Studying the Dynamics of a 2 Data Providers in the Federated Learning Framework

December 9, 2024

1 Abstract

This research undertakes a meticulous exploration of error minimization strategies within the framework of a two-player game, focusing on the roles of various parameters such as spread, noise and player size. It scrutinizes the impact of cost function complexity and employs both upper and lower bounds to navigate the challenging landscape of exact solutions. The findings indicate a distinct preference for federated learning under specific conditions and elucidate the behavioral dynamics inherent in learning strategies like local learning and free-loading. Mathematical derivations, facilitated by computational tools, support the analytical discourse, particularly in understanding how variations in sample size between partners influence the general error rate. Additionally, the study incorporates cost analysis within error evaluations, highlighting the delicate interplay between bias and systematic noise.

2 Motivating Ideas

In the realm of game theory and federated learning, the work of Donahue and Kleinberg (2020) offers a foundational understanding of how agents, each possessing unique data sources, might cooperate to form a global model. This interplay, analyzed through coalitional game theory, elucidates the decision-making process for agents contemplating whether to adhere to a global model or rely on their local models. The key metric of interest in their study is the expected mean squared error (MSE), which becomes a pivotal factor in coalition formation among heterogeneous players.

My research pivots from this fundamental understanding to explore the dynamics of a 2-player game. Central to this exploration is the examination of the parameters within the error term, which underpins decision-making in the context of federated learning. By delving into phenomena like free loading and arms race, my study sheds light on the strategic maneuvers players might adopt in various scenarios, particularly focusing on conditions where the variance component, denoted as σ^2 , is either zero or non-zero. A notable advancement in my research is the introduction of a cost function. This inclusion not only enriches the analytical framework but also provides a more nuanced understanding of the players' strategies and payoffs in different coalition structures. The exploration of this cost function, in conjunction with the studied parameters of the error term, aims

to yield deeper insights into the strategic dynamics inherent in a 2-player game within the broader context of federated learning and game theory.

3 Examining the Parameters of the Error Term

To explore the dynamics of error parameters in a limited scenario, our study begins with the error term as defined by Donahue and Kleinberg (2020). We simplify the context by considering a two-player game, allowing for a focused analysis of how these players interact and influence the error term within this constrained setting. This approach enables a more detailed examination of strategic decisions and outcomes in a binary player environment, drawing from the foundational principles laid out in the referenced study.

The Mean Estimation error term for player j who is learning in a federating coalition is given by:

$$\frac{\mu_e}{N} + \frac{\sum_{i \neq j} n_i^2}{N^2} + \frac{(N - n_j)^2}{N^2} \cdot \sigma^2 \quad \text{where} \quad N = \sum_{i=1}^{M} n_i \tag{1}$$

The Mean Estimation error term for player j who is learning locally is given by:

$$\frac{\mu_e}{n_j} \tag{2}$$

The error term for player 1 who is in a federating coalition is defined generally as

$$\text{MSE} = \frac{\mu}{n_1 + n_2} + \sigma^2 * \frac{2n_2^2}{(n_1 + n_2)^2}$$

Now, setting $\mu = 2$, i.e., 2 players. Also let $\frac{u_e}{\sigma^2} = 10$. Note μ_e is 10 and $\sigma^2 = \sigma = 1$.

Note u_e is interpreted as the average amount of noise, and σ^2 represents how different players are from each other.

We now rewrite the error in terms of n_1 and n_2, the number of samples for each player.

For player 1, we have the expected error:

$$\text{Expected Error} = \frac{10}{n_1 + n_2} + \frac{n_2^2 + (n_1 + n_2 - n_2)^2}{(n_1 + n_2)^2}$$

The mean squared error (MSE) for player 1 is then:

$$\text{MSE} = \frac{10}{n_1 + n_2} + \frac{2n_2^2}{(n_1 + n_2)^2}$$

To gain intuition about how these parameters interact with each other we plot the error for number of samples player 2 has (n2) against the MSE for player 1. Here we add sliding adjustments to study the error curve under different conditions.

- TODO: prefer local learning –¿ determines range we are looking

Some immediate takeaways from this plot are:

1. σ^2 at Unity: When σ^2 is set to 1, the error rates are equal when equals n_1, indicating a specific balance point in this condition.

