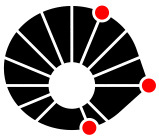

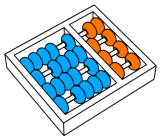

Impacto da Privacidade Diferencial em técnicas de medição de distância

Bruno Henrique Emidio Leite *Luiz Fernando Bittencourt*
Filipe Maciel Roberto

Relatório Técnico - IC-PFG-24-04
Projeto Final de Graduação
2024 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Impacto da Privacidade Diferencial em técnicas de medição de distância

Bruno Henrique Emidio Leite Luiz Fernando Bittencourt*
Filipe Maciel Roberto*

Resumo

Nos últimos anos, diversos algoritmos e métricas foram desenvolvidos para definir de maneira eficiente a distância entre diferentes grupos de dados, técnicas geralmente aplicadas em algoritmos de clusterização de usuários. Paralelamente, existe também a crescente preocupação em formas de garantir a segurança dos dados recebidos de um usuário, de forma que sua identidade não seja comprometida e que suas informações não sejam reproduzidas por terceiros. Nesse contexto, este trabalho busca analisar como diferentes técnicas de segurança impactam em métricas de distância comumente utilizadas na atualidade, aplicando distintos graus de privacidade sobre elas, a fim de verificar a possibilidade da implementação de privacidade em paralelo a aplicação dessas métricas. Dessa forma, utilizando bibliotecas de simulação, como o *Flower*, e conjuntos de dados mais simples, foi possível observar que as métricas possuem comportamentos bastante sensíveis a aplicação de privacidade, mas que ainda permitem a utilização de graus menos severos dessa.

1 Introdução

O aprendizado federado, campo relativamente recente do aprendizado de máquina, tornou-se uma alternativa promissora para o treinamento de modelos que consigam ser aplicados sobre dados mais diversos e únicos, um dos principais pontos de limitação dos modelos treinados em aprendizado centralizado. A possibilidade de conseguir acessar dados de diferentes fontes de maneira indireta, sem a necessidade de transferi-los para uma localização central, faz com que essa área se torne uma opção vantajosa no aspecto comercial e político em comparação à abordagem centralizada, resultando em um acelerado desenvolvimento do campo nos recentes anos, mesmo com seus desafios e empecilhos únicos. [1]

Um dos principais desafios observados com o progresso da área é o resultado de baixa qualidade obtido quando os clientes possuem distribuição de dados muito desiguais. Uma solução então muito utilizada foi o agrupamento, ou *clustering*, dos clientes participantes próximos a fim de gerar modelos diferentes que consigam atender as distribuições locais de cada *cluster* e, por fim, obter melhores resultados. A distância entre clientes geralmente é medida com alguma métrica matemática simples, sendo o mais comum a similaridade

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

de cossenos, mas outras métricas como *Simhash* ou distância de Hammond também são amplamente utilizadas [2].

Por outro lado, apesar dos diversos avanços, pesquisas e melhorias sobre o aprendizado federado, ele ainda apresenta alguns pontos pouco explorados em sua implementação, principalmente em questões relacionadas à segurança. Apesar de não haver necessidade de enviar diretamente os dados de um cliente para um servidor central, o envio de gradientes e pesos já representam um risco de segurança importante, sendo possível, através de engenharia reversa, reconstruir os modelos dos usuários e descobrir informações sensíveis que podem acabar comprometendo suas identidades. Além disso, o mesmo pode ser feito com o modelo global gerado no treinamento, em que utilizando de seus parâmetros, pode ser feita a inferência de parte do conjunto de dados utilizados durante o treino, adicionando mais uma camada de insegurança no sistema como um todo.

Em paralelo ao desenvolvimento do aprendizado federado, a privacidade diferencial, termo cunhado por Cynthia Dwork em 2006 em seu artigo *Differential Privacy* [3], também ganhou destaque nos últimos anos. Ela consiste em algoritmos aplicados sobre dados de usuários a serem compartilhados que busca retirar informações sensíveis sobre a identidade dos indivíduos que compõem tais dados e fornecer apenas os padrões gerais das informações contidas por eles, a fim de evitar que os usuários sejam completamente expostos para fontes externas. Tais algoritmos podem ser aplicados em diversos contextos e, recentemente, com o avanço do campo de aprendizado de máquina, se tornaram um foco importante para a garantia da segurança dos dados recebidos durante o treinamento de diferentes modelos [4].

Assim, observando a lacuna ainda pouco explorada da aplicação de técnicas de segurança no aprendizado federado e relacionando os três temas apresentados acima, o projeto busca explorar a possibilidade da utilização da privacidade diferencial nas métricas de comparação de clientes, a fim de analisar os impactos sobre essas técnicas e definir limites na relação entre segurança e funcionalidade dentro desses algoritmos. Para isso, múltiplos sistemas de privacidade diferencial são construídos, assim como diferentes métricas são calculadas, a fim de obter resultados mais abrangentes para as mais diversificadas aplicações do aprendizado federado e discussões mais significativas sobre os impactos observados.

Este documento segue a seguinte estrutura: a Seção 2 apresenta os conceitos chaves sobre o qual o trabalho é desenvolvido, junto de trabalhos relacionados; a Seção 3 apresenta a metodologia aplicada para a realização dos testes; a Seção 4 apresenta a discussão dos resultados obtidos através desses testes e a Seção 5 mostra potenciais temas futuros que podem ser abordados com base no discutido durante o projeto.

2 Referencial Teórico

Esta seção tem como objetivo expandir os conceitos abordados durante o projeto e também apresentar artigos e outros materiais que foram relevantes durante o desenvolvimento das metodologias aplicadas no trabalho.

2.1 Privacidade Diferencial

A privacidade diferencial, pode ser definida como algoritmos que buscam compartilhar dados de um usuário ou grupo de usuários sem comprometer suas identidades e privacidade. É muitas vezes colocada como uma definição matemática de privacidade. A ideia principal é que colocar ou retirar um registro de um indivíduo no banco de dados não deve afetar os resultados finais obtidos sobre operações de análise sobre ele, ou seja, os dados de um indivíduo não conseguem ser identificados quando presentes nos dados analisados, garantindo então sua privacidade.

Um conceito importante da privacidade diferencial é o "orçamento de privacidade", geralmente representado pela letra ε, que se refere a quanto o resultado de uma operação pode variar de acordo com a inserção ou retirada de uma entrada de um indivíduo no conjunto de dados analisado. Em termos gerais, ele significa a quantidade de informação individual que pode ser exposta pelo algoritmo sem representar um risco, ou seja, quanto menor seu valor mais seguro é o algoritmo. Apesar de valores de ε próximos de zero serem desejados, tais valores podem fazer com que o sistema em que a privacidade diferencial está sendo aplicado acabe perdendo muita funcionalidade, e por isso nem sempre são utilizados.

