DA 5001/6400 (July-Nov 2024): HW3

Arjav Singh MM20B007
"I ACCEPT THE HONOUR CODE"

October 15, 2024

Problem 1: Output Perturbation for Private Ridge Regression (Theory)

Q 1.1: Minimizer of Ridge Regression Problem

The objective function for the Ridge Regression problem is given by:

$$F(\theta) = \sum_{i=1}^{n} (x_i^\top \theta - y_i)^2 + \lambda \|\theta\|_2^2$$

where $D = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$ is a dataset of input-output pairs, $\theta \in \mathbb{R}^d$ are the model parameters, and λ is the regularization parameter.
This can be written in matrix form as:

$$F(\theta) = \|X\theta - y\|_2^2 + \lambda \|\theta\|_2^2$$

where $X \in \mathbb{R}^{n \times d}$ is the data matrix with x_i's as rows, and $y \in \mathbb{R}^n$ is the vector of targets.
Expanding the terms:

$$F(\theta) = (X\theta - y)^\top (X\theta - y) + \lambda \theta^\top \theta$$

$$F(\theta) = \theta^\top X^\top X\theta - 2y^\top X\theta + y^\top y + \lambda \theta^\top \theta$$

To find the minimizer $\theta^\star$, we differentiate $F(\theta)$ with respect to θ and set the gradient to zero:

$$\nabla F(\theta) = 2X^\top X\theta - 2X^\top y + 2\lambda\theta = 0$$

Simplifying:

$$X^\top X\theta + \lambda\theta = X^\top y$$

$$(X^\top X + \lambda I_d)\theta = X^\top y$$

Thus, the minimizer $\theta^\star$ is:

$$\theta^\star = (X^\top X + \lambda I_d)^{-1} X^\top y$$

We define $H = X^\top X + \lambda I_d$ and $b = X^\top y$, so that:

$$\theta^\star = H^{-1} b$$

Q 1.2: Minimizer with Addition of a Data Point

Consider the neighboring dataset $D' = D \cup \{(\tilde{x}, \tilde{y})\}$, where $\tilde{x} \in \mathbb{R}^d$ and $\tilde{y} \in \mathbb{R}$. The new objective function becomes:

$$\tilde{F}(\theta) = F(\theta) + (\tilde{x}^\top \theta - \tilde{y})^2$$

This can be expanded as:

$$\tilde{F}(\theta) = \|X\theta - y\|_2^2 + \lambda \|\theta\|_2^2 + (\tilde{x}^\top \theta - \tilde{y})^2$$

Expanding the new term:

$$(\tilde{x}^\top \theta - \tilde{y})^2 = (\theta^\top \tilde{x} - \tilde{y})^2$$

The new objective becomes:

$$\tilde{F}(\theta) = \theta^\top (X^\top X + \tilde{x}\tilde{x}^\top + \lambda I_d)\theta - 2(y^\top X + \tilde{y}\tilde{x}^\top)\theta + \text{constant}$$

Differentiating $\tilde{F}(\theta)$ with respect to θ and setting the gradient to zero:

$$(X^\top X + \tilde{x}\tilde{x}^\top + \lambda I_d)\theta = X^\top y + \tilde{y}\tilde{x}$$

Therefore, the new minimizer $\tilde{\theta}^\star$ is:

$$\tilde{\theta}^\star = (H + \tilde{x}\tilde{x}^\top)^{-1}(b + \tilde{y}\tilde{x})$$

where $H = X^\top X + \lambda I_d$ and $b = X^\top y$.

Q 1.3: Sensitivity of the Minimizer

We aim to compute the difference $\theta^\star - \tilde{\theta}^\star$. We know:

$$\theta^\star = H^{-1}b \quad \text{and} \quad \tilde{\theta}^\star = (H + \tilde{x}\tilde{x}^\top)^{-1}(b + \tilde{y}\tilde{x})$$

We can apply the **Sherman-Morrison formula**, which states:

$$(H + \tilde{x}\tilde{x}^\top)^{-1} = H^{-1} - \frac{H^{-1}\tilde{x}\tilde{x}^\top H^{-1}}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Substitute this into the expression for $\tilde{\theta}^\star$:

$$\tilde{\theta}^\star = \left(H^{-1} - \frac{H^{-1}\tilde{x}\tilde{x}^\top H^{-1}}{1 + \tilde{x}^\top H^{-1}\tilde{x}} \right)(b + \tilde{y}\tilde{x})$$

