Capítulo

3

Privacidade do Usuário em Aprendizado Colaborativo: *Federated Learning*, da Teoria à Prática

Helio N. C. Neto (UFF)
Diogo M. F. Mattos (UFF)
Natalia C. Fernandes (UFF)

Abstract

The growing number of Internet of Things devices represents an uninterrupted data source that waits for machine learning mechanisms to run. However, the presence of personal data and the difficulty in ensuring users' privacy make it difficult to extract knowledge under recent restrictions such as the General Law on the Protection of Personal Data. In this chapter, we present the Federated Learning paradigm, which allows collaborative execution of learning models in local data and subsequent aggregation into a centralized global model. The chapter focuses on presenting the principles, applications, as well as challenges and attacks on federated learning, through a practical-theoretical approach with a focus on users' privacy.

Resumo

O número crescente de dispositivos de Internet das Coisas representa uma fonte de dados ininterrupta a espera da execução de mecanismos de aprendizado de máquina. Contudo, a presença de dados pessoais e a dificuldade em garantir a privacidade dos usuários dificultam a extração de conhecimento perante restrições recentes como a Lei Geral de Proteção de Dados Pessoais. Assim, este capítulo apresenta o paradigma de Aprendizado Federado (Federated Learning), que permite a execução colaborativa de modelos de aprendizado em dados locais e posterior agregação em um modelo global centralizado. O capítulo foca em apresentar os princípios, as aplicações, assim como os desafios e os ataques ao aprendizado federado, através de uma abordagem prático-teórica com foco na privacidade dos usuários.

Este capítulo foi realizado com recursos do CNPq, CAPES, RNP, FAPERJ e FAPESP (2018/23062-5).

3.1. Introdução

Previsões apontam que até o final do ano de 2023 o número de dispositivos em rede alcançará um total de 29,3 bilhões de dispositivos, correspondendo a mais de três vezes a total do planeta [2]. No entanto, o número de usuários da Internet deve atingir 5,3 bilhões, o que corresponde a 66% do número de habitantes do planeta. Dessa forma, esses dispositivos apresentam o potencial de implantar operações de sensoriamento coletivo (*crowdsensing*) e fornecer dados para a criação de modelos de aprendizado de máquina para diversos fins [Lim et al., 2020]. Paralelamente, há também um crescente desenvolvimento de técnicas de aprendizado profundo (*Deep Learning* — DL) [Medeiros et al., 2019] e técnicas baseadas em treinamento pelo gradiente descendente estocástico [Lobato et al., 2018, Reis et al., 2020]. Nesse sentido, a coleta de dados das mais diversas fontes permite a realização de diversas pesquisas e aplicações inovadoras [Medeiros et al., 2019]. Na abordagem tradicional [Li et al., 2017], os dados coletados por dispositivos móveis são carregados e processados de forma centralizada em servidores na nuvem, seja um servidor único ou um centro de dados. Contudo, a abordagem centralizada de processamento e fusão de dados implica questões de segurança e privacidade dos dados que são abordadas pelas leis de proteção de dados pessoais em diversos países no mundo.

Políticas de proteção de dados privados, cada vez mais severas, impõem limites para essa abordagem centralizada de extração de conhecimento dos dados. As leis de proteção de dados pessoais estipulam direitos aos titulares dos dados e obrigações às instituições que detêm tais dados. Uma lei de destaque é a GDPR, *General Data Protection Regulation*, vigente em toda a União Europeia (UE) e que estabelece diretrizes quanto ao tratamento, por uma pessoa, empresa ou organização, dos dados pessoais de pessoas na UE [Parlamento Europeu e Conselho da União Européia, 2016]. A legislação enfatiza a preocupação em defender os direitos e as liberdades fundamentais dos indivíduos em relação ao manuseio de seus dados e tem inspirado outros países a assumirem compromissos semelhantes, tais como a Lei Geral de Proteção de Dados Pessoais (LGPD) no Brasil[3], a *California Consumer Privacy Act* (CCPA) nos Estados Unidos[4], a *The Personal Information Protection and Electronic Documents Act* (PIPEDA) no Canadá[5], entre outras. No Brasil, a Lei Geral de Proteção de Dados Pessoais identifica como agentes de tratamento a pessoa natural ou jurídica de direito público ou privado que realiza qualquer operação de tratamento sobre os dados pessoais de outrem [Brasil, 2018]. Dentre os deveres estabelecidos a esses agentes estão a coleta de consentimento explícito do titular do dado e a disponibilização de relatórios que identifiquem as operações de tratamento aplicadas ao dado, incluindo a especificação de seu local de armazenamento, mascaramento do dado e medidas de proteção ao dado.

Outra questão importante relaciona-se ao custo do envio dos dados à nuvem, já que os dispositivos móveis podem estar em uma área com baixa taxa de transferência e altos atrasos. A abordagem centrada na nuvem implica atrasos de propagação que podem incorrer em latências inaceitáveis para determinadas aplicações de decisão em tempo

[2]Disponível em https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[3]Disponível em https://www.lgpdbrasil.com.br/. Último acesso em 25/09/2020.

[4]Disponível em https://www.dlapiperdataprotection.com/?t=law&c=US. Último acesso em 17/09/2020.

[5]Disponível em https://www.dlapiperdataprotection.com/?t=law&c=CA. Último acesso em 17/09/2020.

real [Gupta e Jha, 2015]. A transferência de dados para a nuvem para processamento sobrecarrega tanto as redes de núcleo quanto as redes de acesso. A sobrecarga é ainda mais relevante quando são considerados dados não estruturados, como textos, voz ou vídeo [de Oliveira et al., 2020a, de Oliveira et al., 2020b]. As restrições de envio de dados para nuvem, tanto em banda quanto em atraso, é mais crítica quando o treinamento centrado na nuvem é dependente de redes de acesso sem fio [Mattos et al., 2019, Medeiros et al., 2019]. Assim, propostas atuais consideram o desenvolvimento de aplicações móveis na borda da nuvem (*Mobile Edge Computing* — MEC) [Wang et al., 2019a, Nishio e Yonetani, 2019], em que o aprendizado passa a ser uma tarefa realizada por três atores distintos, dispositivos, borda e nuvem. O aprendizado ancorado no modelo MEC incorre em custos de comunicação significativos e é pouco adequado para aplicações com retreinamento constante [Lim et al., 2020]. Além disso, a terceirização da computação e do processamento de dados em servidores de borda ainda envolvem a transmissão de dados pessoais potencialmente confidenciais, o que pode expor dados sensíveis à privacidade ou mesmo violar leis de proteção de dados pessoais [Parlamento Europeu e Conselho da União Européia, 2016, Brasil, 2018].

O aprendizado federado (*Federated Learning* — FL) é proposto como solução de aprendizado colaborativo com foco na preservação da privacidade e eficiência de comunicação [Brendan McMahan et al., 2017]. O aprendizado federado permite o treinamento com dados reais a partir de dispositivos móveis e a utilização de dados sensíveis à privacidade, sem expor o dono dos dados. O aprendizado federado é uma das principais abordagens para garantir a privacidade dos dados ao passo em que permite a execução de algoritmos de aprendizado de máquina de forma colaborativa sem que os dados privados sejam transferidos para um servidor em nuvem.

O aprendizado federado consiste em um conjunto de tarefas realizadas pelos clientes e servidor do aprendizado federado. Os clientes devem realizar as seguintes tarefas: i) todos os clientes devem recuperar os parâmetros de um modelo treinado do servidor, ii) clientes selecionados aleatoriamente devem atualizar o modelo com seus próprios dados e, após, iii) transferir para o servidor os novos parâmetros do modelo treinado com os dados locais. Ao servidor cabe a tarefa de agregar atualizações providas por diversos clientes, objetivando a melhoraria do modelo global. Após a agregação das contribuições locais, o servidor repassa o novo modelo a todos os clientes, permitindo que todos apliquem o modelo atualizado. As abordagens federadas introduzem a solução de compromisso entre o uso de recursos computacionais (CPU ou GPU) locais para o treinamento dos modelos, ao passo que os clientes podem manter seus dados locais, seguros e privados. Nesse sentido, o aprendizado federado se apresenta como uma abordagem poderosa para manter a privacidade, já que os dados são sempre processados localmente e as operações globais, centralizadas, são voltadas para a agregação de modelos distintos, sem acesso aos dados individuais.

Abordagens ingênuas de aprendizado colaborativo se baseiam nas suposições de que os dados locais dos participantes são independentes e identicamente distribuídos, consistindo um conjunto de dados *iid*. Contudo, a heterogeneidade de dispositivos e a heterogeneidade estatística dos dados contradizem essas suposições, gerando conjuntos de dados que não são independentes e podem ser ter distribuições estatísticas diversas entre os participantes do aprendizado federado, sendo chamados de dados *non-iid*. As

características em um conjunto de dados non-iid são dependentes e não é distribuída de do forma idêntica, ou seja, cada característica não tem a mesma probabilidade de distribuição. Assim, As principais propostas de aprendizado federado se baseiam em modelos de aprendizado profundo e propostas baseadas no treinamento com o algoritmo de gradiente descendente estocástico [Brendan McMahan et al., 2017, Li et al., 2020]. Essas abordagens são prevalentes em detrimento a outros modelos de aprendizado de máquina, pois os principais algoritmos de agregação de modelos federados estão fortemente atrelados as médias ponderadas dos modelos treinados e, portanto, métodos de treinamento que fornecem a direção de otimização do modelo tendem a ser preferidos. Assim, um dos principais algoritmos de aprendizado federado é a média federada (*Federated Average* – FedAvg), que pondera a direção de otimização dos modelos dos clientes de acordo com a quantidade de dados que cada cliente usa para treinar seus modelos.

Embora o aprendizado federado seja proposto como uma solução de privacidade para a realização do aprendizado sobre dados privados, diversos ataques focam no próprio funcionamento do aprendizado para extrair informações dos clientes ou para perverterem o modelo gerado. O primeiro tipo de ataque é o envenenamento do conjunto de dados, em que existem um ou mais participantes maliciosos que tentam enviesar o treinamento utilizando amostras com rótulos falsos. Assim, o atacante pode conduzir o modelo global a falsas inferências. O segundo tipo de ataque é o envenenamento do modelo [Bhagoji et al., 2019], em que o atacante modifica diretamente os parâmetros do modelo local. Esse tipo de ataque é mais efetivo que o anterior, principalmente em ambientes federados com muitos participantes, pois esse ataque não depende do treinamento de um conjunto de dados malicioso. O terceiro é o ataque *Free-Riding* [Weng et al., 2019], no qual um participante visa se beneficiar do modelo global, porém sem contribuir com o processo de treinamento. Nesse caso, o atacante simula ter um conjunto de dados pequeno, realizando pouco ou nenhum esforço computacional e recebe o modelo global treinado colaborativamente pelos outros participantes. O quarto ataque é a inversão do modelo, no qual o atacante possui os modelos de aprendizado de máquina dos usuários e consegue aprender informações sobre os dados. Por fim, há ainda o ataque de inferência dos gradientes, em que o atacante consegue inferir os dados dos usuários através de seus gradientes.

Este capítulo foca em apresentar os princípios, as aplicações e os desafios do aprendizado federado através de uma abordagem prática-teórica com foco na privacidade do usuário. O capítulo aborda os principais conceitos sobre aprendizado federado e os principais tipos de ataques ao ambiente do aprendizado federado, assim como as principais aplicações existentes atualmente. Os principais desafios de pesquisa sobre o aprendizado federado são discutidos em detalhes. Ao final do capítulo, há um roteiro de desenvolvimento de uma aplicação prática visando uma simulação de um ambiente federado em Python. O roteiro da aplicação utiliza a biblioteca TensorFlow[6] e permite que o desenvolvedor crie seus próprios algoritmos de aprendizado federado utilizando os modelos de aprendizado de máquina providos pelo arcabouço Keras[7].

O restante do capítulo está organizado da seguinte forma. A Seção 3.2 apresenta os fundamentos de aprendizado de máquina colaborativo. A Seção 3.3 discute e formaliza

[6]Disponível em https://www.tensorflow.org.
[7]Disponível em https://keras.io/.

o aprendizado federado. A Seção 3.4 descreve as principais aplicações do aprendizado federado. Os ataques ao paradigma de aprendizado federado são elencados na Seção 3.5 e os principais desafios de pesquisa são analisados na Seção 3.6. O roteiro para desenvolvimento de uma aplicação prática é apontado na Seção 3.7. Por fim, a Seção 3.8 conclui o capítulo.

3.2. Fundamentos de Aprendizado de Máquina e Aprendizado Colaborativo

O aprendizado de máquina permite inferir soluções para problemas que possam ser representados por um amplo conjunto de dados. Assim, de forma simplificada, é possível dividir os problemas de aprendizado de máquina em quatro categorias: agrupamentos (*clustering*), classificação, regressão e extração de regras. *Problemas de agrupamento* têm como objetivo aglomerar as amostras em grupos de acordo com a sua similaridade [Medeiros et al., 2019]. Assim, esses algoritmos de aprendizado buscam minimizar a distância entre amostras de dados com características similares em um mesmo grupo, enquanto maximizam a separação entre grupos distintos. *Problemas de classificação* caracterizam-se por, dado um conjunto de possibilidades de saídas, tentar classificar cada entrada do conjunto de dados em uma saída possível. Assim, esse tipo de aprendizagem mapeia as entradas em classes-alvos de saída, que são representadas pelo conjunto discreto de valores de saída. Os *problemas de regressão*, assim como os problemas de classificação, mapeiam uma entrada em um valor de saída. Contudo, ao invés de classificar uma amostra em uma classe discreta (0 ou 1, "cachorro"ou "gato", etc.), mapeiam essa entrada em um valor em um intervalo contínuo. Assim, na regressão, os algoritmos geram modelos matemáticos que tendem a se comportar como funções multivariáveis em que os valores de entrada são mapeados em valores contínuos de saída. Já os *problemas de extração* de regras são diferentes dos demais, pois o objetivo desse problema não é inferir valores de saída para cada conjunto de entrada, mas identificar relações estatísticas entre os dados.

