Loss function notes

Loss functions are typical for handling the error while the model is training, to reduce the loss means to ensure the accuracy. But the problems usually come with those advantages, which are overfitting and high confidence issues. Due to some samples being outliers or rare samples, some loss functions are more preferred to put them into model's memory, such that, privacy and data security thread and membership inference attack will occur.

The **loss** function is just like a guide that directs how the model should adjust its parameters. It measures how far the model's predictions are from the actual values. Based on this, gradients are computed, which indicate which direction to move in and how much to change the parameters. These gradients are then used to upgrade the model in order to improve the accuracy.

The **parameter** is the value that is produced from the model learned from the training data, which means it will be used to make predictions and are updated during training to minimize the loss.

Such as in a linear model: **y=wx+b**,
where w is **weight** (major parameter), b is the **bias** (another parameter), the parameters are not fixed.

Weights: Control how much influence each input has.
Biases: Adjust the output of each neuron.

The workflow will be "**calculate the loss** -> **compute gradient** -> **paraments**"

These parameters get updated through backpropagation and gradient descent. It will be adjusted during training.

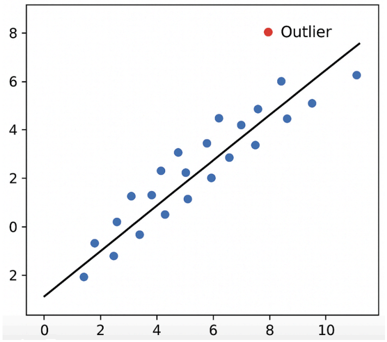
There are few kinds of **LOSS** function to handle whether the model takes the different operations, which are classification and regression.
(Reference Article, the last edited date 12/04/2024, report current date: 05/29/2025)

Outlier data and loss function
An outlier is a data point that is significantly different from the rest of the data. It "lies outside" the overall pattern or distribution of the dataset. The data points are usually very unique, lacking diversity, and isolated, with no surrounding data for generalization. Those special characteristics with 'accuracy-driven loss functions' can cause the model to memorize these samples instead of learning generalized patterns from the datagroup.

Example:
Integer dataset: [1,2,3,4,5,6,7,8,9999], where "9999" is the outlier data.

Medical dataset: [Blood protein concentration: %0.99, White blood cell count: %12.6, **Breakfast menu: bacon omelette**], Where the “**Breakfast menu: bacon omelette**” is outlier data.

Linear regression:

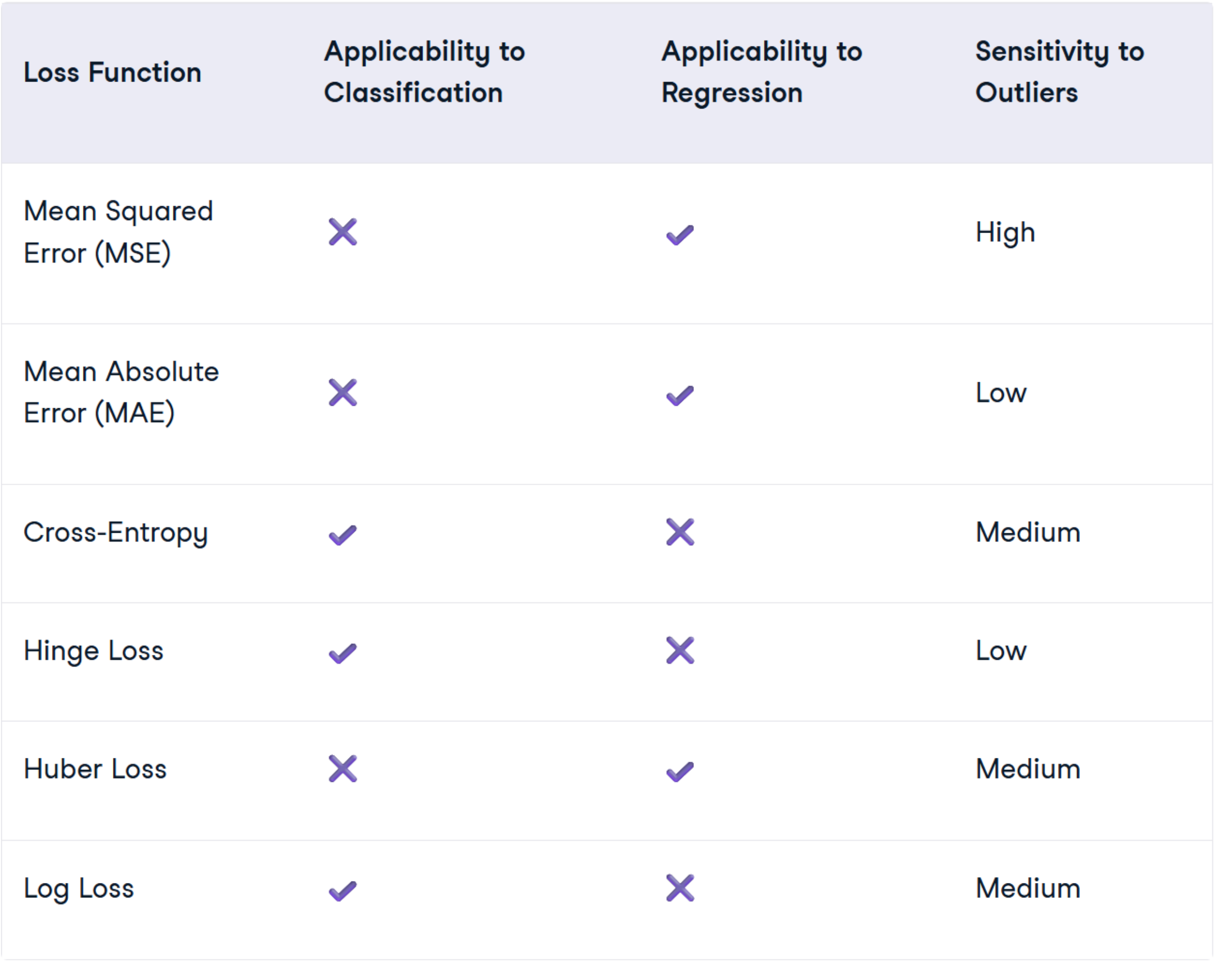


Loss Functions classifier and brief summary

Two kinds of tasks: **Classification** and **Regression**

Classification is to classify the different categories and tags, such as [cat, dog, rat].

Regression is to predict linear regression tasks such as predicting the housing prices such as a group of continuous values.



Loss Function	Applicability to Classification	Applicability to Regression	Sensitivity to Outliers
Mean Squared Error (MSE)	✗	✓	High
Mean Absolute Error (MAE)	✗	✓	Low
Cross-Entropy	✓	✗	Medium
Hinge Loss	✓	✗	Low
Huber Loss	✗	✓	Medium
Log Loss	✓	✗	Medium

Loss for regression

1. Mean Square Error (MSE) / L2 Loss equations：
   $MSE = (1/n) * \Sigma(y_i - \bar{y})^2$
   Why it is High Sensitivity to Outliers
   Workflow:
    1. Subtract the **predicted value** from the **true values** such as yi - ȳi
    2. Then **square the error** which is (yi - ȳi)^2
    3. Apply this for all points, and **sum** them up (the square error)
    4. **Divide** by n, where n is all samples number to take the **average**.

Because in this mode using error by **square**, that large error difference will bring a much larger loss. Unusual error value will significantly affect the model running (heavily distort the overall loss value and gradient direction, and one individual large error could **dominate** the loss to reduce the accuracy), therefore under this situation, the model will primarily fit the outlier data point to ensure the overall performance and accuracy, this is also the advantage of the MSE, Smooth & Differentiable (very applied on the regression task), It has a higher tolerance for small errors, a stronger penalty for large errors, and a strong reaction, and directly forces the data to fit to eliminate performance problems caused by huge errors. But, this mechanism will also bring privacy issues.

Guess: The high penalty that MSE assigns to outliers may cause the model to deliberately fit these few abnormal data points, which can lead to the risk of overfitting.

2. Mean Absolute Error (MAE) / L1 Loss
   $MAE = (1/n) * \Sigma|y_i - \bar{y}|$
   Low Sensitivity to Outliers

   Under this loss calculation, it treats all errors with equal weight regardless of their magnitude. To find the median values and **'ignore'** the large difference impact from the outlier data, with **robust characteristics.** The goal of MAE is to minimize the sum of absolute errors. Mathematically, its optimal solution is the median, but it does not actually exclude any data points. Instead, it minimizes the "total distance from the median." When fitting a constant to a bunch of data, the constant with the smallest MAE is the **median**.
   Reduce the LOSS

   If a group of true label: yi = [10, 20, 30, 40, 100]
   The will try if **ȳ = 30** then:
   MAE $|10-30|+|20-30|+|30-30|+|40-30|+|100-30|$=20+10+0+10+70=110/5 = 22
   If we try **ȳ = 20** then it will be equal to 120/5 = 24
   If we try **ȳ = 40** then it will be equal to 120/5 = 24
   If we try **ȳ = 100** then it will be equal to 300/5 =60

When **$\bar{y} = 30$**, is the smallest median (22).

