Chapter 4

Desaprendizado de Maquinas e a LGPD: da privacidade ao direito ao esquecimento

Daniel O. C. Cota, Daniel Carlos S. de Jesus, Antonio A. de A. Rocha

Abstract

By voluntarily providing huge amounts of personal data to online services such as Facebook, Google and Amazon in exchange for useful services, users are often unaware of the great risk they are taking on. The security aspect associated with access to users' personal information has been a cause for concern, as it can pose a threat to their privacy. Against this backdrop, regulations have given users the option of removing their data from a system, such as the "right to be forgotten". But what does deleting data mean? Normally, data is not isolated in a database, it is used, for example, to train machine learning models. Deleting a user's data may be enough to prevent it from influencing the training of future models, but it doesn't eliminate the influence of the data on existing models. Faced with this challenge, the machine unlearning paradigm has emerged, consisting of algorithms/frameworks that allow selective forgetting of data, comparing the distributions of the unlearned model with the retrained one.
Keywords*: Machine Learning, Machine unlearning, privacy.*

Resumo

Ao fornecer voluntariamente enormes quantidades de dados pessoais a serviços online, como o Facebook, o Google e a Amazon, em troca de serviços úteis, os usuários muitas vezes não estão cientes do grande risco que assumem. O aspecto da segurança associado ao acesso as informações pessoais dos usuários tem sido alvo de preocupação, visto que pode representar uma ameaça a sua privacidade. Diante de tal cenário, regulamentações tem proporcionado aos usuários a possibilidade de optar pela remoção dos seus dados de um sistema, como o intitulado "o direito a serem esquecidos". Mas o que é que significa apagar dados? Normalmente, os dados não ficam isolados numa base de dados, eles são utilizados, por exemplo, para treinamento de modelos de aprendizado de máquinas. A eliminação dos dados de um usuário podem ser suficientes para impedir que

estes influenciem a formação de modelos futuros, mas não elimina a influência dos dados nos modelos existentes. Diante deste desafio, surge o paradigma do desaprendizado de máquinas, do inglês machine unlearning, que consiste em algoritmos/frameworks que permitem o esquecimento seletivo de dados, comparando a distribuições do modelo desaprendido com o retreinado.
***Palavras-chaves**: Machine Learning, Machine unlearning, privacidade.*

4.1. O que é desaprendizado?

Ao longo dos anos, a sociedade tem sido moldada pela aprendizagem, que pode ser entendida como um processo no qual o ser humano modifica suas habilidades e conhecimentos a partir de experiência direta, estudo, observação, raciocínio ou instrução. Entretanto, por inúmeros fatores é possível afirmar que nem toda informação aprendida tem um resultado positivo, sendo necessário o seu esquecimento.

O ato de desaprender que pode definido como o processo de perda ou esquecimento daquilo que aprendera, que sabia [Ferreira 1999], tem sido alvo de estudos, dentre as principais áreas é possível citar nas organizações. O entitulado desaprendizado organizacional pode ser entendido da seguinte maneira, da mesma forma que as organizações desenvolvem processos para que o aprendizado possa ocorrer e ser acumulado, também precisam buscar meios pelos quais os aprendizados passados possam ser revistos. O trabalho de [Holan and Phillips 2004] define esquecimento organizacional como "a perda, voluntária ou não, do conhecimento organizacional". Os autores ainda afirmam que desaprendizagem consiste em um tipo de esquecimento, que ocorre quando o conhecimento organizacional é intencionalmente removido [Buchele et al. 2016].

Assim como nas organizações, o processo de esquecimento intencional das informações dos indivíduos na atual "era dos dados", com o desenvolvimento da Lei Geral de Proteção dos Dados (LGPD), tem gerado um aumento das atenções para lidar com este problema, especialmente por envolver aspecto da privacidade do indivíduo. A estratégia que trata desta problemática e que será alvo deste estudo é conhecida como desaprendizagem de máquina.

4.1.1. O que é desaprendizado de máquina (DM)?

Nos últimos anos, sociedade tem presenciado o notório desenvolvimento das redes móveis, especialmente com o advento das tecnologias 5G e 6G. A evolução tecnológica e o acesso cada vez mais facilitado a dispositivos eletrônicos e à conectividade têm impulsionado o uso crescente destes aparelhos. Hoje, é comum deparar-se com pessoas de diversas faixas etárias utilizando dispositivos computadorizados e navegando na internet para trabalhar, estudar, se comunicar, realizar compras, buscar informações e se entreter. Essas atividades produzem um elevado número de dados pessoais que vão deixando rastros da Internet que podem refletir comportamentos, preferências e interações dos usuários.

A coleta de volume massivo de dados pessoais como: nome, apelido, localização, e-mail, histórico de navegação, informações sobre renda e endereço de IP tem sido utilizada para o avanço da inteligência artificial, especialmente dos modelos de aprendizado de máquinas. Essas informações disponibilizadas pelos conhecidos cookies, tecnologia que fornece "a digital do usuário", possibilitam que os algoritmos utilizados para treinar

os modelos sejam capazes de encontrar padrões de comportamento em função de características em comum dos usuários avaliados. Vale ressaltar que esses "insights" são extremamente valiosos para as grandes corporações, visto que fornecem informações importantes sobre os seus clientes.

Entretanto, o aspecto da segurança associado ao acesso as informações pessoais dos usuários tem sido alvo de preocupação, visto que pode representar uma ameaça a sua privacidade. Diante de tal cenário, os usuários podem optar pela remoção dos seus dados de um sistema, conforme prevê o artigo 17º do Regulamento Geral de Proteção de Dados (RGPD) que obriga as organizações a fornecer aos usuários "o direito a serem esquecidos", ou seja, o direito a que todos ou parte dos seus dados sejam eliminados de um sistema mediante solicitação.

Embora a remoção de dados das bases de dados back-end atenda a nova regulamentação, o mesmo não é suficiente no contexto da IA, uma vez que os modelos de aprendizado de máquinas são criados a partir da compressão dos dados de treinamento, sendo que alguns são demasiadamente adaptados ao seu treino. Além disso, há modelos mais complexos como de aprendizado profundo na qual é difícil identificar a ligação entre os dados e os parâmetros do modelo, mas é possível observar o notório impacto que tal remoção terá sobre a sua performance. Há ainda a opção de retreinar o modelo retirando os dados solicitados, entretanto o processo de retreinamento caracteriza-se por ser computacionalmente dispendioso, o que representa um entrave importante para a sua utilização [Nguyen et al. 2022].

Logo, é evidente o desafio de estabelecer uma estratégia que permite ao modelo de aprendizado de máquina “esquecer” informações previamente aprendidas através dos dados de treinamento. Este processo de remover seletivamente a influência de dados específicos nos modelos treinados é chamada de desaprendizado de máquinas (DM), conhecido pelo seu termo em inglês "machine unlearning".

4.2. Workflow do processo de DM

A Figura 4.1 apresenta o fluxo de trabalho típico de desaprendizado de máquinas, que basicamente pode ser representado por um modelo de aprendizado de máquinas acrescido da presença de um pedido de remoção de dados.

O workflow de desaprendizagem apresenta o fluxo de trabalho típico de um modelo de aprendizado de máquinas na presença de um pedido de remoção de dados. Em geral, um modelo é criado, a partir dos dados de treinamento e em seguida é utilizado para inferência. Após um pedido de remoção, os dados a serem esquecidos são retirados do modelo. O modelo desaprendido é então verificado em relação a critérios de privacidade e, se estes critérios não forem cumpridos, o modelo é treinado de novo, ou seja, se o modelo ainda deixar escapar alguma informação sobre os dados esquecidos. Existem dois componentes principais neste processo: o componente de aprendizagem (à esquerda) e o componente de desaprendizagem (à direita).

A componente de aprendizagem envolve os dados atuais, um algoritmo de aprendizagem e o modelo atual. No início, o modelo inicial é treinado a partir de todo o conjunto de dados utilizando o algoritmo de aprendizagem. A componente de desapren-

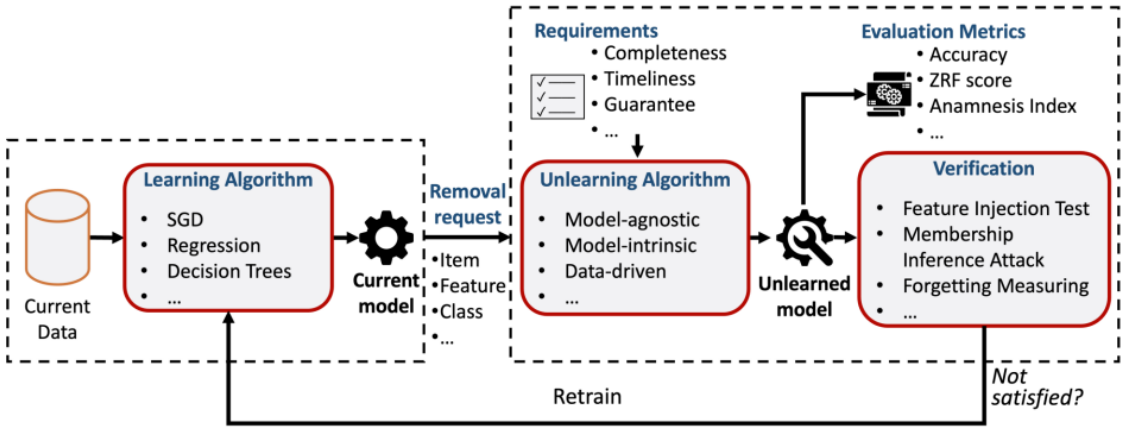

Figure 4.1. Workflow do processo de DM [Nguyen et al. 2022]

dizagem envolve um algoritmo de desaprendizagem, o modelo desaprendido, requisitos de otimização, métricas de avaliação e um mecanismo de verificação. Após um pedido de remoção de dados, o modelo atual será processado por um algoritmo de desaprendizagem para esquecer a informação correspondente a esses dados dentro do modelo. O algoritmo de desaprendizagem pode ter em conta vários requisitos, como a exaustividade, a atualidade e as garantias de privacidade. O resultado é um modelo não aprendido, que será avaliado em função de diferentes métricas de desempenho (por exemplo, completude, pontuação ZRF, índice de anamnese). No entanto, para fornecer um certificado de privacidade para o modelo não aprendido, é necessária uma verificação (ou auditoria) para provar que o modelo esqueceu efetivamente os dados solicitados e que não há fugas de informação. Esta auditoria pode incluir um teste de injeção de características, um ataque de inferência de associação ou medidas de esquecimento. Se o modelo não aprendido passar na verificação, torna-se o novo modelo para tarefas em questão, que podem ser de inferência, previsão, classificação ou recomendação. Se o modelo não passar na verificação, os dados restantes, ou seja, os dados originais excluindo os dados a serem esquecidos, precisam ser utilizados para treinar novamente o modelo. De qualquer forma, o componente de desaprendizagem será chamado repetidamente por meio de um novo pedido de remoção [Nguyen et al. 2022].

4.2.1. Definição formal de AM

O conceito de aprendizado de máquinas já está bastante consolidado na literatura. Dentre os trabalhos que abordam o tema, é possível citar o artigo de [Langley and Carbonell 1984] que o define como um domínio que busca desenvolver métodos e técnicas para automatizar a aquisição de novas informações, novas competências e novas formas de organizar as informações existentes.

A Figura 4.2 representa o diagrama do processo de aprendizado supervisionado, no qual é possível verificar que dentre do conjunto de hipóteses H, o objetivo do aprendizado supervisionadp é encontrar a melhor h, denominada hipótese final g, que de alguma forma se aproxime da função alvo f. Para tal, é necessário definir o algoritmo de aprendizagem A, que inclui a função objetivo (a função a otimizar para procurar g) e os

métodos de otimização [Chao 2011].

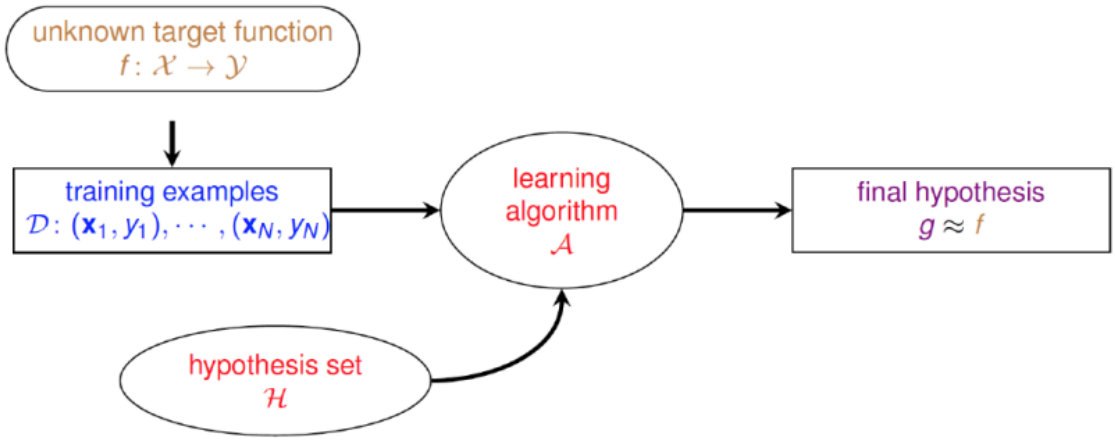

Figure 4.2. Diagrama do processo de aprendizado de máquinas

4.2.2. Definição formal de DM

Como a desaprendizagem de máquina parte do pressuposto que o modelo já "aprendeu", para suportar os pedidos de esquecimento o sistema computacional precisa de um mecanismo de desaprendizagem, denotada pela função U, que recebe como entrada um conjunto de dados de treino $D \in Z^*$, em que Z^* representa o conjunto de todos os datasets de treino possíveis. Além de um conjunto de esquecimento $D_f \subset D$ e um modelo de aprendizado A(D). O resultado será um modelo "desaprendido" $U(D, D_f, A(D)) \in H$, que tem como hipótese final ser um modelo que se aproxima do modelo retreinado $A(D|D_f)$. As principais notações que envolvem o processo de desaprendizado de máquinas podem ser visto na Figura 4.3.

Portanto, é possível dizer que na prática o problema central do desaprendizado de máquina envolve a comparação entre duas distribuições de modelos de aprendizado de máquina, o desaprendido e o retreinado [Bourtoule et al. 2021, Brophy and Lowd 2021, Thudi et al. 2022].

4.3. Por que desaprender?

Em 2010, foi desenvolvido um primeiro projeto legislativo na França que previa a criação de um "direito a ser esquecido" online. Posteriormente, não foram disponibilizadas muitas informações concretas sobre a proposta de lei, que pretendia obrigar as empresas do setor tecnológico a eliminarem as mensagens de correio eletrônico e de texto após um período de tempo acordado ou a pedido da pessoa em causa.Em novembro de 2010, a Comissão Europeia retomou a ideia de introduzir um direito a ser esquecido no contexto proteção de dados; o resultado da vaga proposta é ainda incerto [Weber 2011].

As discussões a respeito do tema retornaram com maior enfâse no ano de 2018, em virtude de um dos maiores escândalos no mundo da tecnologia, protagonizado pela Cambridge Analytica, uma empresa de marketing que teve acesso a dados de 87 milhões de usuários do Facebook indevidamente. Desde então, aspectos envolvendo segurança e privacidade dos dados, como as disposições e as sanções em matéria de proteção de

Symbols	Definition
$\mathcal{Z}$	example space
D	the training dataset
D_f	forget set
$D_r = D \setminus D_f$	retain set
$A(.)$	a learning algorithm
$U(.)$	an unlearning algorithm
$\mathcal{H}$	hypothesis space of models
$w = A(D)$	Parameters of the model trained on D by A
$w_r = A(D_r)$	Parameters of the model trained on D_r by A
$w_u = U(.)$	Parameters of the model unlearned by $U(.)$

Figure 4.3. Notações importantes do processo de desaprendizado de máquinas

dados, tem sido considerado internacionalmente como o correto caminho a seguir.