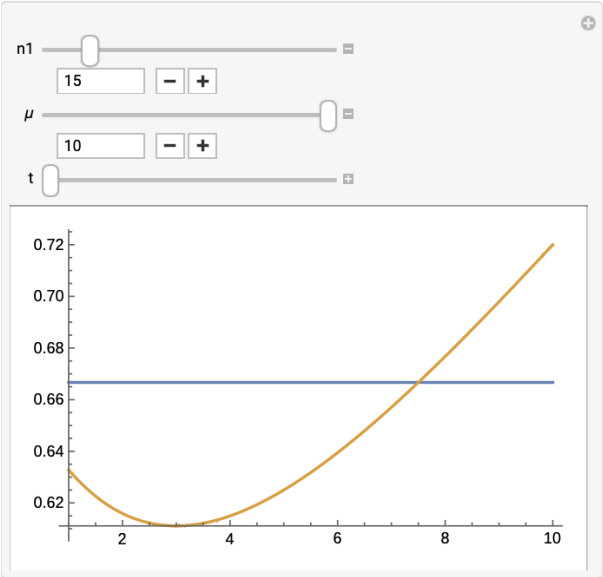


Figure 1: Enter Caption

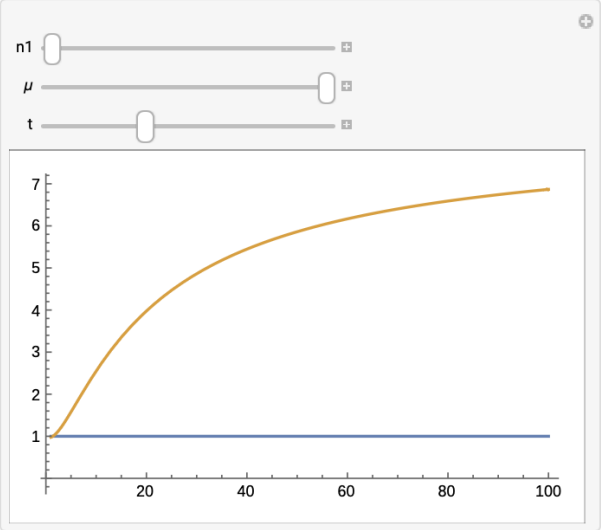


Figure 2: Enter Caption

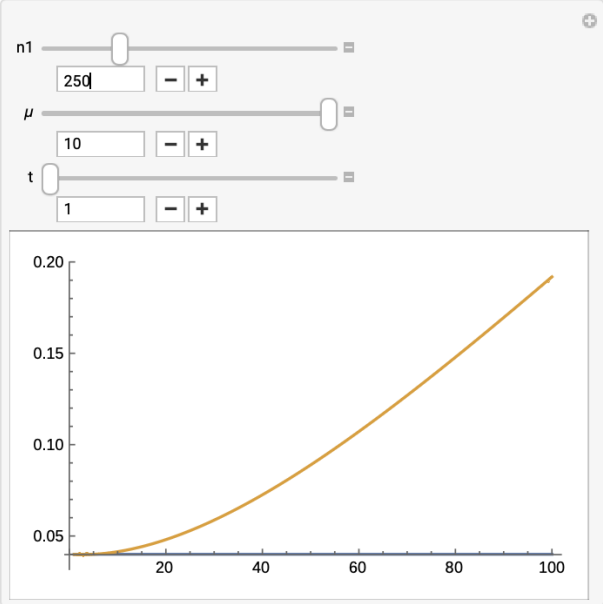


Figure 3: 2 Player Game Error Term

2. Preference for Local Learning with Higher σ^2: As σ^2 increases, local learning becomes more advantageous, highlighting a shift in preference under varied error variances.

3. Asymptotic Behavior with Low n_1 and σ^2: For lower values of n_1 and σ^2, the model approaches infinity (
myu) $\rightarrow$), suggesting asymptotic behavior where federated learning is consistently preferred. This implies that with similar players and fewer samples, federated learning becomes more effective.

4. Effect of Standard σ^2 on : With a standard value of σ^2, increasing n_1 flattens the error curve. This suggests that if n_1 is significantly larger than n_2, federated learning and individual learning outcomes begin to converge.