Existem quatro principais paradigmas de aprendizado de máquina, que são: i) aprendizado supervisionado, ii) aprendizado não-supervisionado, iii) aprendizado semi-supervisionado e iv) aprendizado por reforço (*reinforcement learning*) [Medeiros et al., 2019]. O aprendizado supervisionado necessita de um conjunto de dados rotulados, chamado de conjunto de treinamento, para criar o modelo de classificação ou regressão dos dados. Esse paradigma de aprendizagem requer a existência *a priori* de um conjunto de dados rotulados para a criação e treinamento do modelo de aprendizado de máquina. Já o *aprendizado não-supervisionado* busca padrões nos dados de treinamento e, portanto, não requer a existência prévia de dados rotulados. O aprendizado não-supervisionado é adequado para problemas de agrupamentos (*clustering*), nos quais se quer agrupar as entradas em grupos que não foram definidos (rotulados) *a priori*. O aprendizado não-supervisionado também pode ser utilizado na redução de dimensão de um conjunto de dados, em que encontra-se direções de maximização de variância em que os dados se projetam. Por sua vez, o paradigma de *aprendizado semi-supervisionado* suporta o uso de conjuntos de treinamento com rótulos faltantes ou incompletos. São utilizadas técnicas de aprendizado não-supervisionado nos dados não rotulados na tentativa de normalizar os dados. Ao final da etapa não-supervisionada, o modelo associa cada observação com uma pontuação proporcional à probabilidade do dado ter vindo de um determinado grupo de amostras rotuladas. Em seguida, pode-se usar a parte rotulada dos dados para definir um

limiar na pontuação, a partir do qual é considerado, qual rótulo é mais apropriado para a amostra. Por fim, o paradigma de *aprendizado por reforço* consiste em um processo iterativo, em que agentes aprendem efetivamente novos conhecimentos na ausência de um conjunto de treinamento, com pouco ou nenhum conhecimento do ambiente [Tesauro, 2007]. A ideia central do aprendizado por reforço é que o aprendizado de um agente é baseado em exemplos do conjunto de treinamento, nos quais os agentes interagem com o mundo externo e aprendem com reforços providos pelo ambiente. Os reforços providos podem ser recompensas ou penalidades. Assim, o conjunto de treinamento consiste de pares de amostras de dados e reforços, sejam recompensas ou penalidades. A retroalimentação do ambiente impulsiona o agente para a melhor sequência de ações. O aprendizado por reforço é adequado para problemas de tomada de decisão e planejamento [Tesauro, 2007].

Os algoritmos convencionais de aprendizado de máquina, normalmente, dependem da realização de engenharia de características para processar os conjuntos de dados brutos [Trigeorgis et al., 2016]. A engenharia de características é o processo que transforma dados brutos, ou seja, dados coletados diretamente de uma determinada fonte, em características que melhor representam o problema, para resultar no melhor desempenho do modelo de aprendizado de máquina. O tratamento dessas novas características é feito manualmente por especialistas na construção de um dataset. Dessa forma, a especialização no domínio costuma ser um pré-requisito para a construção de um modelo de aprendizado de máquina eficaz. Paralelamente, as redes neurais profundas são baseadas no aprendizado por representação, ou seja, podem descobrir e aprender automaticamente essas características a partir dos dados brutos [Lecun et al., 1998]. Com isso, muitas vezes, esse tipo de abordagem supera os algoritmos convencionais de aprendizado de máquina, especialmente quando há dados em abundância.

Os principais aspectos das técnicas que se caracterizam por serem de aprendizado profundo (*deep learning*) são os modelos que consistem em rede neurais com múltiplas camadas ou estágios de processamento não-lineares e os métodos para aprendizagem supervisionada ou não supervisionada de representação de características em camadas sucessivamente mais altas e mais abstratas.Dessa forma, a ideia chave do aprendizado profundo é gerar novas representações dos dados a cada camada, aumentando o grau de abstração da representação. A popularidade crescente das técnicas de aprendizado profundo deve-se ao aumento acelerado da capacidade de processamento, e.g. processamento gráfico (Graphical Processing Unit — GPUs); à grande produção de dados; e aos avanços nas pesquisas de aprendizado de máquina [Kwon et al., 2017]. Esses avanços permitiram a exploração de funções complexas não-lineares, a aprendizagem distribuída e uso efetivo de dados rotulados e não rotulados. Os principais algoritmos de aprendizado profundo são as Redes Neurais Convolucionais [Lecun et al., 1998], Redes Neurais Recorrentes [Mikolov et al., 2011] e *AutoEncoders* [Baldi, 2011].

3.2.1. Aprendizado de Máquina Distribuído

O aprendizado de máquina distribuído é um sistema que busca combinar o poder computacional em um aglomerado computacional independente de seu tamanho (gastando mais tempo fazendo cálculos úteis e menos tempo esperando por comunicação). O aprendizado de máquina distribuído, possui no mínimo duas entidades, os nós trabalhado-

res e o servidor de parâmetros [Ho et al., 2013]. O servidor de parâmetros é responsável por gerenciar as tarefas entre os nós trabalhadores. Por sua vez, os nós trabalhadores são responsáveis por executar as tarefas demandadas pelo servidor de parâmetro [Ho et al., 2013]. O servidor de parâmetros, ou nó central, é a ferramenta fundamental para acelerar o processo de treinamento, armazenando dados em nós de trabalho distribuídos e alocando dados e recursos computacionais por meio de interface para treinar o modelo de forma eficiente. No aprendizado de máquina distribuído, o nó central sempre assume o controle e possui total acesso ao conjunto de treinamento; assim, o aprendizado federado se depara com um ambiente de aprendizado mais complexo, pois enfatiza a proteção da privacidade dos dados dos proprietários durante o processo de treinamento. A Figura 3.1 ilustra a arquitetura básica do aprendizado de máquina distribuído.

O aprendizado federado à primeira vista é semelhante ao aprendizado de máquina distribuído. Contudo, o aprendizado de máquina distribuído cobre muitos aspectos não cobertos pelo aprendizado federado, devido à restrição de preservação de privacidade, como, por exemplo, armazenamento distribuído dos dados de treinamento e operação de tarefas computacionais distribuídas. Na maioria dos casos, em aprendizado de máquina distribuído, o servidor de parâmetros possui total controle dos nós trabalhadores e do conjunto de dados. O servidor de parâmetros, em aprendizado federado, é o servidor agregador, que não possui controle dos participantes. Sua função é selecionar os clientes participantes e agregar os parâmetros locais recebidos pelos clientes selecionados. O participante pode se recusar a participar ou até mesmo perder a conexão durante o treinamento. O nó trabalhador, em aprendizado federado, é representado pelo proprietário dos dados.

Em aprendizado de máquina distribuído, o servidor de parâmetros e os nós trabalhadores estão em um mesmo *datacenter* [Lim et al., 2020]. Logo, comunicação não é um problema desse paradigma, pois os enlaces entre o servidor e os nós trabalhadores possuem altas taxas de transferência. Já o aprendizado federado, os participantes, podem estar em áreas em que a taxa de transferência é desfavorável ao envio dos parâmetros para treinamento. Então, técnicas para reduzir o custo de comunicação são frequentemente abordadas nesse paradigma.

3.2.2. Computação na Borda e Banco de Dados Federado

Atualmente, as fontes de dados estão localizadas em dispositivos fora da nuvem. Nesse contexto, a computação móvel na ponta (*Mobile Edge Computing* — MEC) é proposta como uma solução em que os servidores em nuvem são aproximados dos dispositivos finais. Esses servidores ficam localizados nos limites das redes das operadoras, simplificando o acesso aos dados gerados pelos dispositivos móveis. Nesse sentido, a MEC permite que o treinamento do modelo de aprendizado seja trazido para perto de onde os dados são produzidos [Lim et al., 2020], ou seja, nos dispositivos na rede de acesso. A Figura 3.2, apresenta a arquitetura tradicional de uma MEC.

Para o treinamento do modelo de aprendizado de máquina em abordagens MEC convencionais, um paradigma colaborativo foi amplamente utilizado. Nesse paradigma os dados de treinamento são enviados primeiro aos servidores de borda para treinamento do modelo, ao invés de diretamente para os servidores na nuvem para tentar diminuir

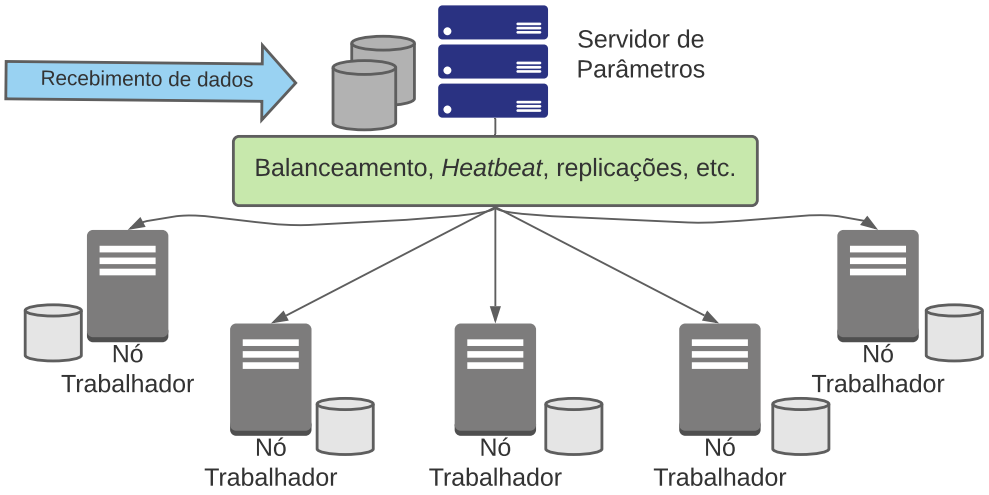


Figura 3.1. Arquitetura do aprendizado de máquina distribuído. Nesse cenário, o servidor de parâmetros recebe os dados para processamento e distribui os dados e tarefas para execução nos nós trabalhadores. O servidor de parâmetros também verifica o estado dos nós trabalhadores enviando *heartbeat*.

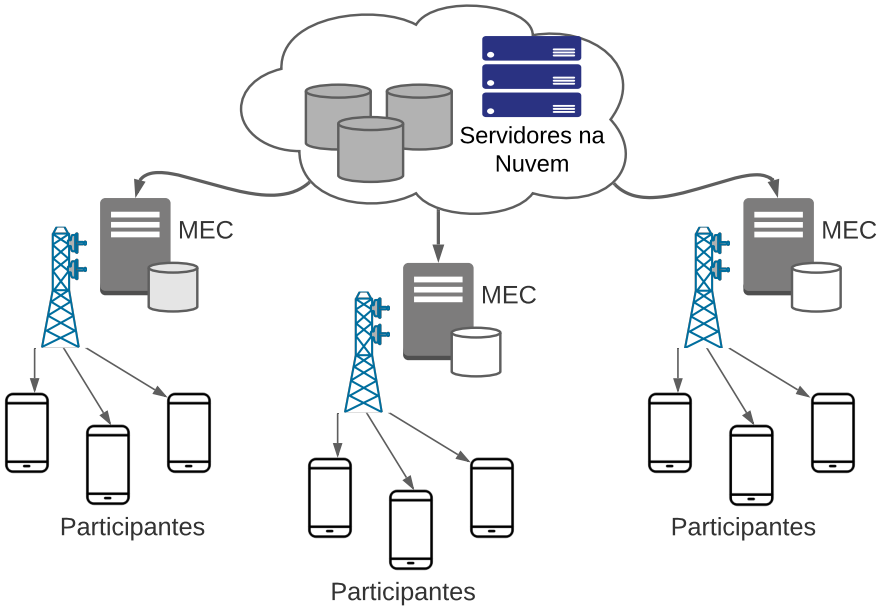


Figura 3.2. Arquitetura da computação móvel na ponta (MEC), onde os servidores estão localizados na estrutura das operadoras, oferecendo, assim, alta taxa de transferência e baixa latência.

o custo de comunicação. No entanto, esse paradigma ainda incorre em altos custos de comunicação e é inadequado especialmente para aplicações que requerem treinamento constante [Wang et al., 2020]. Além disso, o processamento dos dados em servidores de borda ainda implica transmissão de dados pessoais potencialmente sensíveis de celulares para servidores na borda da rede. Isso desencoraja clientes preocupados com a privacidade de seus dados de participar do treinamento do modelo, ou ainda, dependendo de como os dados são armazenados e usados, pode até mesmo violar leis de privacidade cada vez mais rigorosas como, por exemplo, a Lei Geral de Proteção de Dados Pessoais

(LGPD). Para garantir que os dados de treinamento permaneçam nos dispositivos pessoais dos participantes e para facilitar o aprendizado de máquina colaborativo de modelos complexos entre dispositivos distribuídos, aplicações MEC estão cada vez mais adotando o aprendizado federado [Wang et al., 2020].