Nos últimos anos, foram promulgados muitos regulamentos visando a preservação da privacidade, que envolvem o "direito a ser esquecido" [Dang 2021]. Dentre as principais regulamentações, destacam-se o Regulamento Geral sobre a Proteção de Dados (RGPD) da União Europeia [Magdziarczyk 2019], na qual é aplicada sempre que os dados pessoais são processados, o que inclui o seu recolhimento, transformação, consulta ou eliminação, dentro ou fora da União Europeia (UE), bastando que os dados digam a respeito a um residente da UE [Veale et al. 2018]. Dentre outros regulamentos bastante conhecidos é possível citar a Lei da Privacidade do Consumidor da Califórnia (CCPA) [Pardau 2018], que estabelece que os usuários devem ter o direito de apagar os seus dados e informações relacionadas para proteger a sua privacidade. Em parte, esta legislação surgiu na sequência de fugas de informação sobre a privacidade. Por exemplo, casos de fugas de dados dos usuários dos sistemas de computação em nuvem, devido a múltiplas cópias dos dados mantidas por diferentes partes, políticas de cópia de segurança e estratégias de replicação.

Já no Brasil, a regulamentação se deu com a edição da Lei 13.709/2018, a denominada Lei Geral de Proteção de Dados (LGPD). A LGPD coloca o indivíduo como protagonista das relações jurídicas que envolvam o tratamento de dados, não apenas por regular a proteção de dados pessoais, mas, principalmente, elege como fundamento em seu art. 2º, II, a "autodeterminação informativa", que confere a pessoa o direito de escolher quais dados serão usados, bem como os limites e o prazo de sua utilização [Capanema 2020].

Embora seja um tema recente, a preocupação em saber se os sistemas de aprendizagem de máquina estão suficientemente regulamentados tem ganhado destaque no direito e nas políticas envolvendo o setor tecnológico. Dentre os elementos agravadores é possível citar o seu aparente potencial para reproduzir a discriminação social e transformar dados pessoais desprentensiosos em informações sensíveis [Veale et al. 2018]. Diante deste cenário, o processo de desaprendizagem torna-se estritamente necessário.

4.3.1. Motivação: Por que utilizar DM?

Além da questões envolvendo privacidade, é possível citar inúmeras razões pelas quais um usuário pode querer eliminar os seus dados de um sistema, tais motivos podem ser divididos entre grupos a seguir:

- Segurança: Apesar da importância de modelos de aprendizado profundo para o desenvolvimento do uso de IA em problemas do cotidiano, pesquisas recentes apontam que esses modelos são vulneráveis a ataques cibernéticos [Ren et al. 2020]. Dentre as principais estratégias maliciosas utilizadas é possível citar o envenenamento de dados. Para manipular o comportamento do modelo de machine learning (ML), o envenenamento de dados consiste em adicionar dados maliciosos aos conjuntos de dados de treinamento. Estes dados caracterizam-se por ser extremamente semelhantes aos dados originais, a ponto de um ser humano não conseguir distinguir entre os dados reais e os falsos. O principal objetivo deste ataque é manipular o comportamento do modelo treinado e fornecer resultados falsos, o que pode resultar em problemas graves. Por exemplo, na área da saúde, uma previsão errada pode levar a um diagnóstico errado, e consequentemente a um tratamento inadequado, que pode ocasionar a morte do paciente. Logo, a detecção e remoção de dados contraditórios é essencial para garantir a segurança do modelo e, uma vez detectado um ataque, o modelo tem de ser capaz de eliminar os dados contraditórios através de um mecanismo de desaprendizado de máquinas [Cao and Yang 2015, Marchant et al. 2022].

- Usabilidade: Os sistemas de recomendação apontam para as diferentes preferências em aplicações ou serviços onlines para cada usuário. Entretanto, uma aplicação produzirá recomendações inconvenientes se não puder eliminar completamente os dados incorretos relacionados ao usuário, frutos por exemplo, de ruído ou dados maliciosos. Suponha que uma pessoa tenha acidentalmente procurado por um produto ilegal no seu computador e descobriu que continua a receber a recomendação desse produto, mesmo depois de ter apagado o histórico do seu navegador [Cao and Yang 2015]. Esta usabilidade indesejada por não esquecer os dados não só produzirá previsões erradas, como também resultará em menos usuários do sistema. Neste cenário, a desaprendizagem de máquina seria uma alternativa viável para fazer o sistema esquecer esta informação.

- Justiça (Fairness) / IA Ética: No contexto atual, é essencial desenvolver sistemas de forma responsável, que levam em consideração não apenas a eficiência e o desempenho, mas também as implicações éticas e sociais dessas ferramentas, uma vez que elas impactam diretamente a vida das pessoas. Apesar dos avanços recentes, os modelos de aprendizado de máquinas continuam a ser sensíveis a preconceitos, o que significa que os seus resultados podem discriminar injustamente um grupo de pessoas [Mehrabi et al. 2021]. Logo, diante de modelos que foram treinados com algum tipo de viés é necessário desaprender estes dados, incluindo os atributos dos dados afetados.

- Otimização do desempenho: Há ainda o aspecto envolvendo a performance do modelo de aprendizado de máquinas, que dentre outros fatores podem ter desempenho

comprometido em virtude da influência de amostras de dados de baixa qualidade. As amostras utilizadas durante o treinamento podem ter degradado o desempenho geral do modelo e ao desaprender informações desatualizadas ou irrelevantes, os modelos de ML podem se tornar mais precisos.

4.3.2. Desafios no projeto de DM

Antes de conseguir efetivamente realizar a desaprendizagem de máquina, é necessário ultrapassar vários desafios para remover partes específicas dos dados de treino. Essas barreiras podem ser sintetizadas da seguinte forma:

- Estocasticidade do treinamento (stochasticity of training): Em virtude da natureza estocástica do procedimento de formação dos modelos de aprendizado de máquinas, não é possível saber a real influência que cada ponto de dados observado têm durante a formação no modelo [Bourtoule et al. 2021]. As redes neurais, por exemplo, são normalmente treinadas em mini-batches aleatórios que contêm um determinado número de amostras de dados. Além disso, o ordenamento dos lotes de formação também é aleatória [Bourtoule et al. 2021]. Esta característica estocástica gera dificuldades para a desaprendizagem de máquina, uma vez que a amostra de dados específica que precisa ser retirada teria de ser removida de todos os lotes. Logo, conhecimentos residuais ou vieses ainda podem permanecer no modelo

- Desaprendizado catastrófico (catastrophic unlearning): Em geral, um modelo desaprendido tem um desempenho pior do que o modelo retreinado do zero. No entanto, a degradação pode ser exponencial em relação a quantidade de dados a serem desaprendidos. Esta degradação súbita é frequentemente designada por desaprendizagem catastrófica [Nguyen et al. 2020]. Há inúmeros estudos que têm explorado formas de mitigar a desaprendizagem catastrófica através da definição de funções de perda e técnicas especiais para evitar este problema [Du et al. 2019, Golatkar et al. 2020a].

- Incrementalidade do treinamento (incrementality of training): O processo de treinamento de um modelo é um processo incremental, ou seja, a atualização do modelo numa determinada amostra de dados afetará o desempenho do modelo em amostras de dados introduzidas no modelo após essa inserção. O desempenho de um modelo nesta determinada amostra de dados também é afetado por amostras de dados anteriores. Determinar uma forma de eliminar o efeito da amostra de treinamento removida no desempenho do modelo é um desafio para a desaprendizado de máquinas [Nguyen et al. 2022].

- Padronização: Inexistência de métricas gerais para verificar na prática a eficiência e a eficácia do uso das técnicas de desaprendizado de máquinas.

4.4. Tipos de requisição de remoção

A remoção de informações de um modelo de aprendizagem de máquina é uma tarefa não trivial que exige a reversão parcial do processo de treinamento. Esta tarefa é inevitável

quando envolve dados sensíveis, como números de cartões de crédito ou palavras-passe, que acidentalmente entram no modelo e têm de ser removidos posteriormente. Dentre os principais tipos de requisição de remoção destacam-se:

- Remoção de pontos de dados (Item Removal): As solicitações de remoção de pontos de dados específicos são os casos mais comuns envolvendo a desaprendizado de máquina. Depois dos usuários partilharem os seus dados online, é geralmente difícil revogar o acesso à informação e pedir a eliminação dos seus dados.

- Remoção de atributos (Feature Removal): A desaprendizagem não deve limitar-se à remoção de pontos de dados, mas permitir correções em diferentes granularidades dos dados de treino, como a remoção de atributos. O primeiro método para desaprender atributos de um modelo foi proposto por [Warnecke et al. 2021]. A abordagem é inspirada no conceito de funções de influência, uma técnica estatística, que permite estimar a influência dos dados nos modelos de aprendizado [Koh and Liang 2017, Koh et al. 2019]. Ao reformular esta estimativa de influência como uma forma de desaprendizado, obtem-se uma abordagem versátil que mapeia as alterações dos dados de treino em retrospectiva com as atualizações dos parâmetros do modelo. Já o trabalho de [Guo et al. 2022] propõe uma nova abordagem para o desaprendizado de atributos associado a classificação por imagem. A Figura 4.4 apresenta a diferença inerente entre o atual paradigma de desaprendizagem de máquinas (esquerda) e a proposta de desaprendizagem de atributos (direita). A diferença reside principalmente nos objetos de remoção seletiva. A desaprendizagem de máquina existente visa remover seletivamente qualquer influência de certas amostras das representações de características aprendidas. Quando a desaprendizagem de máquinas existente remove amostras de entrada que contêm determinados atributos, o inconveniente é óbvio: para além dos atributos cuja remoção é solicitada, alguns atributos-chave que contribuem para o desempenho do modelo também foram removidos das representações de características aprendidas.

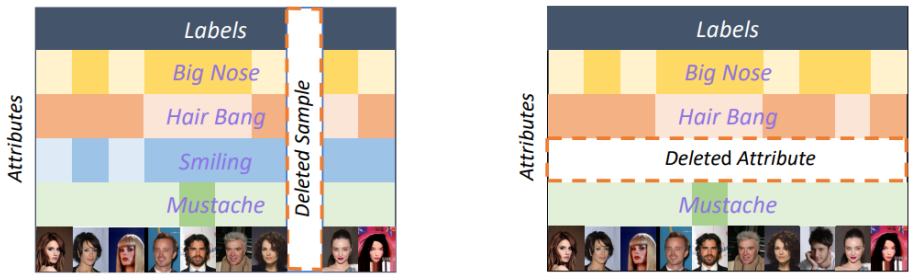

Figure 4.4. Comparação entre remoção por ponto de dados e por atributo

- Remoção de classes (Class Removal): Existem muitos cenários em que os dados a esquecer pertencem a uma ou várias classes de um modelo treinado. Por exemplo, nas aplicações de reconhecimento facial, cada classe é o rosto de uma pessoa, pelo que pode haver milhares ou milhões de classes. No entanto, quando um usuário opta por sair do sistema, a informação sobre o seu rosto deve ser removida sem utilizar uma amostra dele. Neste tipo de requisição a desaprendizagem exige que o

modelo esqueça uma ou mais classes, mas se lembre do resto dos dados. Para que uma ou mais classes sejam esquecidas, se o modelo puder ser atualizado através da observação de padrões que são, de alguma forma, opostos aos padrões aprendidos na altura do treino original, então os pesos atualizados do modelo podem refletir a desaprendizagem desejada e espera-se que preserve a informação sobre as restantes classes. O trabalho de [Tarun et al. 2023] propõe um método para remoção de classes baseado no chamado "data augmentation", que se caracteriza por introduzir ruído na base de treino com o objetivo de maximizar o erro na classe alvo para tratar deste tipo de remoção. Logo, é gerada uma matriz de ruído que é utilizada para manipular os pesos do modelo, com o intuito de desaprender a classe de dados alvo e manter o nível de performance do modelo original. Uma das suas principais vantagens está no fato de não ser necessário ter acesso as amostras de treinamento da classe alvo.

- Remoção de tarefas (Task Removal): Atualmente, os modelos de ML não são apenas treinados para uma única tarefa, mas também para várias tarefas. Este paradigma que busca espelhar o cérebro humano é denominado de aprendizado contínuo. No entanto, há situação em que pode ser necessário remover dados privados relacionados com uma tarefa específica. Por exemplo, considere um robô que foi treinado para assistir um doente em casa durante o seu tratamento médico. Pode ser pedido a este robô que esqueça este comportamento de assistência depois de o doente ter recuperado. Para este efeito, a aprendizagem de uma tarefa e o seu esquecimento no futuro tornou-se uma necessidade para os modelos de aprendizado contínuo.

- Remoção de fluxo (Stream Removal): No contexto da desaprendizado de máquinas, o tratamento de fluxos de dados está associada a lidar com um fluxo de pedidos de remoção. O trabalho de [Gupta et al. 2021] propôs um cenário de desaprendizagem em fluxo contínuo envolvendo uma sequência de pedidos de remoção de dados, motivado pelo fato de muitos usuários poderem estar envolvidos num sistema de ML e decidir apagar os seus dados sequencialmente. É também o caso quando os dados de treino foram envenenados por um ataque e os dados têm de ser eliminados gradualmente para recuperar o desempenho do modelo.

4.5. Métricas de avaliação

Esta seção busca apresentar métricas que avaliam o impacto do desaprendizado de máquinas no desempenho do modelo e que sejam capazes de quantificar o nível de esquecimento do processo de DM.

1. Acurácia: Um modelo desaprendido deve ser capaz de prever corretamente as amostras de teste ou, pelo menos, a sua acurácia deveria ser comparável à do modelo retreinado. No entanto, como o retreinamento é computacionalmente dispendioso, os modelos retreinados nem sempre estão disponíveis para comparação. Para resolver este problema, a acurácia do modelo não aprendido é frequentemente medida num novo conjunto de testes, ou é comparada com a do modelo original antes da desaprendizagem [He et al. 2021].

