258, \infty)$	🟢🔴🟢 $(306, 31890, 741)$
3	🔴🟢🔴	🟢🟢🔴		🔴🟢🟢 $(19685, 321, 741)$
4	🔴🔴🟢	🟢🔴🟢	🔴🟢🟢	

Expected separation times for the local mean based confidence intervals, standard deviation confidence intervals, and kurtosis based confidence intervals are calculated by solving for

$t = T_{sep}$ the following equalities:

$$2\beta_\delta(t) = \mu_a - \mu_{a'} \tag{3.62}$$

$$2\beta_\delta(t) = \sigma_a - \sigma_{a'} \tag{3.63}$$

$$2z_\delta\sqrt{\frac{24}{t}} = \kappa_a - \kappa_{a'}. \tag{3.64}$$

They are given in Table 3.1, as triples corresponding to the mean, standard deviation, and kurtosis, respectively.

Approximate Class Separation Analysis:

1. The expected separation time for the *confidence intervals based on kurtosis* (for a sum of two uniform distributions with kurtosis equal to 2.4 and the Bernoulli distribution with kurtosis equal to 1) is obtained from

$$2.4 - 1.0 = 2 \times 3.89\sqrt{24/t} \tag{3.65}$$

   as $T_{sep} = 741$. The statistical value can be slightly higher since we will not allow disconnection according to this criterion before the local kurtosis is sufficiently accumulated (for example, until $t = 500$).

2. The expected separation time based on *the standard deviation confidence intervals* between the first and third classes will be approximately $T_{sep} = 416$ and between the third and fourth is $T_{sep} = 321$ with an average of 369.

3. The expected separation time based on *the local mean confidence intervals* for the first and second classes will be $T_{sep} = 373$, for the second and third class at $T_{sep} = 161$, and between the second and fourth $T_{sep} = 306$, with an average of 280.

4. Statistical mean separation times slightly differ from the theoretical expected separation times. In our example, the separation based on the local mean starts first with no other concurrent criteria separating edges earlier. Later, separation based on the variance can take some of the disconnections for class two and three, making a lower number of mean-based disconnections for higher times, resulting in a slightly lower statistical average. Since kurtosis-based disconnections start at the last, some of its edges are earlier disconnected by mean or standard deviation, making its average separation time slightly longer.

The separation times are recorded in the program and their histograms are shown in Fig. 3.16, where the mean value of the recorded separation time is also calculated and shown for each type of confidence intervals by vertical dashed lines. They are in agreement with the above approximate theoretical analysis. From Table 3.1 we can see that class 2 will separate first (its highest expected separation time is 373), then class 3 will separate (with the highest expected separation time 416) and finally classes 1 and 4 will separate with the highest expected separation time of 741 (see Fig. 3.19).

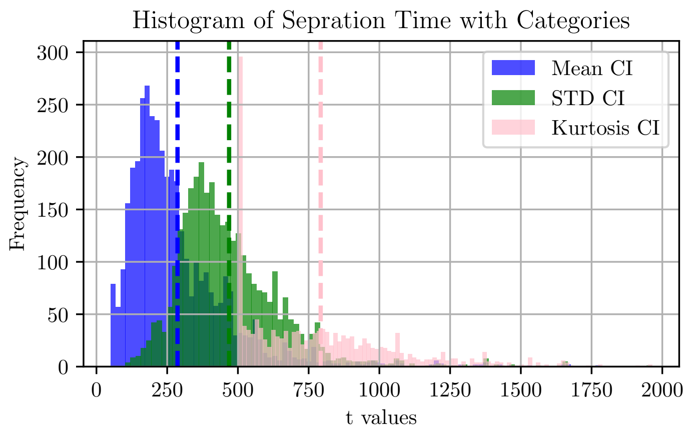


Figure 3.16: Histogram of separation times with four classes of agents and no single confidence interval approach can be used. Mean separation times are given by the vertical dashed lines. They are close to the theoretical expected separation times for the given classes of data.

Here we used a B-colME with $N = 1000$ agents and the mean square error and the number of wrong links over time are shown, along with graphs at a few representative instants. Figs. 3.19, 3.19. and 3.21.

We have also implemented C-colME with agents $N = 1000$ and the mean square error. Fig. 3.22.

Weighted graph: Finally, the weighted graph is used to improve the convergence on both B-colME and C-colME. For weights we used the smallest of three weights, corresponding to the least confidence intervals intersection. We calculated

$$aW_{a,a'}(t) = e^{-\left(2\frac{\bar{x}_{a,a}(t)-\bar{x}_{a',a'}(t)}{2\beta_\delta(t)}\right)^4} \text{ when } A_{a,a'}(t) = 1. \tag{3.66}$$

Then we calculated $e^{-\left(2\frac{\hat{\sigma}_a(t)-\hat{\sigma}_{a'}(t)}{2\beta_\delta(t)}\right)^4}$ and used this value for $AW_{a,a'}(t)$ if it is smaller than the existing $AW_{a,a'}(t)$,

$$aW_{a,a'}(t) = \min\{AW_{a,a'}(t), e^{-\left(2\frac{\hat{\sigma}_a(t)-\hat{\sigma}_{a'}(t)}{2\beta_\delta(t)}\right)^4}\} \tag{3.67}$$

Finally, for $A_{a,a'}(t) = 1$ we calculated $v(t) = e^{-\left(2\frac{\hat{\kappa}_a(t)-\hat{\kappa}_{a'}(t)}{2z_\delta\sqrt{24/t}}\right)^4}$, compared it with the existing $AW_{a,a'}(t)$ and used

$$aW_{a,a'}(t) = \min\{AW_{a,a'}(t), e^{-\left(2\frac{\hat{\kappa}_a(t)-\hat{\kappa}_{a'}(t)}{2z_\delta\sqrt{24/t}}\right)^4}\}. \tag{3.68}$$

The results, with an evident improvement in convergence compared to Figs. 3.20 and 3.22, are shown in Figs. 3.23 and 3.24.

Figure 3.17 shows histograms of the kurtosis for agents across four classes at different time instants. The distributions initially overlap but gradually reflect the differences among the four classes, indicating potential separation based on distribution shape.

Figure 3.18 presents histograms of the estimated standard deviations over time. Variability among classes becomes more distinct as agents evolve, highlighting the role of variance in

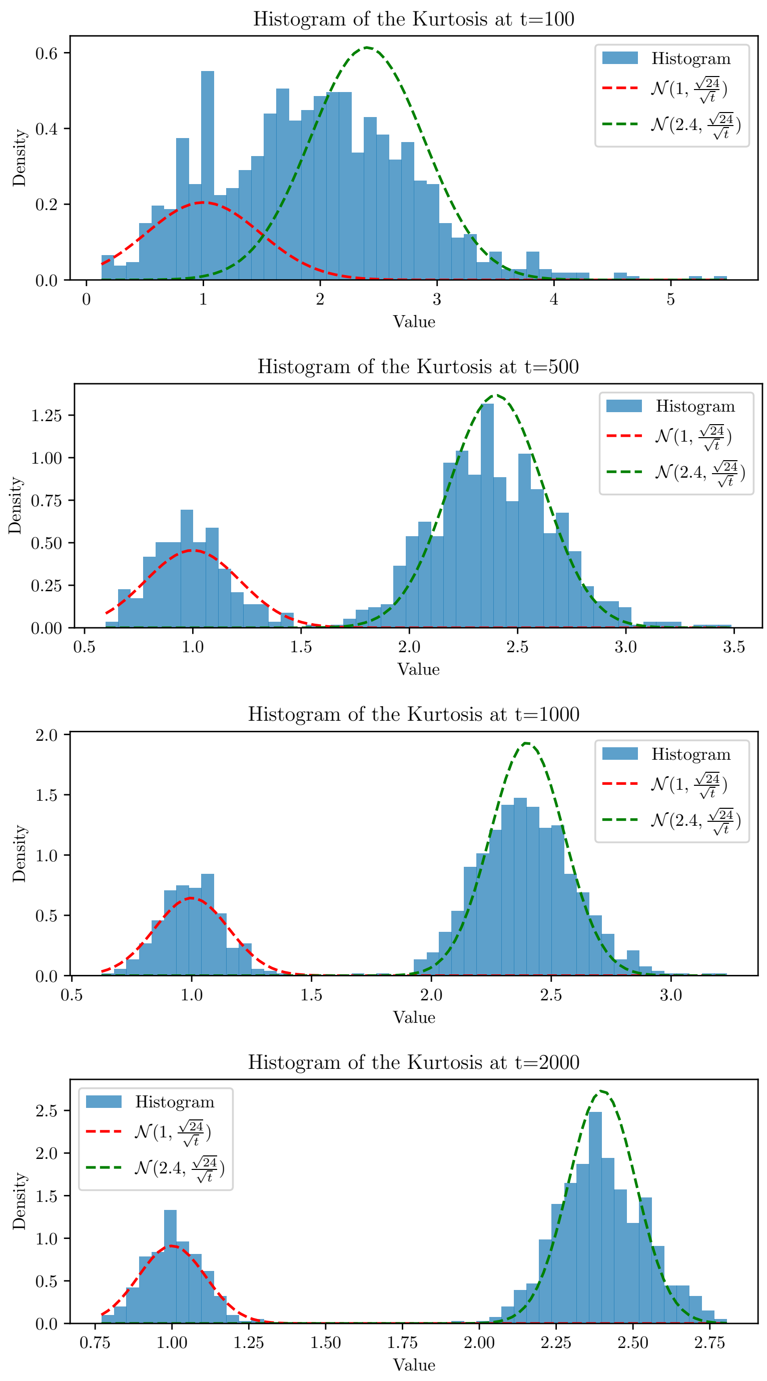


Figure 3.17: Histogram of the kurtosis.

class separation.

