Catalogue

Dengke Chen and qvl920

1 Analyzing $\mathcal{M}_{means}$

(a)

Answer:

To show that $\mathcal{M}_{means}$ satisfies ρ-zCDP given that $\mathcal{M}'_{means}$ does, need to establish that $\mathcal{M}_{means}$ is a post-processing of $\mathcal{M}'_{means}$. According to Lemma 4 (Composition), privacy guarantees are preserved under post-processing; that is, any function applied to the output of a ρ-zCDP mechanism remains ρ-zCDP.

$\mathcal{M}'_{means}$: Outputs noisy versions of intermediate computations:

$\{f_\ell(X) + z_\ell, g_\ell(X) + z'_\ell\}_{\ell=1}^{t}$,

Here, $f_\ell(X)$ and $g_\ell(X)$ are functions of the input data X, and z_ℓ and z'_ℓ are Gaussian noise vectors.

$\mathcal{M}_{means}$**:** Outputs the final cluster centers:

$\left\{c_i^{(t)}\right\}_{i=1}^{k}$,

The $c_i^{(t)}$ are computed using the noisy sums and counts from $\mathcal{M}'_{means}$.

The cluster centers $\left\{c_i^{(t)}\right\}$ in $\mathcal{M}_{means}$ are computed using the noisy outputs from $\mathcal{M}'_{means}$ through deterministic calculations.

Specifically, the update rule for cluster centers in step 2.(a) of $\mathcal{M}_{means}$ is

$$c_i^{(\ell)} = \frac{1}{\max\left(1, n_i^{(\ell-1)}\right)}\left(z_{\ell,i} + \sum\nolimits_{j\in S_i^{(\ell-1)}} x_j\right) = \frac{1}{\max\left(1, n_i^{(\ell-1)}\right)}\left(f_\ell(X)_i + z_{\ell,i}\right)$$

This is a function of $f_\ell(X) + z_\ell$ and $n_i^{(\ell-1)}$ (which depends on $g_{\ell-1}(X) + z'_{\ell-1}$).

Since $\mathcal{M}_{means}$ computes its output by applying deterministic functions to the outputs of $\mathcal{M}'_{means}$, it is a post-processing of $\mathcal{M}'_{means}$.
Lemma 4 (Composition) tells us that post-processing a ρ-zCDP mechanism does not increase the privacy loss.
Therefore, if $\mathcal{M}'_{means}$ satisfies ρ-zCDP, then $\mathcal{M}_{means}$ also satisfies ρ-zCDP.
Since $\mathcal{M}_{means}$ is a deterministic post-processing of $\mathcal{M}'_{means}$, and privacy guarantees are preserved under post-processing, $\mathcal{M}_{means}$ satisfies ρ-zCDP if $\mathcal{M}'_{means}$ does.

(b)

Answer:
X and X' differ in exactly one data point, say at index s.

Since $S_1^{(\ell-1)}$ are fixed and independent of X, the sets remain the same for both X and X'.

For each cluster $i \in [k]$:

$$f_\ell(X)_i = \sum\nolimits_{j \in S_i^{(\ell-1)}} x_j$$

$$f_\ell(X')_i = \sum\nolimits_{j \in S_i^{(\ell-1)}} x'_j$$

Since $x_j = x'_j$ for all $j \neq s$, the only potential difference comes from $\boldsymbol{j} = \boldsymbol{s}$ if $s \in S_1^{(\ell-1)}$.

The difference vector $f_\ell(X) - f_\ell(X')$ has non-zero components only in the cluster iii such that $s \in S_1^{(\ell-1)}$:

$$[f_\ell(X) - f_\ell(X')]_i = \begin{cases} x_s - x'_s, & if\ s \in S_1^{(\ell-1)} \\ 0, & otherwise \end{cases}$$

Since $\|x_s\|_2$, $\|x_s'\|_2$ we have:

$$\|x_s - x_s'\|_2 \leq \|x_s\|_2 + \|x_s'\|_2 \leq 2$$

Therefore, the squared ℓ_2-norm of the difference is:

$$\|f_\ell(X) - f_\ell(X')\|_2^2 = \sum\nolimits_{i=1}^{k} \|[f_\ell(X) - f_\ell(X')]_i\|_2^2 \leq 4$$

Final, taking the square root, get: $\|f_\ell(X) - f_\ell(X')\|_2 \leq 2$

By fixing $S_1^{(\ell-1)}$, the difference $f_\ell(X) - f_\ell(X')$ is non-zero only in one cluster iii where s $s \in S_1^{(\ell-1)}$, and this difference has norm at most 2. Therefore, $\|f_\ell(X) - f_\ell(X')\|_2 \leq 2$ for neighboring $X \sim X'$.