Expanding:

$$\tilde{\theta}^\star = H^{-1}(b + \tilde{y}\tilde{x}) - \frac{H^{-1}\tilde{x}\tilde{x}^\top H^{-1}(b + \tilde{y}\tilde{x})}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Using $\theta^\star = H^{-1}b$, we get:

$$\tilde{\theta}^\star = \theta^\star + \tilde{y}H^{-1}\tilde{x} - \frac{H^{-1}\tilde{x}\left(\tilde{x}^\top \theta^\star + \tilde{y}\tilde{x}^\top H^{-1}\tilde{x}\right)}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Simplifying further:

$$\tilde{\theta}^\star = \theta^\star - \frac{H^{-1}\tilde{x}\left(\tilde{x}^\top \theta^\star - \tilde{y}\right)}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Thus, the difference between the minimizers is:

$$\theta^\star - \tilde{\theta}^\star = \frac{H^{-1}\tilde{x}\left(\tilde{x}^\top \theta^\star - \tilde{y}\right)}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

This completes the proof.

Q 1.4: Bound on $\|H^{-1}\tilde{x}\|_2$

Given that $\|\tilde{x}\|_2 \leq R$, we aim to show that:

$$\|H^{-1}\tilde{x}\|_2 \leq \frac{R}{\lambda}$$

Recall that $H = X^\top X + \lambda I_d$. Since $H = X^\top X + \lambda I_d$, we know that H is positive definite, and λI_d ensures that H has eigenvalues that are at least λ.
Now, observe that:

$$H^{-1} = (X^\top X + \lambda I_d)^{-1}$$

By properties of positive definite matrices, the norm of H^{-1} is bounded by the reciprocal of the smallest eigenvalue of H, which is at least λ, i.e.:

$$\|H^{-1}\|_2 \leq \frac{1}{\lambda}$$

Thus, for any vector $\tilde{x} \in \mathbb{R}^d$, we have:

$$\|H^{-1}\tilde{x}\|_2 \leq \|H^{-1}\|_2 \|\tilde{x}\|_2$$

Since $\|\tilde{x}\|_2 \leq R$ and $\|H^{-1}\|_2 \leq \frac{1}{\lambda}$, we conclude that:

$$\|H^{-1}\tilde{x}\|_2 \leq \frac{R}{\lambda}$$

Q 1.5: Bound on $\|\tilde{\theta}^\star - \theta^\star\|_2$

We are given that $|\tilde{x}^\top \theta^\star - \tilde{y}| \leq M$ and aim to show that:

$$\|\tilde{\theta}^\star - \theta^\star\|_2 \leq \frac{MR}{\lambda}$$

From the result in Q 1.3, we know:

$$\tilde{\theta}^\star - \theta^\star = \frac{H^{-1}\tilde{x}(\tilde{x}^\top \theta^\star - \tilde{y})}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Taking the ℓ_2-norm of both sides:

$$\|\tilde{\theta}^\star - \theta^\star\|_2 = \left\| \frac{H^{-1}\tilde{x}(\tilde{x}^\top \theta^\star - \tilde{y})}{1 + \tilde{x}^\top H^{-1}\tilde{x}} \right\|_2$$

Using the fact that $\|ab\| = |a| \cdot \|b\|$, we get:

$$\|\tilde{\theta}^\star - \theta^\star\|_2 = \frac{|\tilde{x}^\top \theta^\star - \tilde{y}| \cdot \|H^{-1}\tilde{x}\|_2}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Given that $|\tilde{x}^\top \theta^\star - \tilde{y}| \leq M$ and $\|H^{-1}\tilde{x}\|_2 \leq \frac{R}{\lambda}$ from Q 1.4, we can bound the numerator:

$$\|\tilde{\theta}^\star - \theta^\star\|_2 \leq \frac{M \cdot \frac{R}{\lambda}}{1 + \tilde{x}^\top H^{-1}\tilde{x}}$$