Esses algoritmos podem ser aplicados em vários contextos de manipulação de dados e diversos métodos foram desenvolvidos de acordo com o grupo de dados trabalhado e o tipo de computação sendo realizada sobre eles. No campo de aprendizado de máquina, eles geralmente são divididos de acordo com o momento do treinamento em que são aplicados:

- antes, diretamente sobre os dados recebidos pelo modelo a ser treinado, garantindo privacidade do treinamento e sua saída;
- durante, sobre os pesos e gradientes do modelo, geralmente através de aplicação de ruídos sobre os otimizadores do treinamento
- depois, sobre as predições do modelo, aplicando ruído de acordo com alguma distribuição, como a de Laplace.

Para o desenvolvimento do trabalho, foram estudados principalmente algoritmos que atuam durante o treinamento e pós-treinamento, os quais serão expandidos abaixo.

2.1.1 Ruído sobre gradientes

Em modelos treinados com otimizadores baseado em gradientes, uma das maneiras de alcançar a privacidade diferencial é adicionando ruído sobre esses antes da atualização dos pesos. Contudo, apenas esse passo não seria suficiente. Para limitar a influência de cada exemplo sobre o gradiente calculado, é necessário cortar os gradientes de forma que sua norma tenha um valor máximo, definido pelo *clipping*. Dessa forma, cada entrada do modelo terá uma influência menor sobre os pesos finais, garantindo a privacidade.

Utilizando como exemplo o otimizador mais comumente utilizado, a Descida de Gradiente Estocástica (SGD), o pseudo-algoritmo abaixo exemplifica o funcionamento desse método.

Algorithm 1 Algoritmo de privacidade diferencial para otimizador SGD [5]

Entrada: $f(x, \theta)$ função aplicada sobre entrada do modelo
L função de perda sobre a saída
Taxa de aprendizagem α, número de interações N, *batch* B
Norma de *clipping* C, multiplicador de ruído σ
Saída: Pesos do modelo θ

$b \leftarrow 1$
while b $\leq$ N **do**
  for $i \in$B **do**
    $g_{b_i} \leftarrow \Delta_{\theta_b} L_i$ ▷ Cálculo de gradiente
    $g_{b_i} \leftarrow g_{b_i} / max\left(1, \frac{||g_{b_i}||_2}{C}\right)$ ▷ *Clipping* do gradiente
  end for
  $\overline{g_b} \leftarrow \frac{1}{B}(\sum i g_{b_i} + N(0, \sigma^2 C^2))$ ▷ Adição de ruído, pela distribuição Gaussiana
  $\theta_{b+1} \leftarrow \theta_b - \alpha \overline{g_b}$
  $b \leftarrow b + 1$
end while

Nesse caso, a adição de ruído é retirada de uma distribuição Gaussiana de variância igual a $\sigma^2 C^2$. É importante notar que nesse algoritmo não há uma relação direita entre os hiper parâmetros definidos e o ϵ final. Nesse caso, o valor de ϵ depende, além do multiplicador de ruído e da norma definidos, de parâmetros como tamanho de *batch*, número de épocas e quantidade de entradas no conjunto de dados sendo treinado. Atualmente, esse cálculo é feito utilizando de conceitos mais avançados da teoria da privacidade diferencial de Rényi (RDP), que consegue definir limites superiores para ϵ de maneira mais precisa.

Para esse tipo de ruído, valores pequenos de ϵ no geral acabam inviabilizando o treinamento e a convergência dos modelos. Em empresas de grande escala, como Facebook, Apple e Google, que coletam grande quantidades de dados de seus clientes, os valores de ϵ variam entre 2 e 16.[5]

2.1.2 Ruído sobre predições

Uma outra forma de adicionar ruído, de maneira mais simples, é após o treinamento do modelo sobre suas predições de seu conjunto de dados ou sobre alguma de suas camadas. Nesse caso, em vez de utilizar a distribuição Gaussiana para geração de ruído, se utiliza a distribuição de Laplace.

A distribuição de Laplace é uma distribuição de densidade de probabilidade de formato $f(x|\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$, onde μ define a posição em que ela está centralizada e b é o fator de escala, que define quão concentrada a distribuição está. Um conceito importante relacionado a essa distribuição é o de sensibilidade Δf de uma função. Para uma função f aplicada sobre *datasets* diferentes D1 e D2, temos:

$$\Delta f = max||f(D_1) - f(D_2)||$$

Ou seja, é a diferença máxima na saída da função ao ser aplicadas em conjuntos de dados diferentes.

Utilizando μ igual a 0, um ruído aleatório gerado por essa distribuição com escala $\frac{\Delta f}{\epsilon}$ garante uma privacidade diferencial de valor ϵ, ou seja, quanto menor a escala, maior a privacidade gerada pelo ruído.

2.2 Medições de distância

No aprendizado de máquina, o agrupamento de modelos que atuam sobre conjuntos de dados semelhantes, ou seja, cujas imagens, áudios ou textos possuem características próximas num geral, pode ser uma ação interessante para agilizar o processo de treinamento ou avaliar a disposição de dados entre diferentes clientes no aprendizado federado, por exemplo. Para tal fim, primeiro é necessário encontrar a distância entre esses conjuntos, cálculo que pode ser feito através de diferentes equações utilizando algumas das características diretas dos dados ou dos modelos treinados sobre eles.

2.2.1 Similaridade de Cosseno

Dado dois vetores A e B de dimensão n, com um ângulo θ entre eles. A similaridade de cossenos é dada por:

$$SimCos = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||\,||\mathbf{B}||} = cos(\theta)$$

De maneira intuitiva, assim como o comportamento do cosseno de θ, dois vetores paralelos possuem uma similaridade de valor igual a 1 e vetores opostos possuem similaridade -1. Essa similaridade possui uma propriedade útil por não depender do tamanho dos vetores comparados, mas apenas do ângulo entre eles.

A similaridade de cossenos é um conceito bastante aplicado no campo da ciência de dados como uma maneira rápida e eficaz de determinar o quão próximo dois pontos de um conjunto de dados são. No aprendizado de máquina, dois modelos podem ser comparados utilizando o conceito de *embedding*, que são representações matemáticas, em vetores, das características de uma entrada ao ser processada por um modelo. Eles tem como objetivo transformar uma entrada, geralmente um objeto de alta dimensionalidade, em um vetor de baixa dimensionalidade, facilitando e agilizando aplicações de conceitos matemáticos sobre ela.