(c)

Answer:

Privacy Loss per Iteration:

In each iteration ℓ, $\mathcal{M}'_{means}$ releases:

Noisy Sum: $f_\ell(X) + z_\ell$ where $z_\ell \sim \mathcal{N}(0, \sigma^2)^{dk}$

Noisy Counts: $g_\ell(X) + z'_\ell$ where $z'_\ell \sim \mathcal{N}(0, {\sigma'}^2)^k$

Sensitivity Computations:

Sensitivity of $f_\ell(X)$:

From Problem (b), $\|f_\ell(X) - f_\ell(X')\|_2 \le 2$, So, $\Delta_f = 2$

Sensitivity of $g_\ell(X)$:

Changing one data point can affect cluster assignments, changing the counts in up to two clusters by 1 (one increases by 1, another decreases by 1).

The difference vector has +1 and -1 in two components, zeros elsewhere.

$\|g_\ell(X) - g_\ell(X')\|_2 = \sqrt{2}$,

So, $\Delta_g = \sqrt{2}$

Applying the Gaussian Mechanism (Proposition 3):

Privacy Loss for $f_\ell(X) + z_\ell$: $\rho_f = \frac{{\Delta_f}^2}{2\sigma^2} = \frac{2^2}{2\sigma^2} = \frac{2}{\sigma^2}$

Privacy Loss for $g(X) + z'_\ell$: $\rho_g = \frac{{\Delta_g}^2}{2{\sigma'}^2} = \frac{\sqrt{2}^2}{2{\sigma'}^2} = \frac{1}{{\sigma'}^2}$

Total Privacy Loss per Iteration: $\rho_{iter} = \rho_f + \rho_g = \frac{2}{\sigma^2} + \frac{1}{{\sigma'}^2}$

Aggregating Over t Iterations:

Using Lemma 4 (Composition), the total privacy loss over t iterations is additive:

$\rho_{total} = t \cdot \rho_{iter} = t(\rho_f + \rho_g) = (\frac{2t}{\sigma^2} + \frac{t}{{\sigma'}^2})$

Therefore $\mathcal{M}'_{means}$ satisfies ρ-zCDP with $\rho = 2t/\sigma^2 + t/{\sigma'}^2$.

2 Implementing $\mathcal{M}_{means}$

(a)

Answer:

The link to the full code on Colab: [my code on Colab]

From the previous analysis, the algorithm satisfies ρ-zCDP with: $\rho = 2t/\sigma^2 + t/{\sigma'}^2$

To allocate the privacy budget optimally between σ and σ', balance the privacy loss contributed by each. Using the sensitivities of the functions:

Sensitivity of $f_\ell(X)$: $\Delta_f = 2$

Sensitivity of $g_\ell(X)$: $\Delta_g = \sqrt{2}$

Set: $\frac{{\Delta_f}^2}{2\sigma^2} = \frac{{\Delta_g}^2}{2\sigma'^2} = \frac{\rho}{2t}$

Solving for σ^2 and σ'^2: $\sigma^2 = \frac{4t}{\rho}$, $\sigma'^2 = \frac{2t}{\rho}$

The modified code section can be found in the file appendix.

Algorithm Steps:

Initialization:

Randomly assigns each data point to one of the k clusters.

Computes the initial noisy cluster counts $n_i^{(0)}$.

Iterative Updates (for $\ell = 1$ to t):

Cluster Center Update [Step 2.(a)]:

Computes the sum of points in each cluster.

Adds Gaussian noise $z_{\ell,i}$ to the sum.

Updates the cluster centers $c_1^{(\ell)}$ using the noisy counts $n_i^{(\ell-1)}$.

Cluster Assignment [Step 2.(b)]:

Assigns each point to the nearest cluster center.

Noisy Count Update [Step 2.(c)]:

Computes the new cluster sizes.

Adds Gaussian noise $\mathcal{Z}'_{\ell,i}$ to obtain $n_i^{(\ell)}$.