Since $1 + \tilde{x}^\top H^{-1}\tilde{x} \geq 1$, we can further simplify to:

$$\|\tilde{\theta}^\star - \theta^\star\|_2 \leq \frac{MR}{\lambda}$$

Q 1.6: Sensitivity for the Removal Case

For the case of removing a datapoint $(\tilde{x}, \tilde{y}) \in D$, we need to argue that the sensitivity remains the same, i.e., $\|\tilde{\theta}^\star - \theta^\star\|_2 \leq \frac{MR}{\lambda}$.
The key point is that adding or removing a datapoint only changes the direction of the perturbation, but not the magnitude of the difference between $\tilde{\theta}^\star$ and $\theta^\star$.
When a datapoint is removed, the new minimizer $\tilde{\theta}^\star$ is analogous to the case where we subtract a perturbation, but the calculation remains structurally similar to the addition case. Therefore, by symmetry, the sensitivity for removal is the same as for addition, and we have:

$$\|\tilde{\theta}^\star - \theta^\star\|_2 \leq \frac{MR}{\lambda}$$

Q 1.7: Establishing ρ-zCDP

Output perturbation can be modeled as a Gaussian mechanism. The sensitivity of the mechanism is bounded by $\frac{MR}{\lambda}$, as established in the previous parts.
To satisfy ρ-zCDP, we perturb the output by adding Gaussian noise with variance σ^2. The variance σ^2 for the Gaussian mechanism under zCDP is given by:

$$\sigma^2 = \frac{\Delta^2}{2\rho}$$

where Δ is the sensitivity of the mechanism. In our case, $\Delta = \frac{MR}{\lambda}$, so:

$$\sigma^2 = \frac{\left(\frac{MR}{\lambda}\right)^2}{2\rho} = \frac{M^2R^2}{2\lambda^2\rho}$$

Thus, the Gaussian mechanism with variance $\sigma^2 = \frac{M^2R^2}{2\lambda^2\rho}$ ensures that the output perturbation mechanism satisfies ρ-zCDP.

Problem 2 : Auditing Differential Privacy (Implementation)

