Cooperative Kernel regression

AKIL Zakaria, GRABET Rémy, NEL Louis

March 2025

Sommaire

Introduction

For this project we implemented various distributed optimization algorithms to solve the kernel regression problem across a network of communicating agents. The project is divided into three parts, each focusing on different aspects of the problem and algorithms.

1 Part I: Decentralized Optimization

1.1 Data Visualization

We started by loading and visualizing the data from the first database. The data consists of one million points.

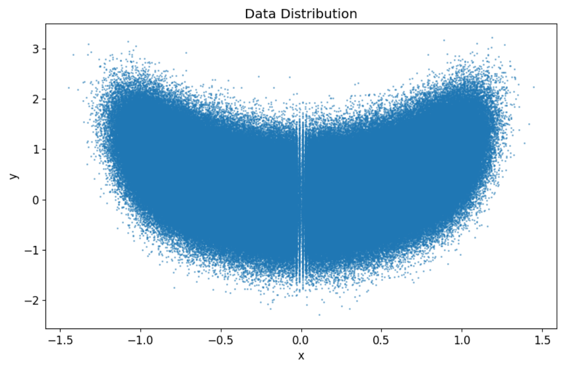


Figure 1: Visualization of the data points from the first database.

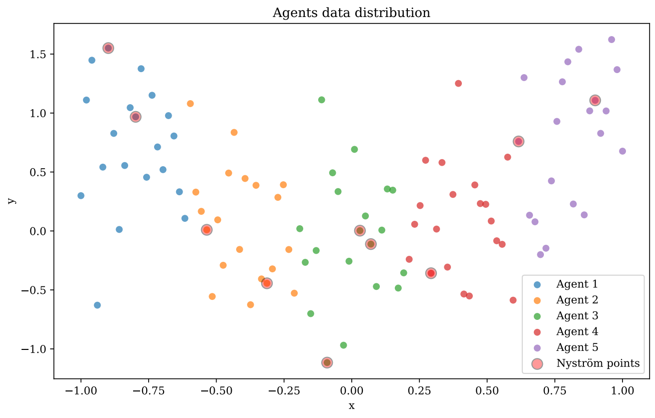
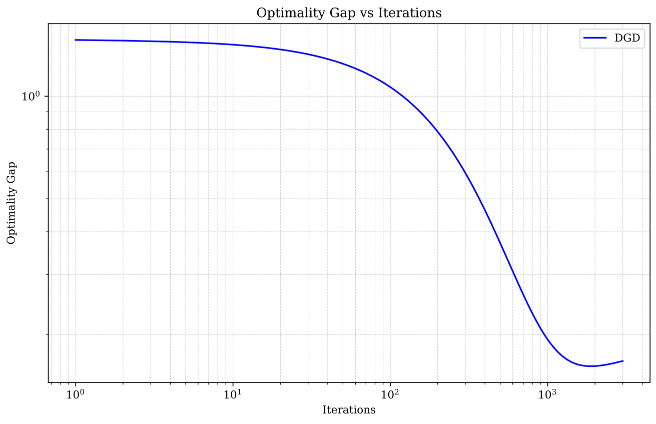


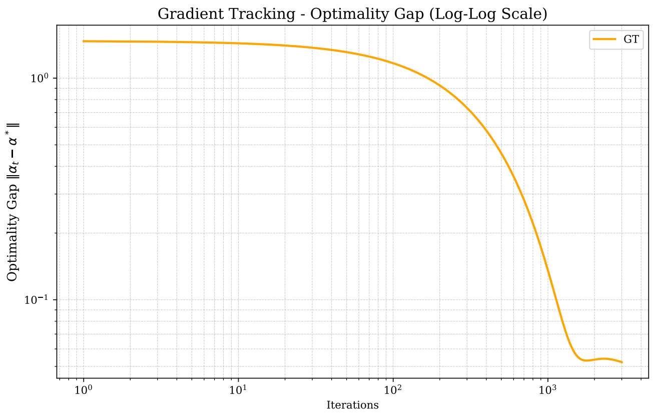
Figure 2: Visualization of 100 data points attributed to 5 agents.

1.2 Algorithm Comparison

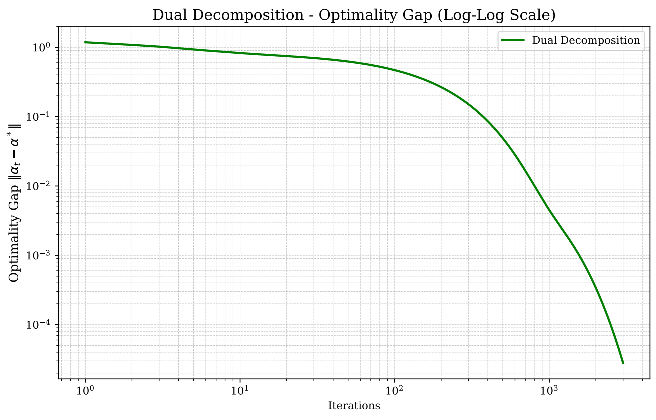
We implemented the four algorithms and we applied them in order to minimize the kernel regression objective function. We observe that for the same number of iterations, dual methods (Dual Decomposition and ADMM) reach very lower gaps compared to the primal methods.