2. Completude (Completeness): Uma característica fundamental de um algoritmo de desaprendizagem é que ele deve ser completo. Esta métrica é capaz de avaliar se o esquecimento do sistema foi realmente integral, por meio da comparação entre os modelos desaprendidos e retreinados. Se o sistema atingiu a completude, os dois modelos farão as mesmas previsões sobre qualquer amostra de dados, estando elas corretas ou não. Para medir empiricamente a completude, quantifica-se a percentagem de amostras de entrada que recebem os mesmos resultados de previsão tanto do sistema desaprendido como do retreinado, utilizando um conjunto de dados de teste representativo. Quanto maior a percentagem, mais completa é a desaprendizagem [Cao and Yang 2015]. Há várias medidas de distância ou divergência que podem ser usadas para quantificar a diferença entre os modelos desaprendidos e retreinados, dentre as medidas representativas incluem a distância L2 e a divergência de Kullback-Leibler (KL) [Xu et al. 2024]. O trabalho de [Cao and Yang 2015] avalia analiticamente o uso desta métrica no processo de desaprendizagem de um algoritmo de recomendação chamado de LensKit. Para esquecer uma classificação, é necessário atualizar a sua matriz de semelhança de cada item. Os resultados para atualizar a semelhança entre os itens k e l são calculados do zero durante o retreinamento e comparados com uma medida conhecida chamada similaridade por cosseno, descrita na fórmula 1 abaixo, em que $||x||2$ representa a norma euclidiana de x, e $a_{*,k}$ é um vetor, (a_{1k}, a_{2k}, ..., a_{nk}), que representa todas as classificações recebidas pelo item k.

$$sim(k,l) = \frac{\overrightarrow{a_{*,k}}.\overrightarrow{a_{*,l}}}{||\overrightarrow{a_{*,k}}||_2||\overrightarrow{a_{*,l}}||} \quad (1)$$

3. Timeliness: Esta também é uma métrica importante para avaliar os sistemas que desaprenderam, pois ela mede o quão mais rápido é o desaprendizado para atualizar as features e o modelo do sistema em relação ao retreino. Ela está associada à rapidez do sistema em restaurar a privacidade, a segurança e a facilidade de utilização. Para medir empiricamente esta métrica, quantifica-se o aumento de velocidade da desaprendizagem em relação ao retreinamento. A desaprendizagem funciona melhor quando os dados a esquecer são pequenos em comparação com o conjunto de treino. Este caso é bastante comum. Por exemplo, os dados privados de um único usuário são normalmente pequenos quando comparados com os dados de treino de todos os usuários. Do mesmo modo, um atacante só precisa de uma pequena quantidade de dados para poluir um sistema de aprendizagem. Quando os dados a sereme esquecidos se tornam grandes, o retreino pode funcionar melhor [Cao and Yang 2015]. Além da completude, o trabalho de [Cao and Yang 2015] avalia o timeless associado ao processo de desaprendizagem do algoritmo LensKit, na qual constata-se que o nível de complexidade da desaprendizagem é $O(m^2)$, enquanto que para retreinamento é $O(nm^2)$. Logo, o algortimo de desaprendizagem tem uma performance melhor na ordem de $O(n)$ em comparação com o retreino.

4. Tempo de reaprendizagem (Relearn Time): Avalia a qualidade do desaprendizado em função do tempo de reaprendizado da informação removida. O tempo de reaprendizagem é um excelente indicador para medir a quantidade de informação de dados

não aprendidos que resta no modelo. Se um modelo recupera seu desempenho nos dados desaprendidos com apenas algumas etapas de retreinamento, é extremamente provável que o modelo tenha retido algum conhecimento dos dados desaprendidos.

5. Eficiência (Time Efficiency): A eficiência de um algoritmo de desaprendizagem é calculada empiricamente pelo razão entre o tempo necessário para obter o modelo desaprendido h^u e o tempo necessário para obter o modelo naıve retreinado h^*, conforme pode ser visto na fórmula 2 [Mercuri et al. 2022]. Uma eficiência alta indica um rápido desaprendizado.

$$Efficiency(h^u) = \frac{\text{time taken to obtain } h^*}{\text{time taken to obtain } h^u} \tag{2}$$

6. ZRF Score (Zero Retrain Forgetting): Esta métrica caracteriza-se por permitir a avaliação de métodos de desaprendizagem sem dependência do modelo retreinado, medindo a aleatoriedade das previsões do modelo, comparando a distribuição de saída do modelo desaprendido a um baseline aleatório [0,1], em que 1 indica modelo desaprendido apresenta comportamento aleatório e 0 exibe um certo padrão [Chundawat et al. 2023a]. A métrica ZRF score está descrita na fórmula 3, na qual é calculada utilizando divergência de Jensen-Shannon (JS) entre o modelo desaprendido M e um modelo inicializado aleatoriamente T_d, em relação à x_i que representa a i-ésima amostra do conjunto a esquecer D_f, notação apresentada anteriormente na Figura 4.3.

$$ZRF = 1 - (1/n_f)\sum_{i=0}^{n} JS(M(x_i), T_d(x_i)) \tag{3}$$

7. Anamnesis Index (AIN): Esta métrica foi introduzida por [Chundawat et al. 2023b], sob o pretexto de que apesar do tempo de reaprendizado ser bastante utilizado como métrica para avaliar a qualidade do desaprendizado ela possui algumas deficiências. Nas experiências realizada pelos autores, foi identificado que por vezes, os modelos desaprendidos recuperam uma acurácia significativa num número muito reduzido de passos de reaprendizagem, mas não convergem para a acurácia original na(s) classe(s) esquecida(s) durante um grande número de passos. Logo, foi proposto a métrica AIN, conforme pode ser visto na fórmula 4, que utiliza um valor percentual α em torno da acurácia original para calcular o tempo de reaprendizagem. Seja rt(M, M_{orig}, α) o número de passos necessários para que um modelo M se aproxime do percentual α da acurácia do modelo original M_{orig} nas classes esquecidas. Se M_u e M_s denotam o modelo não aprendido e o modelo treinado do zero. O valor de AIN varia de 0 a ∞, quanto mais próximo de 1 melhor é a desaprendizagem. Os valores de AIN muito inferiores a 1 correspondem aos casos em que a informação das classes esquecidas ainda está presente no modelo. Também indica que o modelo desaprendido reaprende rapidamente a fazer previsões exatas. Finalmente, caso a pontuação AIN for muito superior a 1, pode sugerir que a abordagem provoca alterações graves nos parâmetros.

$$AIN = \frac{rt(M_u, M_{orig}, \alpha)}{rt(M_s, M_{orig}, \alpha} \tag{4}$$

8. Distância de ativação (Activation Distance): É a separação entre a ativação final dos pesos eliminados e o modelo treinado novamente. Uma distância de ativação mais curta indica uma desaprendizagem melhor.

9. Epistemic Uncertainty: é uma métrica de incerteza que avalia se parâmetros atuais do modelo são ótimos para um determinado conjunto de dados. Com base nesta métrica, foi criado a efficacy score, conforme pode ser observado na fórmula 6, em que $i(w;D)$ determina a quantidade de informação que os parâmetros do modelo w consegue reter do dataset D. Esta pontuação mede a quantidade de informação que o modelo expõe. Quanto melhor for a estratégia de desaprendizagem, ela produzirá um modelo desaprendido com uma pontuação de eficácia mais baixa.

$$efficacy(w;D) = \begin{cases} \frac{1}{i(w;D)}, & \text{if } i(w;D) > 0 \\ \infty, & \text{se } n \text{ otherwise} \end{cases} \tag{5}$$

4.6. Requisitos de projeto

Para desenvolver um algoritmo de DM é necessário identificar alguns requisitos e definir a importância de cada um deles para o projeto. A figura 4.5 apresenta um diagrama contendo todos os requisitos, que estão descritos a seguir:

- Agnosticismo (Model-agnostic): Um processo de desaprendizagem deve ser genérico, ou seja, não estar condicionado a nenhum parâmetro ou especificações de qualquer algoritmo. Logo, ele deveria capaz de ser utilizado em diferentes modelos de aprendizagem de máquina [Bourtoule et al. 2021].

- Leveza (Light-weight): Outro fator fundamental para preparar o processo de desaprendizagem, muitas técnicas precisam armazenar pontos de controle do modelo, histórico de atualizações, dados de treinamento e outros dados temporários [He et al. 2021, Liu et al. 2020]. Um bom algoritmo de desaprendizagem deve ser leve e escalável com grandes volumes de dados. Ele deve ser capaz de lidar com sobrecarga computacional, para além do tempo de desaprendizagem e do custo de armazenamento [Bourtoule et al. 2021].

- Garantias comprováveis (Provable guarantees): É prático que um método de desaprendizagem forneça uma garantia comprovável sobre o modelo desaprendido. Para este efeito, muitos trabalhos conceberam técnicas de desaprendizagem com aproximações limitadas ao retreinamento, como trabalho [Guo et al. 2019] que criou uma garantia teórica muito forte de que um modelo do qual os dados são removidos não pode ser distinguido de um modelo que nunca observou os dados para começar. No entanto, abordagens com essa partem da premissa de que modelos com parâmetros comparáveis terão uma acurácia comparável.

- Capacidade de verificação (Verifiability): Para além dos pedidos de desaprendizagem, outra exigência dos usuários é verificar se o modelo desaprendido protege agora a sua privacidade. Para tal, um bom framework de desaprendizagem deve fornecer aos usuários finais um mecanismo de verificação. Por exemplo, os ataques backdoor podem ser utilizados para verificar a desaprendizagem, injetando amostras backdoor nos dados de treino [Sommer et al. 2020]. Se o backdoor não puder ser detectado no modelo original e for detectado no modelo desaprendido, a verificação é considerada um sucesso. No entanto, esta verificação pode ser demasiado intrusiva para um sistema de aprendizagem de máquina e a verificação pode ainda introduzir falsos positivos devido à incerteza inerente à detecção de backdoors.

- Completude (Completeness): Um bom algoritmo de desaprendizagem deve ser completo, ou seja, os resultados das previsões obtidas no modelo não aprendido e o retreinado devem ser consistentes sobre qualquer amostra de dados. Uma forma de medir esta consistência é calcular a percentagem dos mesmos resultados de previsão num dado de teste. Este requisito pode ser concebido como um objetivo de otimização numa definição de desaprendizagem, formulando a diferença entre o espaço de saída dos dois modelos. Há muitos trabalhos sobre ataques adversários podem ajudar nesta formulação [Chen et al. 2021, Sommer et al. 2022].

- Timeless: Outro importante requisito para bom algoritmo de desaprendizagem está associado a rapidez com ele desaprende. O timeless é utilizado para medir a rapidez da desaprendizagem em relação ao retreinamento após a solicitação de um pedido de desaprendizagem.

- Acurácia (Accuracy): Outra característica importante do algoritmo de desamprendizagem é a acurácia com que realiza as tarefas(previsão, classificação,...). Um modelo não aprendido deve ser capaz de prever corretamente as amostras de teste. A acurácia do modelo não aprendido é frequentemente medida num novo conjunto de testes, ou é comparada com a do modelo original antes da desaprendizagem.

4.6.1. Como os modelos “esquecem”?

Após um pedido de remoção de dados, o modelo atual será processado por um algoritmo de desaprendizagem para esquecer a informação correspondente a esses dados dentro do modelo. O algoritmo de desaprendizagem que permite o esquecimento seletivo, pode ter em conta vários requisitos, como a exaustividade, a atualidade e as garantias de privacidade. O resultado é um modelo não aprendido, que será avaliado em função de diferentes métricas de desempenho, descritas na seção 2.5. A Figura 4.6 apresenta a visão geral de como os modelos "esquecem".

4.7. Taxonomia e Algoritmos de DM

Dentre as possíveis classificações que as diferentes abordagens de desaprendizado de máquinas podem ter, este trabalho optou pela divisão entre os seguintes critérios:

1. Quanto ao grau de remoção de influência alcançado

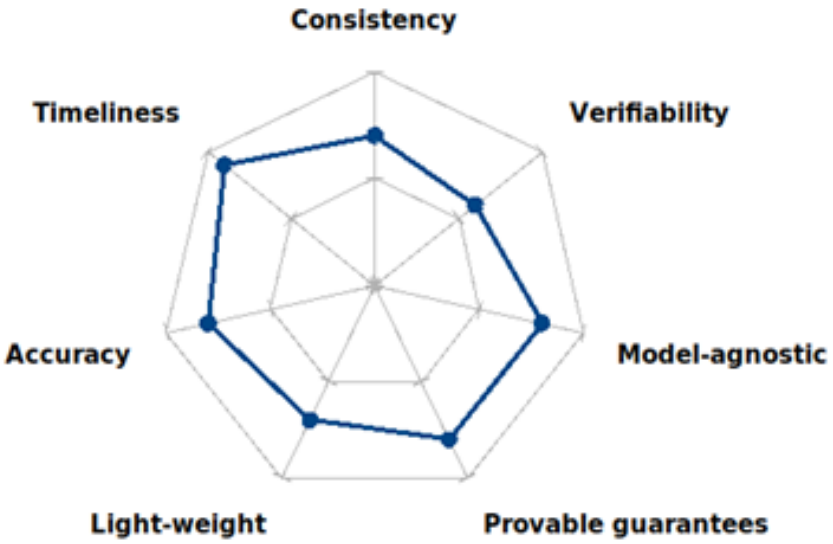

Figure 4.5. Requisitos de projeto para um algoritmo de desaprendizagem

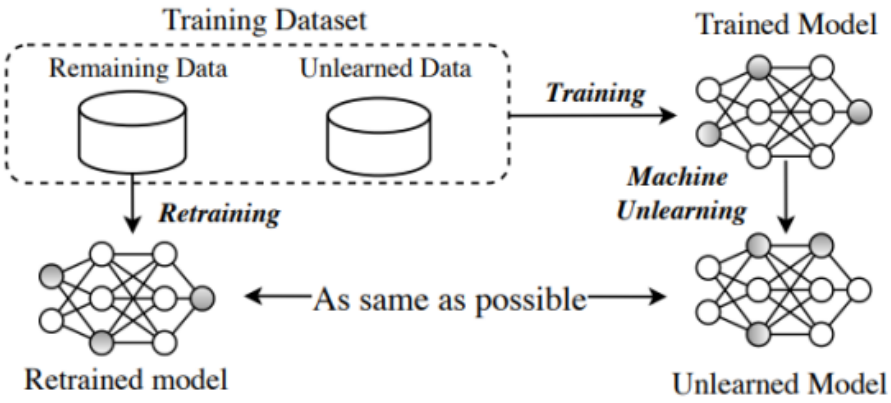

Figure 4.6. Visão geral de comos os modelos "esqucem"

- Exact Unlearning
- Approximate Unlearning

2. Quanto ao agnosticismo do algoritmo de aprendizado
    - Abordagens Model-agnostic
    - Abordagens Model-intrinsic
    - Abordagens Data-driven
3. Quanto à necessidade de dados de treinamento
    - Zero-Shot Unlearning

- One-Shot Unlearning
- Few-Shot Unlearning

4.7.1. Exact Unlearning

Exact unlearning é uma técnica que visa apagar completamente o impacto de dados específicos de treinamento em um modelo treinado. Quando um usuário solicita a exclusão de seus dados, a simples retirada dos mesmos não é suficiente, pois os modelos treinados ainda podem conter informações sensíveis e os dados retirados podem exercer influência sobre os parâmetros do modelo. Diante de tal problema, o exact unlearning garante que os dados que precisam ser desaprendidos sejam completamente removidos do modelo treinado, tornando o modelo desaprendido e o retreinado distribucionalmente idênticos. Em outras palavras, eles podem ser exatamente iguais diante de qualquer amostra de dados.

Matematicamente, o trabalho [Nguyen et al. 2022] apresenta a formulação geral desta abordagem, que permite definir o espaço métrico ao qual pertencem os modelos e consequentemente as suas distribuições. Dado um algoritmo de aprendizagem A(.), afirma-se que o processo U(.) é um processo de desaprendizagem exato se $\forall T \subseteq H, D \in Z^*, D_f \subseteq D$:

$$Pr(A(D|D_f) \in T) = Pr(U(D, D_f, A(D)) \in T) \tag{6}$$

Em termos das vantagens e desvantagens associado ao uso desta técnica, a seguir foram apresentas as principais características do desaprendizado:

- Remoção completa da influência de pontos de dados específicos (como se os dados removidos nunca tivessem sido vistos);
- Isola a influência de pontos exigindo apenas o retreinamento dos componentes afetados;
- Alto grau de consistência e verificabilidade;
- Necessário algum grau de retreinamento;
- Problema quanto a eficiência do retreinamento / Alto custo computacional;
- Induz a deterioração do desempenho;
- Sujeito a ataques de inferência/privacidade.

Portanto, em virtude dos inúmeros benefícios associados ao emprego desta abordagem, pesquisas têm surgido visando desenvolvimento de novos algoritmos que realizam o desaprendizado exato.

