Differential Privacy: A Primer

Abhishek Tiwari

Citation: *A. Tiwari*, ”Differential Privacy: A Primer”, Abhishek Tiwari, 2024.
doi:10.59350/t6p9d-y6y38

Published on: September 22, 2024

Differential Privacy (DP) is a mathematical framework that protects individual privacy in data analysis while allowing useful insights to be extracted. It works by adding carefully calibrated noise to data or query results, ensuring that including or excluding any single individual's data doesn't significantly change the analysis outcomes. This approach makes it extremely difficult to infer information about specific individuals, providing a formal guarantee of privacy (see [1]). Differential privacy is widely used by companies, researchers, and government agencies to analyze sensitive information such as census data, medical records, and large-scale user behavior while preserving privacy.

Understanding Utility and Privacy Budget

Utility: This refers to the usefulness or accuracy of the data after applying differential privacy techniques. Higher utility means the privatized data more closely resembles the original data.

Privacy Budget (ε): This is a measure of the privacy loss. ε represents the maximum amount of information that can be learned about an individual from the output of a privacy-preserving algorithm. A smaller ε provides stronger privacy guarantees but typically results in lower utility. The challenge in differential privacy is to balance utility and privacy. As we increase privacy (by decreasing ε), we typically decrease utility, and vice versa. In practice, ε values often range from 0.1 to 1.

Types of (ε, δ)-Differential Privacy

Differential privacy comes in several flavors, each with its own privacy guarantees and use cases (see [2]). In this blog post, we will cover following two,

(ε, 0)-Differential Privacy

This is the strongest and simplest form of differential privacy. It provides a strict upper bound on the privacy loss. (ε, 0) differential privacy is also known as pure differential privacy.

Definition: A randomized algorithm M satisfies ε-differential privacy if for all datasets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(M)$:

$$P(M(D_1) \in S) \leq e^{\varepsilon} \cdot P(M(D_2) \in S)$$

where P denotes probability.

Use case: When you need the strongest privacy guarantees and can tolerate more noise in the results.

(ε, δ)-Differential Privacy

This is a relaxation of pure ε-differential privacy. It allows for a small probability δ of violating the ε-privacy guarantee.

Definition: A randomized algorithm M satisfies (ε, δ)-differential privacy if for all datasets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(M)$:

$$P(M(D_1) \in S) \leq e^{\varepsilon} \cdot P(M(D_2) \in S) + \delta$$

where P denotes probability.

Probability of Leak (δ): The parameter which tells the probability that the privacy loss might exceed ε. δ is usually chosen to be very small, often smaller than 1/n, where n is the number of records in the dataset. Common values might be 10^-5 or 10^-6.

Use case: When you need to balance strong privacy guarantees with higher utility, and can accept a very small probability of privacy loss.

Local Differential Privacy

Applies Differential Privacy at the individual data point level before aggregation, rather than to the entire dataset. Local DP provides the strong guarantee of pure ε-differential privacy at the individual data point level. Each individual's privacy is protected even if the data collector is compromised.

Definition: A randomized algorithm M satisfies ε-local differential privacy if for all $x, x' \in X$ and all $S \subseteq Range(M)$:

$$P(M(x) \in S) \leq e^{\varepsilon} \cdot P(M(x') \in S)$$

where X is the input domain.

Use case: When you want to provide privacy guarantees to individual users without trusting the data collector.

Local vs. Central Differential Privacy

With Local DP, noise is added by individual users before data is sent to the collector. The query layer aggregates already-noisy data. No additional noise needs to be added at query time.

With Central DP, Raw data is sent to the trusted curator. The query layer is where differential privacy noise is added. Each query on the raw data is processed through the DP mechanism before results are released.