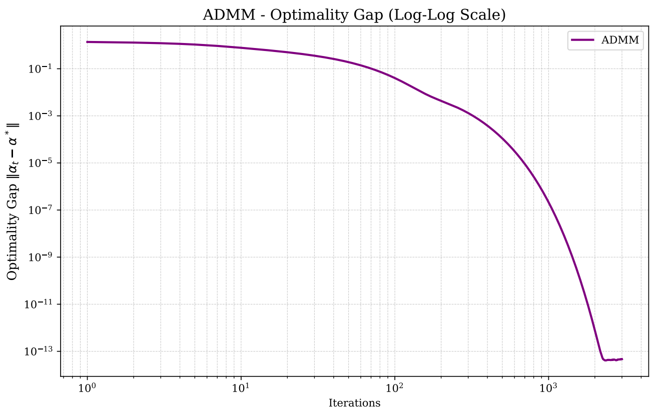
(a) DGD convergence



(b) Gradient tracking convergence



(c) Dual decomposition



(d) ADMM convergence

Figure 3: Optimality gap $\|\alpha_t^i - \alpha^*\|$ for different algorithms (log-log scale)

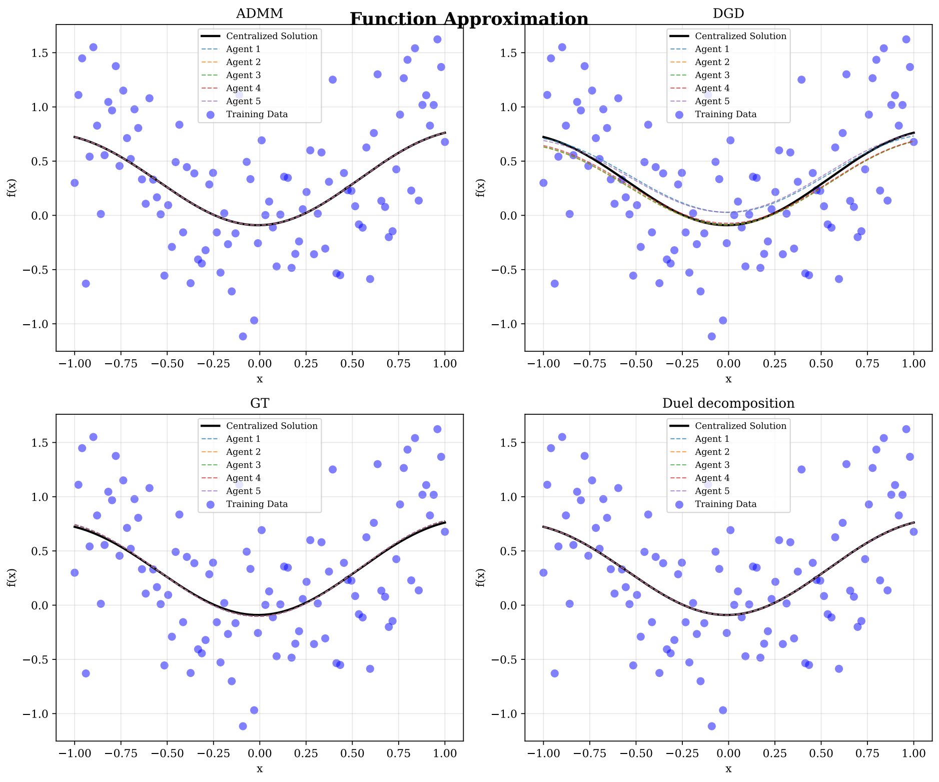
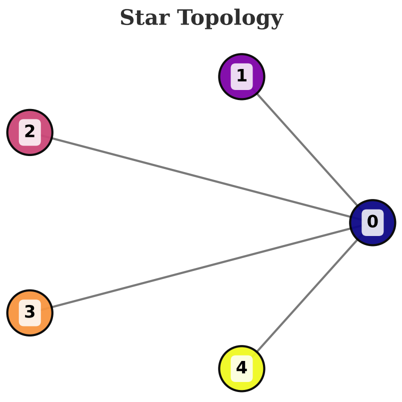
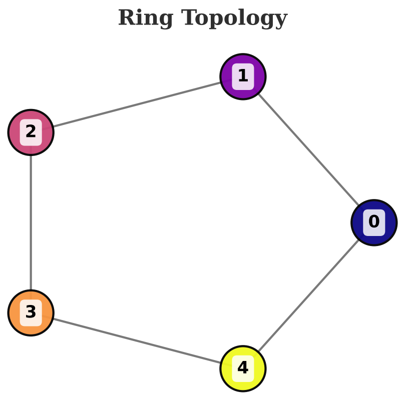