Figure 3.19 illustrates the evolution of the network graph. Initially, agents are arbitrarily connected. Over time, edges are adjusted based on mean, variance, and kurtosis confidence intervals, progressively separating the four classes, with more distinct class clustering by $t = 2000$.

Figure 3.20 shows the total MSE over time for B-colME with four classes. The MSE decreases as agents segregate correctly, with different neighborhood sizes influencing the convergence rate.

Figure 3.21 displays the total number of wrong links, indicating inter-class connections.

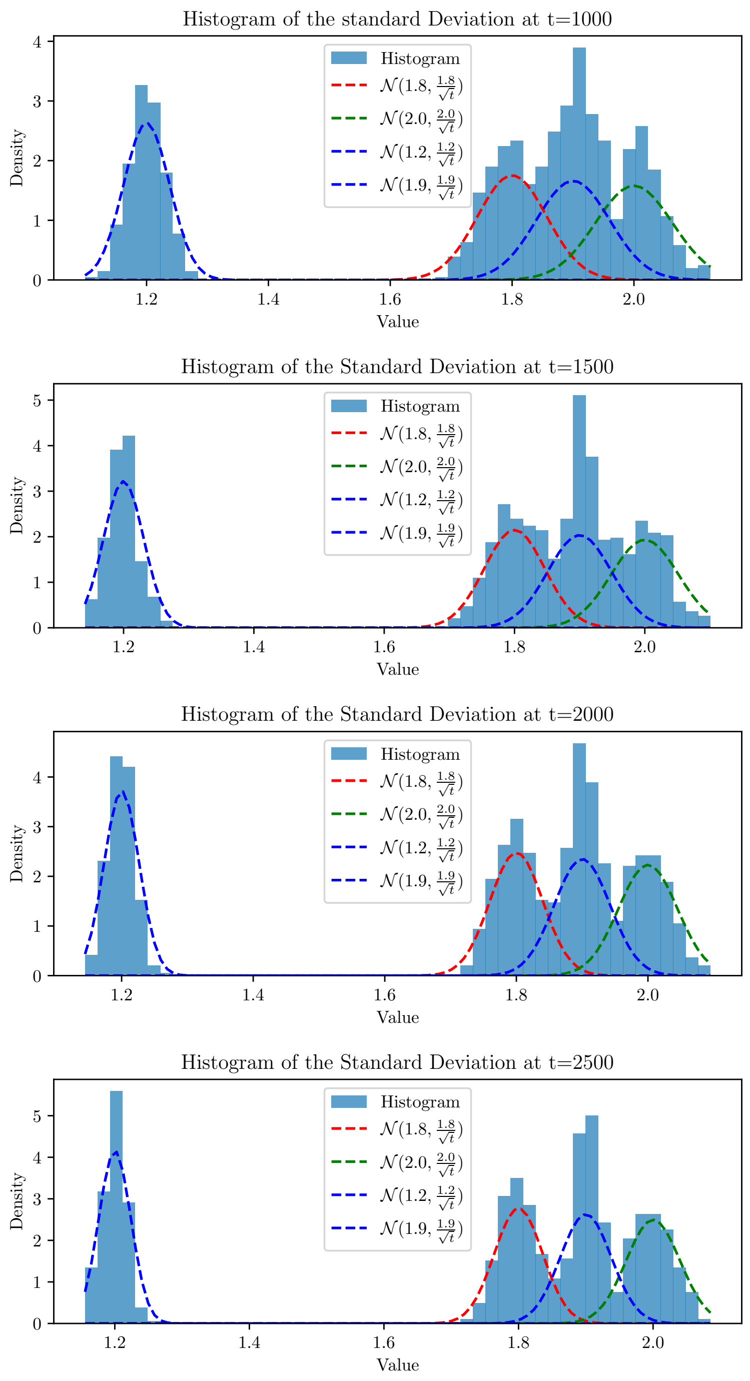


Figure 3.18: Histogram of the standard deviation.

As separation progresses, the number of wrong links diminishes, reflecting improved class distinction.

Figure 3.22 presents the total MSE for C-colME with four classes. The decrease in MSE demonstrates successful collaboration and accurate estimation despite complex class distributions.

Figures 3.23 and 3.24 show the MSE for B-colME and C-colME, respectively, on weighted graphs. Weighting improves convergence by emphasizing the least confident connections, resulting in faster class separation and lower estimation errors.

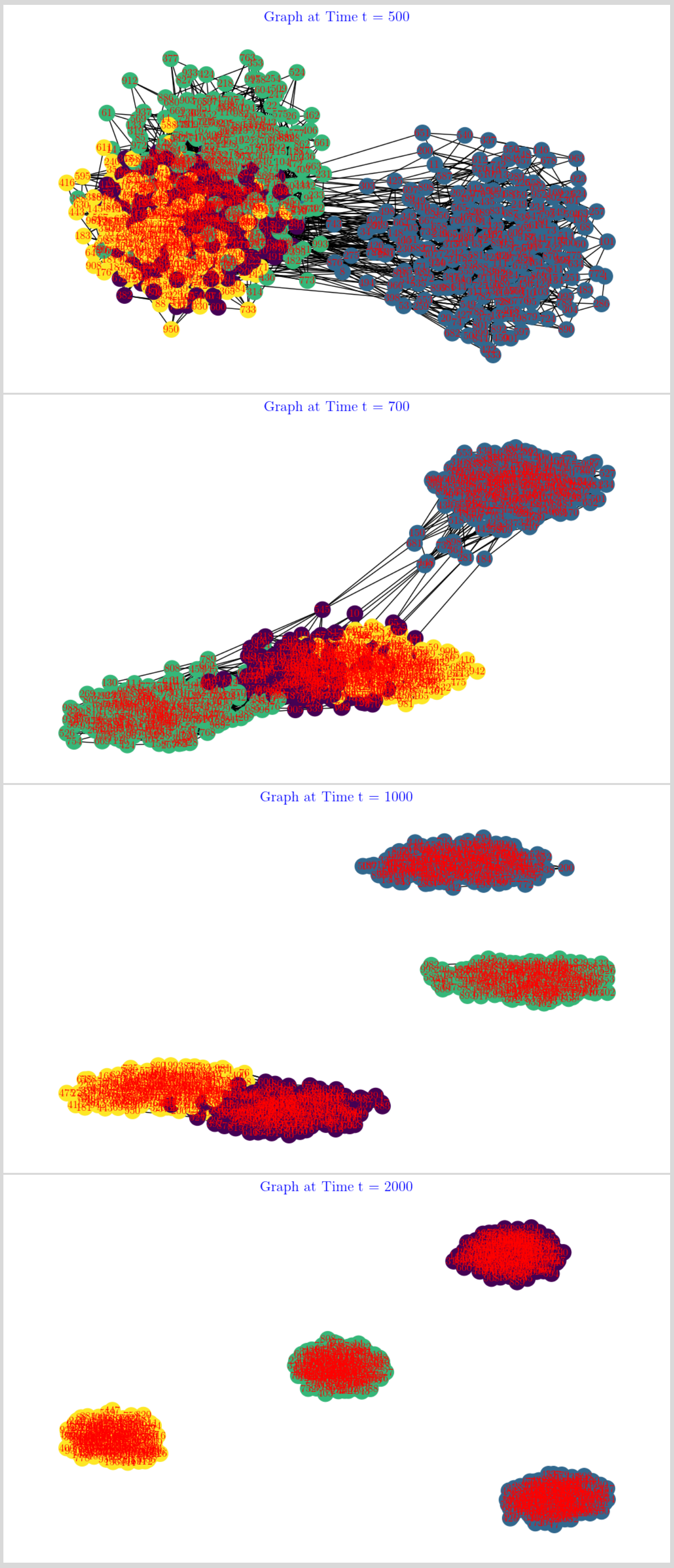
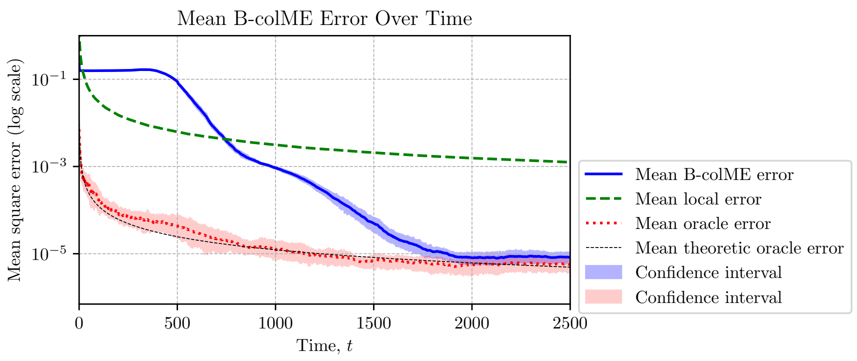


Figure 3.19: B-colME of data with $N = 1,000$ with four classes of agents and no single confidence interval approach can be used. Graphs at $t = 500$, $t = 700$, $t = 1000$, and $t = 2000$, respectively, are shown from top to bottom.



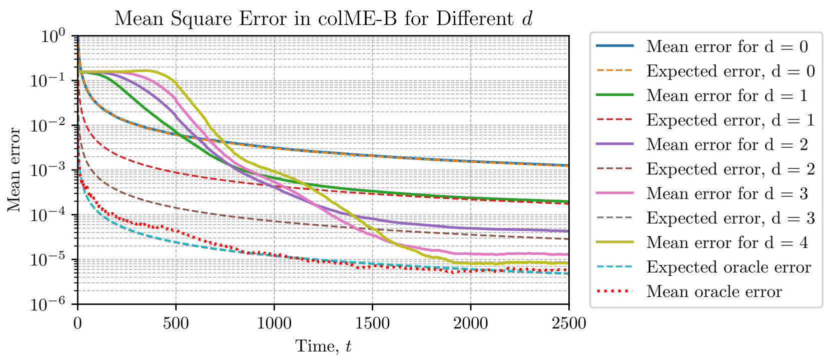


Figure 3.20: B-colME of data with $N = 1,000$ with four classes of agents and no single confidence interval approach can be used. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.