Apesar do predomínio na utilização desse cálculo nos últimos anos para determinar quão próximo dois *embeddings* são, recentemente um estudo publicado pela *Netflix*, *"Is Cosine-Similarity of Embeddings Really About Similarity?"*, mostra que essa técnica é bastante suscetível à transformações lineares aplicadas pelos modelos, as quais podem fazer com que o resultado obtido não represente a real proximidade entre dois *embeddings*. Uma das maneiras sugerida para mitigar esse impacto é através da normalização das camadas do modelo, técnica aplicada nesse trabalho. [6]

2.2.2 Divergência de Jensen-Shannon

Dado duas distribuições probabilísticas P e Q, a divergência de Jensen-Shannon é calculada por:

$$JS(P||Q) = \frac{KL(P||M) + KL(Q||M)}{2} \qquad M = \frac{P+Q}{2}$$

Em que KL representa a divergência de Kullback-Leibler. O resultado é um valor não negativo, em que 0 representa distribuições idênticas e valores maiores distribuições mais diferentes. Diferentemente de Kullback-Leibler, esta métrica possui a vantagem de ser simétrica, ou seja, o resultado não depende de qual distribuição é utilizada como referência e como comparação. Utilizando a divergência, é também possível calcular a distância e similaridade como:

$$JS_{dist} = \sqrt{\frac{KL(P||M) + KL(Q||M)}{2}} = \sqrt{JS(P||Q)} \qquad JS_{sim} = 1 - JS_{dist}$$

obtendo resultados em um intervalo [0, 1].

Em aprendizagem de máquina, este cálculo pode ser realizado sobre, por exemplo, as frequências de cada classe de dois conjuntos de dados distintos sobre quais um modelo deve atuar, avaliando o quão próximo esses conjuntos são.

2.2.3 Distância de Levenshtein

Dada duas sequências a e b, de tamanho $|a|$ e $|b|$, respectivamente, temos:

$$lev(a,b) = \begin{cases} |a|, & \text{se } |b| = 0, \\ |b|, & \text{se } |a| = 0, \\ lev(tail(a), tail(b)) & \text{se } head(a) = head(b), \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & \text{caso contrário} \end{cases} \tag{1}$$

Em que $head(a)$ é o primeiro elemento da sequência a e $tail(a)$ é a subsequência obtida ao retirar o primeiro elemento.

De maneira geral, essa distância pode ser compreendida com o número necessário de operações de inserção, deleção ou substituição de um caractere para igualar as duas sequências. No campo de aprendizado de máquina, é utilizada principalmente para quantificar a diferença entre dois documentos de texto, para encontrar ocorrência entre cadeia de strings semelhantes. Contudo, essa ideia pode ser expandida para sequência de números, como em uma lista, noção utilizada nesse trabalho.

2.2.4 Centered Kernel Alignment (CKA)

Dado duas matrizes de representações de características K e L, o CKA entre elas pode ser calculado como:

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K)HSIC(L, L)}} \qquad HSIC(K, L) = \frac{1}{(n-1)^2} tr(KHLH)$$

ou seja, trata-se da normalização do critério de independência de Hilbert-Schmidt (HSIC). O CKA possui a vantagem de ser invariante à transformações isotrópicas sobre as matrizes analisadas, tornando-se uma métrica mais efetiva em encontrar similaridades em comparação com a análise de correlação canônica (CCA), também muito utilizada. No aprendizado de máquina, esse cálculo pode ser utilizado sobre camadas de modelos distintos para identificar a proximidade entre eles. [7]

3 Metodologia

Como visto acima, há diversas métricas de distâncias matemáticas que podem ser utilizadas no contexto de aprendizado de máquina para quantificar a proximidade entre dois modelos. Em cenários reais, esses cálculos fazem bastante sentido quando se necessita comparar um número grande de modelos diferentes, assim como ocorre em um aprendizado federado. Para esse fim, o primeiro passo foi simular um aprendizado federado em menor escala com ajuda de uma biblioteca chamada *Flower*.

3.1 Flower Framework

Com início em 2020, o *framework* Flower surge como uma biblioteca para facilitar e agilizar a simulação de um aprendizado federado sobre um número significativo de clientes. Sua API fornece métodos de alto nível para que os usuários consigam construir uma arquitetura federada sem se preocupar com os diversas implementações técnicas de comunicação e particionamento de recursos que ocorrem nos bastidores. Além disso, a biblioteca também permite um alto nível de customização sobre a funcionalidade do aprendizado federado de acordo com as necessidades do usuário, ao mesmo tempo que agiliza o treinamento através do suporte à execução de processos paralelos durante a simulação.

3.2 Base de Dados

O primeiro passo para a simulação do aprendizado federado foi a escolha de uma base de dados eficiente e suficientemente simples. Atendendo esses critérios, foram escolhidas duas bases bastante conhecidas e utilizadas na literatura. A primeira foi a MNIST, uma base contendo 60000 imagens de teste de dígitos manuscritos de 0 a 9, ou seja, 10 classes. As imagens possuem tamanho de 28x28 *pixels* e estão no padrão preto e branco, permitindo que modelos mais simples consigam treinar e obter valores altos de acurácia.

O segundo banco de dados escolhido foi o CIFAR-10, o qual também consiste em 60000 imagens divididas em 10 classes. As classes identificam o animal ou objeto retratado em

cada imagem, como por exemplo pássaro, cachorro, avião e carro. Essa base é um pouco mais complexa quando comparada com a anterior, com imagens de dimensões de 32x32 *pixels* e coloridas, sendo escolhida para se aproximar um pouco mais de casos reais de aplicação de aprendizado federado.

3.3 Simulações

Para todas simulações utilizadas, foi utilizado o algoritmo de agregação de média federada, conhecido também como *FedAvg*. Nesse algoritmo, quando os clientes enviam seus pesos após treinamento para o servidor, o peso final do modelo é calculado através da média ponderada dos dados recebidos no servidor, em que os pesos são os números de dados de treinamento de cada cliente. Além disso, para configurar a simulação, são necessários definir alguns hiper parâmetros importantes para o *framework*. Dentre aqueles utilizados, os mais importantes são explicados abaixo:

- **num_clients**: número de clientes que participam do aprendizado federado
- **num_rounds**: número de rodadas que a simulação é executada
- **min_fit_clients**: número mínimo de clientes que são escolhidos para o treinamento e envio de pesos ao servidor por rodada
- **min_evaluate_clients**: número mínimo de clientes escolhidos para avaliar o modelo final obtido na rodada

Além disso, os métodos de seleção de clientes por rodada e de agregação estabelecidos pelo *FedAvg* padrão do *framework* também são modificados de acordo com o cenário sendo testado.