Delivery services like UberEats, Deliveroo or DoorDash may build delivery estimation models to improve customer satisfaction. The time it takes for a delivery service to deliver a meal can be affected by many factors, such as weather, traffic accidents, road construction, etc. These factors can be seen as outliers and affect the prediction.

MAE is discontinuous or not differentiable at 0

3. Huber Loss / Smooth Mean Absolute Error (Medium Sensitivity to Outliers)

```
L(δ, y, f(x))
= (1/2) * (f(x) - y)^2           if |f(x) - y| <= δ  MSE
= δ * |f(x) - y| - (1/2) * δ^2   if |f(x) - y| >   δ  MAE
```

Where:
L represents the Huber Loss function
δ is the delta parameter, which determines the threshold for switching between the quadratic and linear components of the loss function
y is the true value or target value
f(x) is the predicted value

Subtracting (1/2) δ^2 is to ensure that the Huber Loss curve does not jump, interrupt, or break at the inflection point, which is mathematically called "continuous and first-order differentiable."

Combination of mean absolute error and mean square error loss functions, Huber Loss is controlled by a threshold (usually denoted as δ). MSE is used when the error is less than the threshold, and MAE is used when the error is greater than the threshold. Avoid outliers dominating training while pursuing high accuracy.

Loss for classification

4. **Binary Cross-Entropy** Loss / Log Loss

```
L(y, f(x)) = -[y * log(f(x)) + (1 - y) * log(1 - f(x))]
```

Where:
L represents the Binary Cross-Entropy Loss function
y is the true binary label (0 or 1)
f(x) is the predicted probability of the positive class (between 0 and 1)

Medium Sensitivity to Outliers

Under this loss function model, it outputs a prediction with a probability value typically between 0 and 1. Compare with the true values and calculate with log to have the loss values.
Examples:
The true label is 1 the the loss will be calculate by the equation:
$L=-[y\cdot\log(y^\wedge)+(1-y)\cdot\log(1-y^\wedge)]$
If the predicted probabilities is 0.90, then ȳ = 0.90, using the formula:
$L=-[1\cdot\log(0.90)+0\cdot\log(1-0.90)]=-\log(0.90)$, the loss is ~= −0.10536

The true label is 0 the loss will be calculate as the equation:
$L=-[y\cdot\log(y^\wedge)+(1-y)\cdot\log(1-y^\wedge)]$
$L=-\log(1-0.20)=-\log(0.80)$, the loss is ~=-（−0.22314）=~0.22314

For example of large loss value:
If the true label is 0, then the predicted probabilities is 0.88, then the Loss will be:
$\log(0.12)\approx-2.1203$ $L=-(-2.1203)=2.120$, very large.

5. Categorical Cross-Entropy Loss
   Low Sensitivity to Outliers
   Similar to binary cross-entropy but this time is multi-class by using the type of softmax. Such as a true label cat[1.00, 0.00, 0.00] which the predict probabilities is cat[0.90, 0.05, 0.05], then loss will be $\text{Loss} = -[1\cdot\log(0.90)+0+0]=-\log(0.90)\approx0.105$

   The example of extreme outliers datapoint:
   True label = [1.00, 0.00, 0.00], the predict probabilities [0.01, 0.99, 0.00]
   The loss will be $L=-\log(0.01) \approx 4.605$, push and force the model to memorize the samples, overfitting.

6. Hinge Loss
   $L(y, f(x)) = \max(0, 1 - y * f(x)), y\in\{-1,+1\}$
   L represents the Hinge Loss
   y is the true label or target value (-1 or 1)
   f(x) is the predicted value or decision function output

   Medium Sensitivity to Outliers, binary classification

   Similar to the label 1 and 0, but different is in Hinge Loss using +1 and -1 to identify the loss, +1 and -1 are not just classification labels, they are also a directional geometric indicator that allows the model to control the loss by moving closer or farther from the decision boundary on the numerical axis. Very applied for the SVM (vector).

```
 |-------------|-------------|
-1             0            +1
```