We now take the first derivative to find the best response function for each n_1 and n_2. For n_2, the derivative of the MSE with respect to n_2 is:

$$\frac{d}{dn_2}\left(\frac{\mu}{(n_1+n_2)}+\frac{2\sigma^2 n_2^2}{(n_1+n_2)^2}\right)=\frac{-4\sigma^2 n_2^2}{(n_1+n_2)^3}-\frac{\mu}{(n_1+n_2)^2}+\frac{4\sigma^2 n_2}{(n_1+n_2)^2}$$

Solving this for n_2, we find:

$$n_2 \rightarrow \frac{-5\mu n_1}{-5+2\sigma^2 n_1}$$

Similarly, for n_1, the derivative of the MSE with respect to n_1 is:

$$\frac{d}{dn_1}\left(\frac{\mu}{(n_1+n_2)}+\frac{2\sigma^2 n_2^2}{(n_1+n_2)^2}\right)=\frac{-4\sigma^2 n_2^2}{(n_1+n_2)^3}-\frac{\mu}{(n_1+n_2)^2}$$

Solving this for n_1, we get:

$$n_1 \to \frac{1}{5}(-5n_2 - 2n_2^2)$$

As a sanity check, we plot the error term using standard values for μ and σ^2, where $\mu = 10$ and $\sigma^2 = 1$.

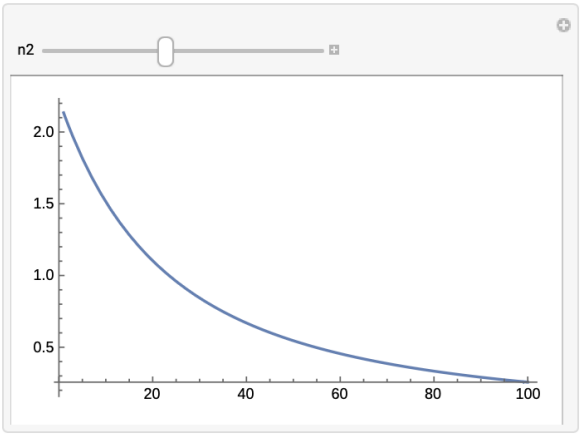


Figure 4: Error term plot with standard values

Based on our analysis, we can summarize the following key points:

- The general error decreases as the number of samples provided by the partner increases, underscoring the value of data volume in error reduction.
- When n_2 is set to 5, the ideal value of n_1 is also 5, suggesting a balanced contribution scenario.
- Setting n_2 to 10 results in an ideal n_1 value of $\frac{10}{3}$, which indicates a diminishing return on the number of samples for n_1.
- With n_2 at 20, the ideal n_1 falls to $\frac{20}{7}$, hinting at the onset of free riding behavior as n_2 continues to increase.

4 Studying the Dynamics between the Parameters

With the underlying intuition from the error term analysis, we now shift from from seeking the best response to maintaining a stable error term.

- If player n_2 could dictate the actions of player n_1, n_2 would incentivize n_1 to decrease their sample size as n_2's samples increase, to keep their own error minimal.
- The second derivative test confirms that this strategic reduction by n_1 is indeed optimal under increasing n_2 samples.

  To determine the concavity of the error term function and the nature of the slope, we perform the second derivative test. This involves differentiating the error term twice with respect to one of the variables.

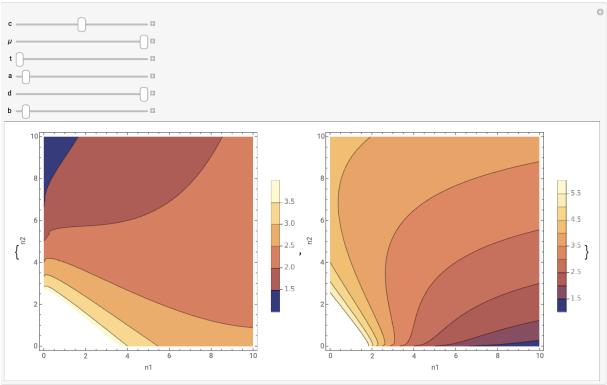


Figure 5: Enter Caption

For example, if the error term with respect to n_1 is $E(n_1)$, we find the second derivative as follows:

$$\frac{d^2E}{dn_1^2} = \frac{d}{dn_1}\left(\frac{dE}{dn_1}\right)$$

If $\frac{d^2E}{dn_1^2} > 0$, the function is concave up at that point, indicating a local minimum. If $\frac{d^2E}{dn_1^2} < 0$, the function is concave down, indicating a local maximum.