Figure 4: Approximation functions of agents for each algorithm

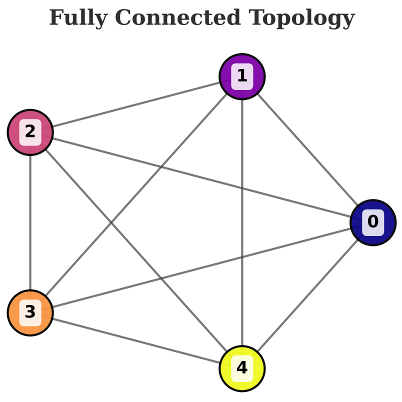
1.3 Topology Impact



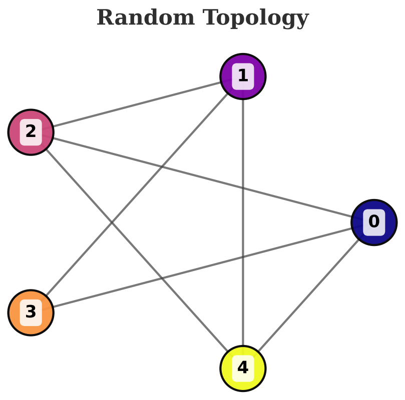
(a) Star topology



(b) Ring topology



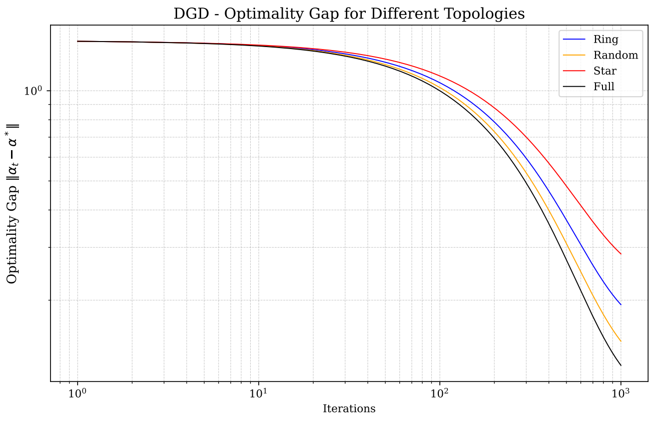
(c) Fully connected



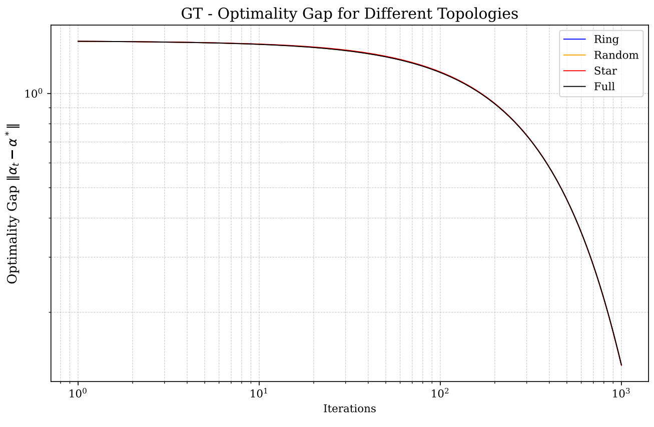
(d) Random

Figure 5: Undirected graph topologies for experiments

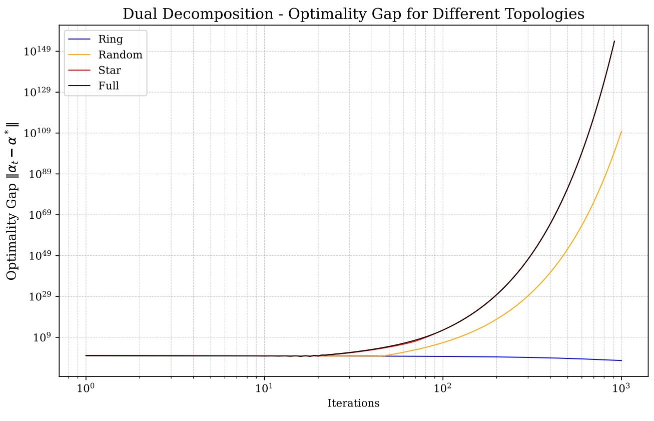
The figures below demonstrate the interplay between network topology and algorithm performance in distributed optimization. All methods—DGD, GT, and ADMM—exhibit decreasing optimality gaps over iterations, confirming convergence, though at varying rates. Fully Connected topologies achieve the fastest convergence due to maximal agent connectivity, while Ring and Star topologies lag, constrained by sequential or centralized communication patterns. ADMM outperforms DGD and GT in sparse networks, leveraging its dual-variable framework to mitigate connectivity limitations. DGD's reliance on frequent neighbor updates makes it highly topology-sensitive, whereas GT's gradient tracking mechanism improves robustness. Notably, ADMM maintains steady progress even in restrictive topologies like Ring, highlighting its suitability for decentralized settings. These results underscore the trade-off between communication efficiency (e.g., Ring/Star) and convergence speed (Fully Connected), emphasizing the need to align algorithm and topology choices with application-specific constraints such as scalability and network reliability.