Dois sistemas de simulações diferentes foram construídos de acordo com as métricas a serem testadas. Os parâmetros dessas simulações e o comportamento específico delas são especificados abaixo. É importante notar que os testes, em ambos os casos, ocorreram para dois cenários possíveis de distribuição de dados entre clientes. O primeiro assume uma distribuição independentes e identicamente distribuídas (IID), enquanto no segundo é realizado um particionamento de Dirichlet sobre o conjunto de dados antes de alocá-los aos clientes, no qual as frequências de cada classe dos clientes são retirados da distribuição de Dirichlet, gerando distribuições mais variadas e aleatórias a fim de verificar o impacto da privacidade em um cenário mais próximo do real. [8]

3.4 Simulações com *Embeddings*

Para a primeira simulação, foi montada uma configuração para testar métricas sobre *embeddings*. Nesse caso, o foco do teste não é o treinamento federado em si, mas o impacto sobre os cálculos realizados pelo servidor ao receber métricas de seus clientes que passaram por um algoritmo de privacidade diferencial.

Nessa primeira configuração, foi utilizado a base de dados CIFAR-10 e um modelo composto por camadas convolucionais simples. Nesse modelo, foram incluídas camadas de

normalização com o objetivo de diminuir o impacto das transformações sobre a métrica de similaridade de cosseno. O objetivo foi treinar rapidamente os modelos para que os *embeddings* gerado por eles conseguissem da melhor maneira identificar as características presentes nas imagens processadas. Para isso, o aprendizado constava com 100 clientes e 1 rodada, em que cada cliente treinava seu modelo por 30 épocas locais. O treinamento foi realizado com um otimizador Adam, com taxa de aprendizado de 1×10^{-3}, betas de 0.9 e 0.999 e ϵ de 1×10^{-7}.

No momento de treinamento, todos 100 clientes eram escolhidos e participavam do aprendizado. Ao terminar o treinamento, no momento de enviar seus pesos para o servidor para a agregação, era enviado os *embeddings* obtidos na saída da penúltima camada do modelo com a adição de um ruído que garanta uma privacidade diferencial de valor ϵ. Esse processo foi feito com a distribuição de Laplace, como mostrado acima. De maneira semelhante, também foi enviado as contagens do número de classes contidos por cada cliente, nas quais também são aplicadas um ruído de Laplace. Por fim, no momento de agregação, o servidor calcula as métricas de similaridade de cossenos sobre os *embeddings* e divergência de Jensen-Shannon sobre os distribuições probabilísticas de cada cliente.

3.5 Simulações com Treinamento Privado

A segunda simulação visa testar a aplicação da privacidade diferencial durante o treinamento do aprendizado federado. Para isso, foi utilizado uma configuração mais habitual, com 100 rodadas de treinamento com 1 época local cada. O servidor escolhe 10 clientes para realizar o treinamento por rodada e 5 clientes clientes para avaliar o modelo gerado em seu conjunto de dados.

Para o conjunto de dados, foi utilizado o MNIST e um modelo de aprendizado mais simples, composto por menos camadas. Nas rodadas 1, 33, 67 e 100, o servidor escolhe todos clientes para realizar o treinamento para que assim ele também calcule o CKA entre clientes nestes momentos. A camada escolhida para ser comparada com o CKA foi a penúltima, uma camada densa com ativação ReLu, *rectified linear unit*. O objetivo é avaliar a evolução do CKA durante o treinamento e compará-la com o resultado apos ser aplicado determinados graus de privacidade, verificando se o treinamento privado consegue alcançar os mesmos resultados da configuração não-privada e, se sim, avaliando o número de rodadas para que isso ocorra.

Para determinar os hiper parâmetros do otimizador privado, foram seguidos os passos sugeridos em [5]. Primeiro, é realizado um treinamento sem privacidade para determinar valores de parâmetros como taxa de aprendizado e *momentum*. Depois, é colocado o otimizador com privacidade sem adicionar um ruído e o *clipping* da norma é variado até encontrar um valor que não altere muito o comportamento do treinamento, começando por potências de 10, processo denominado *ClipSearch* [5]. No caso específico, o valor obtido foi 8. Por fim, adiciona-se um valor de multiplicador de ruído para alcançar o ϵ desejado, o qual pode ser calculado com algoritmos que utilizam conceitos da privacidade diferencial de Rényi, fornecendo parâmetros de simulação mostrados em 2.1.1.

4 Resultados e discussões

Os resultados obtidos para cada um dos experimentos são exibidos abaixo, em formato de gráficos.

4.1 *Embeddings*

A partir das métricas de similaridade de cosseno e de Jensen-Shannon, foram calculados os valores médios de ambas para diferentes valores de ϵ a fim de verificar o comportamento delas em diferentes graus de privacidade. Além disso, os *embeddings* obtidos sob diferentes valores de ϵ foram comparados com os *embeddings* originais através da similaridade de cosseno, para analisar o impacto da privacidade nos resultados obtidos. Por fim, utilizando as métricas de Jensen-Shannon, com o objetivo de avaliar o impacto na ordenação de proximidade entre clientes, foi calculado a distância de Levenshtein entre a ordenação com privacidade e sem privacidade. Para isso, para cada cliente, foram obtidos os índices que ordenam o vetor contendo a similaridade de Jensen-Shannon dele em relação a todos outros clientes participantes. Após realizar esse processo para todos clientes, utiliza-se o algoritmo de Levenshtein entre os índices obtidos de um mesmo cliente antes da aplicação de ruído e após a aplicação, de forma a comparar o quão distante a ordenação torna-se.

Todos essas cálculos foram realizados para diferentes valores de sensitividade, para também avaliar o impacto que uma escolha inadequada desse valor pode ter sobre os resultados obtidos. Os valores de ϵ utilizados foram $\epsilon = \{0.1, 0.5, 1, 5, 10\}$ e de sensitividade $\Delta f = \{0.1, 1, 50, 100\}$. É importante notar que, para fins de comparação, nos gráficos com as médias das métricas, o valor da similaridade de cossenos está normalizado, para ficar na mesma escala da similaridade de Jensen-Shannon.

4.1.1 Distribuição idêntica

Na Figura 1, é possível observar que, para uma sensitividade muito pequena, os resultados quase não são afetados pela aplicação de privacidade. Mesmo um valor muito rigoroso de ϵ, de 0.1, faz com que a média das métricas tenham uma queda pequena e os *embeddings* mudem muito pouco.