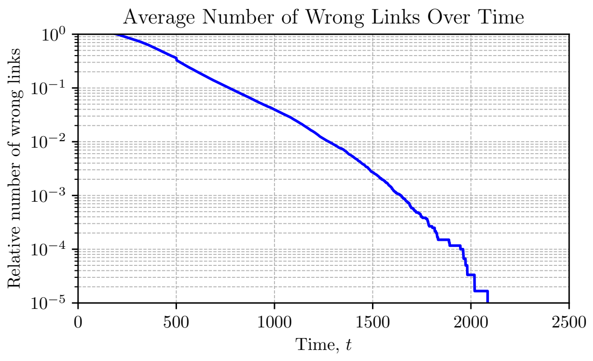


Figure 3.21: B-colME of data with $N = 1,000$ with four classes of agents and no single confidence interval approach can be used: Total number of wrong links.

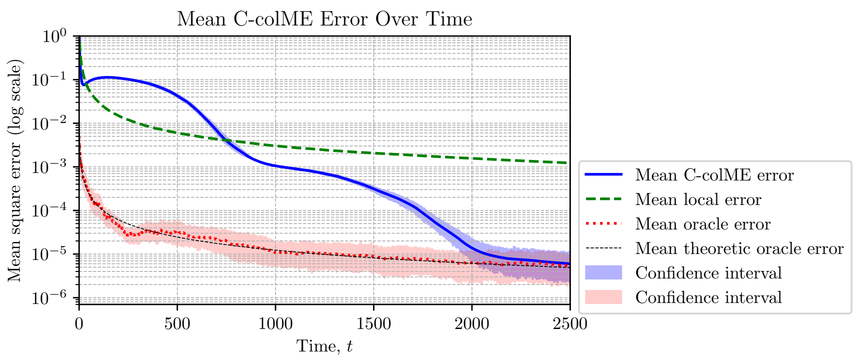
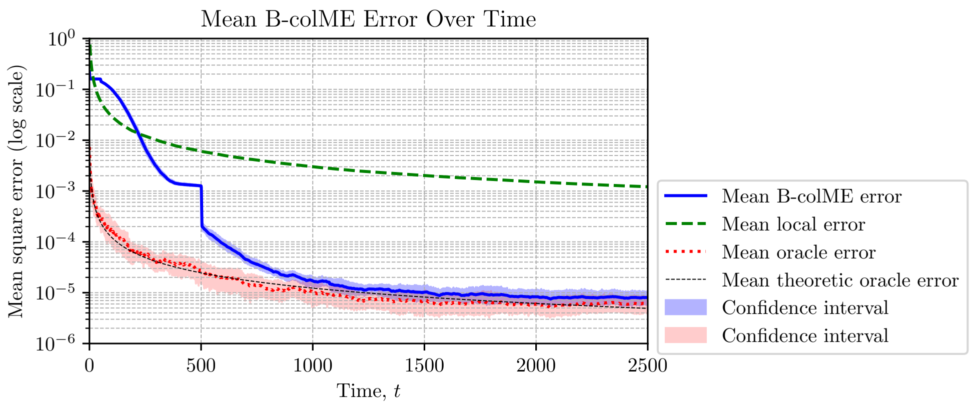


Figure 3.22: C-colME of data with $N = 1,000$ with four classes of agents and no single confidence interval approach can be used. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.



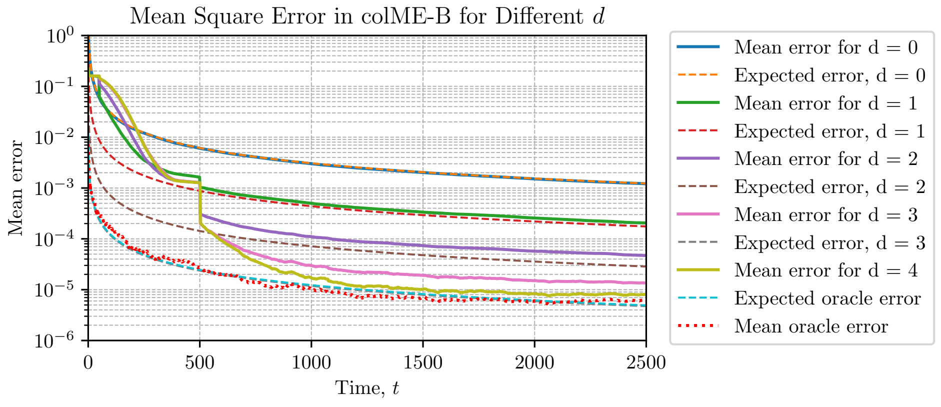


Figure 3.23: B-colME with weighted graph of data with $N = 1,000$ with four classes of agents and no single confidence interval approach can be used. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.

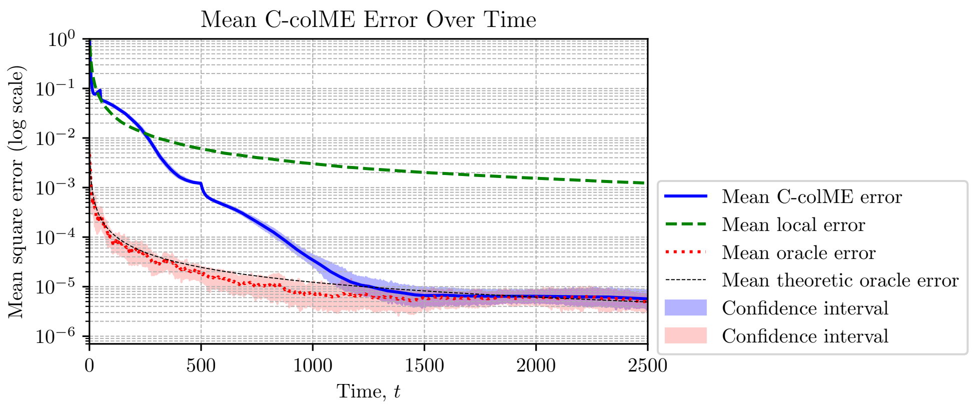


Figure 3.24: C-colME with weighted graph of data with $N = 1,000$ with four classes of agents and no single confidence interval approach can be used. Total MSE of the estimation averaged over all agents at time instants t in 10 realizations.

3.5 Image Collaborative Processing Using C-colME

The image can be represented by its pixels as features. Common two dimensional form of an image can be rewritten in row of pixels. It means that an image is considered as MK-dimensional data vector (row), where M and K are the image sizes, that is

$$\mathbf{x}_a(t) \in \mathbb{R}^{MK}. \tag{3.69}$$

Elements of vector $\mathbf{x}_a(t)$ are $x_a(m, n; t)$ for $m = 1, 2, \dots, M$ and $n = 1, 2, \dots, K$.

Assume that an image received by agent a is a noisy form of the original image $\mu_a(m, n)$, with

$$\mu_a(m, n) = \mathbb{E}_t\{x_a(m, n; t)\}. \tag{3.70}$$

If the images are available for instants $t = 1, 2, \dots, T$ then the noise can be reduced by averaging the images (for each pixel (m, n) separately) available to the agents a

$$\bar{x}_{aa}(m, n; t) = \frac{1}{t}\sum_{i=1}^{t} x_{aa}(m, n; i), \tag{3.71}$$

are the local means for the pixel and agent. The images obtained by these local means will be given in the examples, as references to the other methods.

Estimation of $\mu_a(m, n)$ can be improved if we have agents $a \in \{1, 2, \dots, A = N\}$ who can collaborate. We will assume that the agents receive one of C different images. Therefore, for a successful collaboration, the system must determine which agents receive an image of the same similarity class. To this aim, we will use multidimensional confidence intervals.

The confidence intervals for agents and pixels, are

$$\mathbb{I}_{a,1,1}(t) = [\bar{x}_{aa}(1, 1; t) - \beta_\delta(t), \bar{x}_{aa}(1, 1; t) + \beta_\delta(t)], \tag{3.72}$$

$$\mathbb{I}_{a,1,2}(t) = [\bar{x}_{aa}(1, 2; t) - \beta_\delta(t), \bar{x}_{aa}(1, 2; t) + \beta_\delta(t)], \tag{3.73}$$

$$\vdots \tag{3.74}$$

$$\mathbb{I}_{a,M,K}(t) = [\bar{x}_{aa}(M, K; t) - \beta_\delta(t), \bar{x}_{aa}(M, K; t) + \beta_\delta(t)], \tag{3.75}$$

where $\beta_\delta(t)$ specifies the width of the confidence interval, assuming that the variances are the same for all agents and pixels. The cases with different variances for agents (images) can be implemented with straightforward extension of the presented analysis.

In theory, we may say that the confidence intervals of two images do not intersect if at least one coordinate (pixel) does not intersect. Since even the low-resolution images have 10s of thousands of pixels, in order to achieve a more robust solution we require that the confidence intervals do not intersect for at least, for example 1000 pixels. Then we will disconnect the agents, with images when more than 1000 confidence intervals for the pixels do not intersect.