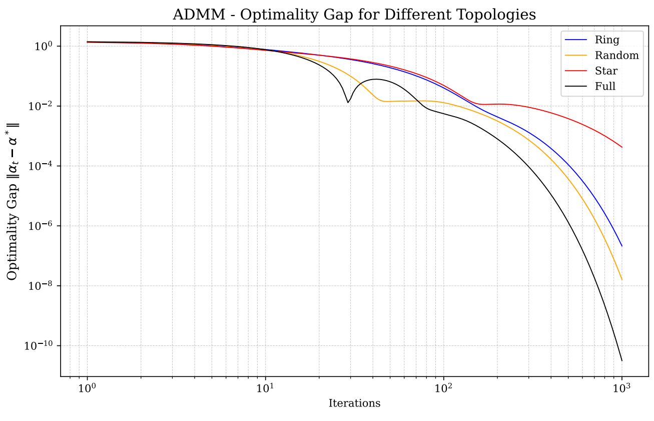
(a) DGD



(b) GT



(c) Dual Decomposition

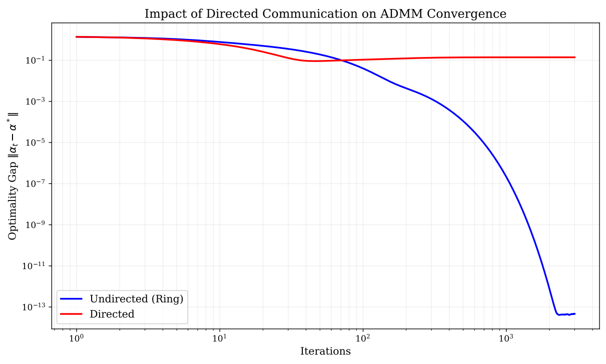


(d) ADMM

Figure 6: Algos convergence accross different Undirected graph topologies

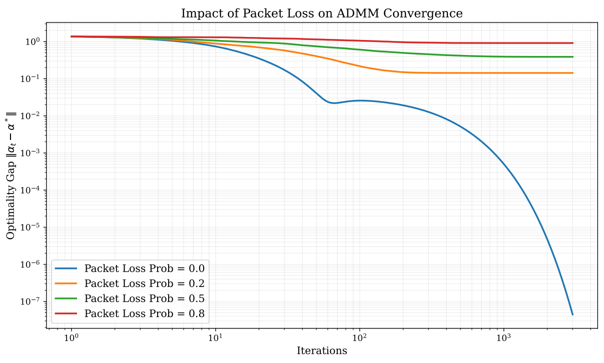
1.4 Adversarial Conditions

We examined our algorithms with undirected graphs, packet loss, and asynchronous updates, as discussed in class [1]. Figure 7 illustrates ADMM's behavior under these conditions.

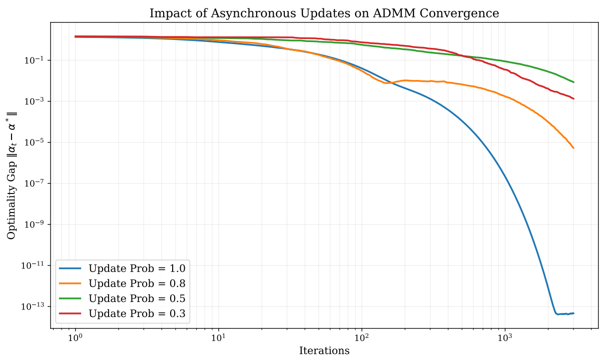


(a) Directed vs undirected graphs

Figure 7: Algorithm robustness analysis



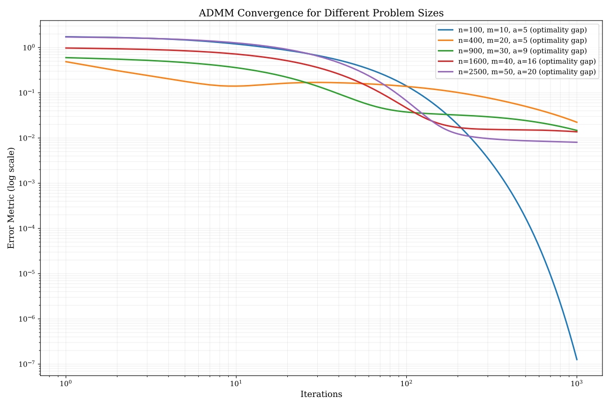
(a) Packet loss impact (ADMM)