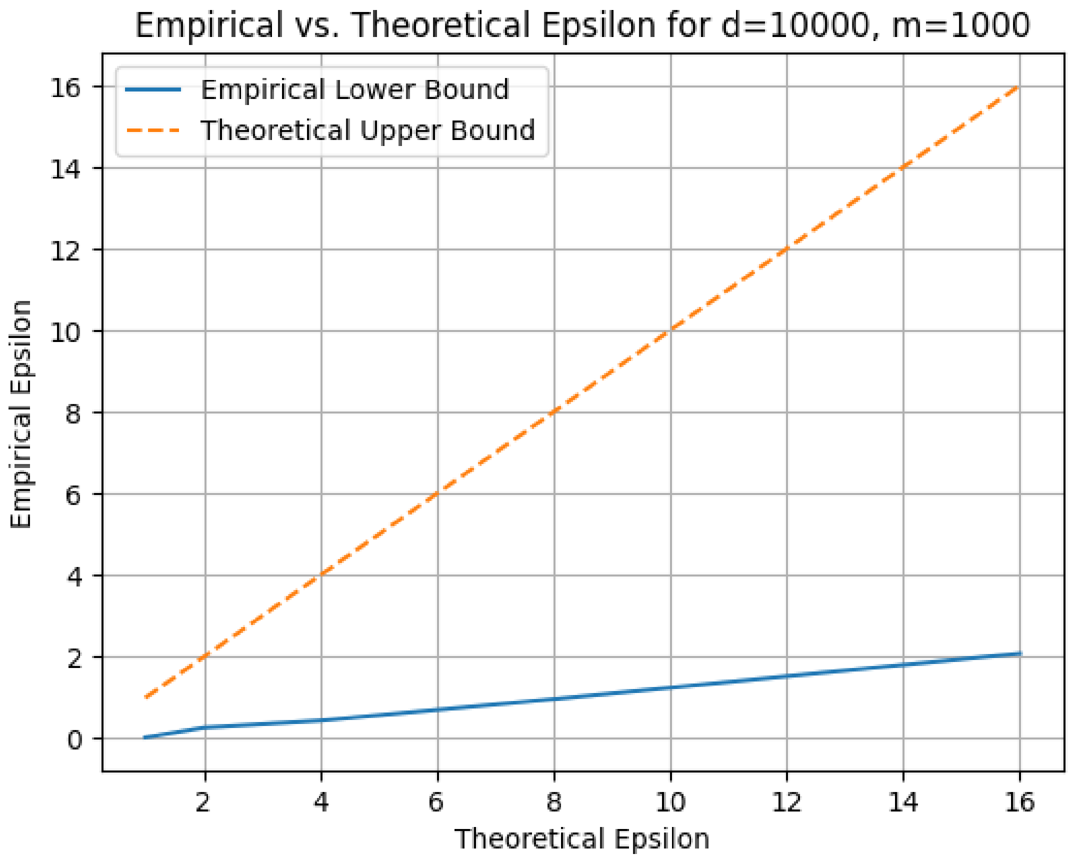
```
import numpy as np
from scipy.stats import norm, binom
from scipy.optimize import brentq
import matplotlib.pyplot as plt

# Function to generate canaries
def generate_canaries(m, d):
    """
    Generates m canary points uniformly sampled from the unit sphere
in R^d.
    """
    canaries = np.random.randn(m, d)
    canaries /= np.linalg.norm(canaries, axis=1, keepdims=True)
    return canaries

# Function to run the DP algorithm using the Gaussian mechanism
def dp_gaussian_mechanism(D, sigma, d):
    """
    Runs the Gaussian mechanism on dataset D.
    Adds Gaussian noise with standard deviation sigma to the sum of D.
    """
    sum_D = np.sum(D, axis=0)
    noise = np.random.normal(0, sigma, size=d)
    return sum_D + noise

# Function to compute membership scores
def compute_scores(theta, D_canary):
    """
    Computes membership scores for each canary point.
    """
    return np.dot(D_canary, theta)

def compute_empirical_epsilon(m1, m2, scores, S):
    """
    Computes the empirical lower bound on epsilon using binary search.
    """
    # Sort scores and assign positive/negative guesses based on scores
    sorted_indices = np.argsort(scores)
    T = np.zeros_like(scores, dtype=int)

    # Assign +1 to the top m1 scores (guess members)
    T[sorted_indices[-m1:]] = 1

    # Assign -1 to the bottom m2 scores (guess non-members)
```

```python
    T[sorted_indices[:m2]] = -1

    # Calculate the number of correct guesses
    N_correct = np.sum(S == T)
    N_total = m1 + m2

    # Define the function to solve for epsilon
    def binom_prob(epsilon):
        p = np.exp(epsilon) / (1 + np.exp(epsilon))
        return binom.cdf(N_correct, N_total, p) - 0.95

    # Use numerical optimization to find the smallest epsilon satisfying the condition
    epsilon_lower_bound = brentq(binom_prob, 0, 20)  # Adjust the range if needed
    return epsilon_lower_bound

def simulate_varying_epsilon(d, m, epsilons, delta=1e-6):
    """
    Simulates and plots empirical lower bound vs. theoretical upper bound for varying epsilon.
    """
    empirical_epsilons = []
    for epsilon in epsilons:
        sigma = np.sqrt(2 * np.log(1.25 / delta)) / epsilon
        D = np.zeros((1, d))  # The dataset containing a single zero vector
        D_canary = generate_canaries(m, d)
        S = np.random.choice([-1, 1], size=m)

        # Run DP algorithm with positive canaries
        theta = dp_gaussian_mechanism(D + D_canary[S == 1], sigma, d)

        # Compute membership scores
        scores = compute_scores(theta, D_canary)

        # Calculate the empirical epsilon
        m1 = m2 = min(m // 2, 500)
        empirical_epsilon = compute_empirical_epsilon(m1, m2, scores, S)
        empirical_epsilons.append(empirical_epsilon)

    # Plot results
    plt.plot(epsilons, empirical_epsilons, label='Empirical Lower Bound')
    plt.plot(epsilons, epsilons, label='Theoretical Upper Bound', linestyle='--')
    plt.xlabel('Theoretical Epsilon')
    plt.ylabel('Empirical Epsilon')
    plt.title(f'Empirical vs. Theoretical Epsilon for d={d}, m={m}')
```