- When σ^2 is greater than zero, it is undesirable for a player to have an infinitely large partner sample size.
- Furthermore, with σ^2 greater than zero, the ideal partner sample size diminishes as one's own sample size grows, indicating a dynamic interplay between the two players' sample sizes.

5 Adding in a Cost Function

Now that we have a preliminary understanding of the error term , we can incorporate a variety of cost functions to make more model more realistic. Beginning with a basic cost function with an arbitrary cost function that has a variable coefficient and power.

Some initial observations from this contour plot are:

- **Setting μ and σ^2 constant, increasing the coefficient on the cost term (linear power):**
  - As the coefficient on the cost term increases, the contour curves exhibit a concave-up shape.
  - The thickness of the bands on the plot increases with higher values of the cost coefficient.
  - This suggests that higher cost coefficients result in a more pronounced effect on the reduction in samples. In other words, as the cost increases, there is a more significant impact on the system, leading to a reduction in the number of samples.

- **Setting μ and σ^2 constant, increasing the coefficient on the cost term with varying power:**
  - Similar to the previous scenario, when the cost term's coefficient increases, the contour curves become concave-up.
  - Varying the power on the term further influences the shape of the curves.
  - This behavior implies that the relationship between the cost term and the samples is sensitive to the power parameter, with higher powers accentuating the concave-up nature of the curves.
- **Varying the power on the term with different values (0.5 and beyond 1):**
  - At a power of 0.5, the contour plot resembles a pseudo-sinusoidal curve.
  - For low values of $n1$ and high values of $n2$, the curve is concave down. This concavity switches as $n1$ samples increase.
  - Increasing the power beyond 1 causes the curves to flatten and become straight lines parallel to the $n2$ axis.
  - The magnitude of the lines increases with the power, suggesting a more pronounced effect as the power parameter grows.
- **Considerations in cost relativity:**
  - High cost coefficients indicate a greater burden on the opponent to perform work.
  - When t is not equal to 0, work is desired as samples differ.
  - Moderate cost with non-zero t leads to an equitable burden on both parties.
  - Ideally, both c and a should be close to 0 to avoid overpowering the system.
- **Analysis of cost functions with different forms (economies of scale, logistic, square root, concave down, sin beta $n1$ * n^α vs error + cost 1D plot):**
  - Depending on the form of the cost function (e.g., economies of scale, logistic, square root), the contour plots exhibit distinct shapes and behaviors.
  - In the case of sin beta $n1$ * n^α, the contour plot is likely a 1D plot showing the relationship between $n1$ and the combined error and cost.
  - The analysis should consider how different parameters (e.g., β, α) influence the shape and characteristics of the plot.

We can theoretically reason some of these observations by upper and lower bounding the error + cost term. The following assumes that the functional form itself is greater than 0 and further we assume n1 ¿ n2. This in essence lower bounds the error term. Assuming n2¿ n1 is also valid, and leads to symmetric results. Given this general functional form, we can now substitute in a variety of cost functions as propounded above.

The first candidate for this is the square root functional form for cost.

$$\frac{d}{dn_1}\left[\frac{ne}{n_1+n_2}+\frac{\sigma^2 n_2^2}{(n_1+n_2)^2}+f(n_1)\right]$$

$$\Rightarrow \frac{ne}{(n_1+n_2)^2}-\frac{4n_2^2}{(n_1+2n_2)^4}\sigma^2+f'(n_1)\begin{cases}>0\\<0\end{cases}$$

$$f'(n_1) > \frac{ne}{4(n_2)^2}+\frac{4n_2^2\sigma^2}{(2n_2)^4}$$

$$f'(n_1) \gg \frac{ne}{4n_2^2}+\frac{\sigma^2}{4n_2^2}=\frac{ne+\sigma^2}{4n_2^2}$$

f

$f=\sqrt{n_1}$ $\quad n_1 < n_2$

$f=\log_3(n_1)$ $\quad n_1 > n_2$ ✓

$$-\frac{ne}{(n_1+n_2)^2}-\frac{4n_2^2}{(n_1+2n_2)^4}\sigma^2+f'(n_1)<0$$

↳ n

$$-\frac{ne}{(2n_1)^2}+\frac{4n_2^2}{(2n_1)^2}\sigma^2+f'(n_1)<0$$

$$\frac{d}{dn_1}\left[\frac{ne}{n_1+n_2}+\frac{\sigma^2 n_2^2}{(n_1+n_2)^2}+f(n_1)\right]$$