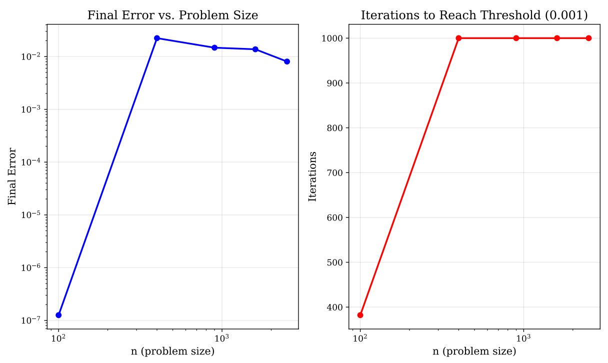
(b) Asynchreounous Update (ADMM)

Figure 7 evaluates the performance of the algorithm under adversarial conditions (e.g., packet loss, directed graphs). ADMM exhibits poor robustness, losing stable convergence when dealing with directed graphs. The dual decomposition, in contrast, proved robust to this. For an asynchronous update scenario, ADMM keeps its convergence, but with a slower rate, which is expected.

1.5 Large scale experiments



(a) Directed vs undirected graphs



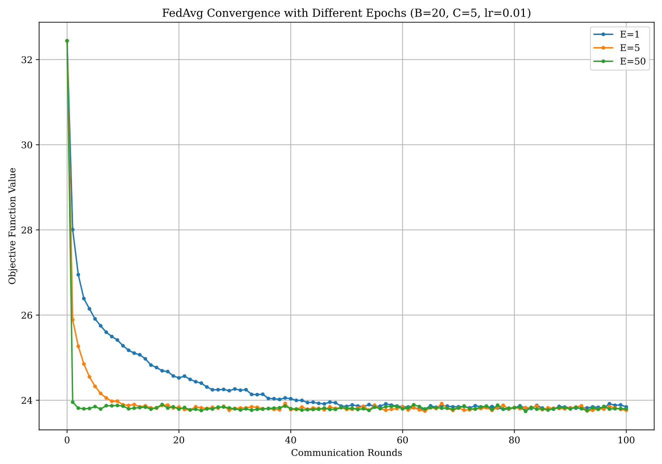
(b) Packet loss impact (ADMM)

Figure 9: ADMM in large scale

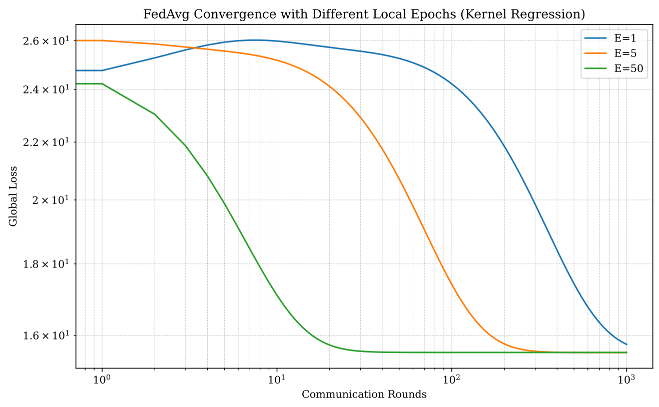
Figure 9 demonstrates ADMM's robust convergence across increasing problem sizes (n=100to n=2500), maintaining stable optimality gaps despite scaling and agent counts (a). Larger problems require marginally more iterations, reflecting ADMM's linear scalability in distributed settings. The results validate ADMM's suitability for large-scale distributed optimization, with topology choice critically impacting performance.

2 Part II: Federated Learning Analysis

2.1 FedAvg Implementation and Convergence



(a) Impact of local epochs (E=1,5,50)

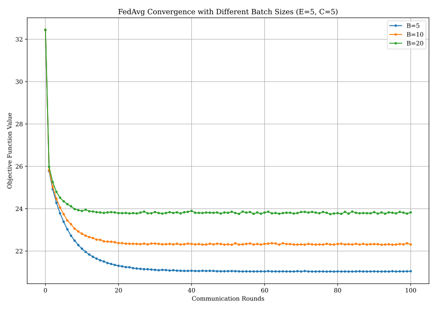


(b) Global objective convergence log-log scale

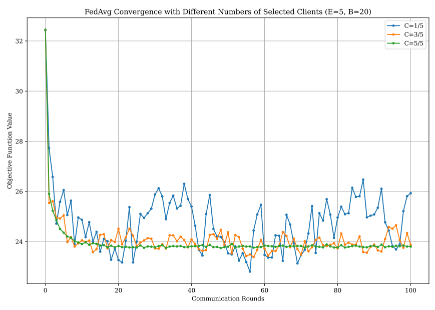
Figure 10: FedAvg performance with $B = 20$, $C = 5$, and constant LR $\eta = 0.01$

Analysis: The convergence rate aligns with the theoretical bound $\mathcal{O}(1/\sqrt{T})$ for non-convex federated optimization. Increasing local epochs (Fig. 10a) reduces communication rounds but introduces client drift, as evidenced by the oscillating pattern when $E = 50$. This matches the trade-off between computation and communication costs discussed in class. The linear convergence phase (Fig. 10b) confirms the effectiveness of client sampling ($C = 5$) in reducing variance.