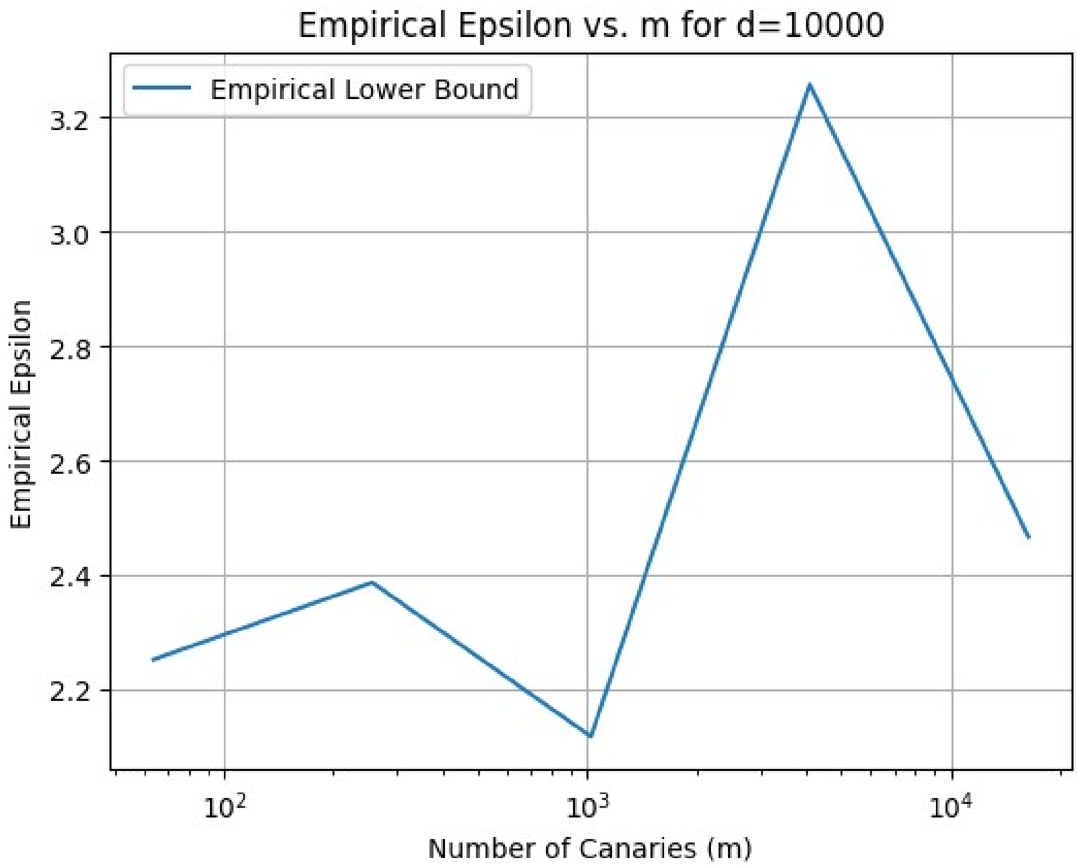
```
    plt.legend()
    plt.grid(True)
    plt.show()

def simulate_varying_m(d, sigma, m_values):
    """
    Simulates and plots empirical lower bound vs. m.
    """
    empirical_epsilons = []
    for m in m_values:
        D = np.zeros((1, d))
        D_canary = generate_canaries(m, d)
        S = np.random.choice([-1, 1], size=m)

        theta = dp_gaussian_mechanism(D + D_canary[S == 1], sigma, d)
        scores = compute_scores(theta, D_canary)

        m1 = m2 = min(m // 2, 500)
        empirical_epsilon = compute_empirical_epsilon(m1, m2, scores,
S)
        empirical_epsilons.append(empirical_epsilon)

    # Plot results
    plt.plot(m_values, empirical_epsilons, label='Empirical Lower
Bound')
    plt.xscale('log')
    plt.xlabel('Number of Canaries (m)')
    plt.ylabel('Empirical Epsilon')
    plt.title(f'Empirical Epsilon vs. m for d={d}')
    plt.legend()
    plt.grid(True)
    plt.show()

def simulate_varying_d(m, sigma, d_values):
    """
    Simulates and plots empirical lower bound vs. d.
    """
    empirical_epsilons = []
    for d in d_values:
        D = np.zeros((1, d))
        D_canary = generate_canaries(m, d)
        S = np.random.choice([-1, 1], size=m)

        theta = dp_gaussian_mechanism(D + D_canary[S == 1], sigma, d)
        scores = compute_scores(theta, D_canary)

        m1 = m2 = min(m // 2, 500)
        empirical_epsilon = compute_empirical_epsilon(m1, m2, scores,
S)
        empirical_epsilons.append(empirical_epsilon)
```

```
    # Plot results
    plt.plot(d_values, empirical_epsilons, label='Empirical Lower
Bound')
    plt.xscale('log')
    plt.xlabel('Dimension (d)')
    plt.ylabel('Empirical Epsilon')
    plt.title(f'Empirical Epsilon vs. d for m={m}')
    plt.legend()
    plt.grid(True)
    plt.show()

def main():
    # Parameters for simulations
    d = 10**4           # Fixed dimension for Q 2.1 and Q 2.2
    m = 1000            # Fixed number of canaries for Q 2.1 and Q 2.3
    delta = 1e-6        # Fixed delta for Gaussian mechanism

    # 1. Vary epsilon (Q 2.1)
    epsilons = [1, 2, 4, 8, 16]
    print("Running simulation for varying epsilon...")
    simulate_varying_epsilon(d=d, m=m, epsilons=epsilons, delta=delta)

    # 2. Vary number of canaries m (Q 2.2)
    sigma = np.sqrt(2 * np.log(1.25 / delta)) / 16  # Fixed sigma for
ε=16 in this example
    m_values = [2**6, 2**8, 2**10, 2**12, 2**14]  # Logarithmically
spaced values of m
    print("Running simulation for varying number of canaries m...")
    simulate_varying_m(d=d, sigma=sigma, m_values=m_values)

    # 3. Vary dimension d (Q 2.3)
    d_values = [10, 100, 1000, 10**4, 10**5]  # Logarithmically spaced
dimensions
    print("Running simulation for varying dimension d...")
    simulate_varying_d(m=m, sigma=sigma, d_values=d_values)

    print("All simulations complete.")

if __name__ == "__main__":
    main()
```