Example when calculate the loss of +1/-1 label

Label y = +1,			
Output f(x)	**$1 - y \cdot f(x) = 1 - f(x)$**	**Loss = max(0, ...)**	**Margin**
0.0	1.0 - 0.0 =1.0	1.0	No margin, misclassified or unsure
0.5	1.0 - 0.5 = 0.5	0.5	Low margin, underconfident
0.9	1.0 - 0.9 = 0.1	0.1	Near margin threshold, small loss
1.0	1.0 - 1.0 = 0.0	0.0	At margin boundary, loss-free
1.5	1.0 - 1.5 = -0.5	0.0	Beyond margin, confident prediction

Label y = -1,			
Output f(x)	**$1 - (-1) \cdot f(x) = 1 + f(x)$**	**Loss = max(0, ...)**	**Margin**
-0.1	1.0 + (-0.1) = 0.9	0.9	Low margin, unconfident
-0.5	1.0 + (-0.5) = 0.5	0.5	Insufficient margin, partial loss
-0.9	1.0 + (-0.9) = 0.1	0.1	Near decision boundary, small loss
-1.0	1.0 + (-1.0) = 0	0.0	Meets margin requirement, no loss
-2.0	1.0 + (-2.0) = 1.0	0.0	Confident prediction, large margin

Example, identify a spam email:
y = +1: it is a spam
y = -1: it's not spam
Output if +2.5 means it is definitely spam, if output -2.5 then means it is definitely not spam.

But how is margin controlled? Margin is a manually set "safe distance threshold", usually 1, which is used to require the model to not only make correct predictions, but also to be "separated far enough" to improve robustness and privacy. It controls the model training direction through "1 - y·f(x)" in Hinge Loss.

7. Log Loss
   Medium Sensitivity to Outliers

Log Loss is mathematically equivalent to Binary Cross-Entropy (BCE).

They both use the same formula to calculate the difference between the predicted probability and the true label in a binary classification model.

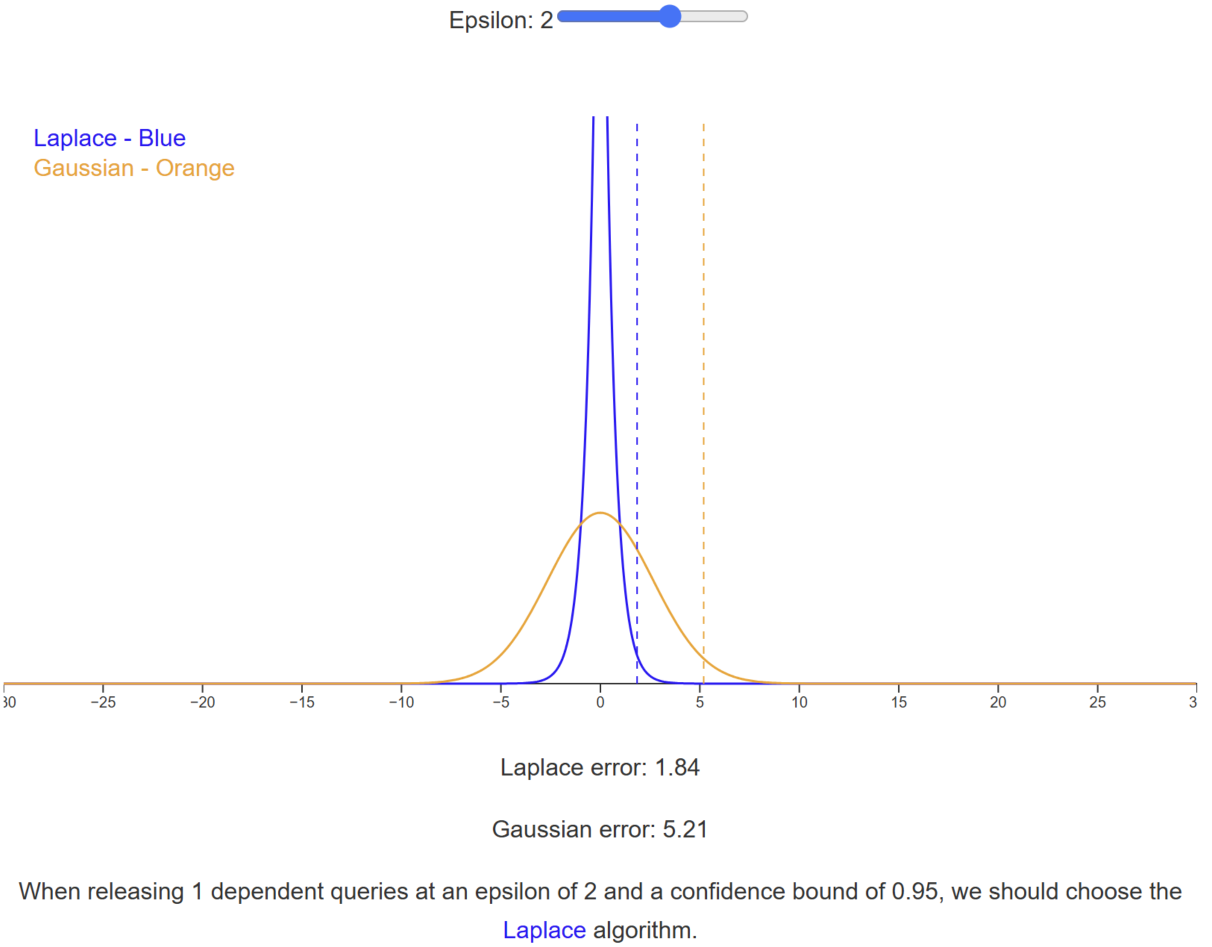
In different applications and frameworks, "Log Loss" is a common name for this loss function in traditional statistics and Kaggle competitions, while "Binary Cross-Entropy" is more commonly used in deep learning frameworks (such as PyTorch and TensorFlow).
Despite the different names, their calculation logic, gradient behavior, and impact on model training are exactly the same.