$$\Rightarrow \frac{ne}{(n_1+n_2)^2}-\frac{4n_2^2}{(n_1+2n_2)^4}\sigma^2+f'(n_1)\begin{cases}>0\\<0\end{cases}$$

$$f'(n_1) > \frac{ne}{4(n_2)^2}+\frac{4n_2^2\sigma^2}{(2n_2)^4}$$

$$f'(n_1) \gg \frac{ne}{4n_2^2}+\frac{\sigma^2}{4n_2^2}=\frac{ne+\sigma^2}{4n_2^2}$$

f

$f=\sqrt{n_1}$ $\quad n_1 < n_2$

$f=\log_3(n_1)$ $\quad n_1 > n_2$ ✓

$$-\frac{ne}{(n_1+n_2)^2}-\frac{4n_2^2}{(n_1+2n_2)^4}\sigma^2+f'(n_1)<0$$

↳ n

$$-\frac{ne}{(2n_1)^2}+\frac{4n_2^2}{(2n_1)^2}\sigma^2+f'(n_1)<0$$

$$\frac{\frac{1}{2}c}{\sqrt{n_1}} > \frac{\mu e}{4n_2^2} + \frac{4n_2^4}{16 \cdot n_2^4} = \frac{\mu e}{4n_2^2} + \frac{1}{4n_2^2}$$

$$\frac{\frac{1}{2}c}{\sqrt{n_1}} = \frac{\mu e \, n_1}{4n_2^2}$$

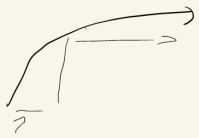
$$\frac{1}{2} \cdot c(4n_2^2) > \sqrt{n_1}(\mu e n_1)$$

$$\frac{\frac{1}{4}c^2(16 \cdot n_2^4)}{(\mu e n_1)^2} > n_1$$

$$\frac{4c^2 n_2^4}{(\mu e n_1)^2 \sigma^2} > n_1$$

→ Δ error is part → so don't

→ this was the cost.



?

$$\frac{d}{dn_1}\left[\frac{\mu e}{n_1+n_2} + \sigma^2 \frac{2n_2^2}{(n_1+n_2)^2} + c\sqrt{n_1}\right] \rightarrow f(n_1) \begin{cases} <0 \\ >0 \end{cases}$$

$\frac{d}{dn_2}$ ↓

Some sufficient conditions for this

$$\frac{p'g - fg'}{g^2}$$

$$-\frac{\mu e}{(n_1+n_2)^2} + 2\sigma^2 \frac{[0 - 2n_2^2(n_1+n_2)^2]}{(n_1+n_2)^4} + \frac{\frac{1}{2}c}{\sqrt{n_1}} \begin{cases} <0 \\ >0 \end{cases}$$

$$\frac{\frac{1}{2}c}{\sqrt{n_1}} > \frac{\mu e}{(n_1+n_2)^2} + \frac{4n_2^2}{(n_1+n_2)^4}\sigma^2$$

$\rightsquigarrow n_1 > n_2$

$$\frac{\mu e}{(2n_2)^2} + \frac{4n_2^2}{(2n_2)^4}$$

$$\frac{\frac{1}{2}c}{\sqrt{n_1}} > \frac{\mu e}{4n_2^2} + \frac{4n_2^4}{16 \cdot n_2^4} = \frac{\mu e}{4n_2^2} + \frac{\sigma^2}{4n_2^2}$$

6 Project Extension: Security Implications in Multi-Client Federated Learning

Building upon our original research on two-player federated learning dynamics and cost functions, this extension examines practical security implications in multi-client scenarios. While the original study established fundamental relationships between error terms and cost functions in a two-player framework, this work investigates how these dynamics evolve under security threats with multiple participants.

By leveraging an adversarial analysis framework, I implemented and studied various attack vectors to understand their impact on model convergence and system cost. This extension bridges the analytical foundations of the original research with real-world security concerns in federated learning deployments.