Therefore, we will form

$$d(m, n) = |\bar{x}_{aa}(m, n; t) - \bar{x}_{bb}(m, n; t)| - \beta_\delta(n; t) - \beta_\delta(m; t) \tag{3.76}$$

for all $m \in \{1, 2, \dots, M\}$, $n \in \{1, 2, \dots, K\}$ and declare that two agents (images) do not belong to the same class if at least 1000 times we get $d(m,n) > 0$.

```python
CIx2 = (sigma_EST[i]+sigma_EST[j]) * np.sqrt(2/(it+1)*(1+1/(it+1))*np.log(np.sqrt(it
    +2)/(0.01/2)))
if np.sum((np.abs(local_image[i] - local_image[j])).flatten() > CIx2)>1000:
    A_current[i,j] = 0
    A_current[j,i] = 0
```

Listing 3.1: Python code for Confidence Intervals Check for Images with Matrix Prunning

Standard Deviation Estimation: The value of the standard deviation, $\hat{\sigma}_a$, can be very easily estimated, since each agent here has an image with MK pixels, with assumed the same variance. Only one image is sufficient for each agent a to estimate the variance,

$$\hat{\sigma}_a = \mathrm{Std}\{(x_a(m,n;0) - x_a(m,n;1))| \;\; m = 1,2,\dots,M, \;\; n = 1,2,\dots,K\}. \tag{3.77}$$

In this way all agents can estimate its own varaince, allowing the case that similarity classes do not have the same varaince of data.

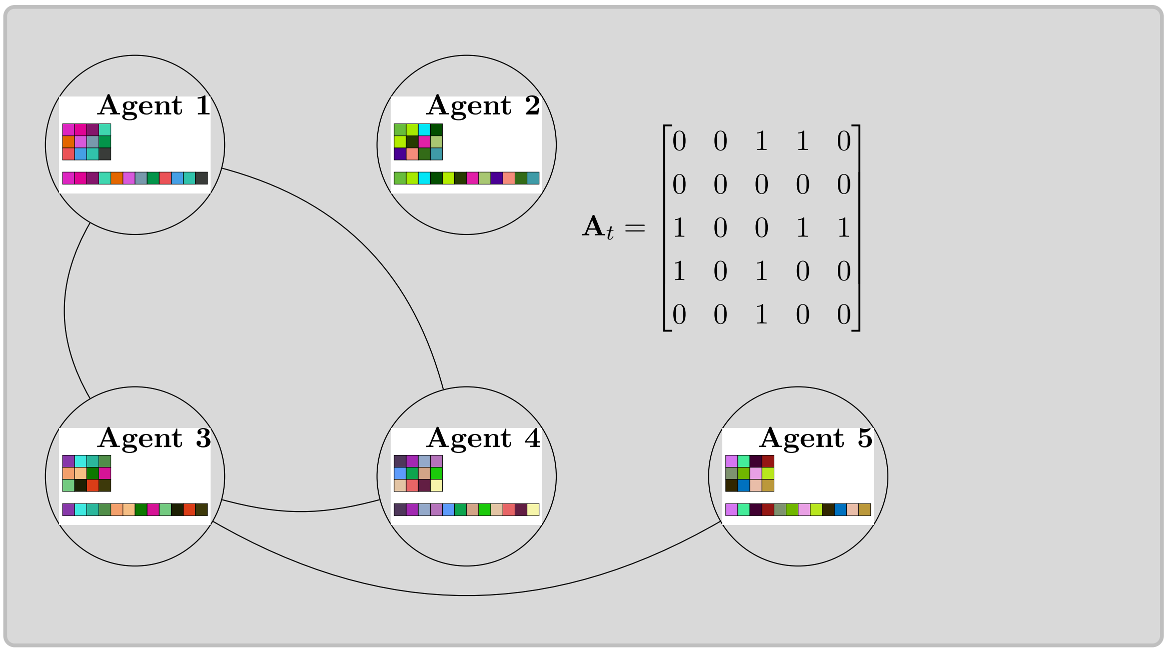
A more complex case would be if the pixels have noise with different variances. We have discussed and simulated this scenario within the previous example, with three-dimensional data.

For the implementation we will use C-colME, and its simplified row stochastic variant. Then

$$\boldsymbol{\mu}(t+1) = (1-\alpha(t))\mathbf{X}(t) + \alpha(t)\mathbf{W}_t\boldsymbol{\mu}(t). \tag{3.78}$$

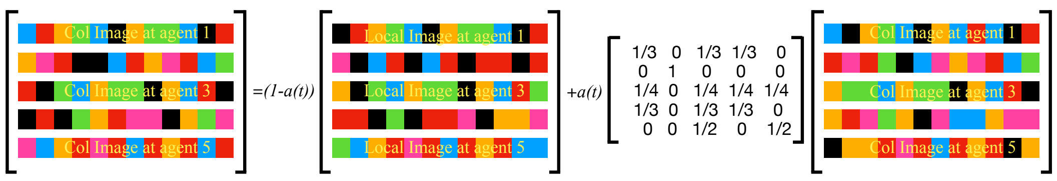
The weight matrix $\mathbf{W}_t$ is formed using the adjacency matrix, updated using confidence intervals to determine the similarity class.

An illustration oof a 4×3 image with 12 pixels (as row images), and five agents connected according to $\mathbf{A}_t$ is given next.



$$\mathbf{A}_t = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

For this system the C-colMe system (with $\mathbf{W}_t$ being 5×5 matrix obtained from the previous matrix $\mathbf{A}_t$ with identity matrix) is shown next, for a row stochastic matrix $\mathbf{W}_t$:



$$\boldsymbol{\mu}(t+1) \quad = \quad (1-\alpha(t)) \quad \mathbf{X}(t) \quad + \quad \alpha(t) \quad \mathbf{W}_t \quad \boldsymbol{\mu}(t), \tag{3.79}$$

Note that the iteration starts with $\boldsymbol{\mu}(t) = \mathbf{X}(0)$ being a matrix with noisy images at N agents as it grows. The matrix has N rows.

3.5.1 Example of Image Denoising using C-colME

Next we will consider a system of $N = 1000$ agents with an initial regular random graph with degree $r = 20$. We will use $t = 0, 1, 2, \dots, T = 2000$, with $\delta = 0.01$.

In the first example, we consider two black-and-white images with 450×300 pixels, representing two similarity classes. Half of the agents receive one (noisy) image and half receive another noisy image. Noise is extremely high, with standard deviation 255, fully degrading a large number of pixels. The system estimates the standard deviation, calculates the means of the images for each agent, checks the confidence intervals, and updates the adjacency matrix. Then it calculates collaborative images for each agent using C-colME. The results for several instants are given next, along with the PSNR showing that the final collaborated image quality is almost 60 dB, meaning an error, lower than 1 amplitude step.

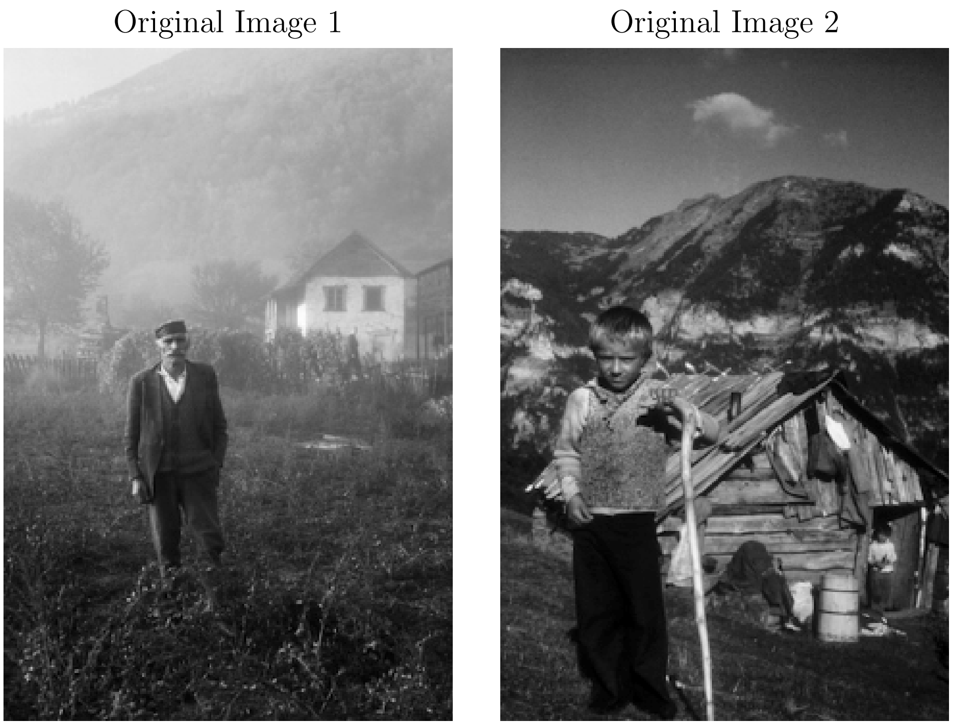


Figure 3.25: Original images from two classes whose noisy forms are received by agents.