Outro conceito relacionado ao de aprendizado federado é o de sistemas de banco de dados federados [Sheth e Larson, 1990]. Bancos de dados federados são sistemas que integram várias unidades de banco de dados e gerenciam o sistema integrado como um todo. O conceito de banco de dados federado é proposto para alcançar a interoperabilidade entre vários bancos de dados independentes. Um banco de dados federado usa armazenamento distribuído para as unidades de banco de dados e, na prática, os dados em cada unidade de banco de dados são heterogêneos. Portanto, tem muitas semelhanças com o aprendizado federado em termos de tipo e armazenamento de dados. No entanto, o sistema de banco de dados federado não envolve nenhum mecanismo de proteção de privacidade no processo de interação entre os sistemas e todas as unidades de banco de dados são completamente visíveis para o sistema de gerenciamento. Além disso, o foco do sistema de banco de dados federado está nas operações básicas de dados, incluindo inserção, exclusão, pesquisa e fusão, enquanto o objetivo do aprendizado federado é estabelecer um modelo conjunto para cada proprietário de dados sob a premissa de proteção da privacidade dos dados [Yang et al., 2019].

3.2.3. Aprendizado multipartidário seguro com preservação de privacidade

O aprendizado federado pode ser considerado um aprendizado de máquina colaborativo descentralizado e com preservação de privacidade [Yang et al., 2019]. Portanto, o aprendizado federado está estreitamente relacionado ao aprendizado de máquina multipartidário seguro (*Secure Multiparty Computacion* — SMC). O SMC é um protocolo de criptografia que distribui uma computação entre várias partes, em que nenhuma dessas partes pode ter acesso aos dados de outras partes. Os protocolos de computação multipartidários seguros permitem que sejam computados dados distribuídos de forma compatível, segura e privada [Bogetoft et al., 2009]. Muitos esforços de pesquisa foram dedicados ao aprendizado de máquina com preservação de privacidade em trabalhos acadêmicos. Por exemplo, trabalhos anteriores [Vaidya et al., 2008, Du e Zhan, 2002] propuseram algoritmos para árvores de decisão multipartidárias seguras para dados particionados verticalmente. Vaidya e Clifton propuseram regras de mineração de associação segura [Vaidya e Clifton, 2002], k-médias seguros [Vaidya e Clifton, 2003] e um classificador Bayesiano ingênuo [Vaidya e Clifton, 2004] para dados particionados verticalmente. Algoritmos de máquinas de vetor suporte seguro foram desenvolvidos para dados particionados verticalmente [Yu et al., 2006b] e dados particionados horizontalmente [Yu et al., 2006a]. Du *et al.* propuseram protocolos seguros para regressão linear e classificação multipartidária [Du et al., 2004]. Wan *et al.* propuseram métodos de gradiente descendente multipartidários seguros [Wan et al., 2007]. Todos esses trabalhos utilizaram SMC [Yao, 1982, Micali et al., 1987] para garantias de privacidade.

3.3. Estratégias para o Aprendizado Federado

O aprendizado federado envolve o treinamento de um modelo estatístico global utilizando dados de dispositivos remotos. A principal motivação do aprendizado federado

é a privacidade dos dados dos participantes durante o treinamento colaborativo. Durante o processo de treinamento, os participantes treinam um modelo compartilhado colaborativamente mantendo os dados no dispositivo. O objetivo do aprendizado federado é treinar o modelo global processando os dados localmente nos dispositivos. Assim, os dispositivos enviam atualizações intermediárias a um servidor central durante cada iteração de comunicação. O servidor agrega os modelos intermediários e distribui o novo modelo agregado a todos os participantes.

3.3.1. Formulação Matemática

O aprendizado federado é uma tecnologia que viabiliza o treinamento de modelos de aprendizado de máquina em redes de dispositivos móveis mantendo a privacidade dos usuários [Lim et al., 2020]. Existem duas entidades principais no sistema de aprendizado federado, os participantes, que são os donos dos dados, e o servidor agregador, que é responsável por criar o modelo global [Li et al., 2020]. Denota-se $N = \{1, ..., n\}$ como o conjunto de participantes, onde cada participante possui seu próprio conjunto de dados privado $D_n, n \in N$. Cada participante n utiliza seu conjunto de dados D_n para treinar um modelo local w_n^t e realiza o envio apenas dos parâmetros do modelo local para o servidor do aprendizado federado a cada certo período de tempo. Em cada iteração de comunicação, um subconjunto S^t, $S^t \in N$ dos dispositivos é selecionado aleatoriamente para fornecer os parâmetros de seus modelos locais para o servidor. Em seguida, todos os parâmetros dos modelos locais selecionados são agregados para gerar um modelo global denominado w_G^t. Os modelos locais são atualizados localmente por τ atualizações locais, antes do envio dos parâmetros dos modelos locais para o servidor realizar a agregação global.

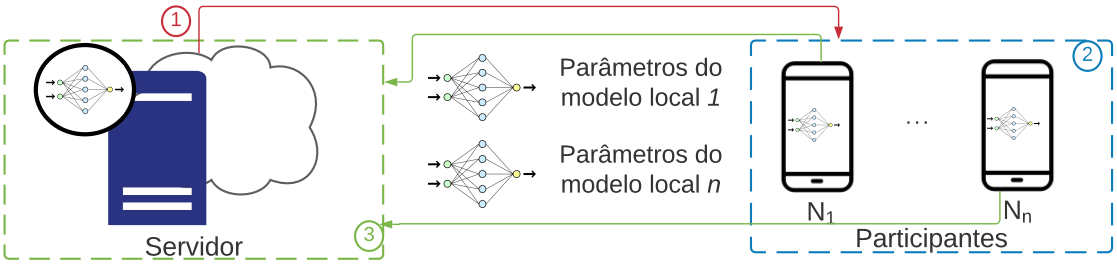


Figura 3.3. Arquitetura do aprendizado federado. No primeiro passo, o servidor envia o modelo inicial. Posteriormente, cada participante selecionado atualiza o modelo com seus dados locais e, por fim, o servidor agrega os parâmetros recebidos.

A Figura 3.3 ilustra o processo de treinamento que é adotado no aprendizado federado. O processo de treinamento difere de acordo com a necessidade de cada cenário, porém o conceito base se mantém. Uma suposição subjacente é que os participantes são honestos, o que significa que eles usam seus dados privados reais para fazer o treinamento e enviam os parâmetros verdadeiros para o servidor. O aprendizado federado possui três passos base que são enumerados a seguir.

1. Inicialização. O servidor inicia a tarefa de treinamento e especifica os hiperparâmetros do modelo global como, por exemplo, taxa de aprendizagem, quan-

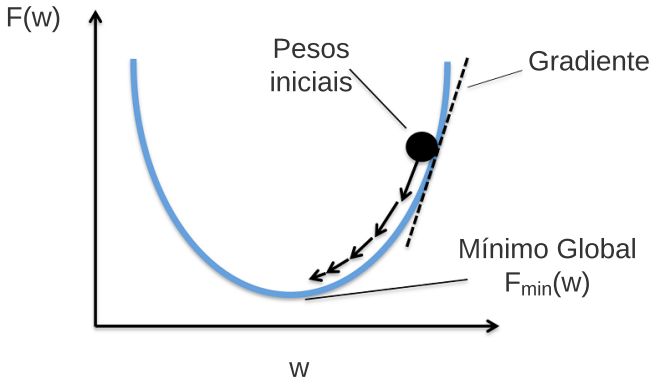
tidade de atualizações locais, tamanho do mini-lote etc. Em seguida, o servidor transmite o modelo global inicializado w_G^0 aos participantes selecionados.

2. Atualizações Locais. Com base no modelo global w_G^t recebido pelo servidor, em que t denota o índice de iteração global atual, os participantes selecionados, respectivamente, utilizam seus dados e dispositivos locais para atualizar os parâmetros do modelo local w_n^t. O objetivo do participante n na iteração t é encontrar os parâmetros otimizados w_n^t que minimizam a função de perda $L(w_n^t)$.

3. Agregação Global. O servidor agrega os parâmetros dos modelos locais dos participantes selecionados e, em seguida, envia o modelo global atualizado w_G^{t+1} de volta a todos os participantes. O objetivo do servidor é minimizar a função de perda global $L(w_G^t)$.

Os passos de atualização dos modelos locais e agregação do modelo global são repetidos até a convergência da função de perda global ou que outra condição de parada seja atendida. O aprendizado federado funciona com modelos que utilizam métodos de gradiente descendente estocástico, (*Stochastic Gradient Descent* — SGD) [Brendan McMahan et al., 2017], como redes neurais, regressão linear e máquinas de vetor suporte. Gradiente, em termos simples, significa inclinação de uma superfície. Portanto, o gradiente descendente é a direção que indica o declive para chegar ao ponto mínimo dessa superfície. A Figura 3.4(a) ilustra esse processo. Então, o principal objetivo do algoritmo gradiente descendente é encontrar os melhores parâmetros, os pesos no caso das redes neurais, para minimizar uma função. Para isso, é calculada a derivada da função para cada amostra no conjunto de dados. Todavia, esse processo causa sobrecarga, caso o conjunto de dados possua grande número de amostras. O SGD é uma variante do gradiente descendente que foi proposta para minimizar a sobrecarga. Um comportamento estocástico implica um sistema ou processo que está vinculado a uma probabilidade aleatória. Portanto, no gradiente descendente estocástico, algumas amostras são selecionadas aleatoriamente em vez de todo o conjunto de dados a cada iteração. O lote denota o número total de amostras de um conjunto de dados que é usado para calcular o gradiente em cada iteração [Bottou, 2010]. A Figura 3.4(b) apresenta a diferença entre gradiente descendente e SGD.

Os modelos de aprendizado de máquina possuem um conjunto de parâmetros que são atualizados com base nos dados de treinamento. Uma amostra j dos dados de treinamento possui duas partes. A primeira parte é o vetor x_j que são as características da amostra j, que são passadas como entrada ao modelo de aprendizado de máquina; a outra parte é um escalar y_j que é a saída desejada pelo modelo. Para otimizar o aprendizado, cada modelo possui uma função de perda definida em seu vetor de parâmetros w para cada amostra de dados j. A função de perda calcula o erro do valor previsto pelo modelo em relação ao valor desejado y_j de cada amostra. O processo de aprendizagem do modelo é minimizar a função de perda em uma coleção de amostras de dados de treinamento. Para cada amostra de dados j, é definida a função de perda $f(w, x_j, y_j)$, que será denotado em resumo como $f_j(w)$. Cada conjunto de dados D_n possuirá uma função de perda local $F_n(w_n)$, que é definida pela equação:



(a) Representação gráfica do algoritmo gradiente descendente.

Gradiente Descendente

id	peso	altura	Gênero	
0	75 kg	1.73 m	M	i = 1
1	98 kg	1.61 m	F	i = 2
2	61 kg	1.56 m	F	i = 3
3	103 kg	1.98 m	M	i = 4
4	81 kg	1.85 m	F	i = 5
5	110 kg	2.10 m	M	i = 6

Gradiente Descendente Estocástico

id	peso	altura	Gênero	
0	75 kg	1.73 m	M	
1	98 kg	1.61 m	F	i = 1
2	61 kg	1.56 m	F	
3	103 kg	1.98 m	M	
4	81 kg	1.85 m	F	i = 2
5	110 kg	2.10 m	M	

(b) No gradiente descendente, os gradientes são calculados a cada amostra contida no conjunto de dados. Já no gradiente descendente estocástico, o cálculo e feito em uma parte menor chamada de lote.

Figura 3.4. Gradiente descendente e sua variação gradiente descendente estocástico.

$$F_n(w) = \frac{1}{D_n} \sum_{j \in D_n} f_j(w). \tag{1}$$

É importante ressaltar que para facilitar a visualização das equações será definido $D_n = |D_n|$, em que $|\cdot|$ denota a cardinalidade de um conjunto e $D = \sum_{n=1}^{N} D_n$. Assumindo que $D_n \cap D_{n'} = \emptyset \; \forall \; n \neq n'$, a função de perda global para todos os conjuntos de dados é definida pela equação

$$F(w) = \sum_{n=1}^{N} \frac{D_n}{D} F_n(w). \tag{2}$$

Note que $F(w)$ não pode ser calculada diretamente sem compartilhar informações entre os participantes [Wang et al., 2019].

O problema de aprendizagem é, então, minimizar $F(w)$, ou seja, encontrar

$$w^* = \arg\min F(w). \tag{3}$$

Devido à complexidade inerente à maioria dos modelos de aprendizado de máquina, é impossível encontrar uma solução de forma fechada para Equação 3. Assim, a Equação 3 é frequentemente resolvida usando técnicas de gradiente descendente [Wang et al., 2019].