Laplace noise and Gaussian noise

Overall review.
The Laplace mechanism is more suitable for "small amount, low dimension, one-time" data release scenarios, emphasizing privacy strength; (e.g. Demographics Population)

The Gaussian mechanism is more suitable for "multi-round, high dimension, iterative" machine learning scenarios, emphasizing the balance between privacy and utility. (e.g. DP-SGD)

https://lpanavas.github.io/mechanism-comparison/



Epsilon: 2

Laplace error: 1.84

Gaussian error: 5.21

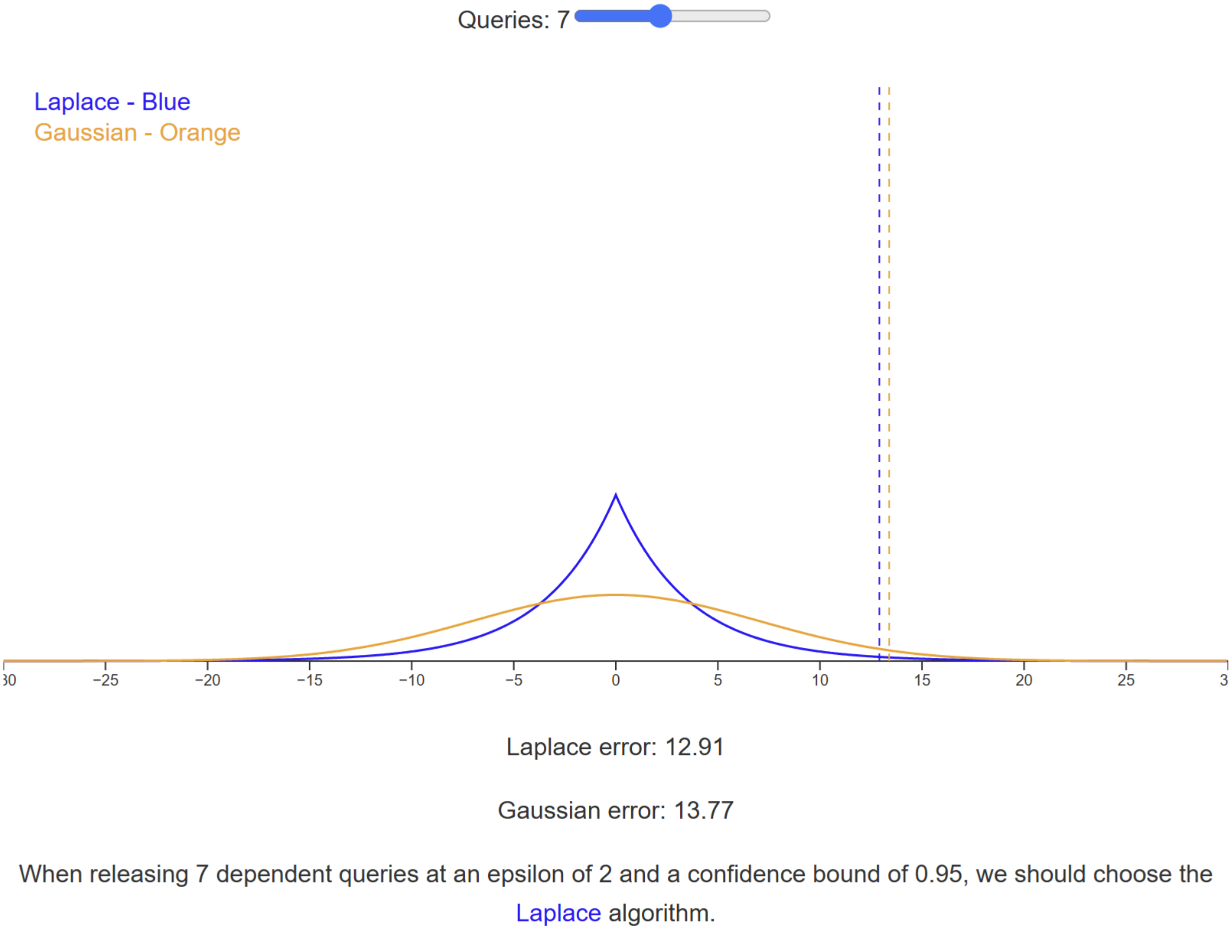
When releasing 1 dependent queries at an epsilon of 2 and a confidence bound of 0.95, we should choose the Laplace algorithm.