4.7.1.1. Exact Unlearning: SISA

Diante de dificuldade de lidar com um pedido de remoção de pontos de dados, o trabalho de [Bourtoule et al. 2021] apresentou o denominado treinamento SISA, sigla que indica:

- Sharded: particiona o dataset em conjuntos disjuntos (shards). Cada shard treina um submodelo
- Isolated: cada ponto de dados e sua influência está restrita a um único shard
- Sliced: adiciona mais um nível de fragmentação
- Aggregation: estratégia de votação majoritária

Este framework caracteriza-se por reduzir a sobrecarga computacional associada à desaprendizagem, mesmo no pior dos casos, em que os pedidos de desaprendizagem são efetuados uniformemente em todo o conjunto de treino. O treinamento SISA diminui o número de parâmetros do modelo afetados por um pedido de desaprendizagem e armazena em cache os resultados intermediários do algoritmo de treinamento para limitar o número de atualizações do modelo que têm de ser calculadas para que estes parâmetros sejam desaprendidos. Na Figura 4.7 é possível observar o funcionamento deste framework, em que os dados são divididos em shards, que por sua vez são divididos novamente em slices. Um modelo constituinte M é treinado em cada shard, apresentando-lhe um número crescente de slices e guardando os seus parâmetros antes do conjunto de treino ser aumentado com uma novo slice. Quando os dados precisam de ser desaprendidos, apenas um dos modelos constituintes cujos shards contém o ponto a desaprender precisa de ser treinado de novo - o treino pode começar a partir dos últimos valores de parâmetros guardados antes de incluir o shard que contém os pontos a serem desaprendidos.

A capacidade de generalização deste framework é notória, a ponto do trabalho de [Xu et al. 2024] definir o SISA como a abordagem geral para a desaprendizagem exata, dividindo as demais estruturas entre as que se baseiam no SISA ou não (non-SISA). O artigo separa as estruturas nos seguintes grupos:

- Exact Unlearning para Random Forest
- Exact Unlearning para modelos baseados em grafos
- Exact Unlearning para k-Means
- Exact Unlearning para Federated Learning
- Modelos não baseados no SISA (non-SISA)

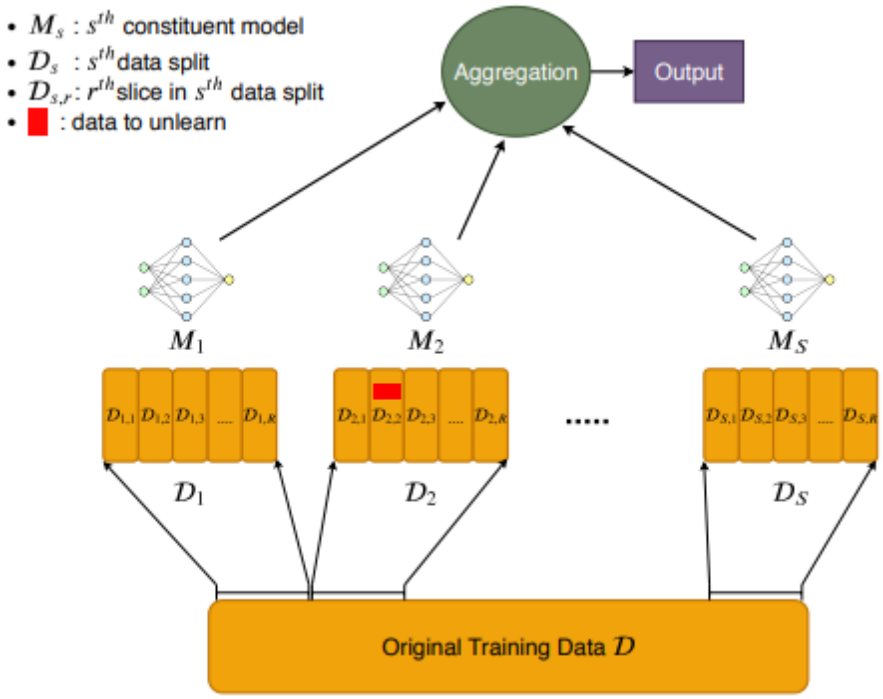

Figure 4.7. Treinamento SISA

4.7.1.2. Exact Unlearning para Random Forest

Cada árvore é treinada num subconjunto diferente de dados, atuando como um shard na estrutura SISA. As previsões das árvores individuais são agregadas para obter a previsão final da random forest. A influência de um ponto de dados é isolada dentro das árvores treinadas no subconjunto que contém esse ponto de dados. Ao desaprender um ponto de dados, apenas as árvores treinadas no subconjunto relevante necessitam de ser novamente treinadas

Dentre as abordagens para random forest, a proposta de [Brophy and Lowd 2021] permite a remoção de dados de treinamento com o mínimo de retreinamento. As atualizações do modelo para cada árvore DaRE na floresta são exatas, o que significa que a remoção de instâncias de um modelo DaRE produz exatamente o mesmo modelo que o retreino a partir do zero.

As árvores DaRE utilizam a aleatoriedade e o armazenamento em cache para tornar a eliminação de dados eficiente. Os níveis superiores das árvores DaRE utilizam nós aleatórios, que escolhem atributos de divisão e limiares uniformemente ao acaso. Estes nós raramente necessitam de atualizações porque dependem minimamente dos dados. Nos níveis inferiores, as divisões são escolhidas para otimizar avidamente um critério de divisão, como o índice de Gini ou a informação mútua. As árvores DaRE armazenam estatísticas em cada nó e dados de treino em cada folha, de modo a que apenas as subárvores necessárias sejam atualizadas à medida que os dados são removidos.

Portanto, as florestas DaRE utilizam várias técnicas para tornar as eliminações efi-

cientes (1) apenas retreinam partes do modelo em que a estrutura tem de mudar para corresponder à base de dados atualizada; (2) consideram no máximo k thresholds selecionados aleatoriamente por atributo; (3) introduzem nós aleatórios no topo de cada árvore que dependem minimamente dos dados e, portanto, raramente precisam ser retreinados. As Figuras 4.8 representa treinamento de uma árvore DaRE.

Algorithm 1 Building a DaRE tree / subtree.

Input: data D, depth d
if stopping criteria reached **then**
  $node \leftarrow$ LEAFNODE()
  save instance counts($node$, D) ▷ $|D|$, $|D_{\cdot,1}|$
  save leaf-instance pointers($node$, D)
  compute leaf value($node$)
else
  if $d < d_{\text{rmax}}$ **then**
    $node \leftarrow$ RANDOMNODE()
    save instance counts($node$, D) ▷ $|D|$, $|D_{\cdot,1}|$
    $a \leftarrow$ randomly sample attribute(D)
    $v \leftarrow$ randomly sample threshold $\in [a_{min}, a_{max})$
    save threshold statistics($node$, D, a, v)
  else
    $node \leftarrow$ GREEDYNODE()
    save instance counts($node$, D) ▷ $|D|$, $|D_{\cdot,1}|$
    $A \leftarrow$ randomly sample $\tilde{p}$ attributes(D)
    for $a \in A$ **do**
      $C \leftarrow$ get valid thresholds(D, a)
      $V \leftarrow$ randomly sample k valid thresholds(C)
      for $v \in V$ **do**
        save threshold statistics($node$, D, a, v)
    $scores \leftarrow$ compute split scores($node$)
    select optimal split($node$, $scores$)
  $D.\ell, D.r \leftarrow$ split on selected threshold($node$, D)
  $node.\ell$ = TRAIN(D_ℓ, $d+1$) ▷ Alg. 1
  $node.r \leftarrow$ TRAIN(D_r, $d+1$) ▷ Alg. 1
Return node

Figure 4.8. Treinamento de uma árvore DaRE

Os experimentos propostos pelos autores do DaRe, buscou medir a eficiência relativa comparando o tempo que um modelo DaRe elimina o número de instâncias com a abordagem de retreinamento naive. A avaliação da abordagem foi feita utilizando dois adversários diferentes: Random e Worst-of-1000. O adversário aleatório seleciona as instâncias de treino a serem eliminadas uniformemente ao acaso, enquanto o adversário pior-de-1000 seleciona cada instância, primeiro selecionando 1.000 instâncias candidatas uniformemente ao acaso e, em seguida, escolhendo a instância que resulta no maior retreino.

Tal como DaRe, HedgeCut [Schelter et al. 2021] é uma baseado no SISA e uma variante do random forest. Esta técnica se concentra em requisições de desaprendizado com baixa latência em árvores extremamente aleatórias. Ela introduz o conceito de ro-

bustez de divisão para identificar decisões de divisão que podem mudar com dados removidos. O HedgeCut mantém variantes de subárvores para esses casos e, ao desaprender um ponto de dados, substitui a divisão correspondente pelas variantes de subárvores preparadas. Esta operação é rápida e direta, garantindo um pequeno atraso na

4.7.1.3. Exact Unlearning para modelos baseados em grafos

A desaprendizagem exata de modelos baseados em grafos tem como objetivo remover de forma eficiente e precisa a influência de pontos de dados individuais nas previsões do modelo, tendo em conta as características únicas dos dados estruturados em grafos.

Dentre as principais abordagens está o GraphEraser [Chen et al. 2022], que expande a estrutura SISA a dados em grafos. O GraphEraser foi concebido para a desaprendizagem de redes neurais em grafos (GNNs). Ela consiste em três fases: divide o grafo de treino original em shards disjuntos, treina paralelamente um conjunto de modelos de shards F_i e aprende uma pontuação de importância ótima α_i para cada modelo de shard. Quando um nó w precisa de uma previsão, o GraphEraser envia w a todos os modelos de shards e obtém os posteriores correspondentes, que são depois agregados utilizando a pontuação de importância ótima α_i para fazer a previsão. Quando um nó u efetua um pedido de desaprendizagem, o GraphEraser remove u do shard correspondente e volta a treinar o modelo do shard.

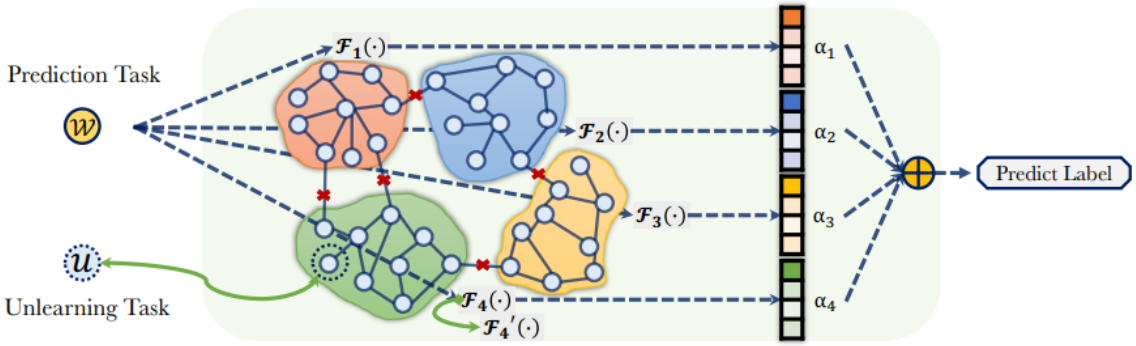

Figure 4.9. Framework do GraphEraser

4.7.1.4. Exact Unlearning para k-Means

Para a técnica baseada no k-means, tem-se a abordagem DC-k-means [Ginart et al. 2019], cuja sigla DC significa "dividir-e-conquistar" (Divide-and-Conquer). Esta técnica se caracteriza por ser baseada no SISA, mas que utiliza um método de agregação hierárquica em forma de árvore. Em linhas gerais, ele funciona através da partição do conjunto de dados em pequenos subproblemas, resolvendo cada subproblema como uma instância k-mean independente e fundindo recursivamente os resultados.

Os dados de treino são divididos aleatoriamente em vários subconjuntos, cada um representado por um nó folha numa árvore. Um modelo k-means é treinado em cada

subconjunto, com cada nó de folha correspondendo a um modelo k-means. O modelo final é uma agregação de todos os modelos k-means nos nós das folhas da árvore, obtida através da fusão recursiva dos resultados dos nós das folhas até à raiz. Para desaprender um ponto de dados, o nó folha relevante é localizado e o modelo k-means correspondente é atualizado para excluir esse ponto de dados. O modelo atualizado substitui então o modelo antigo no nó folha, e as alterações propagam-se pela árvore para atualizar o modelo agregado final. A Figura 4.10 apresenta o pseudocódigo desta abordagem.

Esta técnica utiliza a hierarquia da árvore para modular a dependência do cálculo em relação aos dados. Logo, no momento da eliminação dos dados, só é necessário recomputar os subproblemas de uma folha até à raiz, o que a possibilita suportar operações de eliminação rápidas [Ginart et al. 2019].

Algorithm 2 DC-k-means

Input: data matrix $D \in \mathbf{R}^{n \times d}$
Parameters: $k \in \mathbf{N}$, $T \in \mathbf{N}$, tree width $w \in \mathbf{N}$, tree height $h \in \mathbf{N}$
Initialize a w-ary tree of height h such that each node has a pointer to a dataset and centroids
for $i = 1$ **to** n **do**
  Select a leaf node uniformly at random
  node.dataset.add(D_i)
end for
for $l = h$ **down to** 0 **do**
  for each node in level l **do**
    $c \leftarrow$ k-means++(node.dataset,k,T)
    node.centroids $\leftarrow c$
    if $l > 0$ **then**
      node.parent.dataset.add(c)
    else
      save all nodes as metadata
      return c //model output
    end if
  end for
end for

Figure 4.10. Pseudocódigo DC-kmeans

4.7.1.5. Exact Unlearning para Federated Learning

Para este tipo de abordagem, o trabalho de [Su Ningxin 2023] desenvolveu o KNOT, que adota o framework da SISA para a desaprendizado federado assíncrona a nível do cliente

durante o treinamento. Um mecanismo de agregação em clusters divide os clientes em vários clusters. O servidor só efetua a agregação de modelos dentro de cada grupo, enquanto os diferentes grupos treinam de forma assíncrona. Quando um cliente pede para remover os seus dados, apenas os clientes dentro do mesmo cluster precisam de ser novamente treinados, enquanto os outros clusters não são afetados e podem continuar normalmente. Para obter uma atribuição ótima cliente-cluster, o KNOT efetua a formulação como um problema de minimização lexicográfica. O objetivo é minimizar a classificação de correspondência entre cada cliente e o cluster atribuído, considerando tanto a velocidade de treino como a semelhança do modelo. Este problema de otimização de inteiros pode ser resolvido eficientemente como um Programação Linear (LP) usando um solver LP.

A Figura 4.11 apresenta um exemplo de agregação assíncrona em clusters em que quatro clientes foram atribuídos a dois clusters. Se o cliente n.º 4 pedir para apagar os efeitos dos seus dados do servidor, apenas os clientes do agregado n.º 2 precisam de ser novamente treinados, enquanto os clientes do agregado n.º 1 podem prosseguir normalmente com o seu processo de formação FL.

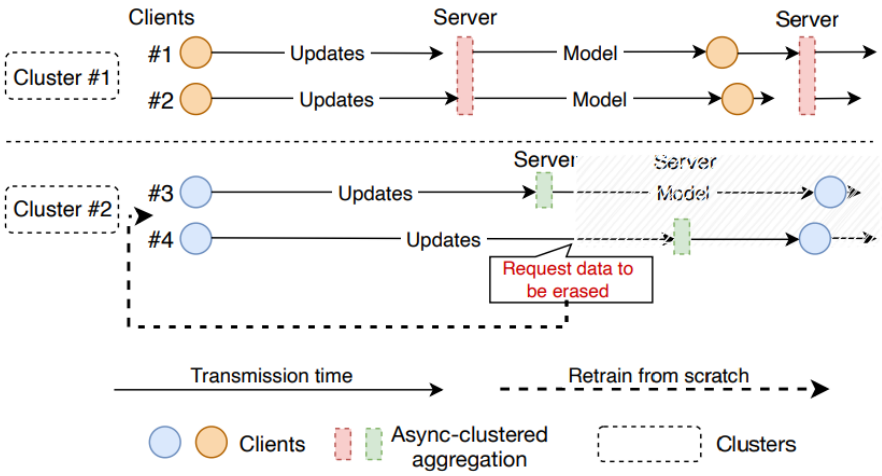

Figure 4.11. Exemplo de KNOT

Ao se analisar comparativamente as diferentes abordagenes de desaprendizado exato que baseiam no SISA, a Figura 4.12 apresenta o quadro geral contendo elementos como: objetivos, modelagem, pontos fortes e limitações de cada método.