Atualmente, com o surgimento do *big data*, a quantidade de dados gerados cresce em um ritmo acelerado. Logo, um único servidor não possui poder computacional suficiente para processar os dados envolvido em nesses problemas de otimização. Então foi proposta a otimização distribuída, que divide a tarefa de otimização em pequenas subtarefas para serem processadas por um aglomerado computacional [Zhang e Xiao, 2018]. A otimização distribuída assume que os dados são independentes e identicamente distribuídos (*Independent and Identically Distributed* — IID) [Zhang e Xiao, 2018]. Na otimização em *data center*, os custos de comunicação são pequenos e os custos computacionais são as principais preocupações, com grande parte da ênfase recente no uso de GPUs para reduzir esses custos. Em contraste, na otimização federada [Brendan McMahan et al., 2017], os custos de comunicação são dominantes, já que normalmente há limitação na largura de banda de envio e, dependendo da rede em que o dispositivo se encontra, o envio é praticamente inviável. O termo otimização federada é utilizado para se referir ao problema de otimização implícito no aprendizado federado, que possui uma conexão e também um contraste com a otimização distribuída [Brendan McMahan et al., 2017]. A otimização federada tem várias propriedades-chave que a diferenciam de um problema típico de otimização distribuída que são:

1. Dados não independentes e identicamente distribuídos (*non-Independent and Identically Distributed* — non-IID): São conjunto de dados em que cada conjunto de dados locais não possui a mesma distribuição de probabilidade que os outros e tendem a ser dependentes entre si. Isso se dá devido ao contexto de uso de cada participantes.

2. Dados desbalanceados: Da mesma forma, alguns usuários possuirão conjuntos de dados grandes com muitas amostras e outros com poucas amostras;

3. Grande quantidade de participantes: É esperado que o número de participantes de uma otimização federada seja grande, logo o algoritmo de aprendizado federado deve lidar com quantidade massiva de participantes;

4. Comunicação limitada: Os dispositivos móveis, típicos de um ambiente de aprendizado federado, estão frequentemente desconectados ou em conexões com baixa taxa de transferência.

Os participantes móveis participam do treinamento quando estiverem com bateria carregada, carregando e/ou em uma conexão wi-fi. Ademais, é esperado que cada usuário participe apenas de um pequeno número de rodadas de atualização por dia. Por outro lado, uma vez que qualquer conjunto de dados no dispositivo é pequeno em relação ao tamanho total de todos os conjuntos de dados e *smartphones* modernos possuem processadores relativamente rápidos e muitos possuem GPUs, a computação torna-se essencialmente barata em comparação com os custos de comunicação para muitos modelos

de dispositivos móveis. Portanto, de modo a diminuir o custo de comunicação utiliza-se o poder computacional dos dispositivos dos participantes para diminuir o número de rodadas de comunicação necessárias para treinar o modelo de aprendizado de máquina.

3.3.2. Algoritmo *Federated Averaging*

O método mais utilizado para agregação de modelos locais em aprendizado federado é a média federada (*Federated Averaging* — FedAvg) [Brendan McMahan et al., 2017], um método de treinamento baseado em gradiente descendente estocástico (SGD). O FedAvg foi o primeiro algoritmo de agregação de modelos locais para aprendizado federado [Lim et al., 2020]. O funcionamento do FedAvg foi comprovado empiricamente, especialmente para problemas onde a função de perda utilizada é não convexa. Funções convexas são funções em que mínimos locais são também um mínimo global, já as funções não convexas possuem diversos mínimos locais que não são mínimos globais. Contudo, o FedAvg não possui garantias de convergência e pode divergir em cenários práticos quando os dados são heterogêneos [Li et al., 2020].

Em 2018, o FedAvg foi implementado no Gboard [Hard et al., 2018], o teclado inteligente da Google, para melhorar o modelo de previsão de próxima palavra. Desde então, outros estudos exploraram o uso de aprendizado federado em uma série de cenários em que os dados são de natureza sensível, por exemplo, para desenvolver modelos preditivos para diagnóstico de saúde [Brisimi et al., 2018], para promover a colaboração entre hospitais [Li et al., 2019] e Agências governamentais [Verma et al., 2018].

O algoritmo FedAvg se baseia no SGD devido ao avanço das aplicações de aprendizado profundo, em que o método de aprendizado converge na direção do SGD [Brendan McMahan et al., 2017]. No algoritmo FedAvg é selecionada uma porção S de participantes em cada rodada de comunicação e calcula-se o gradiente da perda sobre todos os dados mantidos por esses participantes. Assim, S controla o tamanho do lote global. Caso $S = 1$, isso corresponderá, então, ao gradiente descendente determinístico, já que nesse caso ocorre a seleção de todos os participantes. Cada participante computa $\nabla F_n(w_t)$, que são os gradientes em seus dados locais para o modelo atual w_t, e o servidor agrega esses gradientes aplicando a atualização $w_{t+1} \leftarrow w_t - \eta \sum_{n=1}^{N} \frac{D_n}{D} \nabla F_n(w_t)$. O hiper parâmetro η é taxa de aprendizado, que influencia diretamente na velocidade da convergência. Então, uma taxa de aprendizado pequena implica numa trajetória suave e com pequenas mudanças nos pesos a cada iteração. Já uma taxa de aprendizado muito alta implica uma mudança maior nos pesos, aumentando a velocidade da convergência. Contudo, pode levar também a oscilações em torno de um mínimo global. Um tipo de agregação equivalente e mais utilizado é $\forall n$, $w_{t+1}^n \leftarrow w_t^n - \eta \nabla F_n(w_t^n)$ e então $w_{t+1} \leftarrow \sum_{n=1}^{N} \frac{D_n}{D} w_{t+1}^n$. Cada cliente realiza localmente uma ou mais etapas de gradiente descendente no modelo atual usando seus dados locais e o servidor, então, obtém uma média ponderada dos modelos resultantes. Uma vez que o algoritmo é escrito dessa forma, pode-se adicionar mais computação para cada cliente, realizando a atualização local várias vezes antes da etapa de agregação. A quantidade de computação é controlada por três parâmetros principais: S, a porção de clientes participantes que realizam a computação em cada rodada de comunicação; τ, o número de iterações de treinamento que cada cliente faz em seu conjunto de dados local; e B, o tamanho do mini-lote local usado para as atualizações locais do cliente. O pseudocódigo para o FedAvg é apresentado no Algoritmo 1.

Algoritmo 1: O algoritmo de média federada (*Federated Averaging*) [Brendan McMahan et al., 2017].

Input: Tamanho do mini-lote local B, número de épocas locais τ, número de participantes por iteração de comunicação m, taxa de aprendizagem η, quantidade de iterações de comunicação T

Output: Modelo global w_G

1 *[Participante n - Atualiza o modelo local]*
2 **TreinoLocal**(n, w):
3 Divide o conjunto de dados local D_n em mini-lotes de tamanho B criando o conjunto B_i
4 **for** *each epoca_local de 1 até* τ **do**
5 | **for** *each* $b \in B_i$ **do**
6 | | $w_{t+1}^n \leftarrow w_t^n - \eta \nabla F_n(w_t^n; b)$
7 | **end**
8 **end**
9 *[Servidor - Realiza uma média ponderada global das atualizações locais]*
10 Inicializa w_G^0
11 **for** *each iteração t de 1 até* T **do**
12 | Escolhe aleatoriamente um subconjunto $S_n \in N$ de tamanho m
13 | **for** *each participante* $n \in S_n$ **do**
14 | | $w_{t+1}^n \leftarrow$ **TreinoLocal**(n, w_G^t)
15 | **end**
16 | $w_G^t = \sum_{n=1}^{N} \frac{D_n}{D} w_{t+1}^n$
17 **end**

O algoritmo segue os passos descritos anteriormente. No passo 1, o servidor inicializa a tarefa (linhas 10 - 15). Então, no passo 2, o participante n realiza o treinamento local e otimiza sua função de perda nos mini-lotes de conjunto de dados local (linhas 2 - 8). Um mini-lote se refere a um subconjunto aleatório do conjunto de dados de cada participante. Na t-ésima iteração (linha 16), o servidor minimiza a perda global agregando a média dos gradientes recebidos dos participantes. O processo de treinamento do aprendizado federado continuará até que a função de perda global convirja ou alcance uma acurácia desejável.

3.3.3. Arquiteturas de Referência do Aprendizado Federado

Existem três arquiteturas gerais para um sistema de aprendizado federado que são: i) horizontal, ii) vertical e iii) de transferência [Yang et al., 2019]. Cada arquitetura é descrita como uma matriz, na qual as linhas representam o espaço de amostras e as colunas, o espaço de características. O espaço de características será denotado como X, o espaço de rótulos como Y e o I será o espaço de amostras. As características X, rótulos Y e os IDs de amostras I constituem o campo do conjunto de dados de treinamento completo (I, X, Y).

3.3.3.1. Aprendizado Federado Horizontal

É a arquitetura típica e mais utilizada de um sistema de aprendizado federado. A Figura 3.3 apresenta a arquitetura do aprendizado federado horizontal. A principal característica dessa arquitetura é que os n participantes possuem a mesma estrutura de dados, ou seja, os conjuntos de dados dos participantes compartilham o mesmo espaço de características, porém possuem espaço de amostras diferentes, como pode ser visto na Figura 3.5. Um exemplo é o caso de dois bancos regionais que podem ter grupos de usuários diferentes de suas respectivas regiões e o conjunto de interseção de seus usuários é muito pequeno. Contudo, seu negócio é semelhante, então os espaços de características são praticamente os mesmos [Yang et al., 2019]. Na arquitetura horizontal, os participantes aprendem de forma colaborativa um modelo de aprendizado de máquina com a ajuda de um servidor na nuvem. Uma suposição típica é que os participantes são honestos, enquanto o servidor é honesto, mas curioso; portanto, nenhum vazamento de informação de nenhum participante para o servidor é permitido [Phong et al., 2018]. Nesse contexto, o aprendizado federado horizontal pode ser definido como

$$X_n = X_j; Y_n = Y_j; I_n \neq I_j; \forall D_n, D_j, n \neq j. \tag{4}$$

O aprendizado federado horizontal foi a primeira arquitetura para aprendizado federado [Brendan McMahan et al., 2017]. Bonawitz *et al.* propõem um esquema de agregação segura que protege os parâmetros compartilhados com o servidor [Bonawitz et al., 2017]. Phong *et al.* propõem a utilização de criptografia homomórfica nos parâmetros dos modelos dos participantes [Phong et al., 2018].

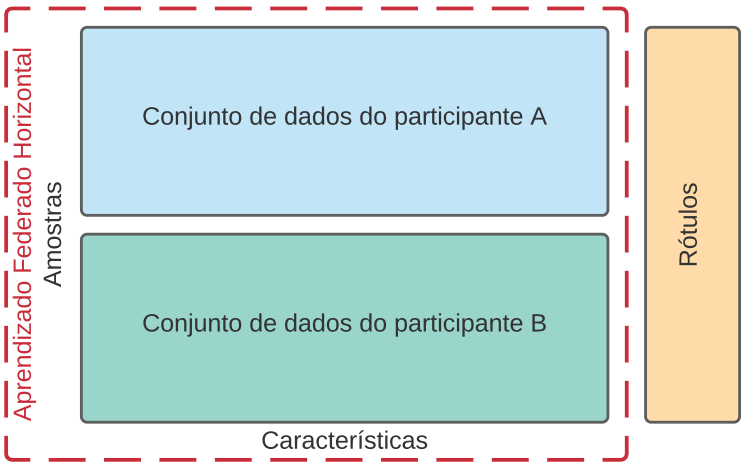


Figura 3.5. No aprendizado federado horizontal, são utilizados conjuntos de dados que possuem o mesmo espaço de características, mas diferem no espaço de amostras.

Analise de segurança: Na arquitetura do aprendizado federado horizontal, uma das preocupações é proteger os dados sensíveis de um servidor honesto, mas curioso [Yang et al., 2019]. As técnicas mais utilizadas são a criptografia homomórfica [Phong et al., 2018] e a computação multipartidária segura [Bonawitz et al., 2017]. No entanto,

essa arquitetura pode estar sujeita a ataques por um participante mal-intencionado durante o processo de aprendizado colaborativo.

3.3.3.2. Aprendizado Federado Vertical

Alguns algoritmos de aprendizado de máquina que preservam a privacidade para dados particionados verticalmente foram propostos na literatura, incluindo análise estatística cooperativa [Wenliang Du e Atallah, 2001], mineração de regras de associação [Vaidya e Clifton, 2002], regressão linear segura [Karr et al., 2009], classificação [Du et al., 2004] e gradiente descendente [Wan et al., 2007]. O aprendizado federado vertical ou aprendizado federado baseado em características, como pode ser visto na Figura 3.6, é aplicável aos casos em que dois conjuntos de dados compartilham o mesmo espaço de amostras, mas diferem no espaço de características. Por exemplo, duas empresas que operam em uma mesma cidade podem ter clientes semelhantes, mas possuem diferentes informações sobre esses clientes. O aprendizado federado vertical é o processo de agregar essas diferentes características e computar a perda e os gradientes do treinamento de maneira a preservar a privacidade para construir um modelo com dados de ambas as partes de forma colaborativa [Yang et al., 2019].

Trabalhos recentes propõem um esquema de aprendizado federado vertical para treinar um modelo de regressão logística que preserva a privacidade [Hardy et al., 2017, Nock et al., 2018]. Os autores aplicaram a aproximação de Taylor às funções de perda e gradiente descendente de modo que a criptografia homomórfica possa ser adotada para cálculos de preservação de privacidade.