Para uma sensitividade de 1, visto na Figura 2, já é possível notar uma diferença maior. Há uma queda significativa na semelhança para valores muito pequenos de ϵ, mas que voltam a se aproximar do original rapidamente. O mesmo ocorre com os *embeddings*, de maneira mais drástica.

No caso da sensitividade 50, na Figura 3, a mais próxima da sensitividade real de ambas funções sobre as quais o ruído é aplicado, o comportamento visto anteriormente é ainda mais acentuado, principalmente para a similaridade de cossenos, que começa a se aproximar do valor original em valores de ϵ maiores, enquanto a semelhança de Jensen-Shannon se recupera mais rapidamente. Os *embeddings* se tornam mais afastados dos *embeddings* originais e, mesmo para privacidades menos severas, ainda apresentam um impacto perceptível.

Por fim, para uma sensitividade muito maior, como 100 na Figura 4, a similaridade de cossenos é drasticamente afetada por qualquer valor de ϵ, não se recuperando mesmo com

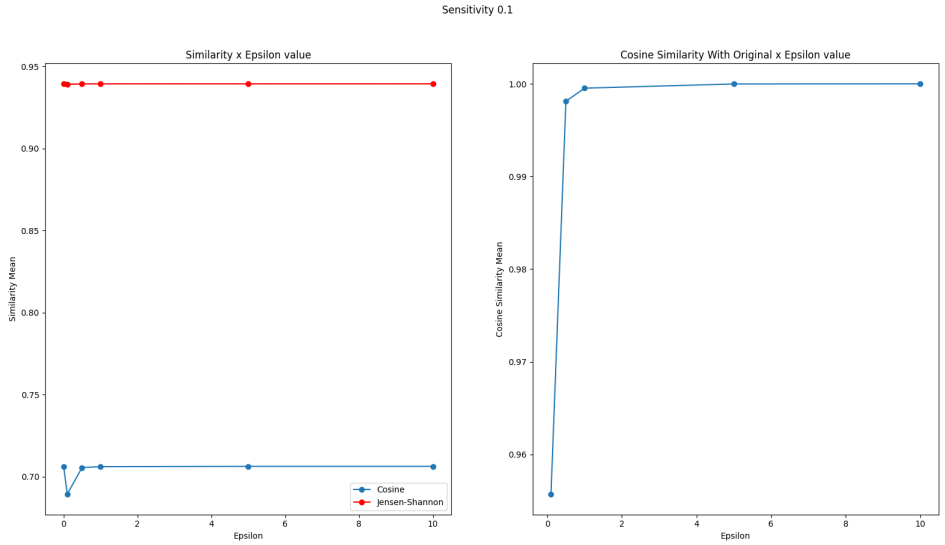

Figura 1: Média das métricas de similaridade e similaridade dos *embeddings* com os originais para uma sensitividade de 0.1

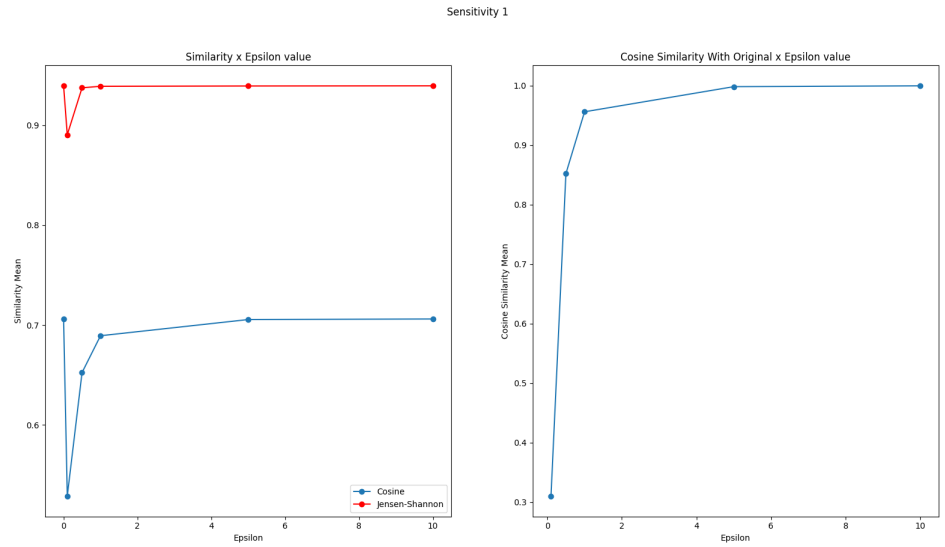

Figura 2: Média das métricas de similaridade e similaridade dos *embeddings* com os originais para uma sensitividade de 1

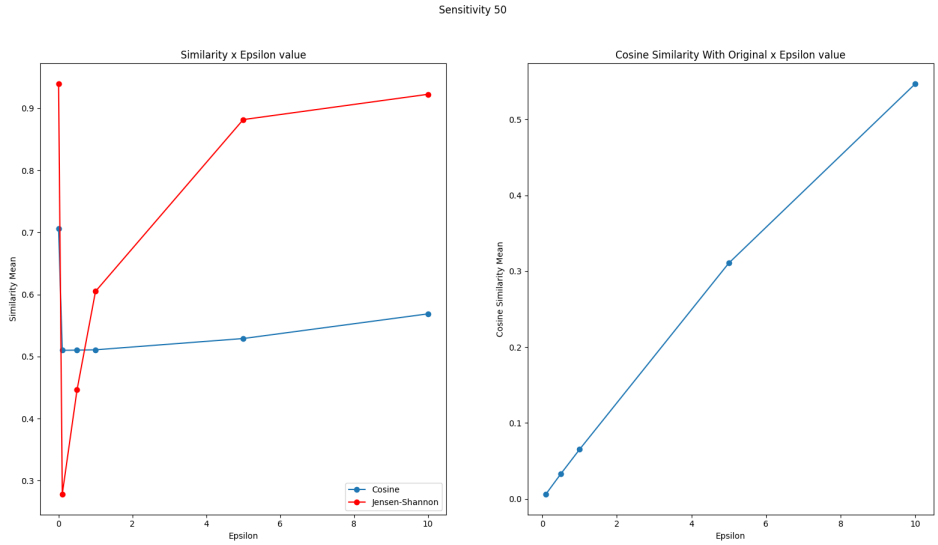

Figura 3: Média das métricas de similaridade e similaridade dos *embeddings* com os originais para uma sensitividade de 50