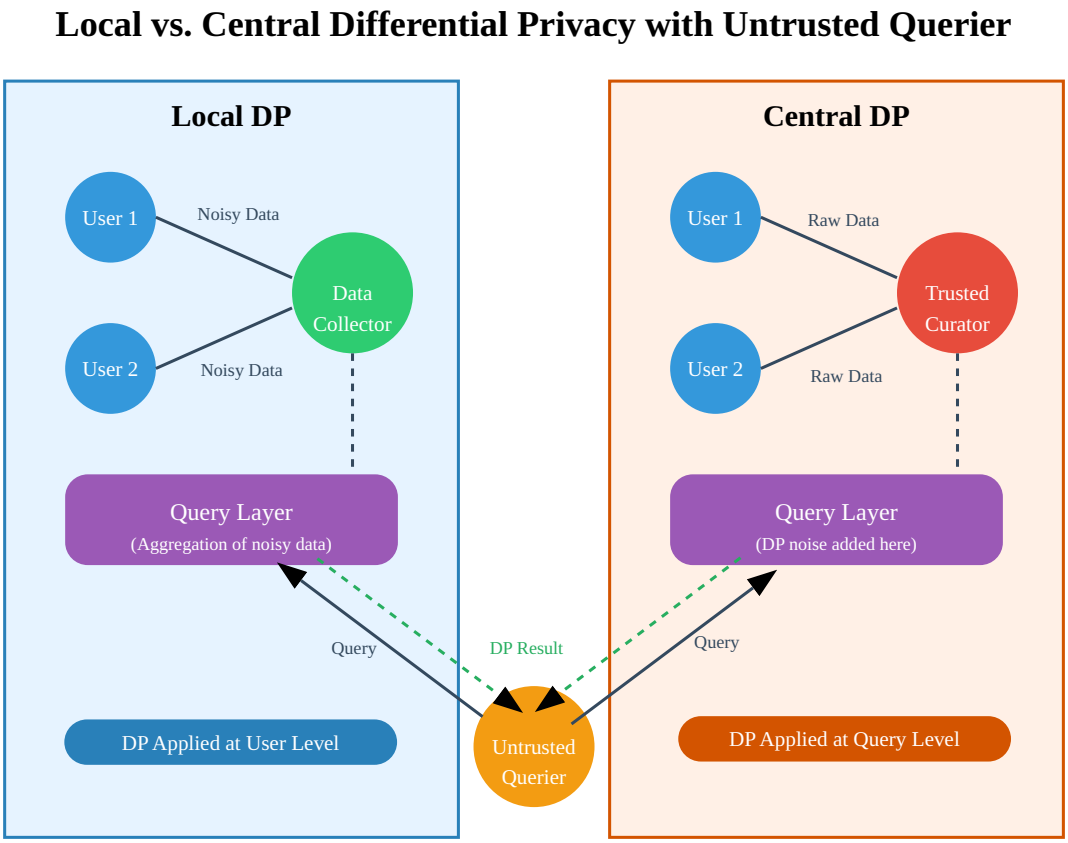

Figure 1: In local DP the random noise is applied at the start of the process(local) level i.e when the data is sent to the data curator/aggregator. In Global DP the random noise is applied at the global level i.e when the answer to a query is returned to the User.

Techniques

Laplace Mechanism

The Laplace Mechanism is widely used for pure ε-differential privacy. It works by adding noise drawn from a Laplace distribution to the true output of a function.

Listing 1: Example Python code for Laplace Mechanism

```python
import numpy as np

def laplace_mechanism(true_value, sensitivity, epsilon):
    scale = sensitivity / epsilon
    noise = np.random.laplace(0, scale)
    return true_value + noise

# Example usage
true_count = 1000
sensitivity = 1  # Assuming count query
epsilon = 0.1  # Privacy parameter

private_count = laplace_mechanism(true_count, sensitivity, epsilon)
print(f"True count: {true_count}")
print(f"Private count: {private_count}")
```

The Laplace Mechanism is particularly useful for numeric queries where we can easily calculate the sensitivity (maximum change in the output when one record is added or removed from the dataset).

Listing 2: Example output

```
True count: 1000
Private count: 1002.0886527668904
```

Emoji Suggestions Apple has been using local differential privacy since iOS 10 to collect user data while preserving individual privacy. Local differential privacy adds noise to the data on the user's device before it's sent to Apple's servers. One of Apple's applications of differential privacy is in improving emoji suggestions. Here's a basic implementation:

```python
import numpy as np

def privatize_emoji_count(true_count, epsilon):
    sensitivity = 1  # Each user can affect the count by at most 1
    noise = np.random.laplace(0, sensitivity / epsilon)
    return max(0, int(round(true_count + noise)))  # Ensure non-negative
       integer

# Simulate emoji usage data
emojis = ["", "", "❤", "", ""]
true_counts = {emoji: np.random.randint(1000, 10000) for emoji in emojis}

epsilon = 0.1  # Privacy budget per emoji

# Privatize counts
private_counts = {emoji: privatize_emoji_count(count, epsilon) for emoji,
   count in true_counts.items()}

# Print results
```

```
for emoji in emojis:
    print(f"{emoji}: True count = {true_counts[emoji]}, Private count = {
        private_counts[emoji]}")

# Calculate total privacy budget used
total_budget = len(emojis) * epsilon
print(f"\nTotal privacy budget used: {total_budget}")
```

In this example, we simulate the process of collecting emoji usage data. Each emoji count is privatized using the Laplace mechanism, similar to Apple's approach. The privacy budget (epsilon) is set to 0.1 per emoji, which is a relatively strong privacy guarantee. The total privacy budget used is the number of emojis multiplied by the epsilon per emoji.