Epsilon is set as 2.

(Both Laplace and Gaussian mechanisms introduce a certain degree of perturbation centered around the true label, typically 0. However, they have different behaviors significantly.
As shown in the figure (**where the x-axis represents the amount of deviation and the y-axis indicates the probability of that perturbation**), the Laplace distribution is highly sensitive to changes in perturbation probability—its probability density drops sharply with each step, resulting in a steep, narrow peak followed by a long extension toward larger deviations.
In contrast, the Gaussian distribution appears much smoother. Its tail decays more gradually, exhibiting a soft and gentle curve, which suggests that extreme deviations are less likely to occur.)

Both the Laplace and Gaussian mechanisms introduce perturbations centered around the true value (e.g., label 0), within a certain range. As illustrated in the figure—where the x-axis represents the magnitude of deviation and the y-axis indicates the probability density—the Laplace distribution is highly sensitive to perturbation probability. It exhibits a steep decline near the center, forming a sharp peak, but maintains relatively

high probability density in the tails. This "delayed decay" implies that large deviations still have a non-negligible chance of being sampled. In contrast, the Gaussian distribution behaves more smoothly. Its probability density decays gradually from the center toward the tails, forming the characteristic bell-shaped curve. Although its peak is slightly lower than that of the Laplace distribution, its tails decay more rapidly, which reduces the likelihood of extreme perturbations and leads to more stable training performance.