4.7.1.6. Exact Unlearning: non-SISA

Dentre as abordagens que não se baseiam no SISA, o trabalho de [Cao and Yang 2015] concentra-se num dos desafios mais difíceis: fazer com que os sistemas de aprendizado de máquinas se esqueçam. A abordagem desenvolvida teve a seguinte ideia de desaprendizagem, em vez de fazer com que um modelo dependa diretamente de cada amostra de dados de treino (à esquerda), convertemos o algoritmo de aprendizagem numa forma de somatório (à direita) usando o conceito de SQL e Map-Reduce, o algoritmo de aprendizado

Paper	Goal	Design Ideas	Strengths	Weaknesses
SISA [36]	Efficient unlearning for general models	• Sharded, isolated, sliced, aggregated training	• Applicable to any model • Improves unlearning efficiency • Scalable	• Breaking data dependencies • Additional storage cost for model parameters • Accuracy degradation
DaRE [58]	Efficient unlearning for random forest	• Use cached statistics to only retrain affected subtrees • Consider subsets of thresholds per attribute	• Achieves efficiency by retraining affected subtrees • Analyzes the effect of layers and thresholds	• Additional storage cost for caching statistics • Worst case performance no better than retraining
HedgeCut [34]	Low-latency unlearning for ensemble of random forest	• Use split robustness to identify splitting decisions • Prepare subtree variants for non-robust splits	• Low-latency unlearning • Predictive accuracy similar to random forest	• Rely on the accuracy of predicted unlearning requests • Storage cost for subtree variants
DC-k-means [61]	Efficient unlearning for k-means	• Divide-and-Conquer approach • Hierarchical aggregation	• Theoretical guarantees on removal efficiency • Negligible quality loss	• State storage cost • High removing time complexity in high dimensions
GraphEraser [59]	Efficient unlearning for for GNNs	• Community detection for balanced sharding • Shard model importance scores	• Maintains graph structure • Balanced partitioning ensures unlearning efficiency	• Additional cost of maintaining massive shards. • Not satisfy the adaptive setting
RecEraser [60]	Efficient unlearning for recommender systems	• Different data partition strategies • Attention-based adaptive aggregation method	• Efficient unlearning with balanced data partition • Good global model utility with adaptive aggregation	• Specific to recommendation task • Open problem in the batch setting
KNOT [62]	Federated unlearning during training	• Cluster clients and perform aggregation within clusters • Clusters train asynchronously	• Limits retraining to a cluster • Asynchrony allows unaffected clusters to continue training	• Performance relies on cluster assignments • Asynchrony can negatively impact model accuracy
ARCANE [18]	Efficiently and accurately unlearning	• Multiple one-class classification tasks • Data preprocessing	• Maintains accuracy even with large unlearning requests • Data preprocessing accelerates unlearning	• Only applies to supervised models • Rely on preprocessing for efficiency

Figure 4.12. Visão geral dos métodos de desaprendizagem exata baseada no SISA [Xu et al. 2024]

é modificado em um sistema formado por somatórios. Especificamente, cada somatório é a soma de amostras de dados transformados, onde as funções de transformação g_i são eficientemente computáveis. Há apenas um pequeno número de somas e o algoritmo de aprendizado depende apenas delas. Para esquecer uma amostra de dados, basta atualizar os somatórios e depois calcular o modelo atualizado. Esta abordagem é muito mais rápida do que o retreinamento a partir do zero. O conceito desta abordagem pode ser visto na Figura 4.13.

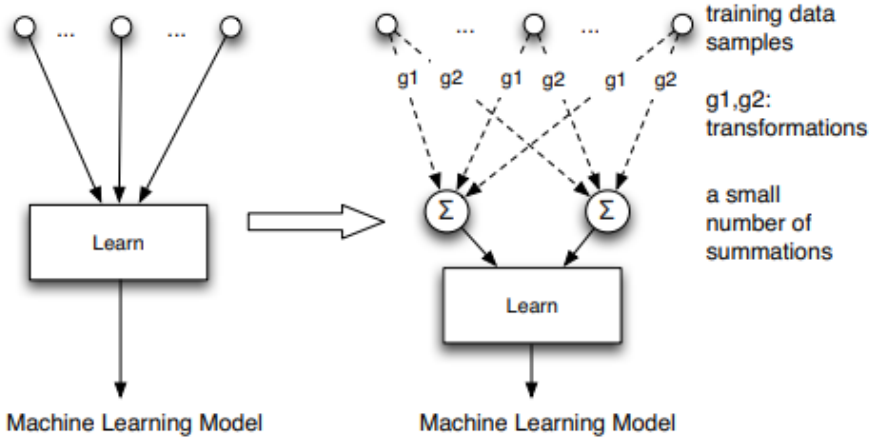

Figure 4.13. O conceito de desaprendizagem

Esta técnica é geral porque a forma de soma provém da aprendizagem da consulta estatística, do inglês "Statistical Query" (SQ) [48]. Muitos algoritmos de aprendizagem

de máquina, como os classificadores naıve Bayes, support vector machines e a técnica de clusterização k-means, podem ser implementados como aprendizado SQ. A nossa abordagem também se aplica a todas as fases da aprendizagem de máquina, incluindo a seleção de características e a modelização.

A abordagem de desaprendizado foi avaliada em quatro sistemas de aprendizado diferentes: o LensKit [Ekstrand et al. 2011], um sistema de recomendação de código aberto utilizado por vários sítios Web para recomendações de conferências e livros. O segundo foi uma reimplementação independente do Zozzle, detetor de malware JavaScript de código fechado cujo algoritmo foi adotado pelo Microsoft Bing. Já o terceiro foi um filtro de spam de redes sociais de código aberto [Gao et al. 2012]. Por último, teve também o PJScan, um detetor de malware PDF de código aberto [Laskov and Šrndić 2011]. Além disso, foi utilizado cargas de trabalho do mundo real, como mais de 100 mil amostras de malware JavaScript da Huawei.

			Analytical Results				Empirical Results				
System	**Attack**	**Summation?**	Complete-ness	Unlearn Speed	Retrain Speed	Speedup	Complete-ness	Unlearn Speed	Retrain Speed	Speedup	Modification LoC (%)
LensKit	Old	✓	100%	$O(m^2)$	$O(nm^2)$	$O(n)$	100%	45s	4min56s	6.57	302 (0.3%)
Zozzle	New	✓	100%	$O(q)$	$O(Nq)$	$O(N)$	100%	987ms	1day2hours	9.5×10^4	21 (0.4%)
OSNSF	New	✗	<100%	$O(logN)$	$O(NlogN)$	$O(N)$	99.4%	21.5ms	30mins	8.4×10^4	33 (0.8%)
PJScan	New	✓	100%	$O(N^p)$	$O(N^l)$	$O(N^k)$	100%	156ms	157ms	1	30 (0.07%)

Figure 4.14. Comparação dos resultados do desaprendizado

A Figura 4.14 apresenta o resumo desta abordagem de desaprendizado aplicado a todos os algoritmos de aprendizagem. Dentre os principais resultados, destaca-se o 100% de completude, um dos principais métricas para avaliação dos algoritmo de desaprendizado, para três sistemas LensKit, Zozzle e PJScan. Embora OSNF não tenha obtido tal performance, ele foi o sistema que desaprendeu mais rápido com 21.5 ms.

4.7.2. Approximate Machine Unlearning

Embora os métodos de desaprendizado de máquinas exata tenham dado passos significativos no sentido de alcançar o direito ao esquecimento, pesquisadores identificaram novos desafios a partir de duas outras perspectivas. A primeira é que o desaprendizado de máquinas exato resulta inevitavelmente numa degradação do desempenho do modelo, mesmo que essa degradação seja, por vezes, menor. O segundo é o potencial de fuga de privacidade. Por exemplo, se a "Alice" for removida e o modelo bruto diferir do modelo recalibrado, um adversário pode perceber que a diferença se deve à remoção da "Alice" e lançar um ataque de inferência para comprometer ainda mais a privacidade. Os dois desafios acima referidos podem ser atenuados através da utilização de métodos de desaprendizado de máquinas aproximados, que ocultam a diferença entre os modelos antes e depois da remoção dos dados, otimizando simultaneamente o desempenho do modelo [Qu et al. 2023].

Logo, o desaprendizado aproximada é um processo que tem por objetivo minimizar a influência dos dados não aprendidos para um nível aceitável, em vez de os eliminar completamente. Para isso ele envolve as seguintes etapas:

- Cálculo da influência: Calcular a influência dos pontos de dados que precisam de ser desaprendidos no modelo original. Isto implica determinar como estes pontos de dados afetam o modelo

- Ajuste dos parâmetros do modelo: Modificar os parâmetros do modelo para inverter a influência dos dados que estão a ser removidos. Este ajustamento envolve normalmente métodos como a reponderação ou o recálculo de parâmetros ótimos e a modificação do modelo para que se comporte como se tivesse sido treinado no conjunto de dados sem os pontos de dados não aprendidos.

- Adição de ruído (opcional): É adicionado ruído cuidadosamente calibrado para evitar que os dados removidos sejam inferidos a partir do modelo atualizado. Este passo assegura a confidencialidade do conjunto de dados de treino.

- Validação do modelo atualizado: Avaliar o desempenho do modelo atualizado para garantir a sua eficácia. Esta etapa de validação pode envolver uma validação cruzada ou um teste num conjunto de espera para avaliar a exatidão e a generalização do modelo.

Em virtude do desaprendizado aproximado ser uma abordagem que constitui uma alternativa prática à desaprendizagem exata, particularmente em cenários em que fatores como o custo computacional, o custo de armazenamento e a flexibilidade sejam cruciais [Qu et al. 2023]. As seguintes subseções apresentaram diferentes métodos que utilizam essa abordagem, com suas respectivas características e limitações.

4.7.2.1. Approximate Unlearning baseada na função de influência dos dados removidos

A ideia central deste tipo de abordagem consiste em utilizar a função de influência para estimar o nível de impacto que a remoção de pontos de dados têm sobre o modelo.

O primeiro trabalho envolvendo a abordagem foi o de [Guo et al. 2019], que propôs um método que ajusta os parâmetros do modelo com base na influência calculada dos dados removidos.envolve a aplicação de regularização nos parâmetros do modelo, como os coeficientes em modelos lineares, para reduzir a influência de dados específicos. O trabalho obtém uma remoção certificada de modelos lineares L2-regularizados, que envolve a aplicação de regularização nos parâmetros do modelo, como os coeficientes em modelos lineares, para reduzir a influência de dados específicos. Entretanto, a proposta apresentou limitações para lidar com modelos de perdas não convexas, porque a função de influência se basear no pressuposto de convexidade para garantir que a influência estimada reflete com precisão a verdadeira influência dos pontos removidos.

Com base no trabalho de Guo, [Sekhari et al. 2021] não requer acesso total ao conjunto de dados de treinamento durante o processo de desaprendizagem. Eles permitem um desaprendizado eficiente sem a necessidade de todos os dados de treinamento, reduzindo os requisitos computacionais e de armazenamento. A Figura 4.15 apresenta o pseudocódigo da proposta, em que A_{sc} consiste no algoritmo de aprendizado, recebe

como entrada o conjunto de pedidos de eliminação U, o ponto w e as estatísticas de dados T(S). Utilizando estes dados, A_{sc} começa por estimar a matriz H que denota o Hessiano da função empírica no conjunto de dados S/U quando avaliada no ponto wb. Em seguida, A_{sc} calcula o ponto w removendo a contribuição dos pontos U eliminados de wb utilizando a atualização em (8). Finalmente, A_sc perturba w com ruído retirado de $N(0, \sigma^2 I_d)$ e devolve o ponto perturbado w.

Algorithm 1 Unlearning algorithm ($\bar{A}_{sc}$)

Input: Delete requests: $U = \{z_j\}_{j=1}^{m} \subseteq S$, output of $A_{sc}(S)$: $\widehat{w}$, additional statistic $T(S)$: $\{\nabla^2 \widehat{F}(\widehat{w})\}$, loss function: $f(w, z)$.

1: Set $\gamma = \frac{2Mm^2L^2}{\lambda^3 n^2}$, $\sigma = \frac{\gamma}{\varepsilon}\sqrt{2\ln(1.25/\delta)}$.

2: Compute

$$\widehat{H} = \frac{1}{n-m}\left(n\nabla^2 \widehat{F}(\widehat{w}) - \sum_{z\in U} \nabla^2 f(\widehat{w}, z)\right). \tag{7}$$

3: Define

$$\bar{w} = \widehat{w} + \frac{1}{n-m}(\widehat{H})^{-1} \sum_{z\in U} \nabla f(\widehat{w}, u). \tag{8}$$

4: Sample $\nu \in \mathbb{R}^d$ from $\mathcal{N}(0, \sigma^2 \mathbb{1}_d)$.

5: **Return** $\widetilde{w} := \bar{w} + \nu$.

Figure 4.15. Pseudocódigo desaprendizado aproximado [Sekhari et al. 2021]

Outro trabalho que merece destaque foi desenvolvido por [Mehta et al. 2022], que aborda desaprendizado aproximado especificamente para redes neurais profundas (DNN). É proposto um esquema de seleção, L-CODEC, que identifica um subconjunto de parâmetros a atualizar, eliminando a necessidade de inverter uma grande matriz. Isto evita a atualização de todos os parâmetros, concentrando-se apenas nos mais influentes. Com base nisto, propõem L-FOCI para construir um conjunto mínimo de parâmetros influentes utilizando L-CODEC de forma incremental. Uma vez identificado o subconjunto de parâmetros a atualizar, aplicam uma atualização de Newton no sentido dos ponteiros do relógio a esse subconjunto. Ao concentrar os cálculos apenas nos parâmetros influentes, a sua abordagem torna viável a desaprendizagem aproximada para grandes redes neurais profundas, anteriormente inviáveis.

Na Figura 4.16 apresenta o exemplo do funcionamento quando uma amostra é perturbada e passada através da rede. As ativações são agregadas juntamente com as perdas e alimentam o L-FOCI. As linhas seleccionadas representam fatias da camada correspondente que são suficientes para a desaprendizagem.