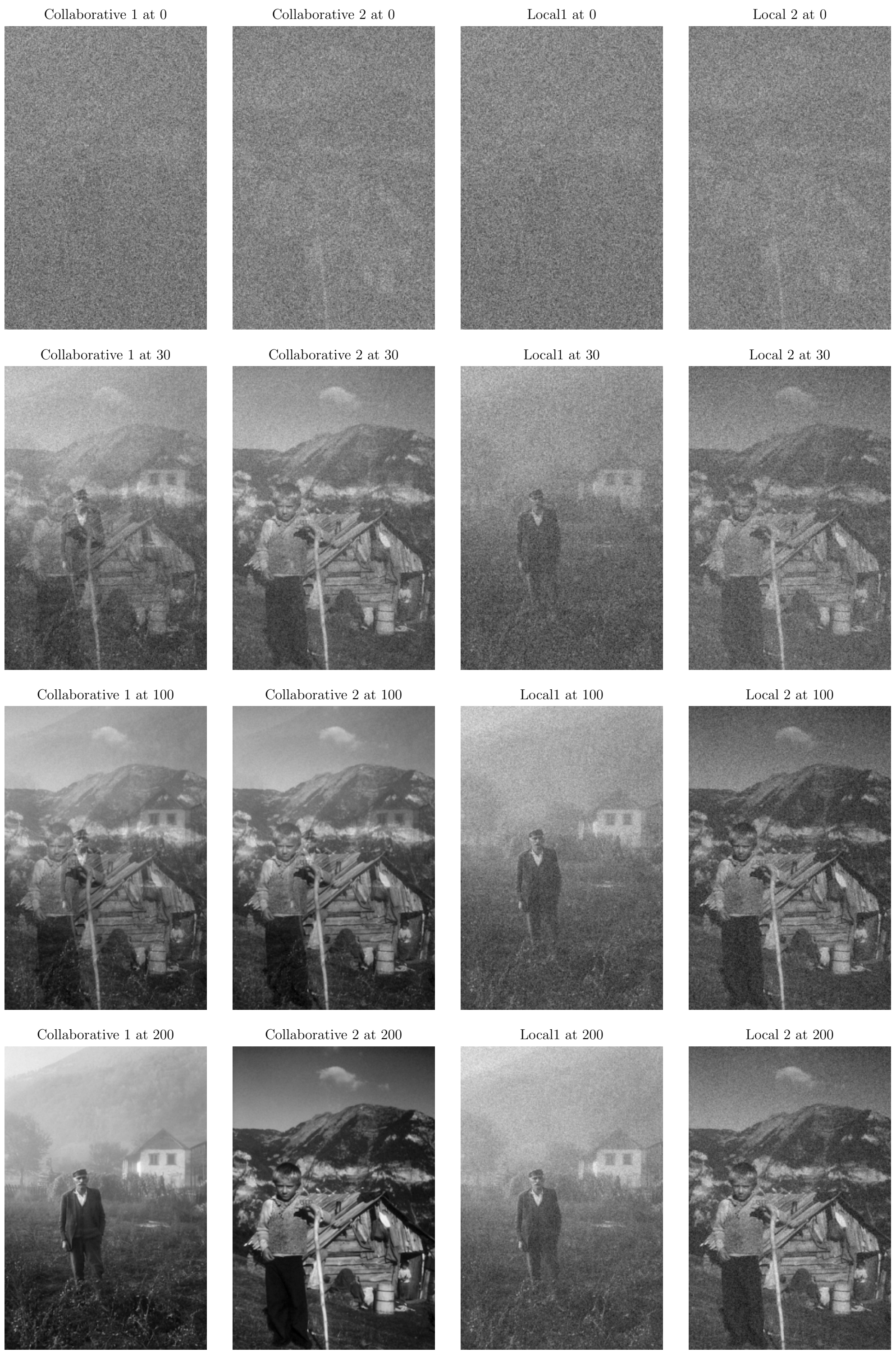


Figure 3.26: Local and collaborative images from two classes at two agents at few time instants, $t = 0$ (input data), $t = 30$, $t = 100$, and $t = 200$.

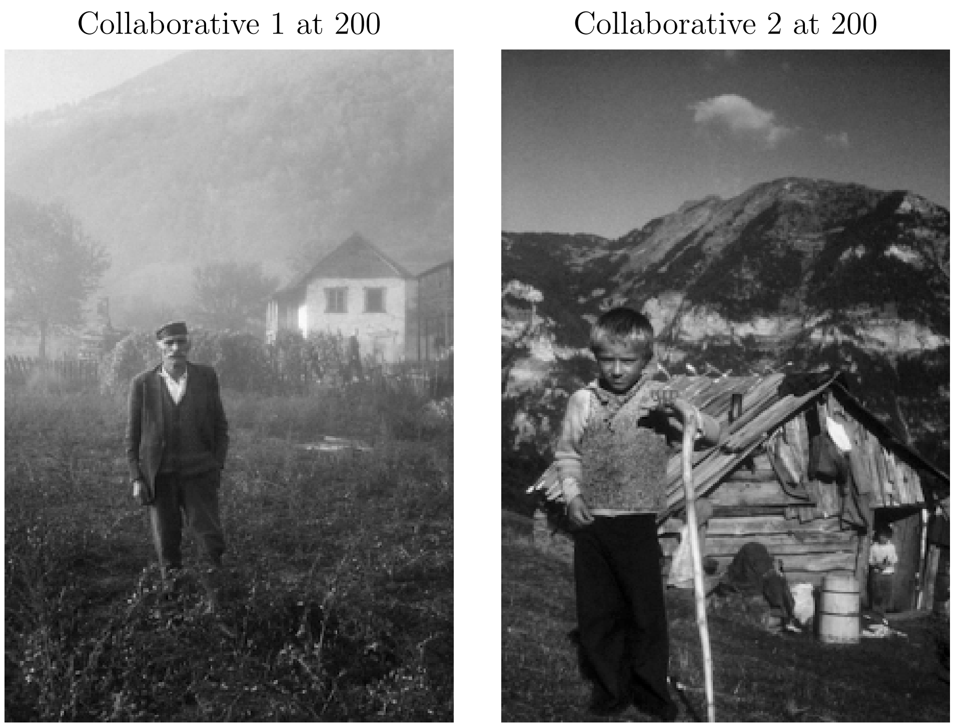


Figure 3.27: Final local and collaborative images from two classes at two agents at $t = 200$.

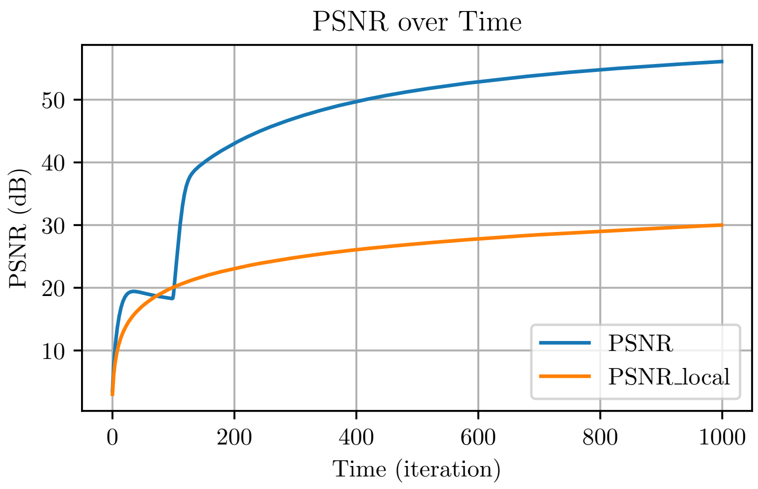
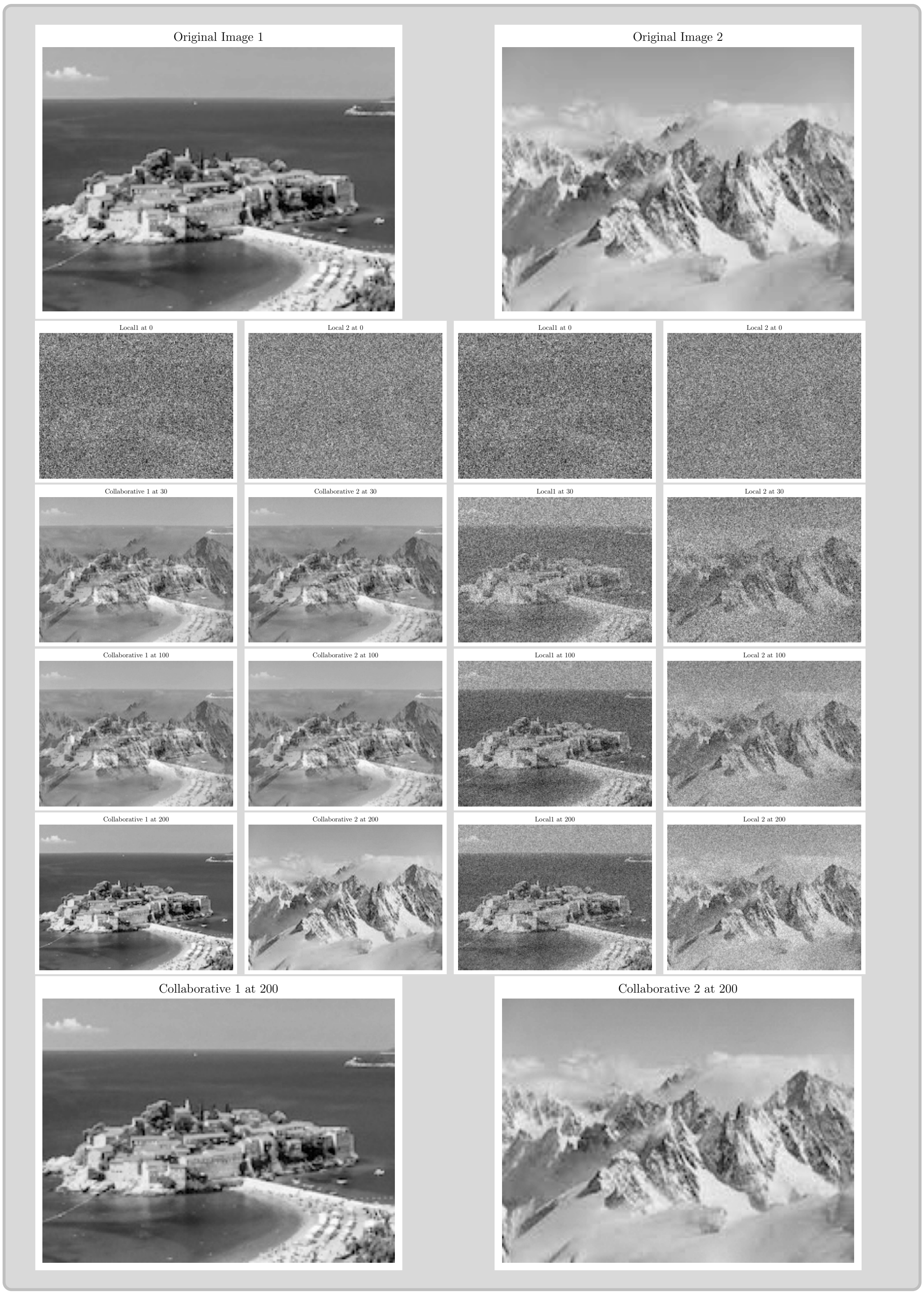


