Siam hackathon

Alberto Bocchinfuso, Ginnie Renz, Thomas Saigre, Kevin Valencia

February 26, 2023

1 Notations

We have a dataset of m individuals, with N features each. We denote the dataset as $A \in \mathbb{R}^{m\times N}$, where each row is a feature vector of an individual. The mean value of the dataset, over features is denoted as $\bar{a} \in \mathbb{R}^N$, and the covariance matrix is denoted as $\Sigma \in \mathbb{R}^{N\times N}$.

2 Distance function

Given

$$A = \begin{bmatrix} a_1^\top \\ \vdots \\ a_m^\top \end{bmatrix} \in \mathbb{R}^{m\times N} \tag{1}$$

find a distance function

$$d(a_i, \bar{a}, \Sigma, a^M) \tag{2}$$

that weights how far a_i is form $\bar{a}$, mean value of A, taking into consideration the covariance matrix Σ and the vector of maximum differences

$$a_j^M = \max_{i=1\dots m} |a_{i,j} - \bar{a}_j| \quad j = 1...N. \tag{3}$$

Possible distances are based on energy norms with respect to the covariance

$$d_1(a, \bar{a})^2 = (a - \bar{a})^\top \Sigma^{-1} (a - \bar{a}) \tag{4}$$

or with respect to the square of the biggest difference:

$$d_2(a, \bar{a})^2 = (a - \bar{a})^\top M^{-1} (a - \bar{a}) \tag{5}$$

with M:

$$\begin{bmatrix} \left[a_1^M\right]^2 & 0 & 0 & \dots & 0 \\ 0 & \left[a_2^M\right]^2 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & & 0 \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & 0 & \dots & \left[a_N^M\right]^2 \end{bmatrix} \in \mathbb{R}^{N\times N}. \tag{6}$$

The last possible distance is a convex combination of d_1, d_2:

$$d_3(a, \bar{a})^2 = \theta d_1(a, \bar{a}) + (1 - \theta) d_2(a, \bar{a}) \,,\ \theta \in (0, 1) \tag{7}$$

The metric to measure how good the privatization is, we may use:

$$M(A, A^1) = \sum_{i=1}^{m} d(a^1, \bar{a}^1)^2 (1 - \mathbb{P}(a_j | a_i^1)) \tag{8}$$

where d is one of the distances above, and $\mathbb{P}(j|a_i^1)$ is a measure of the probability to recognize the that i comes from the j-th row of A, assuming that j is really the correct row. **Alternatives** Probably a better function would be using the square of the probability term as well:

$$M_{best}(A, A^1) = \sum_{i=1}^{m} d(a^1, \bar{a}^1)^2 (1 - \mathbb{P}(a_j|a_i^1))^2 \tag{9}$$

What we do:

- If the matrix is the same, the measure $M(A, A) \approx 0$
- If the matrix is the same and the masking is shuffling the rows, the measure $M(A, LA) \approx 0$ (L is a permutation matrix)
- We weight more the data that are further from the mean value.
- If there is a high probability to retrieve j from a_i^1 for data near the average $\bar{a}$, we still weight it, since $(1 - \mathbb{P}(j|a_j^1))$ will be close to 0.

What we expect:

- d_2 works well in the presence of many outliers because it is not weighting them too much, since it considers the maximum difference and not just the standard deviation, which may lead to weighting a lot the single outlier.

This metric shouldn't give too much advantage of the fact that a masking method masks the outliers very well since considers all the points and weights them both according to the distance from the mean (how peculiar that particle is in the dataset.), but also according to the probability of being recognized.

We expect that the values that are close to the mean are the most present, so if the algorithm doesn't mask well those data, we will weight this fact: we sum many small numbers $(1 - P(j|a_j^1))$ so we keep the metric close to 0), too.

$M(A, A^1)$ close to 0 $\implies$ the matrices are very similar, in the sense that an attacker can retrieve the specific particles. $M(A, A^1)$ big $\implies$ the masking matrix A^1 doesn't allow an attacker to retrieve specific particles, in general. It is still possible to recognize some specific people.

In practice, there should be a trade off between data quality and $M(A, A^1)$.

Probability

$P(j|A^1)$ can be computed using the correlation. For every a_i^1, compute

$$C_i j = \frac{(a_j - \bar{a})^\top (a_i^1 - \bar{a}^1)}{\|(a_j - \bar{a})^\top (a_i^1 - \bar{a}^1)\|} \tag{10}$$

Exclude the maximum and compute the mean

$$\tilde{C}_i = \frac{1}{m-1} (\sum_j C_i j - \max_j C_i j) \tag{11}$$

our probability is:

$$P(a_j|a_i^1) = \max_j C_i j - \tilde{C}_i \tag{12}$$

IMPORTANT: it is not clear whether this is a good way of computing the probability. PROBABLY NOT. (very high probability, every time.)

ALTERNATIVES

In practice, on the given dataset, d_1 worked better.