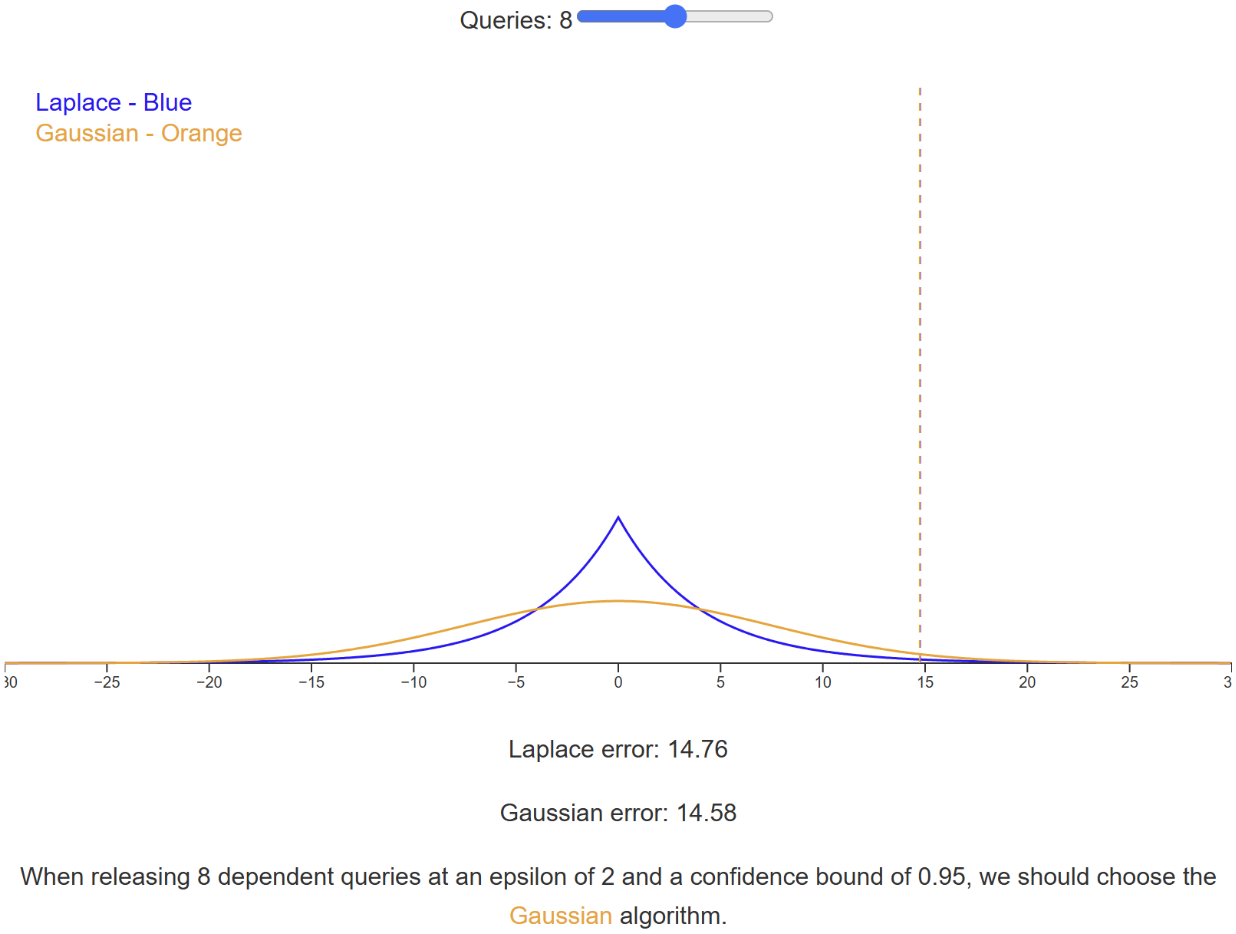


Laplace error: 12.91

Gaussian error: 13.77

When releasing 7 dependent queries at an epsilon of 2 and a confidence bound of 0.95, we should choose the Laplace algorithm.

When the 7 dependent queries, we use Laplace



Queries: 8

Laplace error: 14.76

Gaussian error: 14.58

When releasing 8 dependent queries at an epsilon of 2 and a confidence bound of 0.95, we should choose the Gaussian algorithm.

When the 7 dependent queries, we use Gaussian

Since each query operation in differential privacy requires the model to inject noise into the output, an increasing number of queries leads to a cumulative degradation in accuracy.
As illustrated in the figure, after 8 rounds of noise addition, the Gaussian mechanism results in a lower error (14.58) compared to the Laplace mechanism (14.76), indicating that Gaussian noise provides better stability and precision in multi-query scenarios.

If Laplace noise is used in DP-SGD, due to the heavy-tailed nature of Laplace, the gradient updates in each round of training will fluctuate more violently, and these violent disturbances will quickly accumulate errors in high-dimensional space, causing the model accuracy to drop rapidly or even fail to converge.

Laplace noise applied for pure DP (Differential Privacy):

$$\boxed{M(d) = f(d) + \mathrm{Lap}(\frac{s}{\epsilon})}$$

Where the
ϵ is the privacy budget, smaller **ϵ** more protection, but stronger noise
s is the sensitivity, how much will changed after any datapoint been changed in the dataset
s/ϵ is the laplace noise parament reference, to control how much noise will be.
OutputM(d) = True(f(d)) + Noise(Lap)

Gaussian noise applied for DP-SGD:

$$\boxed{M(d) = f(d) + \frac{s * \sqrt{2 * \log \frac{1.25}{\delta}}}{\epsilon}}$$