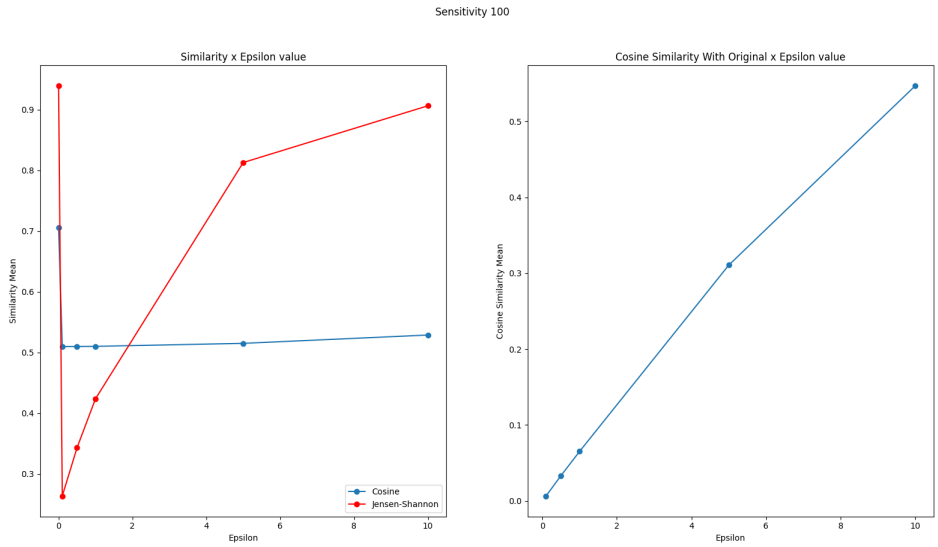

Figura 4: Média das métricas de similaridade e similaridade dos *embeddings* com os originais para uma sensitividade de 100

privacidades pouco rigorosas e sendo consideradas pouco semelhantes no geral, com valores próximos de 0.5. A similaridade de Jensen-Shannon ainda se mostra robusta, apesar de precisar de uma privacidade mais baixa para se aproximar do original. Os *embeddings* também se tornam muito distantes dos originais, mostrando a importância da escolha correta para o valor de sensitividade.

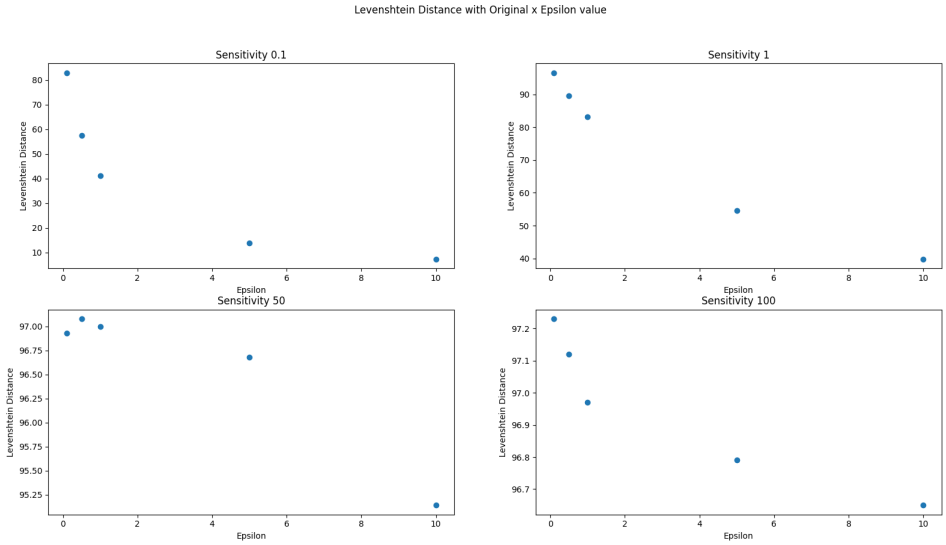

Figura 5: Distância de Levenshtein com ordenação original para diferentes sensitividades

Realizando os testes com a distância de Levenshtein, na Figura 4, em um primeiro momento, a métrica parece ter uma performance razoável, principalmente em sensitividades baixas, onde para valores de ϵ mais altos, ela consegue se aproximar da ordenação original. Contudo, como a sensitividade real da função de contagem de classes sobre o qual o ruído foi aplicada está próximo de 50, é possível perceber que a distância passa a ter um desempenho muito baixo, com valores muito altos e próximos do máximo de 100 operações de mudanças para transformar uma ordenação em outra.

Dessa forma, uma hipótese levantada foi a de que, como o particionamento do banco foi feito de forma IID, os clientes obtém resultados muito próximos, o que os torna sensíveis a mudanças mínimas em seus valores, afetando muito a ordenação. Para testar isso, foi realizado uma partição mais aleatória, como visto abaixo.

4.1.2 Particionamento aleatório

Utilizando o particionamento de Dirichlet com valor de *alpha* de 0.5, foi realizado o mesmo experimento com a distância de Levenshtein para testar a performance da função em casos reais e verificar a hipótese levantada anteriormente. As condições de treinamento foram mantidas para a configuração IID.

O resultado obtido foi bastante diferente do esperado. Para qualquer valor de ϵ e sensitividade, a distância de Levenshtein fica muito próxima da máxima e se comporta de maneira

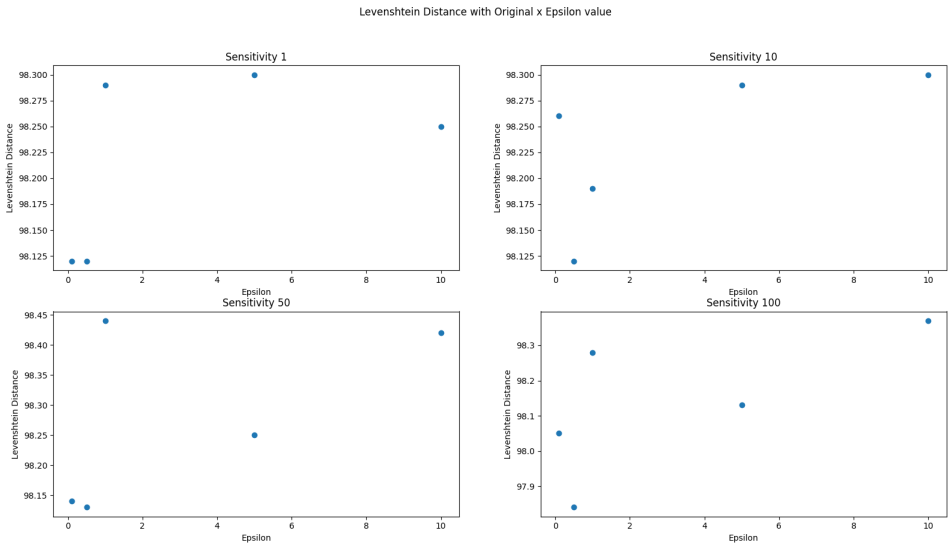

Figura 6: Distância de Levenshtein com ordenação original para diferentes sensitividades

quase aleatória. Além disso, é importante notar que os valores de sensitividade são bastante baixos em comparação ao real, que é próximo de 900, ou seja, a métrica se mostra bastante sensível a qualquer adição de ruído e não parece muito adequada para comparar clientes.