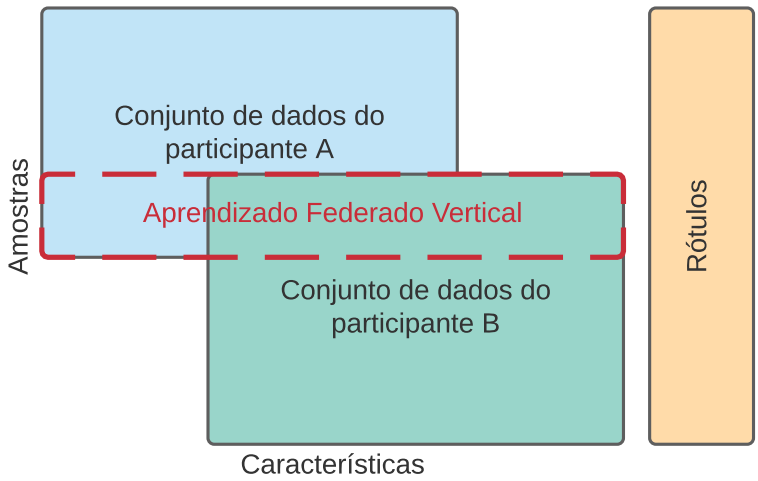


Figura 3.6. No aprendizado federado vertical, o conjunto de dados possui algumas amostras semelhantes, porém com características diferentes. Antes do início do treinamento, os participantes A e B selecionam, de uma forma segura, a interseção dos espaços de amostras.

Suponha que as empresas A e B desejem treinar colaborativamente um modelo de aprendizado de máquina em seus sistemas de negócios, cada uma com seus próprios dados. Além disso, a Empresa B também possui dados de rótulos que o modelo precisa prever. Por motivos de privacidade e segurança de dados, A e B não podem trocar

dados diretamente. Para garantir a confidencialidade dos dados durante o processo de treinamento, uma terceira entidade C é envolvida [Yang et al., 2019]. Assume-se que o colaborador C é honesto e não está em conluio com A ou B, mas as partes A e B são honestas, mas curiosas uma para com a outra. Uma terceira parte confiável C é uma suposição razoável, uma vez que a parte C pode ser desempenhada por autoridades como governos ou substituída por um nó de computação seguro [Yang et al., 2019]. O aprendizado federado vertical possui duas partes, como mostrado na Figura 3.7. Na parte 1, é feito um alinhamento entre as entidades utilizando criptografia. Uma vez que os grupos de usuários das duas empresas não são os mesmos, o sistema usa as técnicas de alinhamento de IDs de usuário baseadas em criptografia para confirmar os usuários comuns mútuos em A e B [Liang e Chawathe, 2004, Scannapieco et al., 2007]. Durante o alinhamento das entidades, o sistema não expõe usuários que não se sobrepõem entre si. Na parte 2, é feito o treinamento de modelo criptografado. Depois de determinar as entidades comuns, pode-se usar os dados dessas entidades comuns para treinar o modelo de aprendizado de máquina. O processo de treinamento pode ser dividido nas seguintes quatro etapas:

1. A entidade C cria pares de chaves de criptografia homomórfica aditiva e envia a chave pública para A e B, além de criptografar máscaras para A e B aplicarem.

2. A e B criptografam e trocam seus resultados intermediários utilizando a máscara de C para cálculos de gradiente e perda;

3. A e B calculam gradientes criptografados e adicionam uma máscara adicional. B também calcula a perda criptografada. A e B enviam valores criptografados para C.

4. C descriptografa e envia os gradientes descriptografados e a perda de volta para A e B. A e B desmascaram os gradientes e atualizam os parâmetros do modelo.

O sistema de aprendizado federado vertical ajuda os participantes a estabelecer uma estratégia de "riqueza comum" sem afetar a privacidade dos dados, razão pela qual esse é considerado um sistema de aprendizado colaborativo. Portanto, esse sistema vertical pode ser definido como

$$X_n \neq X_j; Y_n \neq Y_j; I_n = I_j; \forall D_n, D_j, n \neq j. \tag{5}$$

Análise de segurança: Um sistema de aprendizado federado vertical normalmente assume participantes honestos, mas curiosos. Em um caso de duas partes, por exemplo, ambas as partes não estão em conluio e uma das partes pode estar comprometida com um adversário. O ponto de segurança é que o adversário pode aprender dados apenas do participante corrupto e não pode aprender dados dos outros participantes. Para facilitar os cálculos seguros entre as duas partes, às vezes um terceiro participante honesto, mas curioso é introduzido, C no caso do exemplo anterior. Nesse caso, assume-se que esse terceiro não está em conluio com nenhuma das partes. A computação multipartidária segura fornece prova de privacidade formal para esses protocolos [Goldreich et al., 2019]. Ao final do aprendizado, cada parte detém apenas os parâmetros do modelo associados às suas características. Portanto, no momento da inferência, as duas partes também precisam colaborar para gerar o modelo de saída.

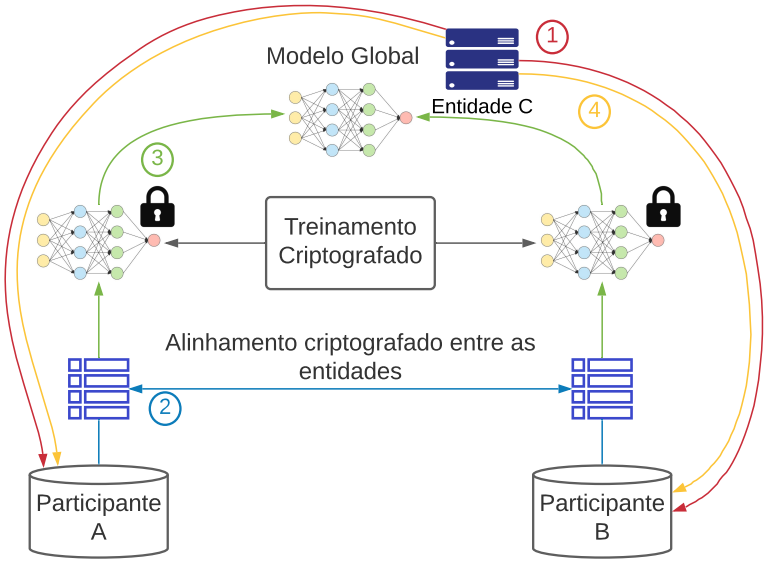


Figura 3.7. Arquitetura de um sistema de aprendizado federado vertical. No primeiro passo, a entidade cria os pares de chaves e envia a chave pública para os participantes. Já no passo 2, A e B realizam a verificação segura para encontrar interseções em suas amostras, ou seja, clientes em comum. Os participantes enviam seus parâmetros criptografados e com uma máscara para agregação pela entidade C no passo 3. Por fim, no passo 4, a entidade C retorna o resultado para os participantes.

3.3.3.3. Aprendizado por Transferência Federada

O aprendizado por transferência federada se aplica aos cenários nos quais dois conjuntos de dados diferem não apenas nas amostras, mas também no espaço de características. Um exemplo é o caso de duas instituições diferentes operando em países diferentes [Yang et al., 2019]. Nesse caso, as técnicas de aprendizado por transferência [Pan e Yang, 2010] podem ser aplicadas para fornecer soluções para toda a amostra e espaço de características em um ambiente de aprendizado federado, como pode ser visto na Figura 3.8.

Suponha que no exemplo de aprendizado federado vertical discutido anteriormente, as partes A e B tenham apenas um conjunto muito pequeno de amostras sobrepostas e o principal interesse é aprender os rótulos para todo o conjunto de dados do participante A. A arquitetura descrita na seção acima funciona até agora apenas para o conjunto de dados sobreposto. Para estender sua cobertura a todo o espaço amostral, o aprendizado por transferência é proposto [Yang et al., 2019]. O aprendizado por transferência federada não muda a arquitetura geral mostrada na Figura 3.7, porém muda os detalhes dos resultados intermediários que são trocados entre as partes A e B. O aprendizado por transferência normalmente envolve aprender uma representação comum entre as características das partes A e B e minimizar o erro na previsão dos rótulos para a parte do domínio de destino, aproveitando os rótulos na parte do domínio de origem (B, nesse caso). Portanto, os cálculos de gradiente para as partes A e B são diferentes daqueles cenário de aprendizado federado vertical. O aprendizado por transferência federada é

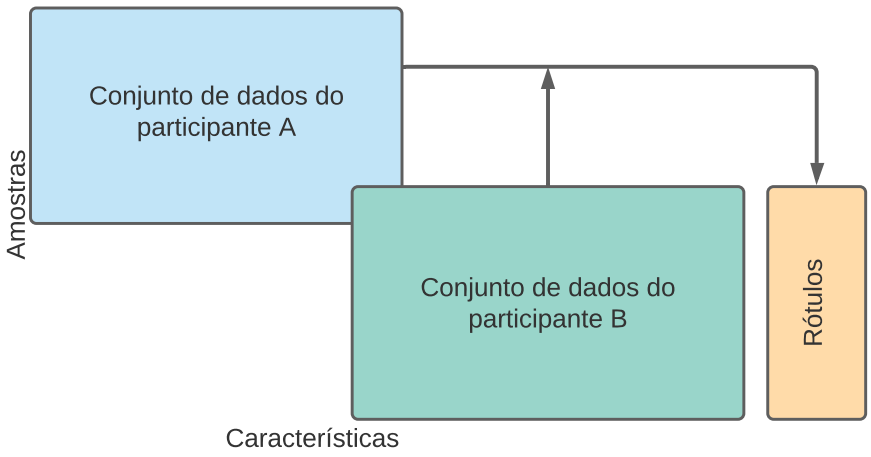


Figura 3.8. No aprendizado por transferência federada, o participante A quer aprender com todo o conjunto de dados do participante B. Para isso, são realizadas técnicas de aprendizado por transferência, que transfere o conhecimento de um modelo para outro sem expor o conjunto de dados que foi utilizado para treinar o modelo de origem.

uma extensão para os sistemas de aprendizado federado existentes, pois lida com problemas que excedem o escopo dos algoritmos de aprendizado federado existentes. Assim, o aprendizado por transferência federada é definido por

$$X_n \neq X_j; Y_n \neq Y_j; I_n \neq I_j; \forall D_n, D_j, n \neq j. \tag{6}$$

3.4. Principais Aplicações

O aprendizado federado é uma técnica promissora em diversas aplicações móveis como redes veiculares [Samarakoon et al., 2018], detecção de ataques cibernéticos [Nguyen et al., 2018], cache de borda e descarregamento de computação (*Edge Caching and Computation Offloading*) [Wang et al., 2019b] e associação de estações base [Chen et al., 2020]. O aprendizado federado também possui aplicações para dispositivos em redes tradicionais como detecção de intrusão [Preuveneers et al., 2018], vendas [Yang et al., 2019], aplicações de saúde [Brisimi et al., 2018], entre outras. O principal foco dessa seção são as aplicações de aprendizado federado em redes móveis e tradicionais.

3.4.1. Ataques Cibernéticos

A detecção de ataques cibernéticos é uma etapa importante para prevenir e mitigar as consequências de ataques em redes. Atualmente, diversos trabalhos utilizando aprendizado de máquina para detecção de ataques foram propostos [Andreoni Lopez et al., 2019]. Entre as diferentes abordagens para detecção de ataques cibernéticos, o aprendizado profundo é considerado a ferramenta mais eficaz para detectar uma ampla gama de ataques com alta acurácia [Nguyen et al., 2018]. Contudo, a acurácia do modelo de detecção irá depender do conjunto de dados utilizado. Os modelos de aprendizado profundo só superam os modelos tradicionais de aprendizado de máquina quando há um conjunto de dados

relativamente grande para o treinamento. No entanto, esses dados podem ser de natureza confidencial. Portanto, alguns modelos de detecção de ataques baseados em aprendizado federado foram propostos para resolver esse problema de privacidade.

Nguyen *et al.* propõem um modelo de detecção de ataque cibernético para uma rede de borda utilizando aprendizado federado [Nguyen et al., 2019]. Nesse trabalho, os *gateways* IoT operam como participantes do aprendizado federado e um provedor de serviços de segurança IoT funciona como o servidor para agregar modelos treinados de forma compartilhada pelos participantes. Os autores apresentam uma aplicação de um sistema de detecção de intrusão utilizando aprendizado federado que foi avaliado em um ambiente real de casas inteligentes e conseguiu detectar com sucesso 95,6% dos ataques em aproximadamente 257 ms sem disparar nenhum alarme falso. No trabalho, assume-se que os participantes são honestos e estão dispostos a contribuir no treinamento utilizando os parâmetros de seus modelos atualizados. No entanto, se alguns dos participantes forem maliciosos, eles poderão corromper toda a detecção de intrusão. Em contrapartida, Preuveneers *et al.* propõem o uso da tecnologia de cadeia de blocos para o gerenciamento dos dados compartilhados pelos participantes [Preuveneers et al., 2018]. Ao usar cadeia de blocos, todas as atualizações incrementais do modelo de detecção de anomalias do aprendizado de máquina são armazenadas no livro-razão facilitando, portanto, a identificação de um participante malicioso. Além disso, com base nos modelos compartilhados dos participantes honestos armazenados no livro-razão, o modelo global pode ser facilmente recuperado por qualquer usuário da rede federada. Os autores identificaram que o uso da cadeia de blocos gera latência no treinamento ao comparar com o treinamento utilizando o algoritmo FedAvg sem cadeia de blocos. O trabalho, portanto, foca apenas em utilizar aprendizado federado para detecção de intrusão, mas não é escopo do trabalho a utilização de modelos eficientes. O trabalho possui uma arquitetura frágil que depende que os nós da rede monitorada pelos participantes do treinamento federado enviem todos os seus fluxos para análise. A Figura 3.9 mostra uma arquitetura de referência para sistemas de detecção de ataque utilizando aprendizado federado.