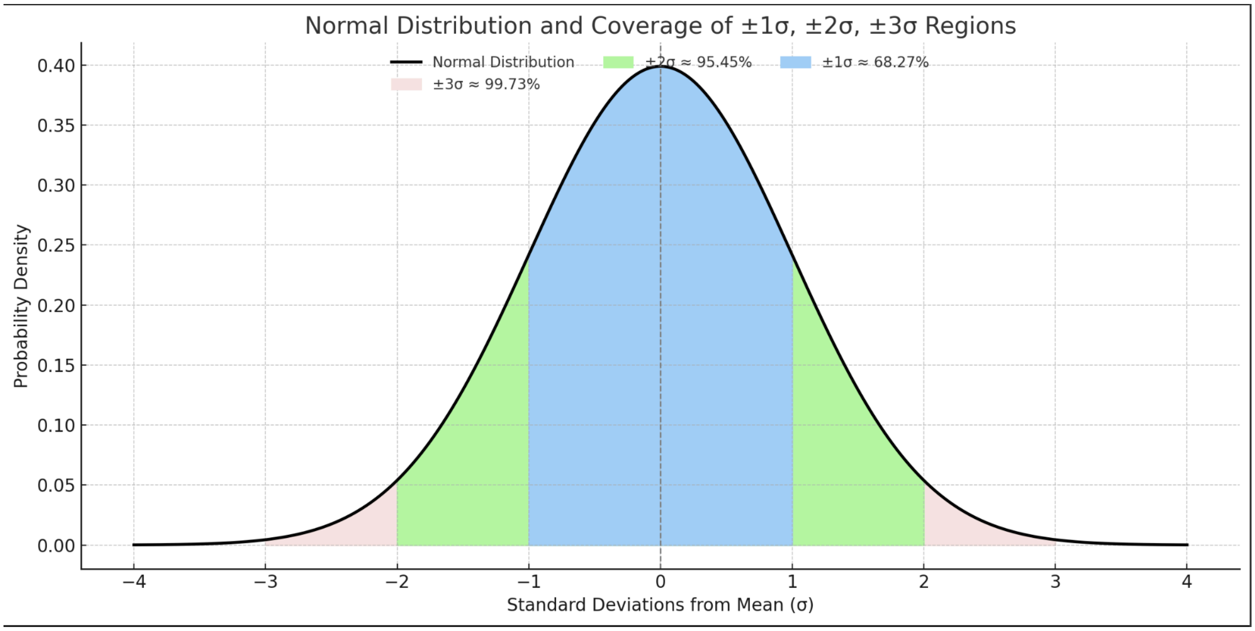
Gaussian using **(ε, δ)** to ensure the differential privacy
Where the
f(d) is the true label.
s is the sensitive, this parameter will decide how many noises will be added to “cover” this individual.
ϵ is the privacy budget that smaller means stronger protection, noises will be stronger
δ is the failure probability that allows a small probability of occasionally failing to preserve privacy, usually set to 1/n or less.
log(1.25/δ) This logarithmic term will become larger as δ becomes smaller, which means that the smaller δ is, the stronger the privacy is, but the greater the noise is.
Sqrt{2log(1.25/δ)} The term that controls the amplification of the noise standard deviation (the core factor of the Gaussian mechanism).

OutputM(d) = True(f(d)) + Noise(Gaus)

“The square root term $\sqrt{2 * \log \frac{1.25}{\delta}}$ determines how much Gaussian noise needs to be added in order to achieve a target failure probability δ. The smaller the δ, the more noise is required.”

Example (maybe):
s = 1, every individual samples can only affect 1 unit
ϵ = 1, medium strength privacy
δ =10^-5: strong privacy, low probabilities failure
We got **σ** =~ 4.84
N(μ,σ^2) normal distribution
μ is 0 since the noise will be turbate from the 0 (true label)
Pick one number from N(0,4.84^2) randomly, and add to the true label, such as 3.0.
If the true label is 20, then the output will be 23.

Normal Distribution and Coverage of ±1σ, ±2σ and ±3σ. (We only need 1 σ)



Where

±1σ x +- 1 density y high,   area cover 68.27%
±2σ x +- 2 density y medium, area cover 95.00%
±3σ x +- 3 density y low,   area cover 99.73%

The most of perturbation on 68.27%, ±1σ, x +-1
The larger σ is, the wider the distribution is and the stronger the disturbance is.

Laplace = "I always keep my answers very vague and never reveal extra information" → safe but difficult to communicate

Gaussian = "I am vague most of the time, but sometimes I say a little more" → more efficient communication, controllable risks

Reference

Loss

https://www.datacamp.com/tutorial/loss-function-in-machine-learning
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html
https://docs.pytorch.org/docs/stable/generated/torch.nn.BCELoss.html
https://arxiv.org/html/2310.13786v4#:~:text=to%20the%20data%20distribution%20is,Bousquet%20and%20Elisseeff%2C%202002

Noises

https://arxiv.org/html/2408.07021v2
https://lpanavas.github.io/mechanism-comparison/
https://pmc.ncbi.nlm.nih.gov/articles/PMC8662814/#:~:text=Note%20that%20the%20Laplace%20distribution,drawn%20from%20the%20Gaussian%20distribution