Já dentre as abordagens diferentes de [Guo et al. 2019], que apenas ajusta a camada de decisão linear de um modelo, o trabalho proposto por [Wu et al. 2022] desenvolveu o PUMA, que caracteriza-se por modificar todos os parâmetros de treinamento, oferecendo uma solução mais completa para a remoção de dados. O principal objetivo do PUMA é manter o desempenho do modelo após a remoção dos dados, em vez de apenas monitorar se o modelo modificado consegue produzir previsões semelhantes às de um

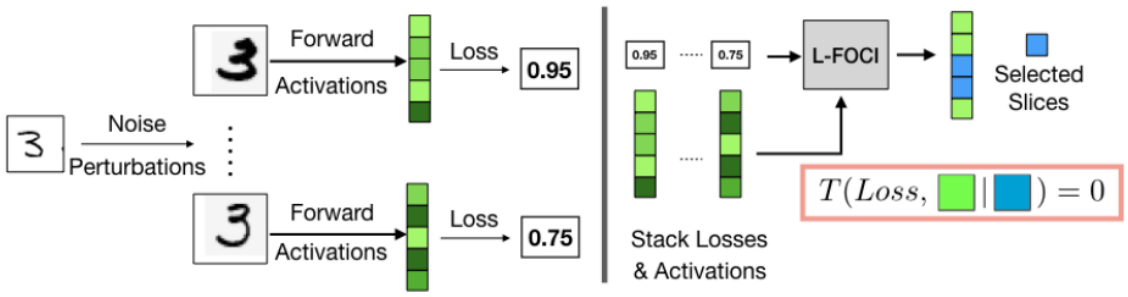

Figure 4.16. Exemplo de funcionamento L-FOCI

modelo treinado com os dados restantes, como faz o método de Guo et al. Para o conseguir, o PUMA utiliza a influência para medir o impacto que de cada ponto de dados tem no desempenho do modelo e, em seguida, ajusta o peso dos dados restantes para compensar a remoção de um ponto de dados específico. A Figura 4.17 apresenta o resultado da remoção dos pontos de treino marcados por cruzes do modelo. Como demonstrado no gráfico da direita, o PUMA removeu com sucesso a informação de todos os pontos marcados.

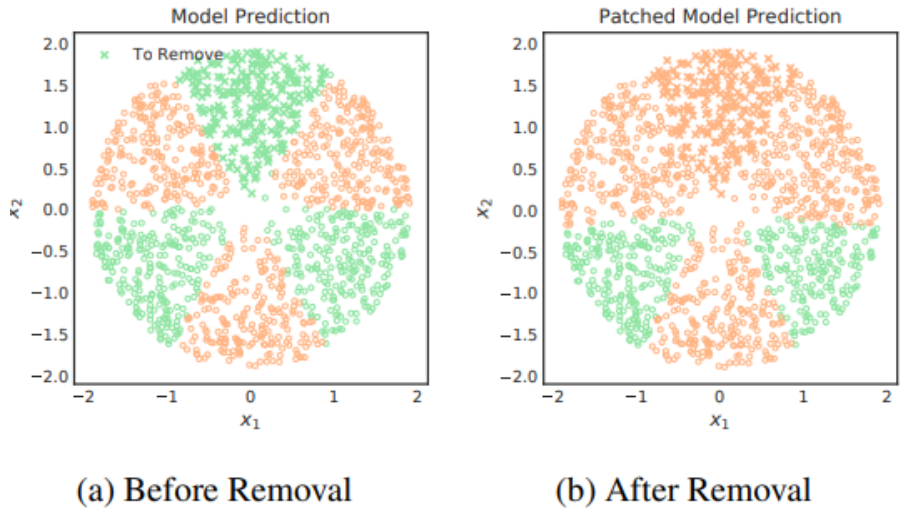

Figure 4.17. Removendo pontos dos dados usando PUMA

A Figura 4.18 apresenta o quadro geral das diferentes abordagens de desaprendizagem aproximada baseada na função de influência dos dados removidos, contendo características como: objetivos, modelagem, pontos fortes e limitações de cada método.

4.7.2.2. Approximate Unlearning baseada na reotimização após a remoção dos dados

A ideia central da desaprendizagem aproximada baseada na reotimização é ajustar iterativamente um modelo para esquecer efetivamente pontos de dados específicos, man-

Paper	Goal	Design Ideas	Strengths	Weaknesses
Guo *et al.* (2020) [43]	Certified removal	• One-step Newton update/influence function • Difference privacy	• Efficient training data removal • Strong theoretical certified removal guarantee	• Relies on convexity • High computational cost for inverting the Hessian matrix
Sekhari *et al.* (2021) [64]	Efficient unlearning with generalization guarantees	• Use influence functions to identify important data points • Store cheap data statistics	• Considers test loss instead of just training loss • Reduce computational and storage costs • Give the deletion capacity	• Relies on convexity • Relies on storage of data statistics • Does not handle finite/discrete hypothesis classes
Suriyakumar *et al.* (2022) [65]	Efficient unlearning for ERM models	• Infinitesimal jackknife • Newton update	• Computationally efficient online unlearning • Accommodating non-smooth regularizers	• Specific to ERM models • Inefficient for batch removal
Mehta *et al.* (2022) [66]	Efficient unlearning for DNN	• Conditional independence-based parameter selection	• Avoids full Hessian inverse • Improves unlearning efficiency in DNN	• Relies on weighted sampling based on Lipschitz constants of filters/layers • Dependence on newly developed optimization tools
PUMA [67]	Maintain performance during unlearning	• Performance unchanged model augmentation	• Maintain performance after removal • Computationally efficient	• Limited evaluation on simple datasets • Sensitive to hyperparameters
Tanno *et al.* (2022) [68]	Repair model by data removal	• Identify detrimental data via influence function • Remove via posterior approximation	• Model-agnostic framework • Identify causes of failures	• Limited to detrimental data removal • Cause identification can be computationally intensive
Warnecke *et al.* (2023) [70]	Unlearn features and labels	• Use influence functions as updates for features or labels	• Effective unlearning of features/labels • Efficient closed-form updates	• Efficacy drops as affected features/labels increase • No guarantee for non-convex models

Figure 4.18. Visão geral dos métodos de Desaprendizagem aproximada baseada na função de influência dos dados removidos [Xu et al. 2024]

tendo o desempenho global. Pesquisas neste domínio propõem diferentes técnicas de remoção/esquecimento seletivo com base nos objetivos da aplicação.

O primeiro trabalho sobre o tema foi feito por [Golatkar et al. 2020a], que propõem um algoritmo ótimo de depuração quadrática para conseguir o esquecimento seletivo em redes profundas. O esquecimento seletivo é definido como um processo que modifica os pesos da rede usando uma função de depuração S(w) para tornar a distribuição indistinguível dos pesos de uma rede nunca treinada com os dados esquecidos. O esquecimento seletivo é medido pela divergência KL, que dado p(x) e q(x) duas distribuições, essa métrica pode ser representado pela fórmula 7 .

$$KL(p(x)||q(x)) := E_{x\sim p(x)}[log(p(x)/q(x))] \tag{7}$$

Se a divergência KL entre a distribuição de pesos da rede após o esquecimento seletivo e a distribuição de pesos da rede que nunca viu os dados esquecidos for zero, as duas distribuições são exatamente iguais, o que indica um esquecimento completo (completeness). Logo, ela é capaz de medir a quantidade máxima de informação que um atacante pode extrair.

No seu trabalho de seguinte, [Golatkar et al. 2020b] observam que as alterações de peso podem não afetar os resultados das redes profundas devido à sobreparametrização. Consequentemente, os atacantes ainda podem extrair dados removidos D_f das saídas. Para resolver esse problema, eles se concentram nas ativações finais. Essas ativações representam a resposta do modelo aos dados de entrada e refletem mais diretamente os

processos de memória e remoção. Para isso, eles utilizam um Neural Tangent Kernel (NTK) para correlacionar pesos e activações e introduzem um processo de depuração baseado em NTK para conseguir a remoção, minimizando a diferença entre a ativação da rede no conjunto de dados de remoção e o modelo alvo.

Shibata et al. apresentam a Aprendizagem com Esquecimento Seletivo (LSF) [Shibata et al. 2021] para conseguir o esquecimento seletivo ao nível da classe na aprendizagem contínuo (lifelong learning). Os autores introduzem uma função de perda F com quatro componentes: perda de classificação F_C para garantir a precisão da classificação, perda mnemónica F_M que associa cada classe a um código incorporado, perda de esquecimento seletivo F_{SF} para remover classes marcadas para remoção e perda de regularização F_R para evitar o esquecimento catastrófico, mencionado como um dos desafios do DM presentes na seção 2.3.

Os códigos mnemónicos permitem o esquecimento seletivo de qualquer classe, eliminando o seu código sem os dados originais. O modelo pode esquecer seletivamente certas classes através de uma nova otimização da perda F, como ilustrado na fórmula 8.

$$F = F_C + F_M + F_{SF} + F_R \tag{8}$$

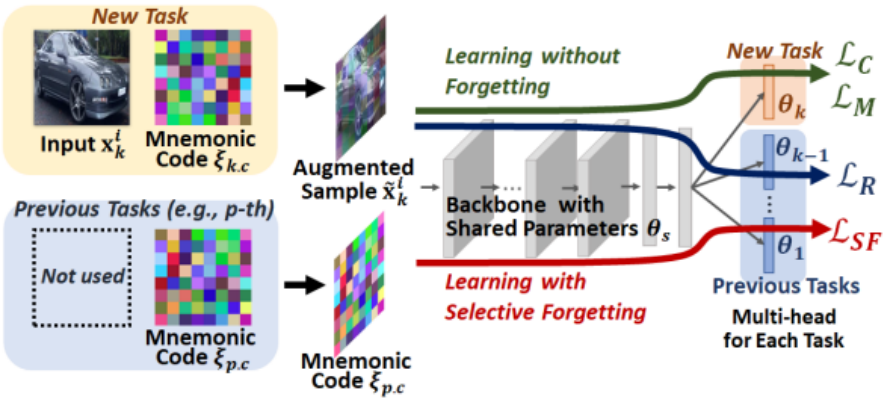

Figure 4.19. Visão geral do método LSF

A Figura 4.20 apresenta o quadro geral das diferentes abordagens de desaprendizagem aproximada baseada na reotimização após a remoção dos dados, contendo características como: objetivos, modelagem, pontos fortes e limitações de cada método.

4.7.2.3. Approximate Unlearning baseada na atualização do gradiente

A desaprendizagem aproximada baseada em atualizações de gradiente faz pequenos ajustes aos parâmetros do modelo para modificá-lo depois de remover ou adicionar incrementalmente pontos de dados. Esses métodos geralmente seguem uma estrutura de duas etapas para atualizar modelos treinados após pequenas alterações de dados sem retreinamento completo: a primeira consiste em inicializar os parâmetros do modelo usando o modelo

Paper	Goal	Design Ideas	Strengths	Weaknesses
Golatkar *et al.* (2020a) [41]	Selective forgetting in deep networks	• Optimal quadratic scrubbing on weights • Add noise tailored to the loss landscape	• Formal definition of selective forgetting • Provides upper bound on remaining info	• Rely on the stability of SGD • Computationally expensive, limited scalability
Golatkar *et al.* (2020b) [72]	Selective forgetting from input-output observations	• Analysis based on final activations • NTK-based scrubbing	• Provides tighter black-box bounds with limited queries • Handles over-parameterized models	• Relies on linearization assumptions • Computationally expensive
Golatkar *et al.* (2021) [73]	Selective forgetting for large deep networks	• Mixed-privacy setting • User weights obtained by minimizing quadratic loss	• Maintaining accuracy on large vision tasks • Provides information bounds	• Relies on the strong convexity assumptions • Forgetting quality depends on data subsets
Shibata *et al.* (2021) [74]	Class-level selective forgetting in lifelong learning	• Mnemonic codes • New loss function with four components	• Class-level removing • No need for original data	• Customized for image data • Computational cost of embedding codes

Figure 4.20. Visão geral dos métodos de desaprendizagem aproximada baseada na reotimização após a remoção dos dados [Xu et al. 2024]

treinado anteriormente. Já o segunda consiste em executar algumas etapas de atualização de gradiente nos novos dados.

Dentre os representantes desta categoria, é possível citar o Deltagrad [Wu et al. 2020] caracterizado por se adaptar os modelos de forma eficiente a pequenas alterações no conjunto de treino, utilizando o gradiente armazenado em cache e a informação sobre os parâmetros durante o processo de treino original. O algoritmo inclui dois casos: iteração de burn-in e outras iterações.

Outra abordagem é chamada de Descent-to-Delete, proposta por [Neel et al. 2021] introduz um algoritmo básico de descida de gradiente que começa com o modelo anterior e executa um número limitado de atualizações de descida de gradiente. Este processo assegura que os parâmetros do modelo permanecem muito próximos dos parâmetros ótimos. O ruído gaussiano é aplicado aos parâmetros do modelo para garantir a indistinguibilidade de qualquer entidade próxima do modelo ótimo. Para dados de elevada dimensão, divide os dados e otimiza independentemente cada partição. A Figura 4.21 consiste no pseudocódigo do processo de desaprendizado do Descent-to-Delete.

Algorithm 2 $\mathcal{R}_{\mathcal{A}}$: ith **Unlearning** for Perturbed Gradient Descent

Input: dataset $\mathcal{D}_{i-1}$, update u_i, model θ_i
Update dataset $\mathcal{D}_i = \mathcal{D}_{i-1} \circ u_i$
Initialize $\theta'_0 = \theta_i$
for $t = 1, 2, \ldots T_i$ **do**
  $\theta'_t = \text{Proj}_\Theta\left(\theta'_{t-1} - \eta_t \nabla f_{\mathcal{D}_i}(\theta'_{t-1})\right)$
Output: $\hat{\theta}_i = \theta'_{T_i}$; // Secret output

Figure 4.21. Pseudocódigo Descent-to-Delete

A Figura 4.22 apresenta o quadro geral das diferentes abordagens de desaprendizagem aproximada baseada na função de influência dos dados removidos, contendo características como: objetivos, modelagem, pontos fortes e limitações de cada método.

Paper	Goal	Design Ideas	Strengths	Weaknesses
DeltaGrad [40]	Efficiently retrain models after minor data changes	• Cached training parameters and gradients • Burn-in iterations + L-BFGS approximation	• Applicable to general models with GD/SGD • Support both addition and removal	• Gradient storage cost • Relies on strong convexity assumptions
FedRecover [75]	FL model recovery after poisoning attacks	• Server estimates updates • L-BFGS approximation +corrections	• Estimate clients' model updates to reduce communication cost • Scalable to numerous clients	• Storing historical information • Convexity assumptions
Descent-to -Delete [76]	Unlearning with efficiency and privacy guarantees	• Gradient descent perturbations • Data partitioning	• Gaussian noise for indistinguishability • Handles arbitrary updates • Improved accuracy for high-dimensional data	• Convexity assumptions • Accuracy/efficiency tradeoff
BAERASER [77]	Remove backdoor	• Trigger pattern recovery • Gradient ascent unlearning	• Removes backdoors without retraining data • Prevents catastrophic forgetting	• Recovered triggers not identical • Applicable to backdoor only

Figure 4.22. Visão geral dos métodos de desaprendizagem aproximada baseada na reotimização após a remoção dos dados [Xu et al. 2024]

4.7.2.4. Approximate Unlearning em grafos

Os dados estruturados em grafos colocam desafios únicos à desaprendizagem de máquinas devido às dependências inerentes entre pontos de dados ligados. Os métodos tradicionais de DM concebidos para dados independentes não têm frequentemente em conta as interações complexas presentes nos dados de grafos.

Em primeiro lugar, a interdependência dos dados é um desafio fundamental na desaprendizagem de grafos. Dado um nó num grafo como alvo de remoção, é necessário remover a sua influência e a sua potencial influência nos vizinhos multi-hop. Para resolver este problema, [Wu et al. 2023] propuseram uma Função de Influência de Grafo (GIF) para considerar essa influência estrutural do nó/aresta/caraterística nos seus vizinhos. A GIF estima as alterações dos parâmetros em resposta a perturbações de massa nos dados removidos, introduzindo um termo de perda adicional relacionado com os vizinhos afetados. O GIF fornece uma forma de explicar os efeitos da desaprendizagem das características dos nós.

Outro trabalho que aborda o problema da interdependência dos dados em grafos, [Cheng et al. 2023] propôs o GNNDELETE, um método que integra um novo operador de eliminação para lidar com o impacto da eliminação de arestas em grafos. Eles introduziram duas propriedades chave, nomeadamente a consistência da aresta eliminada e a influência da vizinhança, para limitar o impacto da eliminação de arestas apenas à vizinhança local. A consistência da aresta eliminada garante que a eliminação de uma aresta não afeta a representação de outras arestas na mesma vizinhança. A influência da vizinhança garante que a eliminação de uma aresta afeta apenas os seus vizinhos directos e não todo o grafo.