3.4.2. Cache de Borda e Descarregamento de Computação (*Edge Caching and Computation Offloading*)

Para tratar das condições dinâmicas e variáveis temporais em um sistema de rede móvel de borda (*Mobile edge network* — MEC), Wang *et al.* propõem o uso de aprendizado por reforço profundo (*Deep Reinforcement Learning* — DRL) com aprendizado federado para otimizar a memória transitória e as decisões de descarregamento de computação em um sistema MEC [Wang et al., 2019b]. O sistema MEC consiste em um conjunto de equipamentos de usuário cobertos por estações base. Para o armazenamento em memória transitória, o agente DRL toma a decisão de armazenar na memória transitória ou não o arquivo baixado e qual arquivo local substituir, caso ocorra o armazenamento em memória transitória. Para o descarregamento de computação, os equipamentos de usuário podem escolher entre descarregar tarefas de computação para o nó de borda por canais sem fio ou realizar as tarefas localmente. Os estados do sistema MEC incluem as condições da rede sem fio, o consumo de energia do equipamento do usuário e os estados de fila de tarefas, enquanto a função de recompensa é definida como qualidade de experiência (QoE) dos equipamentos do usuário. Dado o grande número de estados e o espaço

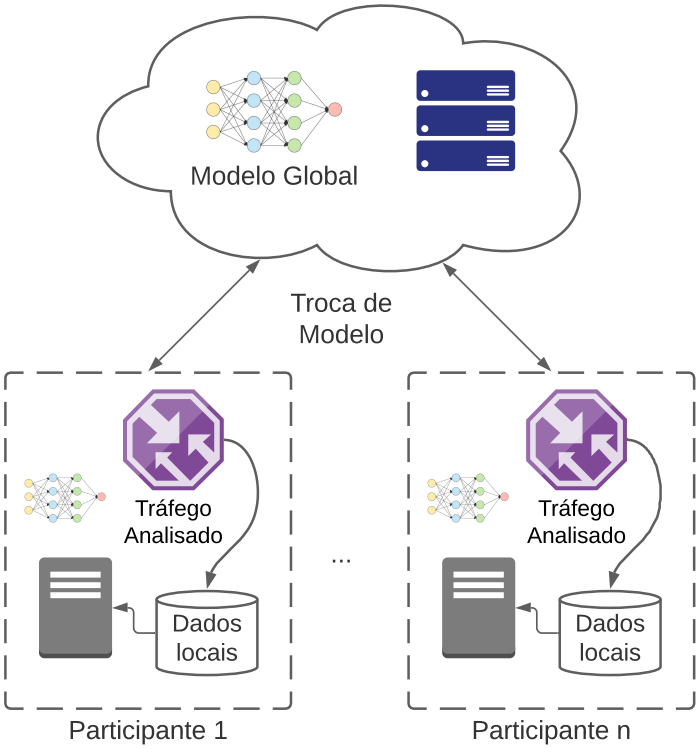


Figura 3.9. Arquitetura de referência do sistema de detecção de ataque baseado em aprendizado federado proposto em [Preuveneers et al., 2018]. Os participantes, nesse cenário, são sistemas de detecção de intrusão intermediários. O tráfego é enviado para os participantes da rede através de uma API. Esse tráfego é utilizado para treinar um modelo global sem expor os tráfegos pertencentes a cada participante.

de ações no ambiente MEC, uma abordagem *Double Deep Q-Network* (DDQN) foi adotada para gerenciar em conjunto os recursos de comunicação e computação. Os autores utilizaram DDQN devido ao bom desempenho comprovado em trabalhos anteriores [Sadeghi et al., 2018, He et al., 2018]. Para proteger a privacidade dos usuários, é proposta uma abordagem de aprendizado federado em que o treinamento pode ocorrer com os dados remanescentes nos equipamentos do usuário. Além disso, os algoritmos de aprendizado federado existentes, por exemplo FedAvg [Brendan McMahan et al., 2017], também garantem que o treinamento seja robusto para os dados não balanceados e non-IID dos equipamentos do usuário. Os resultados da simulação mostram que a abordagem DDQN com aprendizado federado atinge resultados semelhantes entre os equipamentos dos usuários em comparação com a abordagem DDQN centralizada (não utilizando aprendizado federado), consumindo menos recursos de comunicação e preservando a privacidade dos usuários participantes. No entanto, as simulações são realizadas apenas com 10 equipamentos de usuários. Se a implementação for expandida para atingir um número maior de equipamentos de usuário heterogêneos, pode haver atrasos significativos no processo de treinamento, especialmente porque o treinamento de um modelo DRL é computacionalmente intenso [Lim et al., 2020]. De maneira semelhante, Ren *et al.* propõem o uso de DRL na otimização de decisões de descarregamento de computação em sistemas IoT [Ren et al., 2019].

3.4.3. Associação de Estação Base

Chen *et al.* propõem uma abordagem de redes de estados de eco profundo (*Deep Echo State Networks* — DeepESNs) baseada em aprendizado federado para minimizar quebras de presença (*Breaks In Presence* — BIPs) para usuários de aplicações de realidade virtual (*Virtual Reality* — VR) [Chen et al., 2020, Chung et al., 2010]. Um evento BIP pode ser resultado de atraso na transmissão de informações que pode ser causado quando os movimentos do corpo do usuário obstruem o enlace sem fio. Os BIPs fazem com que o usuário perceba que está em um ambiente virtual, reduzindo assim a qualidade de sua experiência. Logo, uma política de associação de usuário deve ser projetada de forma que os BIPs sejam minimizados. O modelo do sistema consiste em estações base que cobrem um conjunto de usuários de VR. As estações base recebem informações de rastreamento carregadas de cada usuário associado, por exemplo, localização física e orientação, enquanto os usuários baixam vídeos VR para seu uso no aplicativo VR. Para transmissão de dados, os usuários de VR devem se associar a uma das estações base. Assim, um problema de minimização é formulado de forma em que os BIPs são minimizados em relação às localizações e orientações esperadas do usuário de VR. Para obter uma previsão das localizações e orientações dos usuários, a estação base precisa contar com as informações históricas dos usuários. No entanto, a informação histórica armazenada em cada estação base coleta apenas dados parciais de cada usuário, ou seja, um usuário se conecta a várias estações base e seus dados são distribuídos entre elas. Então, na implementação de aprendizado federado realizada pelos autores, cada estação base treina um modelo local usando seus dados parciais. Em seguida, os modelos locais são agregados para formar um modelo global capaz de generalização, ou seja, prever de forma abrangente a mobilidade e orientações de um usuário. Os resultados da simulação mostraram que o algoritmo DeepESN federado atinge menos BIPs, melhorando a experiência dos usuários [Chen et al., 2020].

3.4.4. Redes Veiculares

Um pré-requisito essencial para o desenvolvimento de um sistema de transporte inteligente são as redes veiculares de comunicação de baixa latência ultra confiável (*Ultra Reliable Low Latency Communication* — URLLC). No entanto, as técnicas existentes de gerenciamento de recursos de rádio não consideram eventos raros, como grandes comprimentos de fila na distribuição final. Para modelar a ocorrência de tais eventos de baixa probabilidade, Samarakoon *et al.* propõem o uso da teoria dos valores extremos (*Extreme Value Theory* — EVT) [De Haan e Ferreira, 2007], uma ferramenta poderosa que caracteriza as ocorrências de eventos extremos com baixa probabilidade [Samarakoon et al., 2018]. O EVT pode ser usada para modelar a distribuição em termos de três parâmetros conhecidos como localização, escala e forma. O EVT caracteriza assintoticamente as estatísticas de eventos extremos, fornecendo modelos analíticos para analisar o tráfego de rede, atrasos, taxas de pico e comunicação veículo a veículo ultraconfiável em sistemas sem fio. O maior desafio é que a abordagem requer amostras suficientes de informações do estado da fila (*Queue State Information* — QSI) e troca de dados entre veículos para treinar um modelo de inferência eficiente. Os usuários veiculares (*Vehicular Users* — VUEs) têm acesso a um número limitado de QSI que tenha excedido um limiar de sobrecarga, portanto, eles são incapazes de estimar a distribuição final de comprimentos de fila

em toda a rede. Então, os autores propõem uma abordagem de aprendizado federado em que os VUEs treinam o modelo de aprendizagem com dados mantidos localmente e carregam apenas seus parâmetros de modelo atualizados para as unidades de beira de estrada (*Roadside Units* – RSU). O RSU calcula a média dos parâmetros do modelo e retorna um modelo global atualizado para os VUEs. Em uma abordagem síncrona, todos os VUEs carregam seus modelos no final de um intervalo pré-especificado. No entanto, o envio simultâneo por vários veículos pode levar a atrasos na comunicação. Em contraste, para uma abordagem assíncrona, cada VUE apenas avalia e carrega seus parâmetros de modelo após um número predefinido de amostras de QSI coletadas. O modelo global também é atualizado sempre que uma atualização local é recebida, reduzindo assim os atrasos na comunicação. Para reduzir ainda mais a sobrecarga, a otimização de Lyapunov [Neely, 2010] para alocação de energia também é utilizada. Os resultados da simulação mostram que, sob essa estrutura, há uma redução do número de veículos com grandes comprimentos de fila, enquanto o aprendizado federado pode garantir uma troca mínima de dados em relação a uma abordagem centralizada.

Saputra *et al.* propõem uma abordagem chamada de *federated energy demand learning* (FEDL) para gerenciar recursos de energia em estações de carregamento (*Charging Stations* — CSs) para veículos elétricos (*Electric Vehicle* — EVs) [Saputra et al., 2019]. Quando um grande número de EVs se reúne em um CS pode levar a um congestionamento de transferência de energia. Para resolver isso, a energia é fornecida das redes de energia e reservada com antecedência para atender às demandas em tempo real dos EVs [You e Yang, 2014]. Como tal, é necessário prever a demanda de energia para EVs usando dados históricos de carga. No entanto, esses dados são normalmente armazenados separadamente em cada um dos CS que os EVs utilizam e são de natureza privada. Então, na abordagem FEDL proposta, cada CS treina o modelo de previsão de demanda em seu próprio conjunto de dados local antes de enviar apenas as informações de gradiente para o provedor de estação de carga (Charging Station Provider — CSP). Em seguida, as informações de gradiente do CS são agregadas para o treinamento do modelo global. Para melhorar a acurácia do modelo, os CSs são agrupados usando o algoritmo K-médias (*K-Means*) restrito [Bradley et al., 2000] com base em suas localizações físicas. O FEDL baseado em agrupamento reduz a dimensionalidade do conjunto de dados com base na classificação de características úteis e, portanto, a previsão tendenciosa pode ser minimizada [Li et al., 2018]. Os resultados da simulação mostram que a perda de um modelo FEDL agrupado é inferior aos algoritmos convencionais de aprendizado de máquina, por exemplo, regressor *perceptron* multicamadas [Boutaba et al., 2018]. Contudo, a privacidade dos dados do usuário ainda não é protegida por essa abordagem, uma vez que os dados do usuário são armazenados em cada um dos CSs.

3.5. Ataque ao Aprendizado Federado

Um dos principais objetivos do aprendizado federado é proteger a privacidade dos participantes do treinamento colaborativo, pois os participantes só precisam compartilhar os parâmetros do modelo treinado em vez de compartilhar seus dados locais. Entretanto, esse processo é suscetível a uma variedade de ataques, como o envenenamento de dados ou do modelo, em que um participante malicioso pode enviar parâmetros incorretos ou modelos corrompidos para enviesar o processo de aprendizagem durante a agregação glo-

bal. Consequentemente, o modelo global será atualizado incorretamente e todo o sistema de aprendizagem será corrompido. Em particular, isso anula o propósito do aprendizado federado, uma vez que o modelo global resultante pode ser corrompido ou os participantes podem ter a privacidade comprometida durante o treinamento do modelo. Esta seção discute os ataques emergentes sobre o aprendizado federado, bem como algumas contramedidas propostas para lidar com esses ataques.

3.5.1. Envenenamento do Conjunto de Dados

No aprendizado federado, um participante treina seu modelo local com seus dados e envia o modelo treinado ao servidor para posterior processamento. Nesse caso, é inviável para o servidor verificar se os parâmetros de cada participante são reais [Lim et al., 2020]. Assim, um participante malicioso pode envenenar o modelo global, criando dados rotulados de forma errada para treinar o modelo global com o objetivo de gerar parâmetros falsificados e enviesar o treinamento do modelo global [Fung et al., 2018]. Por exemplo, um participante malicioso pode gerar uma série de amostras falsas, e usá-las para treinar o modelo global para atingir seu objetivos.