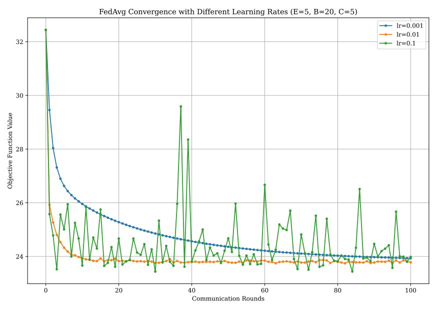
2.2 Hyperparameter Sensitivity Study



(a) Batch size ($B = 5, 10, 20$)



(b) Client participation ($C = 1, 3, 5$)



(c) Learning rates ($\eta = 0.1, 0.01, 0.001$)

Figure 11: Parameter sensitivity analysis ($E = 5$ fixed)

Analysis: Figure 11a demonstrates that larger batches ($B = 20$) stabilize updates, consistent with the variance reduction principle but leads to optimality efficiency even with more communication rounds. Partial participation of clients (Fig. 11b) introduces quantization noise, but convergence remains intact due to the client sampling scheme satisfying $E[w_{t+1}] = w_t - \eta_t \nabla F(w_t)$. The learning rate comparison (Fig. 11c) validates the upper bound $\eta < 1/L$ where L is the smoothness constant.

2.3 Learning Rate Adaptation

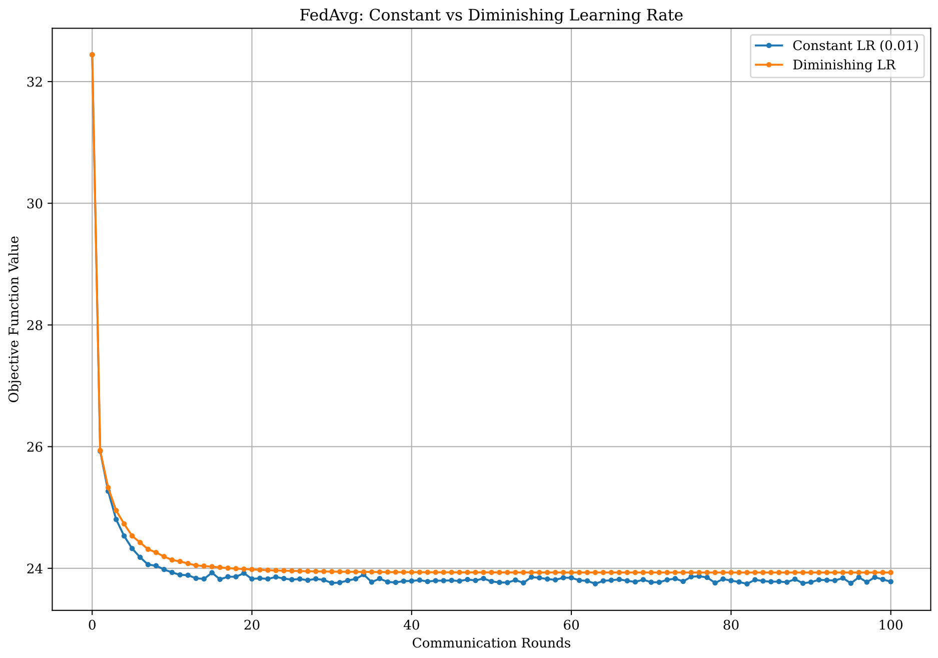


Figure 12: Constant vs diminishing learning rate ($\eta_t = \eta_0/\sqrt{t+1}$)

Analysis: The diminishing schedule achieves final convergence accuracy as predicted by the Robbins-Monro conditions ($\sum \eta_t = \infty$, $\sum \eta_t^2 < \infty$). However, the constant rate yields faster initial progress, supporting the practice of learning rate decay after initial warmup. This matches the theoretical recommendation for non-convex objectives where early large steps escape saddle points.