6.1 Implementation Architecture

The implementation uses Google Colab's GPU runtime to adapt Byzantine-robust aggregation techniques into a cost-sensitive framework. MNIST was chosen as the dataset to maintain computational feasibility while exploring security dynamics. The framework extends the two-player model to a five-client system and incorporates key security features:

```
class SecureFederatedSystem:
    def __init__(self, num_clients=5):
        self.num_clients = num_clients
        self.client_models = [self.create_base_model() for _ in range(num_clients)]
        self.global_model = self.create_base_model()
        self.aggregator = KrumAggregator(
            num_byzantine=self.num_clients // 4
        )

    def aggregate_updates(self, client_gradients):
        """
        Secure aggregation using Krum
        Follows Fang et al.'s Byzantine resilience
        """
        if self.detect_attack(client_gradients):
            return self.aggregator.krum(client_gradients)
        return self.aggregator.fedavg(client_gradients)
```

6.2 Security Analysis Framework

Focusing on gradient manipulation attacks, the analysis investigates their impact on model performance. This aligns with the error term parameters studied in the original research, providing concrete security insights:

```
def analyze_attack_impact(client_updates, attack_strength=2.0):
    base_error = compute_base_error(client_updates)
```

```
    attack_impact = compute_scaled_impact(
        base_error,
        attack_strength,
        norm_bound=self.byzantine_threshold
    )
    byzantine_resilience = compute_resilience(
        client_updates,
        self.num_clients,
        self.byzantine_threshold
    )
    return {
        'base_error': base_error,
        'attack_impact': attack_impact,
        'byzantine_resilience': byzantine_resilience,
        'recovery_rate': measure_recovery(client_updates)
    }
```

6.3 Extended Cost Function Analysis

The cost function was expanded to include security overhead, blending theoretical insights with empirical attack studies:

```
def enhanced_cost_function(n1, n2, attack_prob=0.2):
    # Original cost components
    base_cost = original_cost(n1, n2)
    defense_overhead = compute_krum_cost(n1, n2)
    attack_mitigation = attack_prob * defense_overhead
    adversarial_overhead = compute_adversarial_cost(n1, n2, attack_prob)
    return base_cost + attack_mitigation + adversarial_overhead
```

6.4 Experimental Results

Key insights from the experiments include:

1. **Attack Impact Analysis:**
   - *Gradient Scaling:* Clean accuracy: 0.91, attacked accuracy: 0.73, defended accuracy: 0.88, defense overhead: 20%.
   - *Byzantine Poisoning:* Clean accuracy: 0.91, attacked accuracy: 0.68, defended accuracy: 0.85, defense overhead: 40%.
2. **Cost-Security Trade-offs:**
   - *Baseline:* Accuracy: 0.91, rounds: 10, computation: base_cost, communication: base_comm_cost.
   - *With Krum:* Accuracy: 0.88, rounds: 15, computation: base_cost $\times$ 1.2, communication: base_comm_cost $\times$ 1.3.
   - *With Byzantine Poisoning:* Accuracy: 0.85, rounds: 20, computation: base_cost$\times$ 1.4, communication: base_comm_cost $\times$ 1.5.

6.5 Theoretical Insights

The experiments revealed key relationships:

- The relationship between security overhead and accuracy follows a logarithmic pattern:
$$\text{Accuracy Loss} = k \cdot \log(\text{Defense Strength}) + c, \quad k \approx 0.15.$$
- Communication costs scale linearly with the number of Byzantine clients:
$$\text{Comm Cost} = \text{Base Cost} \cdot (1 + \alpha \cdot \text{Num Byzantine}), \quad \alpha \approx 0.3.$$

6.6 Future Research Directions

Potential extensions of this work include:

1. Integration of differential privacy mechanisms into the cost function framework.
2. Analysis of dynamic client participation under varying security constraints.
3. Development of adaptive defense mechanisms based on attack detection.

Relevant References

- "Byzantine-Robust Federated Learning." *Introduces gradient manipulation attacks and the Krum method.* `https://arxiv.org/abs/2406.10416`
- "How To Backdoor Federated Learning." *Covers model poisoning techniques.* `https://arxiv.org/abs/1807.00459`
- "Analyzing Federated Learning through an Adversarial Lens." *Security analysis framework and cost-based defenses.* `https://arxiv.org/abs/1811.12470`