Fung *et al.* investigam os impactos de um ataque de envenenamento de dados baseado no ataque Sybil em um sistema de aprendizado federado [Fung et al., 2018]. Em particular, no ataque Sybil, um participante malicioso tenta melhorar a eficácia do envenenamento de dados no treinamento do modelo global, criando vários participantes falsos. Os autores mostraram que com apenas dois participantes falsos, a taxa de sucesso do ataque pode atingir até 96,2%, tornando o modelo global incapaz de classificar corretamente os rótulos. Para mitigar os ataques Sybil, os autores propõem uma estratégia de defesa, chamada de *FoolsGold*. A ideia principal dessa abordagem é que os participantes honestos possam ser diferenciados dos participantes falsos com base em seus gradientes atualizados. No aprendizado federado, os dados de treinamento de cada cliente têm uma distribuição única e não são compartilhados. Os atacantes sybils compartilham um objetivo comum e contribuirão com atualizações que são semelhantes entre si, diferente dos clientes honestos. O FoolsGold usa essa suposição para modificar as taxas de aprendizagem de cada cliente em cada iteração de comunicação. A proposta é manter a taxa de aprendizado de clientes que fornecem atualizações exclusivas, enquanto reduz a taxa de aprendizado de clientes que contribuem repetidamente com atualizações de gradiente de aparência semelhante. Com o FoolsGold, o sistema pode se defender do ataque de envenenamento de dados Sybil com mudanças mínimas no processo de aprendizado federado convencional e sem exigir nenhuma informação auxiliar ao processo de aprendizagem. Através de resultados de simulações em três conjuntos de dados diversos (MNIST [Lecun et al., 1998], KDDCup [Cup, 1999], Amazon Reviews [Asuncion e Newman, 2007]), os autores mostram que o FoolsGold pode mitigar o ataque sob uma variedade de condições, incluindo diferentes distribuições de dados de participantes, variando os alvos de envenenamento e várias estratégias de ataque.

3.5.2. Envenenamento do Modelo de Aprendizado de Máquina

Ao contrário dos ataques de envenenamento de dados, que visam gerar dados falsos para causar impactos adversos ao modelo global, um ataque de envenenamento de modelo tenta envenenar diretamente o modelo global enviando parâmetros falsos para a

agregação. Conforme apresentado em trabalhos anteriores [Bhagoji et al., 2019, Bagdasaryan et al., 2020], os ataques de envenenamento de modelo de aprendizado de máquina são muito mais eficazes do que os de ataques de envenenamento de dados para aprendizado federado em grande escala com muitos participantes. O motivo é que, para ataques de envenenamento de dados, as atualizações de um participante malicioso são dimensionadas com base em seu conjunto de dados e no número de participantes do ambiente federado. No entanto, para ataques de envenenamento de modelo, um participante malicioso pode modificar o modelo atualizado, que é enviado ao servidor para agregação, diretamente. Como resultado, mesmo com um único invasor, todo o modelo global pode ser envenenado. Os resultados da simulação confirmam que mesmo um adversário altamente restrito com dados de treinamento limitados pode atingir alta taxa de sucesso na execução de ataques de envenenamento de modelo [Bhagoji et al., 2019]. Portanto, soluções para proteger o modelo global de ataques de envenenamento de modelo devem ser desenvolvidas.

Bhagoji *et al.* sugerem algumas soluções para prevenir ataques de envenenamento de modelo [Bhagoji et al., 2019]. Primeiro, com base nas atualizações dos parâmetros do modelo de um participante, o servidor verifica se o modelo compartilhado contribui na melhoria do desempenho do modelo global ou não. Caso contrário, o participante será marcado como um invasor em potencial e, após algumas rodadas de observação do modelo atualizado desse participante, o servidor pode determinar se esse é um participante malicioso ou não. A segunda solução é baseada na comparação entre os modelos atualizados compartilhados pelos participantes. Se um modelo atualizado de um participante for muito diferente dos demais, o participante é potencialmente malicioso. Então, o servidor continuará observando as atualizações desse participante antes de determinar se esse é um usuário malicioso ou não. No entanto, os ataques de envenenamento de modelo são extremamente difíceis de prevenir porque, ao treinar com grande número de participantes simultâneos, é inviável avaliar a melhoria de cada participante individualmente e em comparação com as dos demais. Como tal, soluções mais eficientes precisam ser investigadas.

Bagdasaryan *et al.* apresentam um ataque de envenenamento de modelo mais eficaz, que demonstrou atingir 100% de acurácia no ataque em apenas uma única rodada de aprendizagem [Bagdasaryan et al., 2020]. Um participante malicioso pode compartilhar seu modelo envenenado, que não apenas é treinado para enviesar o modelo, mas também contém uma função de *backdoor*. Essa função, proposta pelos autores, é chamada de *constrain-and-scale* e pode comprometer um ou mais participantes. Esse algoritmo permite ao invasor produzir um modelo com alta acurácia tanto na tarefa principal quanto na *backdoor*, ainda que não seja rejeitado pelo detector de anomalias do agregador. Essa função maliciosa pode fazer com que o modelo global classifique incorretamente, sem a necessidade de modificar o conjunto de dados local do participante malicioso. Por exemplo, uma função *backdoor* de classificação de imagem pode injetar um rótulo escolhido pelo invasor em todas as imagens com algumas características específicas, por exemplo, todos os cães com listras pretas podem ser classificados erroneamente como gatos. Os resultados das simulações mostraram que este ataque supera os ataques convencionais de envenenamento de dados em aprendizado federado. Bagdasaryan *et al.* mostram que em uma tarefa de previsão de palavras com 80.000 participantes no total, o comprometimento

de apenas oito participantes é o suficiente para atingir 50% de acurácia na classificação maliciosa.

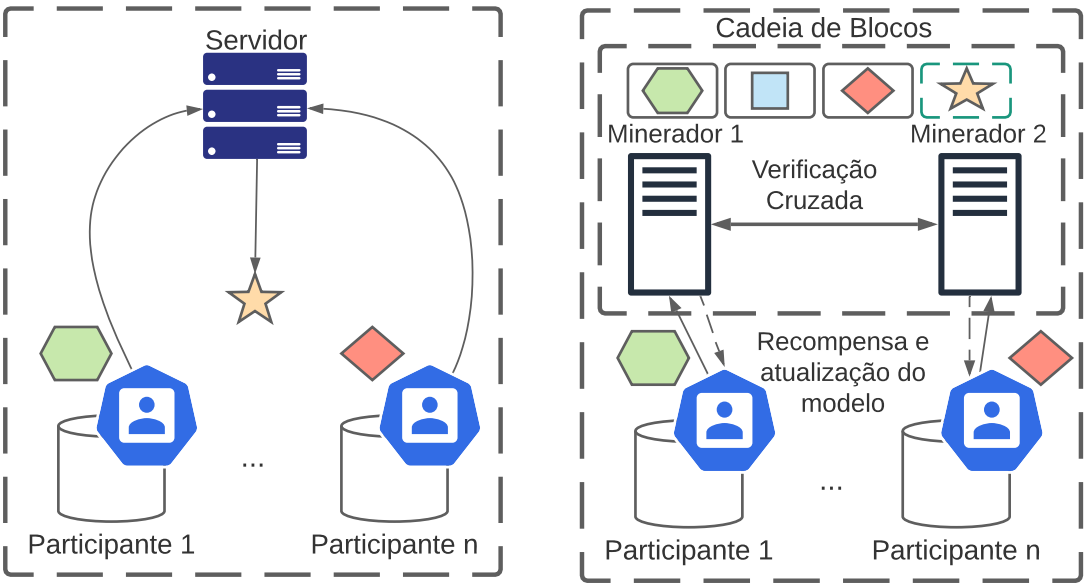
3.5.3. Ataque *Free-Riding*

O *free-riding* é outro ataque em aprendizado federado que ocorre quando um participante deseja se beneficiar do modelo global sem contribuir com o processo de aprendizagem. O participante malicioso, chamado de *free-rider*, finge ter um número pequeno de amostras para treinar ou seleciona um pequeno conjunto menor de seu conjunto de dados real para o treinamento, para economizar seus recursos computacionais. Como resultado, os participantes honestos precisam contribuir com mais recursos computacionais no processo de treinamento do modelo global. Para resolver esse problema, Kim *et al.* apresentam uma arquitetura de aprendizado federado baseada em cadeia de blocos, chamada BlockFL, na qual as atualizações do modelo local dos participantes são trocadas e verificadas por meio da tecnologia de cadeia de blocos [Kim et al., 2018]. Cada participante treina e envia o modelo local treinado para seu minerador associado na cadeia de blocos e, em seguida, recebe uma recompensa que é proporcional ao número de amostras de dados treinados, conforme ilustrado na Figura 3.10(b). Dessa forma, o arcabouço proposto não só evita participantes *free-riders*, mas também incentiva todos os participantes a contribuir para o processo de aprendizagem. Um modelo semelhante, também baseado em cadeia de blocos é introduzido por Weng *et al.*, visando fornecer confidencialidade de dados, capacidade de auditoria computacional e incentivos para os participantes do aprendizado federado [Weng et al., 2019]. No entanto, a utilização da tecnologia de cadeia de blocos implica a implementação e manutenção de mineradores para operar a cadeia de blocos. Além disso, os protocolos de consenso usados em redes de cadeia de blocos, como a prova de trabalho (*Proof-of-Work* — PoW), tendem a causar longo atraso na troca de informações e, assim, não são apropriados para implementar modelos de aprendizado federado.

3.5.4. Inversão do Modelo e Inferência dos Gradientes

A inversão de modelo é o ataque em que um adversário, em posse de um modelo treinado para prever uma variável específica, utiliza os pesos desse modelo para prever o conjunto de dados usados como entrada para treinar esse modelo, caracterizando assim um ataque à privacidade de atributos [Fredrikson et al., 2014]. O ataque busca tirar proveito da correlação entre o alvo, que seriam as características desconhecidas, e o resultado previsto pelo modelo. Esse ataque, em aprendizado federado, pode ser executado pelo servidor agregador que possui os modelos locais atualizados dos participantes. O servidor agregador pode ser honesto, mas curioso e, então, o servidor mantém a funcionalidade do ambiente de aprendizado federado, mas está disposto a descobrir os dados dos clientes. Fredrikson *et al.* propuseram o ataque de inversão de modelo para recuperar imagens de um modelo de reconhecimento facial [Fredrikson et al., 2015]. Os autores desenvolveram uma nova classe de ataque de inversão de modelo que explora os parâmetros de treinamentos revelados com as previsões.

Triastcyn e Faltings propõem um arcabouço chamado FedGP (*Federated Generative Privacy* – Privacidade Generativa Federada) [Triastcyn e Faltings, 2020]. A ideia principal dessa abordagem, é treinar redes adversárias gerativas (*Generative Adversarial*



(a) Aprendizado Federado tradicional, ou seja, média federada (FedAvg).

(b) Arquitetura da proposta BlockFL, onde o modelo global gerado em cada iteração de comunicação é verificado pelos mineradores e, posteriormente, salvo na cadeia de blocos.

Figura 3.10. Comparação entre a arquitetura do algoritmo FedAvg e BlockFL.

Networks — GANs) em clientes para produzir dados artificiais que substituam os dados reais dos clientes. Como alguns clientes podem ter dados insuficientes para treinar um GAN localmente, foi treinado um modelo de GAN federado. Dessa forma, os dados do usuário sempre permanecem em seus dispositivos. Além disso, o GAN federado produzirá amostras da distribuição comum entre usuários e não de um único usuário, o que aumenta a privacidade. A Figura 3.11 mostra a arquitetura do arcabouço. Os autores avaliaram a proteção fornecida executando o ataque de inversão de modelo e mostrando que o treinamento com o GAN federado reduz o vazamento de informações.

Zhang *et al.* propõem um esquema de criptografia homomórfica para preservar os parâmetros dos modelos dos usuários participantes de cada iteração de comunicação do aprendizado federado [Zhang et al., 2019]. Os autores propõem o *Privacy-Enhanced Federated Learning* (PEFL) para proteger os gradientes de um servidor não confiável. Isso é viabilizado principalmente pela criptografia dos gradientes locais dos participantes com o sistema de criptografia homomórfico de Paillier. Para reduzir os custos de computação do sistema de criptografia, é utilizado o método *Distributed Selective Stochastic Gradient Descent* (DSSGD) [Dean et al., 2012] na fase de treinamento local para diminuir o custo computacional da criptografia distribuída. Além disso, os gradientes criptografados são usados para agregação de soma segura no lado do servidor conforme mostrado na Figura 3.12. Dessa forma, o servidor não confiável aprende apenas as estatísticas agregadas de todas as atualizações dos participantes, enquanto as informações particulares de cada indivíduo estarão protegidas. Para a análise de segurança, os autores provam teoricamente que o esquema é seguro. Os resultados experimentais demonstram que o PEFL tem baixos custos de computação e, ao mesmo tempo, atinge alta precisão nas configurações do aprendizado federado. Zhang *et al.* também analisam o tempo para criptografar os parâ-

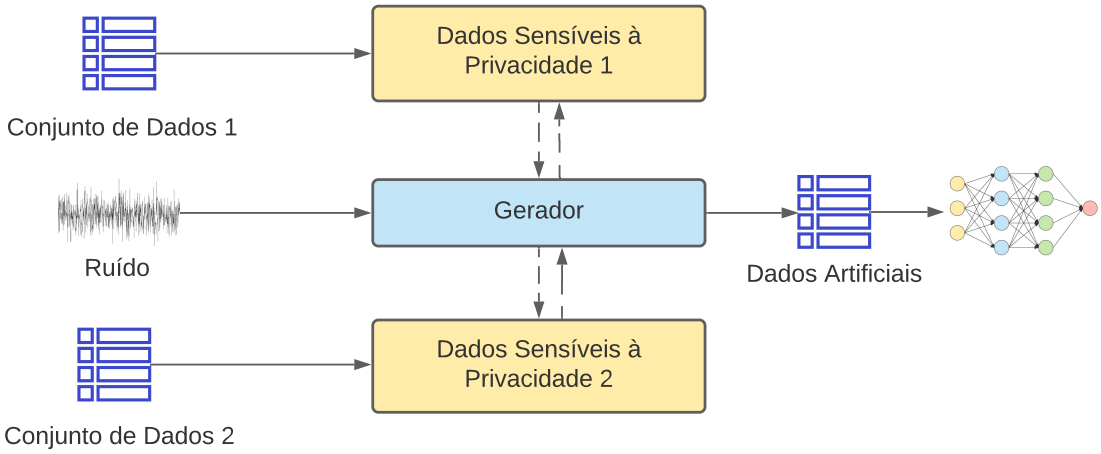


Figura 3.11. Arquitetura do FedGP com dois participantes. Os dados confidenciais são usados para treinar um GAN que posteriormente produz um conjunto de dados artificial privado.