4.2 Treinamento Privado

Após definido o valor de clipping com o algoritmo *ClipSearch*, diferentes valores de multiplicadores de ruídos foram testados empiricamente para obter diferentes valores de ϵ. Dessa forma, para os ruídos selecionados de $\delta = \{0.35, 0.4, 0.6, 1.1, 1.8\}$, são obtidas privacidades de $\epsilon = \{9.09, 6.1, 1.66, 0.19, 0.02\}$. Além dos valores de CKAs entre clientes medidos nas rodadas 1, 33, 67 e 100, após as simulações também são calculados os valores de distância euclidiana média entre os valores de CKAs do treinamento sem privacidade e com privacidade. Assim como realizado com a ordenação de Jensen-Shannon, a distância de Levenshtein também foi calculada em comparação com o resultado original utilizando o CKA.

4.2.1 Distribuição idêntica

É possível observar na Figura 7 que, assim como esperado, valores de ϵ mais altos resultam em uma distância menor em comparação ao treinamento original. Além disso, conforme a simulação avança para os últimos rounds, a distância para todo valor de ϵ diminui consideravelmente. Considerando que a distância máxima euclidiana máxima entre dois vetores de CKA de 100 elementos deve ser 10, considerando que os valores variam entre 0 e 1, é possível observar que para valores de ϵ maiores que 1, já é obtido uma proximidade bem alta em relação aos vetores originais em todos rounds, o que indica que valores altos de privacidade ainda podem ser utilizados sem impactar tanto no funcionamento do algoritmo.

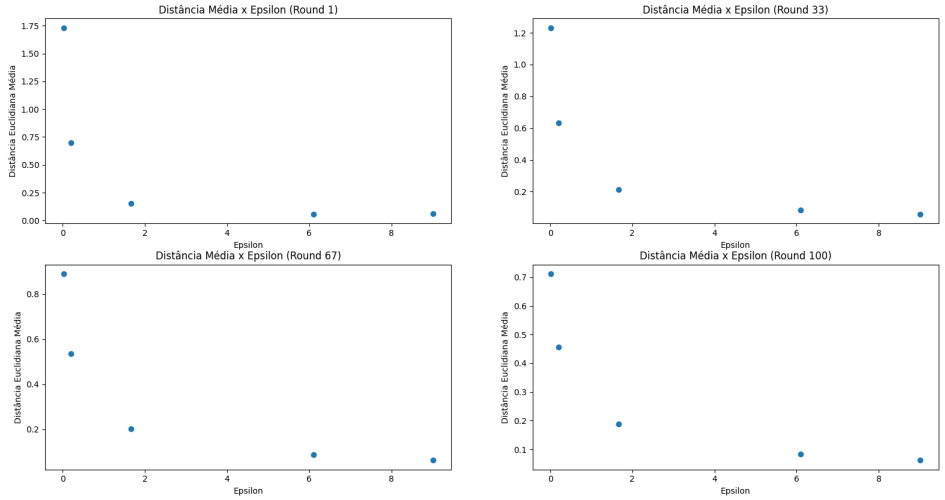

Figura 7: Distância euclidiana média dos CKAs com treinamento sem privacidade em cada rodada de análise

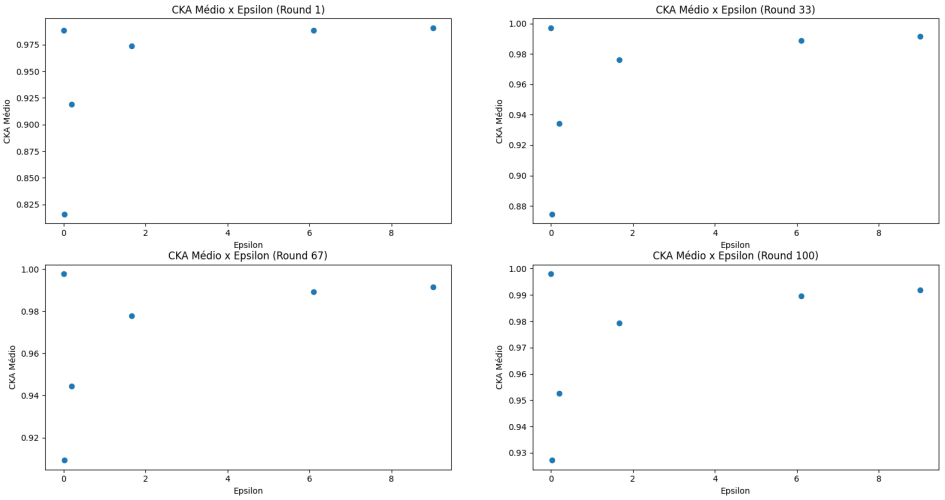

Figura 8: CKA médio em cada rodada de análise em relação ao ϵ

Similarmente, na Figura 8, onde é mostrado a evolução do CKA médio entre os clientes de acordo com ϵ e as rodadas, também se obtém um comportamento bastante próximo do esperado, em que valores mais altos de ϵ, de menor privacidade, tenham uma média mais próxima do original. Nesse caso, esse desempenho é observado independente da rodada.

Assim como na distância, valores de ϵ maiores que 1 já demonstram um desempenho próximo do original e parecem poder ser utilizados sem afetar significativamente a métrica.