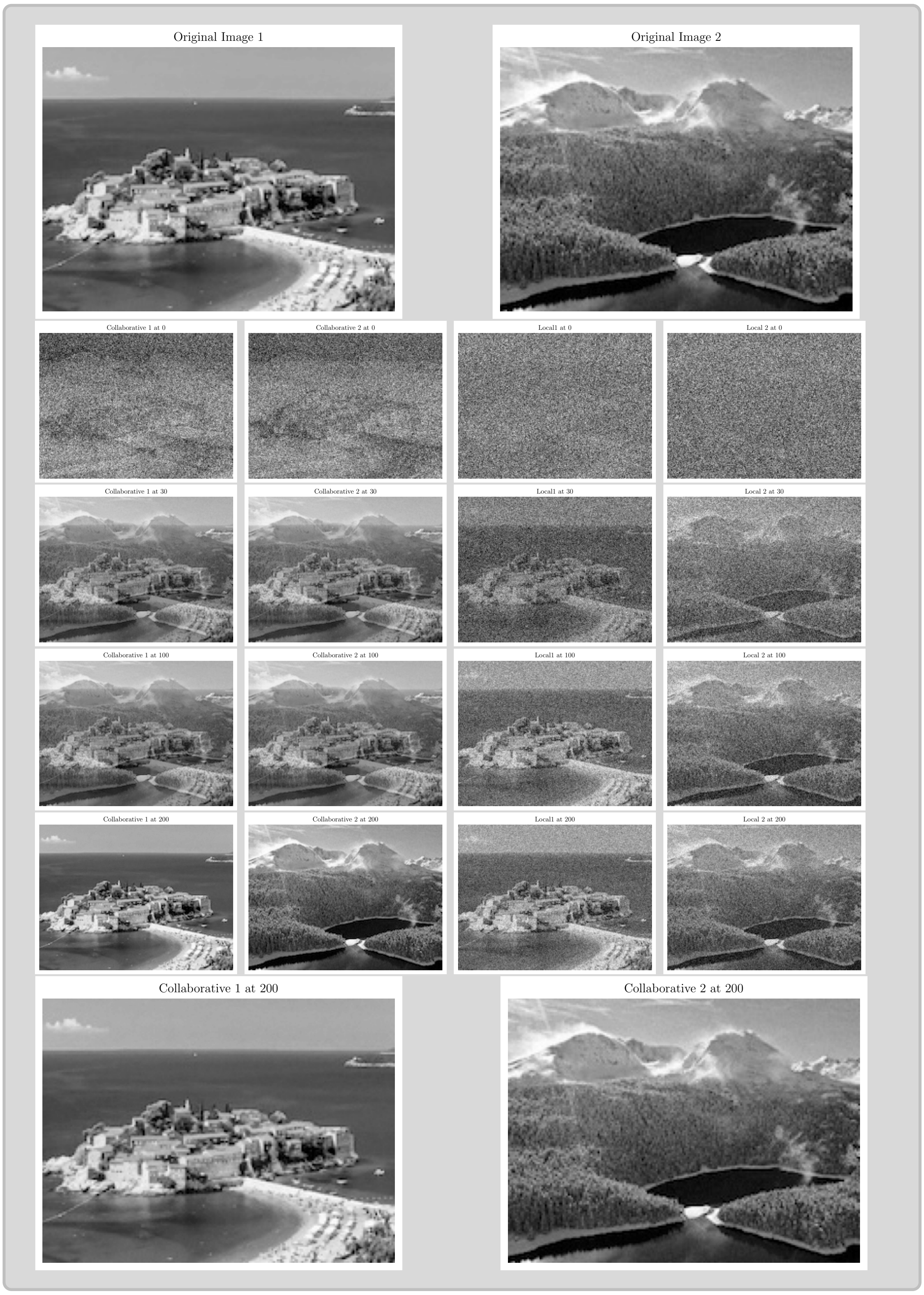
Figure 3.28: PSNR over time for local mean images and C-colMe collaborated images.

One more set of images is considered. The results are presented next.



3.5.2 Example of Image Denoising using B-colME

The same experiment is repeated with other sets of images of 150×200 pixels and B-colME with $d = 3$. The results are presented next.



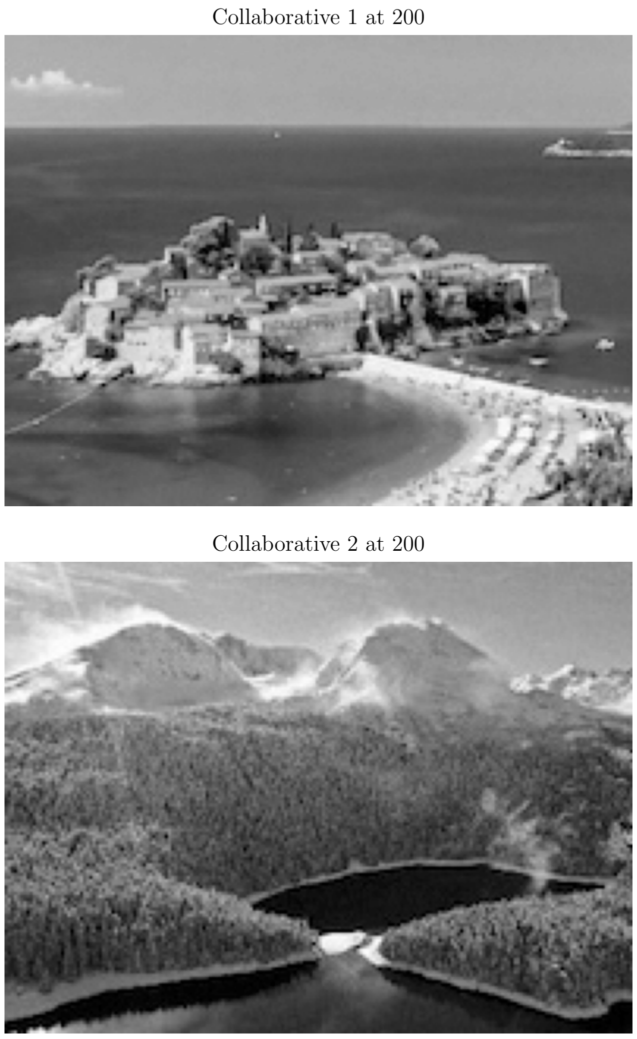


Figure 3.29: Final local and collaborative images from two classes at two agents at $t = 200$.