2.4 Global Model Validation

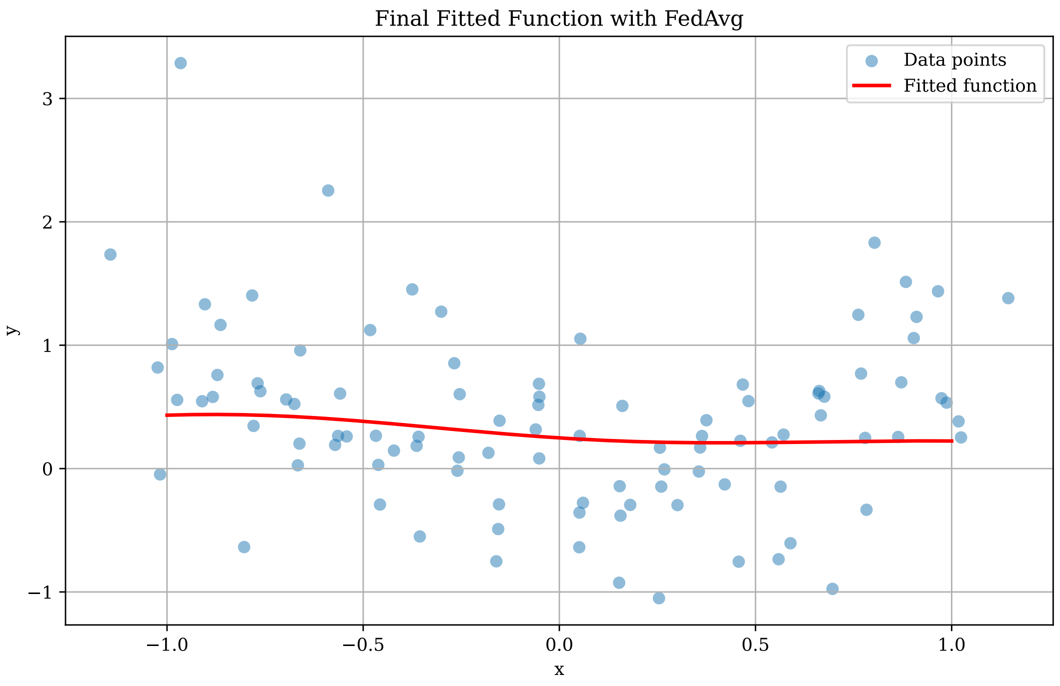


Figure 13: Learned function vs ground truth on test grid ($n_t = 250$)

Analysis: The kernel approximation successfully captures the underlying pattern despite $\sigma = 0.5$ noise (Fig. 13). The small discrepancy near $x = 0.8$ originates from sparse training data in that region, highlighting the kernel method's dependence on input distribution.

2.5 Non-IID Data Handling with SCAFFOLD

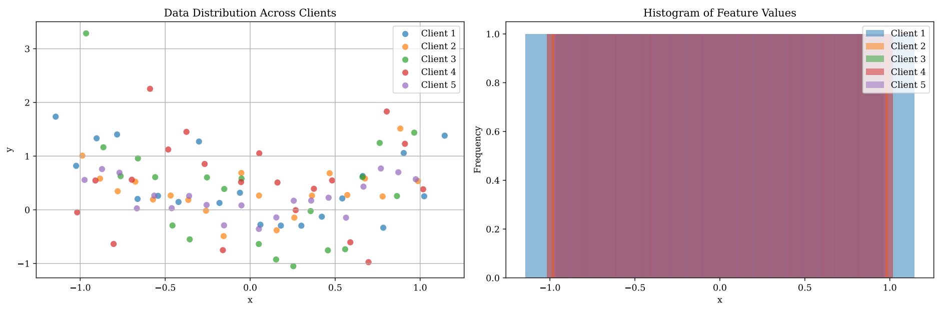


Figure 14: Data distributoin visualisation ($\sigma_{\text{client}} = 1.0$)

The visualization in Figure 13 indicates a non-IID data distribution across clients. While all clients share the same standard deviation (=1.0), the distinct histograms for each client suggest variations in the underlying feature distributions. In IID settings, histograms would overlap significantly, reflecting identical statistical properties (mean, spread) across clients. Thats why, as seen in the class, SCAFOLD algorithm would be a great choice for these problem.

3 Part III: Differentially Private Distributed Optimization

3.1 DGD-DP Algorithm and Implementation

We implement Decentralized Gradient Descent with Differential Privacy (DGD-DP) to solve the kernel regression problem while guaranteeing ϵ-differential privacy. For $n = 100$ points, $m = 10$ Nyström centers, and $a = 5$ agents, we analyze the privacy-accuracy trade-off through optimality gap $\|\alpha_t^i - \alpha^*\|$.