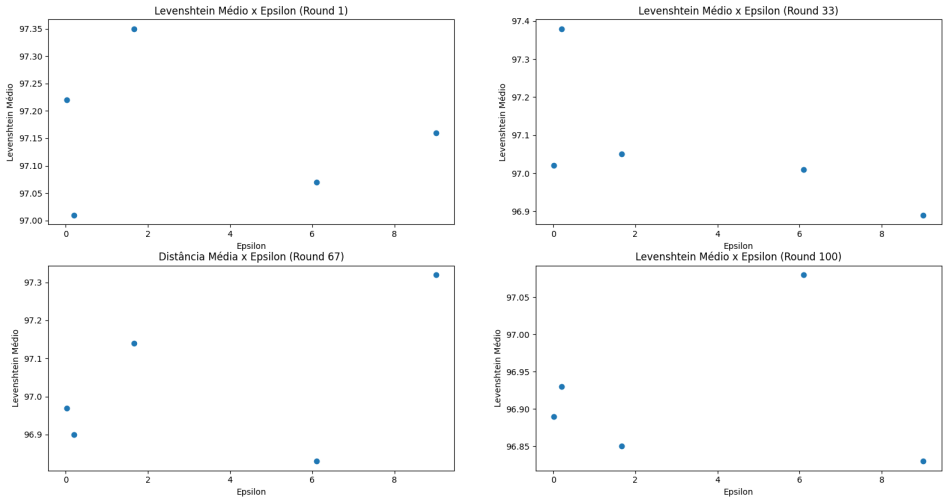

Figura 9: Distância de Levenshtein entre ordenação de clientes em relação ao original

Por fim, na Figura 9, podemos observar um comportamento bastante diferente dos anteriores. Os valores de ϵ e o número de rodadas no treinamento nada influenciam na métrica de Levenshtein, que mantém valores muito altos, próximos do número de trocas máximas de 100, em todos os casos. Além disso, não existe um padrão no desempenho da métrica. Novamente, esse fato fez com que uma distribuição diferente entre clientes fosse testada para avaliar se o comportamento de Levenshtein mudaria.

4.2.2 Distribuição aleatória

Devido as mudanças nos valores de número de dados das partições de cada cliente, os valores máximos de ϵ obtidos foram diferentes comparados com a distribuição IID. Foi escolhido o ϵ de menor privacidade, ou seja, maior valor, entre os clientes para definir o nível de privacidade de cada treinamento.

Assim como ocorre com a distância de Jensen-Shannon, o particionamento aleatório não apresenta resultados melhores, com um comportamento quase aleatório e distâncias muito altas para todos valores de ϵ em todos rounds. Dessa forma, essa métrica se mostra muito sensível à ruídos e pouco efetiva para definir distâncias.

5 Conclusões e trabalhos futuros

Utilizando diferentes abordagens de privacidade diferencial, foi possível analisar de forma efetiva seu impacto sobre algumas métricas comumente utilizadas. A similaridade de cosse-

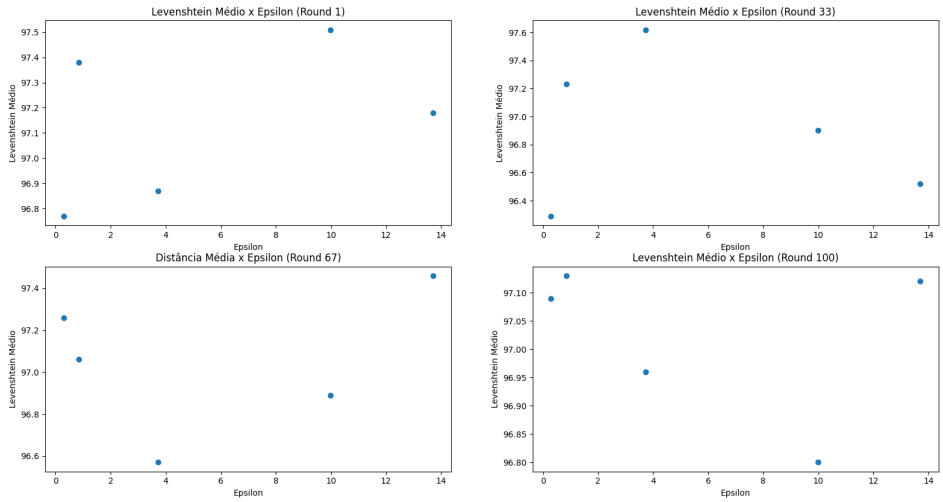

Figura 10: Distância de Levenshtein entre ordenação de clientes em relação ao original para particionamento aleatório

nos se mostra bastante afetada por privacidades mais severas e, de forma geral, apresenta diferenças bem grandes com a presença de qualquer ruído, apesar de conseguir manter ainda um determinado grau de funcionalidade. A distância de Levenshtein é a mais afetada por ruídos, de forma a apresentar um comportamento quase aleatório na presença de graus de privacidade muito baixos, se tornando ineficaz para ser aplicada junto com a privacidade diferencial.

Por outro lado, as similaridades obtidas com CKA e Jensen-Shannon se mostram bastante robustas à aplicação de privacidade, conseguindo ter um comportamento bastante similar ao original mesmo com graus de privacidade considerado altos, tornando-se opções adequadas para calcular distâncias e também respeitar a identidade e segurança dos clientes participantes.

Este trabalho pode ser expandido para verificar o impacto da privacidade em outras métricas comumente utilizadas, como por exemplo em funções de *hashing* sensíveis à localidade (LSH) ou discrepância média máxima (MMD). Além disso, também pode ser analisado os efeitos da aplicação de segurança diretamente em algoritmos de clusterização, como o *k-means* ou *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN), investigando se os *clusters* gerados possuem muita diferença com os *clusters* originais.

Dessa forma, a utilização de privacidade diferencial não precisa ser um empecilho para o cálculo de métricas de distância, sendo possível a aplicação de ambas simultaneamente sem uma perda de desempenho muito significativa. Além disso, mesmo que valores altos de ϵ signifiquem uma privacidade menos efetiva, eles podem ser aplicados para conseguir um nível mínimo de segurança e conseguir quase o mesmo desempenho das métricas comparados com um sistema sem privacidade, o que também já é uma alternativa melhor que configurações

sem nenhuma privacidade.

Referências

[1] Mammen, Priyanka Mary. "Federated learning: Opportunities and challenges."arXiv preprint arXiv:2101.05428 (2021).

[2] F. Sattler, K. -R. Müller, T. Wiegand and W. Samek, "On the Byzantine Robustness of Clustered Federated Learning,"ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 8861-8865, doi: 10.1109/ICASSP40776.2020.9054676.

[3] Dwork, Cynthia. "Differential privacy."International colloquium on automata, languages, and programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[4] Blanco-Justicia, Alberto, et al. "A critical review on the use (and misuse) of differential privacy in machine learning."ACM Computing Surveys 55.8 (2022): 1-16.

[5] Ponomareva, Natalia, et al. "How to dp-fy ml: A practical guide to machine learning with differential privacy."Journal of Artificial Intelligence Research 77 (2023): 1113-1201.

[6] Steck, Harald, Chaitanya Ekanadham, and Nathan Kallus. "Is cosine-similarity of embeddings really about similarity?."Companion Proceedings of the ACM on Web Conference 2024. 2024.

[7] Kornblith, Simon, et al. "Similarity of neural network representations revisited."International conference on machine learning. PMLR, 2019.

[8] Yurochkin, Mikhail, et al. "Bayesian nonparametric federated learning of neural networks."International conference on machine learning. PMLR, 2019.