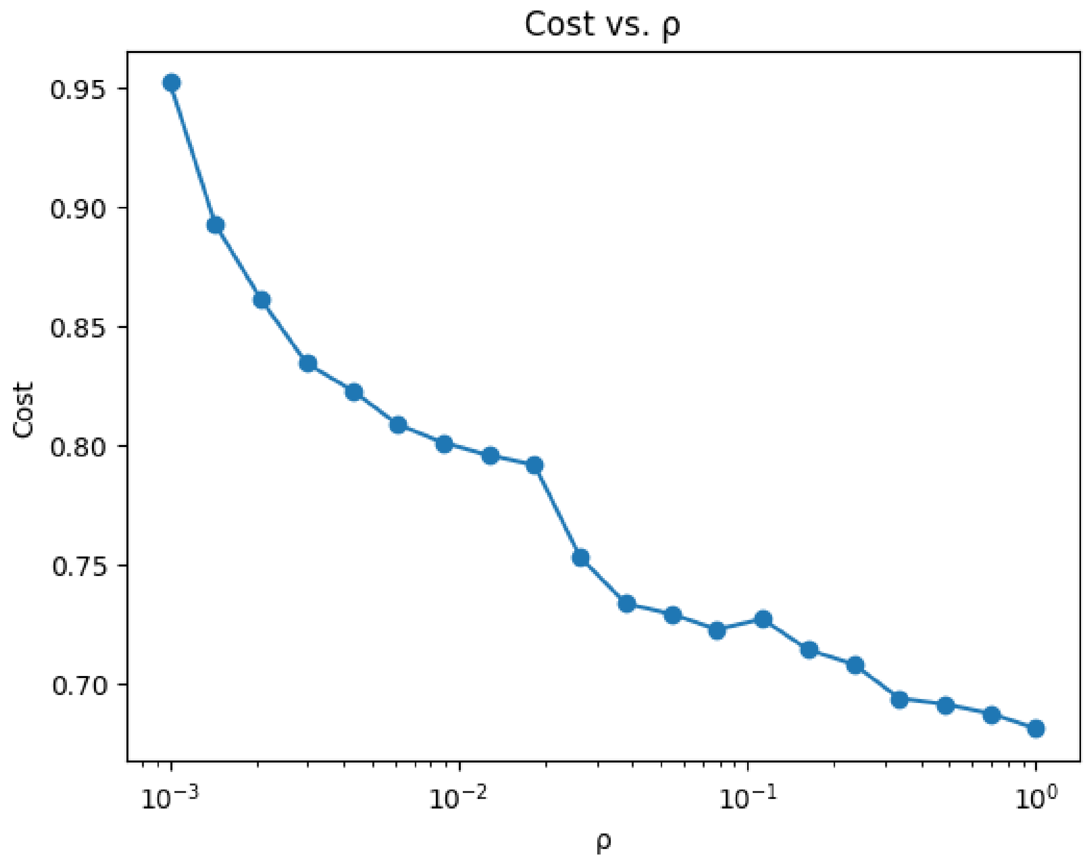
Final Output [Step 3]:

Returns the cluster centers after t iterations.

(b)

Answer:

Result plot (The modified code section can be found in the file appendix.):



High Privacy (Low ρ): At low values of ρ, the added noise is significant, leading to higher clustering costs.
Low Privacy (High ρ): As ρ increases, less noise is added, resulting in better clustering (lower cost).

Appendix

```
# Differentially Private k-means Algorithm
def M_means(points, k, t, rho):
    n, d = points.shape

    # Calculate sigma and sigma_prime based on rho and t
    sigma_squared = (4 * t) / rho
    sigma = np.sqrt(sigma_squared)

    sigma_prime_squared = (2 * t) / rho
    sigma_prime = np.sqrt(sigma_prime_squared)

    # Step 1: Random initialization of disjoint clusters
```

```
    initial_assignment = np.random.choice(range(k), n)
    cluster_indexes = [ np.where(initial_assignment == i)[0] for i in
range(k) ]
    n_i_prev = [ len(cluster_indexes[i]) for i in range(k) ]  #
n_i^{(0)}

    for l in range(1, t + 1):
        # Step 2(a): Update cluster centers with noise
        c_i = []
        for i in range(k):
            S_i_prev = cluster_indexes[i]
            # Sum over points in S_i^{(l-1)}
            if len(S_i_prev) > 0:
                f_l_i = np.sum(points[S_i_prev], axis=0)
            else:
                f_l_i = np.zeros(d)
            # Add Gaussian noise
            Z_l_i = np.random.normal(0, sigma, d)
            denominator = max(1, n_i_prev[i])
            # Compute c_i^{(l)}
            c_i_l = (Z_l_i + f_l_i) / denominator
            c_i.append(c_i_l)
        centers = np.array(c_i)  # Centers for iteration l

        # Step 2(b): Assign points to the nearest cluster center
        distances_squared = np.sum((points - centers[:,
np.newaxis])**2, axis=-1)
        assignment = np.argmin(distances_squared, axis=0)
        cluster_indexes = [ np.where(assignment == i)[0] for i in
range(k) ]

        # Step 2(c): Update cluster sizes with noise
        n_i = []
        for i in range(k):
            cluster_size = len(cluster_indexes[i])
            z_prime_l_i = np.random.normal(0, sigma_prime)
            n_i_l = cluster_size + z_prime_l_i
            n_i.append(n_i_l)
        n_i_prev = n_i  # Update for next iteration
    # Step 3: Output the final cluster centers
    return centers  # Final cluster centers c_i^{(t)}
```

```python
"""# Plot cost as function of number of iterations"""

# Parameters
k = 5  # Number of clusters
t = 5  # Number of iterations
rho_values = np.logspace(-3, 0, num=20)  # 20 values from 0.001 to 1
costs = []

# Run M_means for each rho and compute cost
for rho in rho_values:
    centers = M_means(points, k, t, rho)
    cost = compute_cost(points, centers)
    costs.append(cost)
    print(f"Rho: {rho:.4f}, Cost: {cost:.4f}")

# Plot the cost as a function of rho
fig, ax = plt.subplots()
ax.set_xlabel('ρ')
ax.set_ylabel('Cost')
ax.plot(rho_values, costs, marker='o')
ax.set_xscale('log')
ax.set_title('Cost vs. ρ')
plt.show()
```