A Figura 4.23 apresenta a visão geral do funcionamento do método. Dado um modelo GNN treinado e um pedido de eliminação de arestas, o GNNDELETE produz representações não aprendidas de forma eficiente, aprendendo apenas um pequeno operador de eliminação W_D. Ele também garante a qualidade da representação minimizando uma função de perda que satisfaz as duas propriedades-chave propostas no trabalho.

Um segundo problema envolvendo grafos, é que a maioria dos métodos de de-

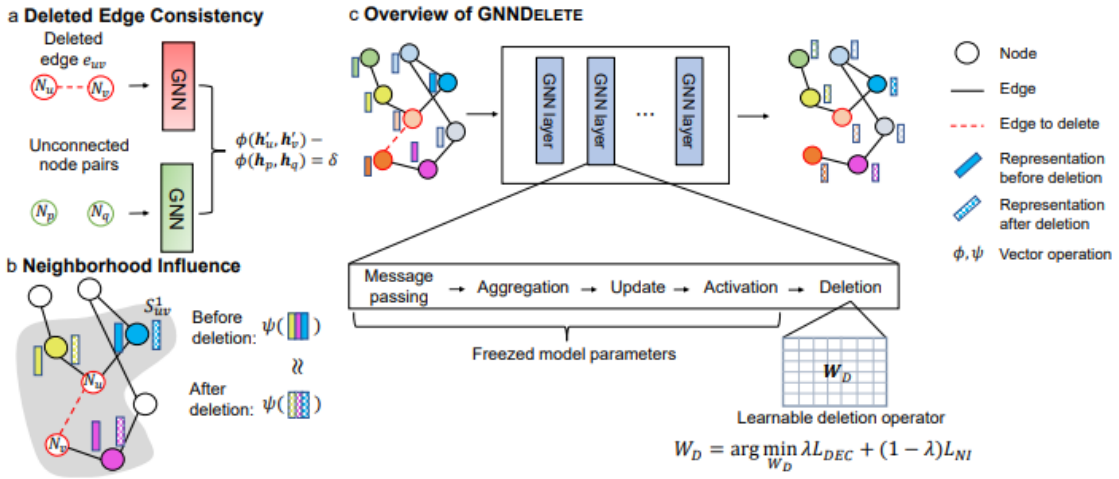

Figure 4.23. Visão geral do GNNDELETE

saprendizagem de grafos foi concebida para o cenário de aprendizado transdutivos, em que o grafo é estático e a informação do grafo de teste está disponível durante o treino. No entanto, muitos grafos do mundo real são dinâmicos, adicionando continuamente novos nós e arestas. Para resolver este problema, [Wang et al. 2023] propuseram o GUIded InDuctivE Graph Unlearning framework (GUIDE) para efetuar a desaprendizagem de grafos dinâmicos. O GUIDE inclui um particionamento justo e equilibrado dos grafos guiados, uma reparação eficiente dos subgrafos e uma agregação baseada na similaridade. O particionamento equilibrado garante que o tempo de reciclagem de cada fragmento seja semelhante, e o reparo de subgrafos e a agregação baseada em similaridade reduzem os efeitos colaterais do particionamento de grafos, melhorando assim a utilidade do modelo. A Figura 4.24 apresenta a visão geral do framework GUIDE.

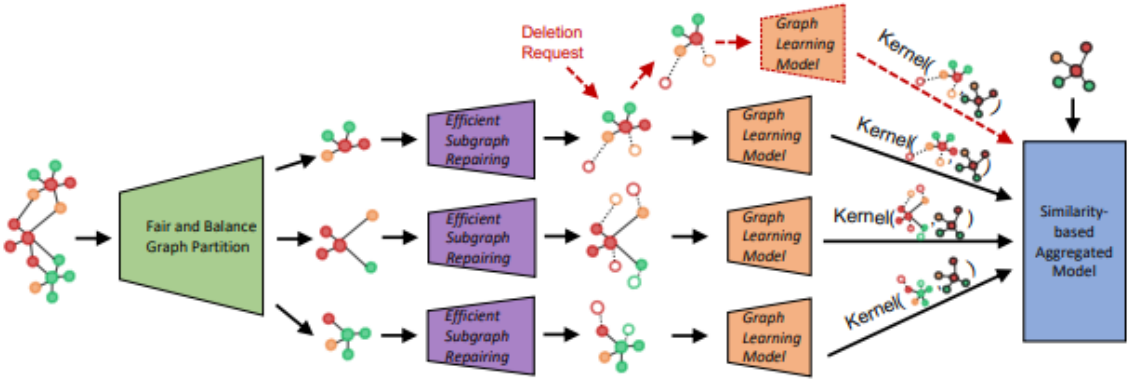

Figure 4.24. Visão geral do GUIDE

Em terceiro lugar, é mais difícil conseguir a desaprendizagem de grafos mantendo o desempenho do modelo quando o número de dados de treino é limitado. Para resolver este problema, [Pan et al. 2023] propuseram um método aproximado não linear de desaprendizagem de grafos baseado na Transformada de Dispersão de Grafos (GST). A GST é estável sob pequenas perturbações nas características e topologias dos grafos,

tornando-a um método robusto para o processamento de dados de grafos. Além disso, as GSTs não são treináveis e todos os coeficientes wavelet nas GSTs são construídos analiticamente, tornando as GSTs computacionalmente mais eficientes e exigindo menos dados de treino do que as GNNs

Há também abordagens no domínio da aprendizagem de embeddings de grafos com conhecimento federados (KG). Para esta área, uma preocupação importante é o tratamento eficaz da heterogeneidade dos dados e dos desafios da retenção de conhecimentos. A FedLU [Zhu et al. 2023] adota um processo de esquecimento em duas fases, de interferência e decaimento. Na primeira etapa de interferência retroativa, o FedLU efectua confusões duras e suaves, uma vez que a teoria da interferência afirma que o esquecimento ocorre quando as memórias competem e interferem com outras memórias [Wixted 2021].

A Figura 4.25 apresenta o quadro geral das diferentes abordagens de desaprendizagem aproximada baseada na função de influência dos dados removidos, contendo características como: objetivos, modelagem, pontos fortes e limitações de cada método.

Paper	Goal	Design Ideas	Strengths	Weaknesses
GIF [81]	General unlearning strategy for GNNs	Graph influence functions considering neighbors' influence	• Applicable to different models • Supports various removal tasks • Closed-form solution	• Focus on the classification • Memory-intensive
GNNDelete [82]	Efficient unlearning for GNNs	• Deleted Edge Consistency • Neighborhood Influence	• Flexible removal operator applicable to any GNN • Supports various removal tasks	• Limited to transductive learning
Chien *et al.* (2022) [83]	Graph-structured data unlearning	• Update model parameters based on gradient difference	• Analyze for node/edge/feature unlearning • Strong theoretical guarantees	• Analysis limited to linear models • Loose worst-case bounds
GUIDE [84]	Inductive graph unlearning	• Guided graph partitioning • Subgraph repairing • Similarity-based aggregation	• Repair subgraphs independently • Enables unlearning on evolving graphs	• Increased memory cost • Generalizing partition fairness needs exploration
Pan*et al.* (2023) [85]	Unlearning for GNNs with limited training data	Use GSTs for graph embeddings	• Nonlinear approximate graph unlearning • Theoretical guarantees	• Limited to classification • Bounds are loose
FedLU [86]	Unlearning for knowledge graphs in FL	• Mutual knowledge distillation • Retroactive interference & passive decay for unlearning	• Unlearning for heterogeneous federated KGs • Mutual knowledge distillation reduces bias in local and global models	• High computational cost • Applicability to other graph data needs exploration

Figure 4.25. Visão geral dos métodos de desaprendizagem aproximada em grafos [Xu et al. 2024]

4.7.3. Outras formas de classificação

Outros surveys como [Nguyen et al. 2022] apresentam diferentes critérios de classificação das abordagens de desaprendizado, baseados no nível de agnosticismo e a necessidade de dados de treinamento.

4.7.3.1. Abordagens Model-agnostic

As metodologias de desaprendizagem de máquinas agnósticas em relação aos modelos incluem processos ou frameworks de desaprendizagem que são aplicáveis a diferentes modelos. No entanto, em alguns casos, as garantias teóricas são fornecidas apenas para uma classe de modelos (por exemplo, modelos lineares). No entanto, continuam a ser consideradas agnósticas em relação aos modelos, uma vez que as suas ideias centrais são

aplicáveis a modelos complexos (por exemplo, redes neurais profundas) com resultados práticos.

O trabalho de [Gupta et al. 2021] propuseram um mecanismo de desaprendizado especificamente para pedidos de remoção de dados em streaming. Estes pedidos são também adaptativos, o que significa que os dados a remover dependem do modelo atual de desaprendizado.

Já o trabalho de [Golatkar et al. 2020a] introduz um limite superior computável para algoritmos de aprendizagem baseados no stochastic gradient descent (SGD), especialmente redes neurais profundas. A ideia central baseia-se na noção de perturbação (ruído) para mascarar o pequeno resíduo incorrido pela atualização baseada no gradiente. A ideia é aplicável a outros casos, embora não sejam dadas garantias teóricas.

Ullah et al. [156] continuaram a estudar a desaprendizagem de máquinas no contexto do SGD e dos pedidos de remoção em fluxo contínuo. Provaram que existe um processo de desaprendizagem que satisfaz a desaprendizagem exacta em qualquer momento do pedido de remoção em fluxo contínuo,

4.7.3.2. Abordagens Model-intrinsic

As abordagens intrínsecas ao modelo incluem métodos de desaprendizagem concebidos para um tipo específico de modelos. Embora sejam intrínsecas ao modelo, as suas aplicações não são necessariamente limitadas, uma vez que muitos modelos de aprendizagem automática podem partilhar o mesmo tipo.

O trabalho do [Izzo et al. 2021] propôs um método de desaprendizagem aproximado para modelos lineares e logísticos baseado em funções de influência. Aproximaram o cálculo da matriz Hessiana com uma atualização dos resíduos do projeto, conhecido pela sigla PRU, que combina métodos de gradiente com dados sintéticos. Este método é adequado para esquecer pequenos grupos de pontos de um modelo aprendido. A Figura 4.26 apresenta o pseudocódigo do PRU.

Algorithm 1 The projective residual update

procedure PRU($X, Y, H, \theta^{\text{full}}, k$)
  $\hat{y}'_1, \ldots, \hat{y}'_k \leftarrow \text{LKO}(X, Y, H, k)$
  $S^{-1} \leftarrow \text{PseudoInv}(\sum_{i=1}^{k} x_i x_i^\intercal)$
  $\nabla L \leftarrow \sum_{i=1}^{k} (\theta^{\text{full}\intercal} x_i - \hat{y}'_i) x_i$
  return $\theta^{\text{full}} - \text{FastMult}(S^{-1}, \nabla L)$
end procedure

Figure 4.26. Pseudocódigo do PRU

Motivados pelo regulamento da UE “Direito a ser esquecido”, iniciamos um estudo dos problemas de eliminação de dados estatísticos nos casos em que os dados dos

utilizadores estão acessíveis apenas durante um período de tempo limitado. O artigo de [Li et al. 2020] formularam um caso especial da configuração online em que os dados só estão acessíveis durante um período de tempo limitado, logo não ocorre um processo de treinamento completo. Mais precisamente, o sistema dispõe de uma memória constante para armazenar dados históricos ou um esboço de dados, e tem de fazer previsões num período de tempo limitado. Embora os dados a esquecer possam ser desaprendidos de um modelo em tempo real utilizando um esquema de arrependimento na memória, este processo específico de desaprendizagem só é aplicável à regressão linear.

Já [Schelter et al. 2021] propuseram uma solução de desaprendizagem para árvores extremamente aleatórias, medindo a robustez das decisões de divisão. Uma decisão de divisão é robusta se a remoção de k itens de dados não reverter essa divisão. Note-se que k pode ser limitado, e é muitas vezes pequeno como apenas um em dez mil usuários que querem remover os seus dados de cada vez). Como resultado, o algoritmo de aprendizagem é redesenhado de forma a que a maioria das divisões, especialmente as de alto nível, sejam robustas. Para as divisões não robustas, todas as variantes de sub-árvore são desenvolvidas a partir de todos os candidatos a divisão e mantidas até que um pedido de remoção reveja essa divisão. Quando isso acontece, a divisão é mudada para a sua variante com maior ganho de Gini. Como resultado, o processo de desaprendizagem envolve o recálculo dos ganhos de Gini e a atualização das divisões, se necessário.

4.7.3.3. Abordagens Data-driven

Dentre os trabalhos que propuseram frameworks de desaprendizado cujas decisões são baseadas nos dados. O trabalho de [Huang et al. 2021] apresenta a ideia de ruído de minimização de erros, que leva um modelo a pensar que não há nada a aprender com um determinado conjunto de dados (ou seja, a perda não muda). No entanto, só pode ser utilizado para proteger um determinado item de dados antes de o modelo ser treinado.

Por outro lado, [Tarun et al. 2023] propôs um ruído que maximiza o erro para prejudicar o modelo numa classe de dados alvo (a ser esquecida). No entanto, esta tática não funciona em itens de dados específicos, uma vez que é mais fácil interferir com a previsão de um modelo numa classe inteira do que num item de dados específico dessa classe.

O trabalho de [Zeng et al. 2023] sugeriram um novo método de modelação da influência dos dados, adicionando termos de regularização ao algoritmo de aprendizado. Embora este método seja independente do modelo, requer a intervenção no processo de formação do modelo original. Além disso, só é aplicável a problemas de aprendizagem convexa e a redes neurais profundas.

4.7.3.4. Abordagens Zero-Shot Unlearning

Este cenário ocorre se os dados de treino não forem acessíveis por um método de desaprendizagem, ou seja, a desaprendizagem ocorre com zero amostras de treino. No entanto, não é trivial resolver este problema em casos genéricos.

Esta abordagem foi introduzida no trabalho de [Chundawat et al. 2023b] estudou um cenário deste tipo para a classificação com classes a serem esquecidas, com a ideia de que os resultados do modelo desaprendido devem assemelhar-se aos de um modelo treinado novamente. Além disso, os autores propões duas novas soluções para a desaprendizagem de máquinas com zero disparos, com base em (a) minimização de erros - maximização de ruído e (b) transferência de conhecimentos. Estes métodos removem a informação dos dados esquecidos do modelo, mantendo a eficácia do modelo nos dados retidos. Dentre os benefícios relacionados a privacidade, os autores apontam que esta abordagem oferece uma boa proteção contra os ataques de inversão do modelo e os ataques de inferência de membros.

4.7.3.5. Abordagens One-Shot Unlearning

Para este cenário, o trabalho [Cong and Mahdavi 2022] propôs uma solução de desaprendizagem one-shot, caracterizada por requerer apenas o acesso aos dados a serem esquecidos. A ideia é inspirada na arquitetura de uma rede neural em grafos linear (GNN), em que as não-linearidades numa GNN típica são substituídas por uma matriz de peso único entre camadas convolucionais consecutivas. Apesar da sua extensão linear sobre todas as características dos nós de entrada, essas GNN lineares têm mostrado um desempenho competente, por exemplo, SGC [Wu et al. 2019] e APPNP [Klicpera et al. 2019]. Ao utilizar esta propriedade, os autores propuseram um processo exato de desaprendizagem a nível algorítmico baseado em operações lineares como a projeção e a recombinação.