Listing 3: Example output

```
□
: True count = 9725, Private count = 9726□
: True count = 6248, Private count = 6238❤
: True count = 7296, Private count = 7287□
: True count = 3788, Private count = 3760□
: True count = 6026, Private count = 6022

Total privacy budget used: 0.5
```

Apple uses a similar approach but with more sophisticated algorithms and a carefully managed privacy budget to ensure that the overall privacy loss remains within acceptable limits.

Gaussian Mechanism

The Gaussian Mechanism is similar to the Laplace Mechanism but uses Gaussian (normal) noise instead. It's often used when we need to satisfy (ε, δ)-differential privacy, a relaxation of pure ε-differential privacy.

```
import numpy as np

def gaussian_mechanism(true_value, sensitivity, epsilon, delta):
    sigma = np.sqrt(2 * np.log(1.25 / delta)) * sensitivity / epsilon
    noise = np.random.normal(0, sigma)
    return true_value + noise

# Example usage
true_average = 50
sensitivity = 1  # Assuming bounded range of data
epsilon = 0.1
delta = 1e-5
```

```
private_average = gaussian_mechanism(true_average, sensitivity, epsilon,
    delta)
print(f"True average: {true_average}")
print(f"Private average: {private_average}")
```

The Gaussian Mechanism is useful when you need more flexibility in balancing privacy and accuracy, especially for complex queries or machine learning applications.

Listing 4: Example output

```
True average: 50
Private average: -8.029981070071308
```

Medical Research The Gaussian Mechanism is often used in medical research to protect patient privacy while allowing meaningful analysis. Let's simulate a study on the effectiveness of a new treatment:

```
import numpy as np

# Simulated patient data
placebo_group = np.random.normal(loc=5, scale=1, size=1000)  # Improvement
    score for placebo group
treatment_group = np.random.normal(loc=7, scale=1, size=1000)  #
    Improvement score for treatment group

def private_t_statistic(group1, group2, epsilon, delta):
    true_t_stat = (np.mean(group1) - np.mean(group2)) / np.sqrt(np.var(
        group1)/len(group1) + np.var(group2)/len(group2))
    sensitivity = 2 / (len(group1) + len(group2))  # Simplified
        sensitivity calculation
    return gaussian_mechanism(true_t_stat, sensitivity, epsilon, delta)

epsilon = 0.1
delta = 1e-5

private_t_stat = private_t_statistic(treatment_group, placebo_group,
    epsilon, delta)
true_t_stat = (np.mean(treatment_group) - np.mean(placebo_group)) / np.
    sqrt(np.var(treatment_group)/len(treatment_group) + np.var(
    placebo_group)/len(placebo_group))

print(f"True t-statistic: {true_t_stat:.4f}")
print(f"Private t-statistic: {private_t_stat:.4f}")

# Calculate utility (as relative error)
utility = 1 - abs(private_t_stat - true_t_stat) / true_t_stat
print(f"\nUtility: {utility:.2%}")
print(f"Privacy budget (epsilon) used: {epsilon}")
```

```
print(f"Delta: {delta}")
```

In this medical research example, we use a privacy budget (epsilon) of 0.1 and a delta of 1e-5. The choice of these parameters depends on the sensitivity of the medical data and the desired privacy guarantees. The utility is calculated as the relative accuracy of the private t-statistic compared to the true t-statistic.

Listing 5: Example output

```
True t-statistic: 43.4309
Private t-statistic: 43.3674

Utility: 99.85%
Privacy budget (epsilon) used: 0.1
Delta: 1e-05
```

Conclusion

As you implement these techniques, remember that the choice of privacy parameters (ε and δ) is crucial and depends on your specific use case and privacy requirements. Always consider the sensitivity of your data and the potential risks of privacy breaches.

References

[1] A. Tiwari, “Mathematical Guarantee,” 2024, *Abhishek Tiwari*. doi: 10.59350/ghs12-1vq60.

[2] C. Dwork and A. Roth, “The Algorithmic Foundations of Differential Privacy,” *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014, doi: 10.1561/0400000042.