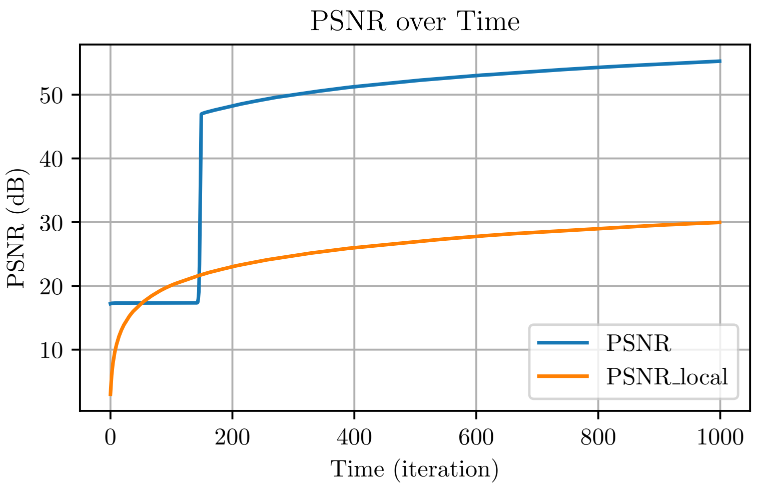
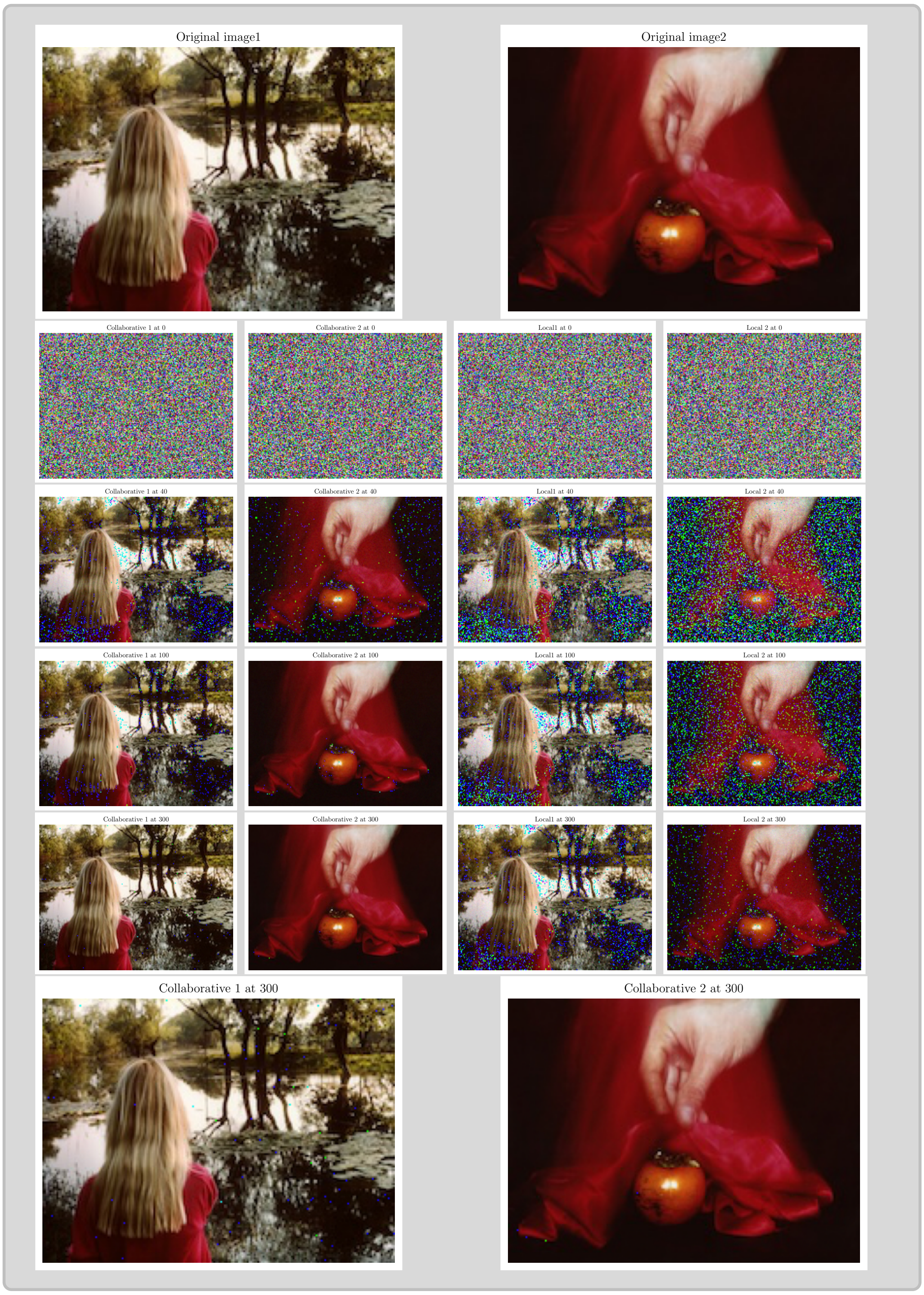


Figure 3.30: PSNR over time for local mean images and B-colMe collaborated images.

3.5.3 Example of Color Image Denoising using C-colME

A set of color images is considered. Color images increase the data size 3 times for R, G, B channels, $\mathbf{x}_a(t) \in \mathbb{R}^{3MK}$. All other calculations are the same. The results are presented next.



Conclusions and Future Work

This thesis investigates the problem of online personalized mean estimation, leveraging a recently developed framework for collaborative learning. We consider a setting where each agent receives data streams from its individual σ-sub-Gaussian distribution and strives to efficiently estimate its mean. Agents collaborate under the assumption of an underlying class structure in which agents within the same class share a similar data distribution. This work examines collaborative algorithms that enable agents to dynamically identify their affiliation to the class and subsequently use estimates from other agents within the same class to enhance the speed and precision of their individual mean estimation. The approach employs AC learning principles, allowing agents to iteratively and confidently exclude agents from different classes. This is analyzed in both the full-scale problem domains and the scalable graph-based analysis domains. We have presented a basic theoretical analysis of these algorithms, focusing on the expected time for an agent to correctly identify its class (separation time) and the time to achieve a target accuracy in its mean estimate. Our analysis emphasizes the importance of confidence intervals and variance estimation in discerning means and similarity classes. We compare the performance of the collaborative algorithms to local, non-collaborative approaches, as well as to an oracle baseline that assumes prior knowledge of the class structure.

The contributions of this thesis are the following.

- A method for the estimation of sample variance is proposed in the collaborative setting of mean estimation, without using the agent's mean value. Estimation of the variance is done locally. Collaboration with neighboring agents can be used to improve the accuracy of the estimate. The cases where the variance is the same for the whole system and when it differs for different classes are considered.
- The sample kurtosis estimation is implemented without using the agents' means.
- The standard deviation of the estimated sample standard deviation and kurtosis is derived. This result, along with sample estimation methods, is used to propose a combined method when confidence intervals are used based on the sample mean, sample standard deviation, and sample kurtosis. By this approach, we were able to consider and use collaborative estimation in very complex cases, like those when the classes have similar means but different standard deviations, or the cases where classes have similar means and standard deviations, but the distribution of data differs, resulting in different kurtosis. Experiments are presented when all three sample estimators must be used.
- In addition to the standard colME, B-colME, and C-colME, we have implemented a simplified row-stochastic C-colME and a Laplacian-Based C-colME. These methods

simplify the calculation and produce very similar results with a shorter calculation time. The Laplacian-Based C-colME does not use computationally demanding divisions.

- We used the forgetting factor in the sums of the B-colME to deal with the time-varying distribution of the data. Considered collaborative methods accumulate data and calculate average based on it. If there is a change in the data, at any instant in time, the data before the change will significantly influence the results for a long time. We introduce the forgetting factor to tackle this issue, as well as the issue of initially accumulated data from wrong classes.
- A reconnection approach with reduced confidence interval widths is described to improve convergence. The convergence of all methods is highly influenced by the width of the confidence interval. By reducing this width, we can improve the convergence, however, we increase the probability of false pruning of the graph and the probability of isolated agents. This can be resolved using the reconnection approach as introduced and analyzed in the thesis. Reconnection avoids isolated terms in all cases.
- To maintain the confidence intervals (thereby avoiding false pruning) while still achieving fast convergence, we can reduce the weight of data points near the ends of these intervals. To achieve this, we use weighted graphs, employing a Gaussian kernel as the graph weight. This approach has been applied to both B-colME and C-colME methods, with experiments confirming the improved convergence .
- Combination of the reconnection and weighted graph, applied to B-colME and C-colME, is used to further improve the convergence and accuracy.
- Estimation of sample variance for multidimensional data with different variances for similarity classes is done.
- The application of collaborative mean estimation and variance estimation to image denoising is implemented. The examples with both black and white and colored images are presented and confirm the methods efficiency.
- A Python code for the implementation of all previous methods is written, from scratch including all functions, using efficient sparse matrix calculations, for one-dimensional data, multidimensional data and for images.

In all our experiments, we have assumed that within the similarity class the distribution of data does not change, that is, the mean, standard deviation, and kurtosis are the same within one class. That is, one-mean estimation methods are used. If the mean values of the data can change slightly within one similarity class, then low-pass filtering approaches would be more appropriate. In that case we should use the low-pass data filtering on graphs, a recently very intensively studied field. We hope that the Laplacian-based C-colMe approach can be a good starting point for that research. Non-stationary analysis is briefly mentioned by using the forgetting factor. This topic would be of significant interest for further research. Another direction for further research would be image de-noising where the noise variance changes pixel-wise within the image. In addition, developing user-friendly software to analyze publicly available data also represents a promising avenue for future work.

Appendix A

Confidence Intervals for Sub-Gaussian Random Variables

Here, we will present the bounds for σ-sub-Gaussian distributions obtained using the Laplace method. We start from the Maillard's Lemma that will be rewritten in its original form, as follows:

Time-uniform sub-Gaussian concentration (Maillard's Lemma) [33]: *Let Y_1, Y_2, $\ldots$,Y_t be a sequence of t independent real-valued random variables where for each $s \leq t$, Y_s has mean μ_s and is σ_s-sub-Gaussian. Then for all $\gamma \in (0,1)$, it holds:*

$$\mathbb{P}\Big(\exists t \in \mathbb{N}, \quad \sum_{s=1}^{t}(Y_s - \mu_s) \geq \sqrt{2\sum_{s=1}^{t}\sigma_s^2(1+\frac{1}{t}\ln(\sqrt{t+1}/\gamma)}\Big) \leq \gamma$$

$$\mathbb{P}\Big(\exists t \in \mathbb{N}, \quad -\sum_{s=1}^{t}(Y_s - \mu_s) \geq \sqrt{2\sum_{s=1}^{t}\sigma_s^2(1+\frac{1}{t}\ln(\sqrt{t+1}/\gamma)}\Big) \leq \gamma.$$

We can write the last probability in the Lemma as

$$\mathbb{P}\Big(\exists t \in \mathbb{N}, \quad |\sum_{s=1}^{t}(Y_s - \mu_s)| \geq \sqrt{2\sum_{s=1}^{t}\sigma_s^2(1+\frac{1}{t}\ln(\sqrt{t+1}/\gamma)}\Big) \leq 2\gamma$$

or by replacing $2\gamma = \delta$ as

$$\mathbb{P}\Big(\exists t \in \mathbb{N}, \quad |\sum_{s=1}^{t}(Y_s - \mu_s)| \geq \sqrt{2\sum_{s=1}^{t}\sigma_s^2(1+\frac{1}{t}\ln(\sqrt{t+1}/(\delta/2))}\Big) \leq \delta.$$