4.7.3.6. Abordagens Few-Shot Unlearning

A desaprendizagem few-shot consiste em um algoritmo de desaprendizagem que apenas recebe uma pequena parte dos dados a serem esquecidos D_f. Esta definição é útil para os casos em que o conjunto a esquecer contém dados mal rotulados ou se pretende remover os efeitos maliciosos de alguns dados num modelo.

Para este cenário, o trabalho de [Yoon et al. 2022] formula o problema da desaprendizagem com poucas amostras dos dados alvo, especificando as intenções por detrás do pedido de desaprendizagem (por exemplo, desaprendizagem pura e simples, correção de erros de rotulagem, proteção da privacidade), e concebemos uma estrutura simples que (i) recupera um proxy dos dados de treino através da inversão do modelo, explorando plenamente a informação disponível no contexto da desaprendizagem; (ii) ajusta o proxy de acordo com a intenção de desaprendizagem; e (iii) atualiza o modelo com o proxy ajustado. Além disos, Demonstram que o método utilizado por eles, com apenas um subconjunto de dados-alvo, pode superar os métodos de desaprendizagem mais avançados, mesmo com uma indicação completa dos dados-alvo.

4.8. Aplicações de DM

Dentre as aplicações do cotidiano que o desaprendizado de máquinas é extremamente útil, é possível mencionar as ações de segurança contra os ataques cibernéticos. Em muitas situações, a natureza dos ciberataques inclui não só o roubo de dados mas também a

penetração nos modelos de inteligência artificial do cliente, utilizadas pela maioria dos servidores e sistemas atuais. Nesses casos, é introduzida no sistema uma falha que contém dados contraditórios, com os quais o modelo acaba por aprender [Wazid et al. 2022]. O DM é utilizado quando todos os dados hostis têm de ser apagados e o modelo tem de ser reconstruído [Chamola et al. 2023].

Além dos ciberataques, outra aplicação importante envolve os sistemas de recomendação. O trabalho de [Xu et al. 2023] aponta que quando os usuários querem esquecer as suas ações passadas, como términos de relacionamento como no exemplo da 4.27, eles esperam que as plataformas de recomendação apaguem os dados seletivos ao nível do modelo. Idealmente, tendo em conta o histórico de um determinado usuário, o sistema de recomendação pode ser desfeito ou "esquecido", como se o registo não fizesse parte do treinamento. Logo, é desenvolvido o Unlearn-ALS para pedidos de desaprendizado no Netflix.

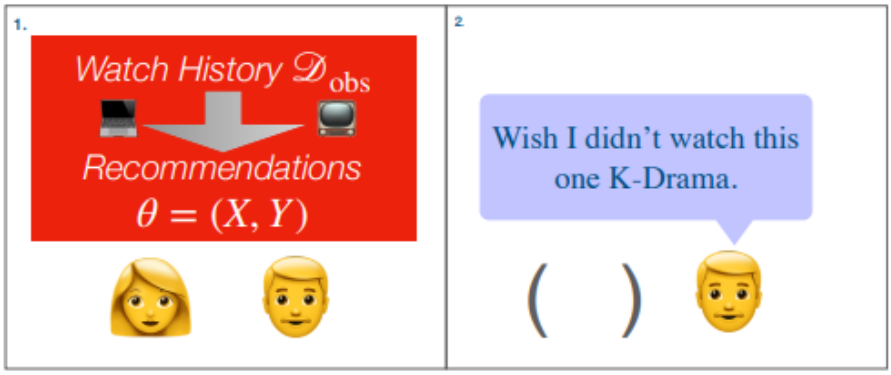

Figure 4.27. Usuário solicita ao Netflix que apague os históricos de dados

Além dos sistemas de recomendação, DM tem tido um papel vital na tradução de idiomas, visto que os algoritmos de desaprendizagem são essenciais para permitir que os modelos invertam palavras e padrões linguísticos previamente adquiridos [Hutchins 1986]. Os métodos DM podem ser utilizados para completar este processo de desaprendizagem. Estes modelos podem fornecer aos clientes traduções mais precisas e atualizadas desaprendizagem de componentes linguísticos desatualizados.

Outra aplicação que vem ganhando destaque na indústria são os modelos de reconhecimento facial baseados em ML, são frequentemente utilizados para controlar o acesso aos espaços de trabalho e impedir a entrada ilegal. As informações de um funcionário têm de ser eliminadas da lista de usuários aprovados após a sua saída da organização. O retreinamento de um modelo com dados novos pode ser computacionalmente exigente. Nesses casos, os dados de um de um empregado ou grupo de empregados podem ser apagados permanentemente utilizando técnicas DM [Rosenfeld 2002].

Portanto, as inúmeras aplicações que podem demandar desaprendizado ratifica a importância deste paradigma para aprimorar modelos de ML, permitindo a remoção seletiva de informações indesejadas. Isso tem implicações importantes em termos de privacidade e segurança.

References

[Bourtoule et al. 2021] Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2021). Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE.

[Brophy and Lowd 2021] Brophy, J. and Lowd, D. (2021). Machine unlearning for random forests. In *International Conference on Machine Learning*, pages 1092–1104. PMLR.

[Buchele et al. 2016] Buchele, G. T., Teza, P., Müller, I. R. F., and Souza, J. A. d. (2016). Desaprendizagem organizacional: um estudo de campo na universidade federal de santa catarina. *Revista de Administração Contemporânea*, 20:64–83.

[Cao and Yang 2015] Cao, Y. and Yang, J. (2015). Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE.

[Capanema 2020] Capanema, W. A. (2020). A responsabilidade civil na lei geral de proteção de dados. *Cadernos Jurídicos, São Paulo, ano*, 21:163–170.

[Chamola et al. 2023] Chamola, V., Goyal, A., Sharma, P., Hassija, V., Binh, H. T. T., and Saxena, V. (2023). Artificial intelligence-assisted blockchain-based framework for smart and secure emr management. *Neural Computing and Applications*, 35(31):22959–22969.

[Chao 2011] Chao, W.-L. (2011). Machine learning tutorial. *Digital Image and Signal Processing*.

[Chen et al. 2021] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. (2021). When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 896–911.

[Chen et al. 2022] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. (2022). Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pages 499–513.

[Cheng et al. 2023] Cheng, J., Dasoulas, G., He, H., Agarwal, C., and Zitnik, M. (2023). Gnndelete: A general strategy for unlearning in graph neural networks. *arXiv preprint arXiv:2302.13406*.

[Chundawat et al. 2023a] Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. (2023a). Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7210–7217.

[Chundawat et al. 2023b] Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. (2023b). Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*.

[Cong and Mahdavi 2022] Cong, W. and Mahdavi, M. (2022). Grapheditor: An efficient graph representation learning and unlearning approach.

[Dang 2021] Dang, Q.-V. (2021). Right to be forgotten in the age of machine learning. In *Advances in Digital Science: ICADS 2021*, pages 403–411. Springer.

[Du et al. 2019] Du, M., Chen, Z., Liu, C., Oak, R., and Song, D. (2019). Lifelong anomaly detection through unlearning. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1283–1297.

[Ekstrand et al. 2011] Ekstrand, M. D., Ludwig, M., Konstan, J. A., and Riedl, J. T. (2011). Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 133–140.

[Ferreira 1999] Ferreira, A. d. H. (1999). *Novo Aurélio Século XXI: o dicionário da língua portuguesa. rev. e ampl.* Nova Fronteira.

[Gao et al. 2012] Gao, H., Chen, Y., Lee, K., Palsetia, D., and Choudhary, A. N. (2012). Towards online spam filtering in social networks. In *NDSS*, volume 12, pages 1–16.

[Ginart et al. 2019] Ginart, A., Guan, M., Valiant, G., and Zou, J. Y. (2019). Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32.

[Golatkar et al. 2020a] Golatkar, A., Achille, A., and Soatto, S. (2020a). Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312.

[Golatkar et al. 2020b] Golatkar, A., Achille, A., and Soatto, S. (2020b). Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 383–398. Springer.

[Guo et al. 2019] Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. (2019). Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*.

[Guo et al. 2022] Guo, T., Guo, S., Zhang, J., Xu, W., and Wang, J. (2022). Efficient attribute unlearning: Towards selective removal of input attributes from feature representations. *arXiv preprint arXiv:2202.13295*.

[Gupta et al. 2021] Gupta, V., Jung, C., Neel, S., Roth, A., Sharifi-Malvajerdi, S., and Waites, C. (2021). Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330.

[He et al. 2021] He, Y., Meng, G., Chen, K., He, J., and Hu, X. (2021). Deepobliviate: a powerful charm for erasing data residual memory in deep neural networks. *arXiv preprint arXiv:2105.06209*.

[Holan and Phillips 2004] Holan, P. M. d. and Phillips, N. (2004). Remembrance of things past? the dynamics of organizational forgetting. *Management science*, 50(11):1603–1613.

[Huang et al. 2021] Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. (2021). Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*.

[Hutchins 1986] Hutchins, W. J. (1986). *Machine translation: past, present, future*. Ellis Horwood Chichester.

[Izzo et al. 2021] Izzo, Z., Smart, M. A., Chaudhuri, K., and Zou, J. (2021). Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR.

[Klicpera et al. 2019] Klicpera, J., Bojchevski, A., and Günnemann, S. (2019). Combining neural networks with personalized pagerank for classification on graphs. In *International conference on learning representations*.

[Koh and Liang 2017] Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

[Koh et al. 2019] Koh, P. W. W., Ang, K.-S., Teo, H., and Liang, P. S. (2019). On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32.

[Langley and Carbonell 1984] Langley, P. and Carbonell, J. G. (1984). Approaches to machine learning. *Journal of the American Society for Information Science*, 35(5):306–316.

[Laskov and Šrndić 2011] Laskov, P. and Šrndić, N. (2011). Static detection of malicious javascript-bearing pdf documents. In *Proceedings of the 27th annual computer security applications conference*, pages 373–382.

[Li et al. 2020] Li, Y., Wang, C.-H., and Cheng, G. (2020). Online forgetting process for linear regression models. *arXiv preprint arXiv:2012.01668*.

[Liu et al. 2020] Liu, G., Ma, X., Yang, Y., Wang, C., and Liu, J. (2020). Federated unlearning. *arXiv preprint arXiv:2012.13891*.

[Magdziarczyk 2019] Magdziarczyk, M. (2019). Right to be forgotten in light of regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec. In *6th International Multidisciplinary Scientific Conference on Social Sciences and Art Sgem 2019*, pages 177–184.

[Marchant et al. 2022] Marchant, N. G., Rubinstein, B. I., and Alfeld, S. (2022). Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7691–7700.

[Mehrabi et al. 2021] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35.

[Mehta et al. 2022] Mehta, R., Pal, S., Singh, V., and Ravi, S. N. (2022). Deep unlearning via randomized conditionally independent hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10422–10431.

[Mercuri et al. 2022] Mercuri, S., Khraishi, R., Okhrati, R., Batra, D., Hamill, C., Ghasempour, T., and Nowlan, A. (2022). An introduction to machine unlearning. *arXiv preprint arXiv:2209.00939*.

[Neel et al. 2021] Neel, S., Roth, A., and Sharifi-Malvajerdi, S. (2021). Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR.

[Nguyen et al. 2020] Nguyen, Q. P., Low, B. K. H., and Jaillet, P. (2020). Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33:16025–16036.

[Nguyen et al. 2022] Nguyen, T. T., Huynh, T. T., Nguyen, P. L., Liew, A. W.-C., Yin, H., and Nguyen, Q. V. H. (2022). A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

[Pan et al. 2023] Pan, C., Chien, E., and Milenkovic, O. (2023). Unlearning graph classifiers with limited data resources. In *Proceedings of the ACM Web Conference 2023*, pages 716–726.

[Pardau 2018] Pardau, S. L. (2018). The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68.

[Qu et al. 2023] Qu, Y., Yuan, X., Ding, M., Ni, W., Rakotoarivelo, T., and Smith, D. (2023). Learn to unlearn: A survey on machine unlearning. *arXiv preprint arXiv:2305.07512*.

[Ren et al. 2020] Ren, K., Zheng, T., Qin, Z., and Liu, X. (2020). Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360.

[Rosenfeld 2002] Rosenfeld, A. (2002). Face recognition: A literature survey. *UMD, 2002*.

[Schelter et al. 2021] Schelter, S., Grafberger, S., and Dunning, T. (2021). Hedgecut: Maintaining randomised trees for low-latency machine unlearning. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1545–1557.

[Sekhari et al. 2021] Sekhari, A., Acharya, J., Kamath, G., and Suresh, A. T. (2021). Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086.

[Shibata et al. 2021] Shibata, T., Irie, G., Ikami, D., and Mitsuzumi, Y. (2021). Learning with selective forgetting. In *IJCAI*, volume 3, page 4.

[Sommer et al. 2020] Sommer, D. M., Song, L., Wagh, S., and Mittal, P. (2020). Towards probabilistic verification of machine unlearning. *arXiv preprint arXiv:2003.04247*.

[Sommer et al. 2022] Sommer, D. M., Song, L., Wagh, S., and Mittal, P. (2022). Athena: Probabilistic verification of machine unlearning. *Proceedings on Privacy Enhancing Technologies*.

[Tarun et al. 2023] Tarun, A. K., Chundawat, V. S., Mandal, M., and Kankanhalli, M. (2023). Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*.

[Thudi et al. 2022] Thudi, A., Deza, G., Chandrasekaran, V., and Papernot, N. (2022). Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE.

[Veale et al. 2018] Veale, M., Binns, R., and Edwards, L. (2018). Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083.

[Wang et al. 2023] Wang, C.-L., Huai, M., and Wang, D. (2023). Inductive graph unlearning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3205–3222.

[Warnecke et al. 2021] Warnecke, A., Pirch, L., Wressnegger, C., and Rieck, K. (2021). Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*.

[Wazid et al. 2022] Wazid, M., Das, A. K., Chamola, V., and Park, Y. (2022). Uniting cyber security and machine learning: Advantages, challenges and future research. *ICT express*, 8(3):313–321.

[Weber 2011] Weber, R. H. (2011). The right to be forgotten: more than a pandora's box. *J. Intell. Prop. Info. Tech. & Elec. Com. L.*, 2:120.

[Wixted 2021] Wixted, J. T. (2021). The role of retroactive interference and consolidation in everyday forgetting. In *Current issues in memory*, pages 117–143. Routledge.

[Wu et al. 2019] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

[Wu et al. 2022] Wu, G., Hashemi, M., and Srinivasa, C. (2022). Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8675–8682.

[Wu et al. 2023] Wu, J., Yang, Y., Qian, Y., Sui, Y., Wang, X., and He, X. (2023). Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, pages 651–661.

[Wu et al. 2020] Wu, Y., Dobriban, E., and Davidson, S. (2020). Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pages 10355–10366. PMLR.

[Xu et al. 2024] Xu, J., Wu, Z., Wang, C., and Jia, X. (2024). Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

[Xu et al. 2023] Xu, M., Sun, J., Yang, X., Yao, K., and Wang, C. (2023). Netflix and forget: Efficient and exact machine unlearning from bi-linear recommendations. *arXiv preprint arXiv:2302.06676*.

[Yoon et al. 2022] Yoon, Y., Nam, J., Yun, H., Lee, J., Kim, D., and Ok, J. (2022). Few-shot unlearning by model inversion. *arXiv preprint arXiv:2205.15567*.

[Zeng et al. 2023] Zeng, Y., Wang, J. T., Chen, S., Just, H. A., Jin, R., and Jia, R. (2023). Modelpred: A framework for predicting trained model from training data. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 432–449. IEEE.

[Zhu et al. 2023] Zhu, X., Li, G., and Hu, W. (2023). Heterogeneous federated knowledge graph embedding learning and unlearning. In *Proceedings of the ACM Web Conference 2023*, pages 2444–2454.