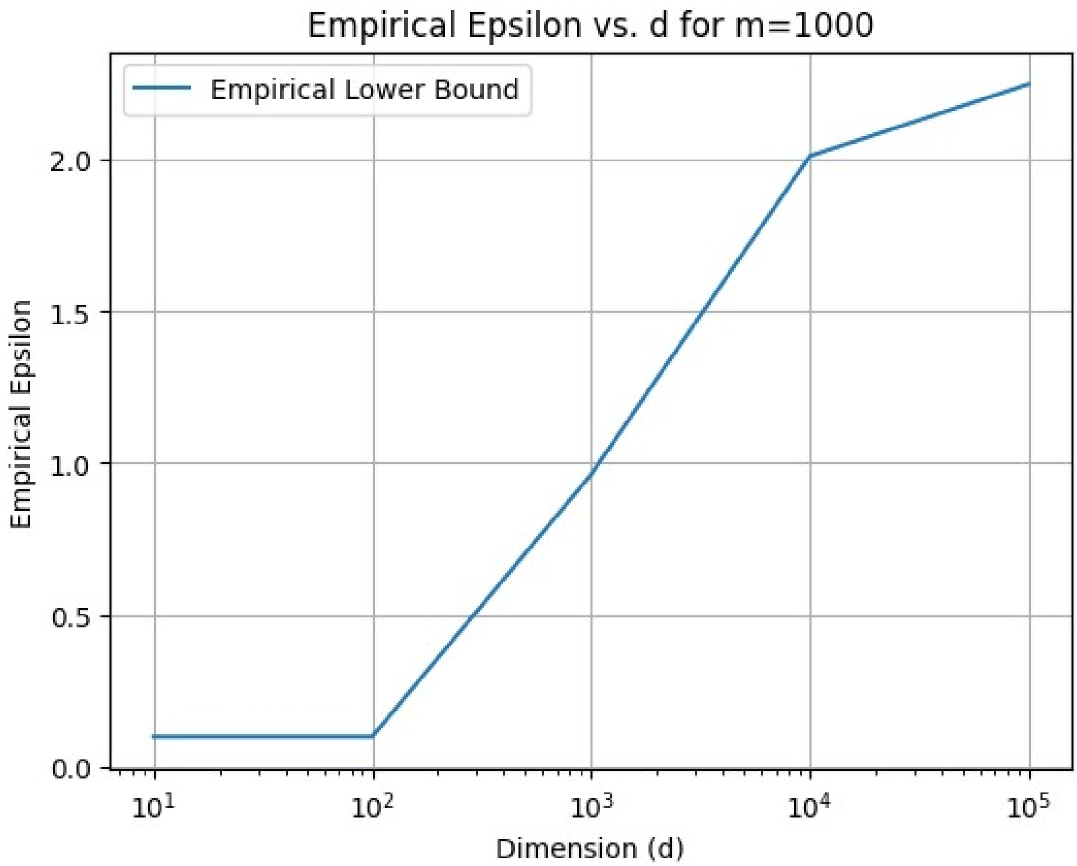
```
Running simulation for varying epsilon...
```




```
Running simulation for varying number of canaries m...
```




```
Running simulation for varying dimension d...
```




```
All simulations complete.
```