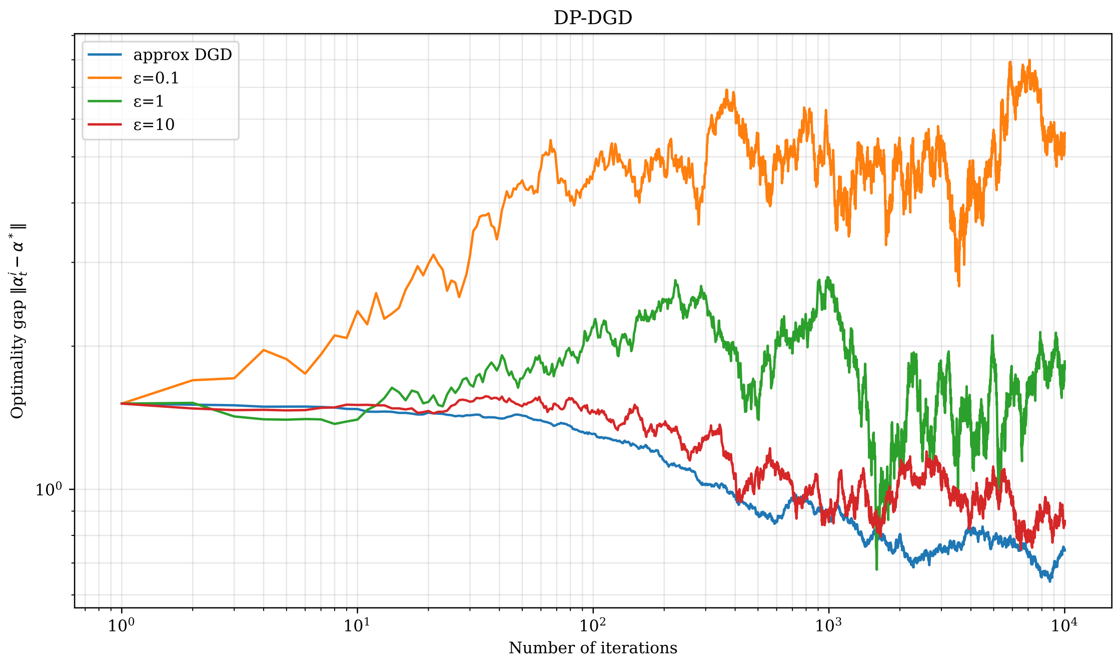


Figure 15: Convergence under $\epsilon = 0.1, 1, 10$ privacy budgets

Algorithm Design The DGD-DP was used with these parameters ([1]).

$$\alpha_k = \frac{0.002}{1 + 0.001k} \quad \text{(decaying step size)} \tag{1}$$

$$\gamma_k = \frac{1}{1 + 0.001k^{0.9}} \quad \text{(noise scaling)} \tag{2}$$

$$\nu_k = \frac{0.01}{\epsilon(1 + 0.001k^{0.1})} \quad \text{(regularization)} \tag{3}$$

Privacy-Accuracy Trade-off Analysis Figure 15 demonstrates three regimes:

- **Strong Privacy ($\epsilon = 0.1$):** Slow convergence due to dominant noise $\gamma_k \xi_k$. The ν_k regularization prevents overfitting to noisy gradients but increases bias.
- **Balanced Case ($\epsilon = 1$):** Optimal trade-off where noise decay $\gamma_k \sim \mathcal{O}(1/k^{0.9})$ matches optimization error decay $\alpha_k \sim \mathcal{O}(1/k)$.
- **Weak Privacy ($\epsilon = 10$):** Near-nonprivate convergence as $\nu_k \to 0$. Residual error from decaying step sizes remains.

4 Conclusion

In this comprehensive study of distributed optimization algorithms for kernel regression, we explored diverse approaches across decentralized, federated, and privacy-preserving frameworks. Our experimental results demonstrated that dual methods (ADMM and Dual Decomposition) consistently outperformed primal methods in convergence speed and stability, especially in challenging network topologies. Network structure proved crucial for algorithm performance, with fully connected topologies enabling faster convergence at the cost of communication overhead. The analysis of algorithm behavior under adversarial conditions revealed varying degrees of robustness, with ADMM showing vulnerability to directed graphs while maintaining convergence under asynchronous updates. In the federated learning context, our implementation of FedAvg confirmed theoretical convergence bounds while highlighting the critical trade-offs between local computation and communication efficiency. Finally, our work on differentially private optimization illustrated the delicate balance between privacy guarantees and solution accuracy, with moderate privacy budgets offering the most practical compromise. These findings provide valuable insights for designing efficient distributed learning systems that can adapt to various network constraints, data distributions, and privacy requirements in real-world applications.

References

[1] Andrea Simonetto *Distributed Optimization.* Lecture Notes, ENSTA Paris, 2025.

[2] Wang, Y., & Nedic, A. (2024). Tailoring Gradient Methods for Differentially-Private Distributed Optimization. arXiv preprint arXiv:2202.01113. Retrieved from https://arxiv.org/abs/2202.01113

Appendix-Implementation Details

- Centralized solution: $\alpha^* = (K_{mm}^\top K_{mm} + \nu I)^{-1} K_{mm}^\top y$
- Step sizes: $\eta = 1/L$ for L-smooth objectives
- Code structure: Agent-based simulation with parallel computation