From this result we can get the Laplace method confidence bound. For our sequence of random variables, we can take

$$Y_s = x_a(s)$$

with $\mathbb{E}\{x_a(s)\} = \mu_s = \mu$ and $\mathrm{Var}\{x_a(s)\} = \sigma_s^2 = \sigma^2$, resulting in

$$\mathbb{P}\Big(\exists t \in \mathbb{N}, \quad |\frac{1}{t}\sum_{s=1}^{t}(Y_s - \mu)| \geq \frac{1}{t}\sqrt{2\sum_{s=1}^{t}\sigma^2(1 + \frac{1}{t}\ln(\sqrt{t+1}/(\delta/2))}\Big) \leq \delta.$$

With

$$\bar{x}_{aa}(t) = \frac{1}{t}\sum_{s=1}^{t} Y_s = \frac{1}{t}\sum_{s=1}^{t} x_a(s)$$

we can write

$$\mathbb{P}\Big(\exists t \in \mathbb{N}, \quad |\bar{x}_{aa}(t) - \mu| \geq \sigma\sqrt{\frac{1}{t}(1 + \frac{1}{t}\ln(\sqrt{t+1}/(\delta/2))}\Big) \leq \delta.$$

It means that $\exists t \in \mathbb{N}$ such that the inequality

$$|\bar{x}_{aa}(t) - \mu| \leq \beta(t)$$

holds with probability $1 - \delta$, where

$$\beta(t) = \sigma\sqrt{\frac{1}{t}(1 + \frac{1}{t}\ln(\sqrt{t+1}/(\delta/2))}.$$

We can write

$$-\beta(t) \leq \bar{x}_{aa}(t) - \mu \leq \beta(t)$$

$$\mu - \beta(t) \leq \bar{x}_{aa}(t) \leq \mu + \beta(t)$$

with probability $1 - \delta$.

This means that the random variable $\bar{x}_{aa}(t)$ will assume a value within the interval $\mu - \beta(t), \mu + \beta(t)$ with probability $1 - \delta$ of that the intervals

$$[\bar{x}_{aa}(t) - \beta, \bar{x}_{aa}(t) + \beta(t)]$$

are the confidence intervals of the σ-sub-Gaussian distributed random variable $\bar{x}_{aa}(t)$ with probability $1 - \delta$.

The probabilistic confidence level can be considered as a function of δ. Since the methods and algorithms used in this thesis are based on the assumption that the confidence interval inequality is true, it means the results will have the PAC convergence, with probability of obtained solution with an accuracy ε is defined by δ.

Bibliography

[1] Peter Kairouz et al. “Advances and open problems in federated learning”. In: *Foundations and trends® in machine learning* 14.1–2 (2021), pp. 1–210 (cit. on pp. 1, 3).

[2] Mahsa Asadi, Aurélien Bellet, Odalric-Ambrym Maillard, and Marc Tommasi. “Collaborative algorithms for online personalized mean estimation”. In: *arXiv preprint arXiv:2208.11530* (2022) (cit. on pp. 1, 3, 4, 10, 25, 67, 76).

[3] Franco Galante, Giovanni Neglia, and Emilio Leonardi. “Scalable Decentralized Algorithms for Online Personalized Mean Estimation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 16. 2025, pp. 16699–16707 (cit. on pp. 1, 3, 4, 10, 30, 68, 82, 86).

[4] Fernando Mateo, Juan José Carrasco, Abderrahim Sellami, Mónica Millán-Giraldo, Manuel Domínguez, and Emilio Soria-Olivas. “Machine learning methods to forecast temperature in buildings”. In: *Expert Systems with Applications* 40.4 (2013), pp. 1061–1068 (cit. on p. 1).

[5] Xiaoyuan Su and Taghi M. Khoshgoftaar. “A survey of collaborative filtering techniques”. In: *Advances in Artificial Intelligence* (2009). 2009a (cit. on p. 1).

[6] Erwin Adi, Adnan Anwar, Zubair Baig, and Sherali Zeadally. “Machine learning and data analytics for the IoT”. In: *Neural computing and applications* 32 (2020), pp. 16205–16233 (cit. on pp. 1, 4, 5).

[7] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. “Towards personalized federated learning”. In: *IEEE transactions on neural networks and learning systems* 34.12 (2022), pp. 9587–9603 (cit. on p. 3).

[8] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran. “An Efficient Framework for Clustered Federated Learning”. In: *Advances in Neural Information Processing Systems*. 2020 (cit. on p. 3).

[9] A. Fallah, A. Mokhtari, and A. Ozdaglar. “Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach”. In: *Advances in Neural Information Processing Systems*. 2020 (cit. on p. 3).

[10] F. Sattler, K.-R. Müller, and W. Samek. “Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2021), pp. 3710–3722 (cit. on p. 3).

[11] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. “Federated Learning: Challenges, Methods, and Future Directions”. In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60 (cit. on p. 3).

[12] T. Li, S. Hu, A. Beirami, and V. Smith. “Ditto: Fair and Robust Federated Learning Through Personalization”. In: *Proceedings of the 38th International Conference on Machine Learning*. 2021 (cit. on p. 3).

[13] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. “Federated Multi-Task Learning under a Mixture of Distributions”. In: *NeurIPS*. 2021 (cit. on p. 3).

[14] S. Ding and W. Wang. “Collaborative Learning by Detecting Collaboration Partners”. In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 15629–15641 (cit. on p. 3).

[15] M. Even, L. Massoulié, and K. Scaman. “On Sample Optimality in Personalized Collaborative and Federated Learning”. In: *Advances in Neural Information Processing Systems*. 2022 (cit. on p. 3).

[16] Eshcar Hillel, Zohar Shay Karnin, Tomer Koren, Ronny Lempel, and Oren Somekh. “Distributed exploration in multi-armed bandits”. In: *NIPS*. 2013 (cit. on p. 3).

[17] Chao Tao, Qin Zhang, and Yuan Zhou. “Collaborative learning with limited interaction: Tight bounds for distributed exploration in multi-armed bandits”. In: *FOCS*. 2019 (cit. on p. 3).

[18] Po-An Wang, Alexandre Proutiere, Kaito Ariu, Yassir Jedra, and Alessio Russo. “Optimal algorithms for multiplayer multi-armed bandits”. In: *International Conference on Artificial Intelligence and Statistics*. 2020a. PMLR, 2020, pp. 4120–4129 (cit. on p. 3).

[19] Abishek Sankararaman, Ayalvadi Ganesh, and Sanjay Shakkottai. “Social learning in multi agent multi armed bandits”. In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3.3 (2019), pp. 1–35 (cit. on p. 3).

[20] David Martínez-Rubio, Varun Kanade, and Patrick Rebeschini. “Decentralized cooperative stochastic bandits”. In: *Proc. of NIPS*. 2019 (cit. on p. 3).

[21] Yuanhao Wang, Jiachen Hu, Xiaoyu Chen, and Liwei Wang. “Distributed bandit learning: Near-optimal regret with efficient communication”. In: *ICLR*. 2020b. 2020 (cit. on p. 3).

[22] Peter Landgren, Vaibhav Srivastava, and Naomi Ehrich Leonard. “Distributed cooperative decision making in multi-agent multi-armed bandits”. In: *Automatica* 125 (2021), p. 109445 (cit. on p. 3).

[23] Udari Madhushani, Abhimanyu Dubey, Naomi Ehrich Leonard, and Alex Pentland. “One more step towards reality: Cooperative bandits with imperfect communication”. In: *NeurIPS*. 2021 (cit. on p. 3).

[24] Etienne Boursier and Vianney Perchet. “Sic - mmab: Synchronisation involves communication in multiplayer multi-armed bandits”. In: *NeurIPS*. 2019 (cit. on p. 3).

[25] Chengshuai Shi, Cong Shen, and Jing Yang. “Federated multi-armed bandits with personalization”. In: *AISTATS*. 2021 (cit. on pp. 3, 4).

[26] Nikolai Karpov and Qin Zhang. “Collaborative best arm identification with limited communication on non-iid data”. In: *arXiv preprint arXiv:2207.08015* (2022) (cit. on p. 3).

[27] Clémence Réda, Sattar Vakili, and Emilie Kaufmann. “Near-optimal collaborative learning in bandits”. In: *NeurIPS*. 2022 (cit. on p. 4).

[28] Tomáš Kocák and Aurélien Garivier. “Best arm identification in spectral bandits”. In: *IJCAI*. 2020 (cit. on p. 4).

[29] Tomáš Kocák and Aurélien Garivier. “Epsilon best arm identification in spectral bandits”. In: *IJCAI*. 2021 (cit. on p. 4).

[30] F. E. Dorner, N. Konstantinov, G. Pashaliev, and M. Vechev. “Incentivizing honesty among competitors in collaborative learning and optimization”. In: *Advances in Neural Information Processing Systems*. Vol. 36. 2024, pp. 7659–7696 (cit. on p. 4).

[31] N. Tsoy, A. Mihalkova, T. N. Todorova, and N. Konstantinov. “Provable Mutual Benefits from Federated Learning in Privacy-Sensitive Domains”. In: *International Conference on Artificial Intelligence and Statistics*. 2024 (cit. on p. 4).

[32] Felix Grimberg, Mary-Anne Hartley, Sai P Karimireddy, and Martin Jaggi. “Optimal model averaging: Towards personalized collaborative learning”. In: *arXiv preprint arXiv:2110.12946* (2021) (cit. on p. 4).

[33] Odalric-Ambrym Maillard. “Mathematics of statistical sequential decision making”. PhD thesis. Université de Lille, Sciences et Technologies, 2019 (cit. on pp. 10, 123).

[34] Ronald Aylmer Fisher. “The moments of the distribution for normal samples of measures of departure from normality”. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 130.812 (1930), pp. 16–28 (cit. on p. 100).