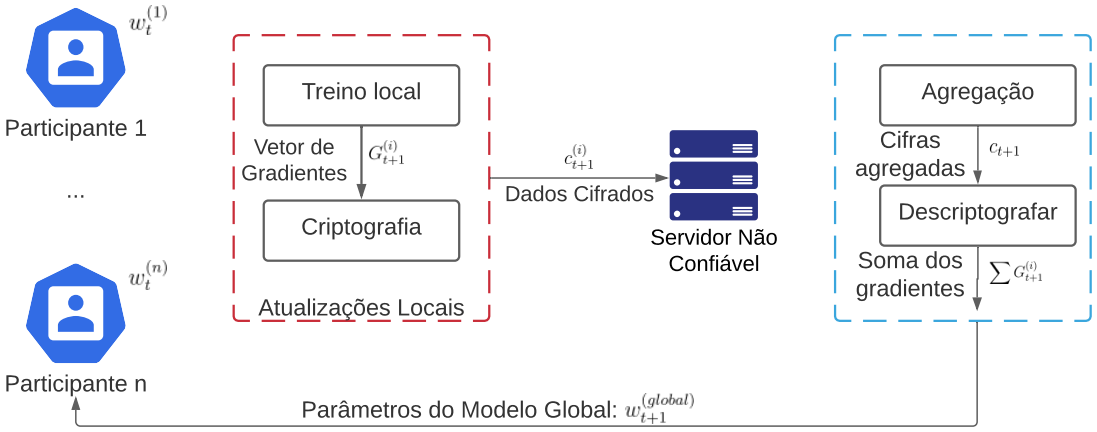


Figura 3.12. Na Arquitetura do PEFL, os participantes enviam os dados criptografados utilizando criptografia homomórfica e o servidor realiza a agregação retornando os pesos agregados a todos os participantes.

metros de um fragmentos dos pesos, utilizando DSSGD, e concluem que é pequeno, por vezes menor que 1 segundo [Zhang et al., 2019]. Contudo, os autores utilizam equipamentos de última geração nas avaliações, o que não corresponde à realidade do ambiente móvel real.

3.6. Desafios e Oportunidades de Pesquisa em Aprendizado Federado

Os desafios tornam a execução do aprendizado federado distinta de outros problemas clássicos, como aprendizado distribuído em configurações de *data center* ou análises de dados privados tradicionais. Os três principais desafios em aprendizado federado são:

1. o custo de comunicação - O ambiente de aprendizado federado compreende um grande número de dispositivos, por exemplo, milhões de *smartphones*, e a comunicação da rede pode ser mais lenta do que a computação local, devido à limitação de

recursos, como largura de banda e energia [Yang et al., 2019]. Para reduzir a comunicação no ambiente federado, dois aspectos principais devem ser considerados: i) reduzir o número total de rodadas de comunicação e ii) diminuir o tamanho das mensagens transmitidas em cada rodada;

2. a heterogeneidade de dispositivos - Os recursos de armazenamento, computação e comunicação de cada dispositivo do ambiente federado podem diferir devido à variabilidade no hardware (CPU e memória), conectividade de rede (2G, 3G, 4G, 5G e Wi-Fi) e energia (nível de bateria) [Yang et al., 2019]. Além disso, o tamanho da rede e as restrições relacionadas aos sistemas de cada dispositivo normalmente resultam em apenas uma pequena fração dos dispositivos ativos ao mesmo tempo [Bonawitz et al., 2019]. É comum que um dispositivo ativo falhe em uma determinada iteração devido a restrições de conectividade ou energia [Bonawitz et al., 2019]. A heterogeneidade de dispositivos intensifica os desafios, como mitigação de retardatários e tolerância a falhas. Os métodos de aprendizado federado desenvolvidos devem, portanto, i) prever a falha dos participantes, ii) tolerar hardware heterogêneo e iii) ser robustos o suficiente para que a falha dos participantes não afete a agregação;

3. a heterogeneidade estatística dos dados - Dispositivos frequentemente geram e coletam dados de maneira non-IID no ambiente federado. Usuários de telefones celulares, por exemplo, têm uso variado de linguagem no contexto de uma tarefa de previsão de próxima palavra. Além disso, o número de pontos de dados (vetor de características) entre os dispositivos pode variar significativamente e pode haver uma estrutura estatística subjacente que captura a relação entre os dispositivos e suas distribuições associadas [Yang et al., 2019]. Esse paradigma de geração de dados viola as suposições de dados independentes e identicamente distribuídos (i.i.d.) usadas frequentemente na otimização distribuída e pode adicionar complexidade na modelagem de problemas, análise teórica e avaliação empírica de soluções.

3.6.1. Custo de Comunicação

Comunicação é um ponto crucial em aprendizado federado. No aprendizado federado, é necessária uma série de rodadas de comunicação entre os participantes e o servidor para atingir a acurácia desejada. O treinamento de modelo de aprendizado profundo complexo, como em redes neurais convolucionais, pode compreender milhões de parâmetros em cada atualização [He et al., 2016]. A alta dimensionalidade das atualizações resulta na ocorrência de altos custos de comunicação e leva a um gargalo no treinamento. O gargalo é agravado devido a i) condições de rede dos dispositivos participantes [Wang et al., 2019] e ii) assimetrias nas conexões à Internet em que a taxa de envio é menor do que a taxa de recebimento, resultando em atrasos no envio dos modelos locais dos participantes [Konečnỳ et al., 2016]. Então, alguns trabalhos na literatura [Liu et al., 2019, Yao et al., 2018, Wang et al., 2019, Konečnỳ et al., 2016, Caldas et al., 2018, Tao e Li, 2018] visam melhorar a comunicação do aprendizado federado de três formas: i) aumentando a computação local, reduzindo, assim, a necessidade de rodadas de comunicação; ii) realizando a compressão do modelo local, reduzindo o tamanho dos dados enviados ao

servidor; iii) realizando atualizações com base na importância, onde é enviado somente os parâmetros que tiveram mudanças relevantes durante o treinamento local.

Para diminuir o número de rodadas de comunicação, a computação pode ser realizada nos dispositivos finais dos participantes antes de cada iteração de comunicação para agregação global. Por exemplo, o protocolo FedAvg [Brendan McMahan et al., 2017] considera duas maneiras de aumentar a computação nos dispositivos participantes, reduzindo o volume de dados a serem transmitidos: i) aumentar o paralelismo, selecionando mais participantes para participar de cada rodada de treinamento e ii) aumentar o esforço computacional por participante, no qual cada participante realiza mais atualizações locais antes da comunicação para agregação global.

Como uma extensão do FedAvg, Liu *et al.* validam um conceito semelhante [Liu et al., 2019]. Liu *et al.* propõem o algoritmo *Federated Stochastic Block Coordinate Descent* (FedBCD), no qual cada dispositivo participante realiza várias atualizações locais antes da comunicação para agregação global. A garantia de convergência é fornecida com uma calibração aproximada do número ideal de atualizações locais calculado a cada intervalo de comunicação. Semelhantemente, Yao *et al.* propõem computação aumentada em cada dispositivo participante, adotando um modelo de dois fluxos(*two-stream model*) [Yao et al., 2018].O modelo de dois fluxos é comumente usado no aprendizado por transferência e adaptação de domínio [Long et al., 2015]. Durante cada rodada de treinamento, o modelo global é recebido pelos participantes e fixado como referência no processo de treinamento local. Durante a atualização local, o participante aprende não apenas com dados locais, mas também com de outros participantes utilizando o modelo global como referência. Isso é feito por meio da incorporação da Discrepância Média Máxima (Maximum Mean Discrepancy — MMD) na função de perda. O MMD mede a distância entre as médias de duas distribuições de dados [Long et al., 2015]. Ao minimizar a perda do MMD entre os modelos local e global, o participante pode extrair características mais generalizadas do modelo global, acelerando assim a convergência do processo de treinamento e reduzindo rodadas de comunicação. As simulações foram realizados utilizando os conjuntos de dados CIFAR-10 e MNIST e usando modelos de aprendizado profundo, como redes neurais convolucionais. Os resultados mostram que o aprendizado federado de dois fluxos proposto pode atingir a acurácia desejável em 20% menos rodadas de comunicação, mesmo quando os dados são non-IID [Yao et al., 2018].

Wang *et al.* propõem um algoritmo para determinar a frequência de agregação global para que os recursos disponíveis sejam usados com maior eficiência [Wang et al., 2019]. Para isso, os autores analisam o limite de convergência do aprendizado federado baseada no gradiente descendente a partir de uma perspectiva teórica, na qual um novo limite de convergência é proposto, que incorpora a distribuição dos dados non-iid entre os nós e um número arbitrário de atualizações locais entre duas agregações globais. Utilizando esse limite de convergência teórico, os autores propõem um algoritmo de controle que aprende a distribuição dos dados, dinamicidade do sistema e características do modelo. Com base no algoritmo proposto, o sistema adapta dinamicamente a frequência da agregação global em tempo real para minimizar a perda de aprendizado. Por fim, os autores avaliam o desempenho do algoritmo de controle proposto através de experimentos utilizando conjuntos de dados reais, tanto em um cenário com protótipo de hardware quanto em um ambiente simulado. Os resultados confirmam que a abordagem proposta

fornece desempenho próximo ao ideal para diferentes distribuições de dados, vários modelos de aprendizado de máquina e configurações do sistema com diferentes números de nós de borda. O sistema obtém uma relação de compromisso desejável entre atualização local e agregação global, de modo a minimizar a função de perda.

Embora os métodos de atualização local possam reduzir o número total de rodadas de comunicação, esquemas de compressão de modelo também podem ser utilizados para reduzir o volume de dados em trânsito no aprendizado federado. Alguns exemplos desses esquemas de compressão incluem "esparsificação´´, subamostragem e quantização, os quais reduzem significativamente o tamanho das mensagens comunicadas em cada rodada. Esses métodos foram amplamente estudados, tanto empiricamente quanto teoricamente, na literatura para treinamento distribuído em ambientes de *datacenter*. Konevcny *et al.* propõem duas maneiras de atualizações do modelo local, a atualização estruturada (*Structured update*) e a atualização esboçada (*Sketched update*) [Konečnỳ et al., 2016]. Os modelos são propostos para reduzir o tamanho das atualizações enviadas dos participantes para o servidor durante cada rodada de comunicação. As atualizações estruturadas restringem as atualizações dos participantes a terem uma estrutura pré-especificada, com baixa classificação (*Low rank*) e máscara aleatória (*Random mask*). Na estrutura de baixa classificação, cada atualização é forçada a ser o produto de duas matrizes. Dessas duas matrizes, uma matriz é gerada aleatoriamente e mantida constante durante cada rodada de comunicação, enquanto a outra é otimizada. Assim, a matriz aleatória pode ser neste caso compactada na forma de uma semente aleatória e apenas a matriz otimizada precisa ser enviada ao servidor. Para a estrutura de máscara aleatória, cada atualização local é restrita a ser uma matriz esparsa seguindo um padrão esparso aleatório predefinido, gerado de forma independente durante cada rodada. Apenas as entradas diferentes de zero devem ser enviadas para o servidor. Por outro lado, as atualizações de esboço referem-se à abordagem de codificar a atualização em uma forma compactada antes da comunicação com o servidor, que subsequentemente decodifica as atualizações antes da agregação. Um exemplo de atualização esboçada é a abordagem de subamostragem, na qual cada participante comunica apenas um subconjunto aleatório da matriz de atualização. O servidor então calcula a média das atualizações sub-amostradas para obter uma estimativa imparcial da média real. Outro exemplo, a abordagem de quantização probabilística [Han et al., 2015] prevê que as matrizes de atualização são vetorizadas e quantizadas para cada escalar. Para reduzir o erro de quantização, uma rotação aleatória estruturada, que é o produto de uma matriz Walsh-Hadamard e uma matriz diagonal binária, é aplicada antes da quantização [Suresh et al., 2017].

Os resultados da simulação realizada por Konevcny *et al.* na tarefa de classificação de imagens utilizando o conjunto de dados CIFAR-10 mostram que, para atualizações estruturadas, a máscara aleatória obteve desempenho melhor do que a abordagem de classificação baixa [Konečnỳ et al., 2016]. A abordagem de máscara aleatória também atinge maior acurácia do que atualizações esboçadas, uma vez que esta envolve a remoção de algumas informações obtidas durante o treinamento. No entanto, a combinação de todas as três ferramentas de esboço, ou seja, subamostragem, quantização e rotação, pode atingir maior taxa de compressão e convergência mais rápida, embora com alguns sacrifícios de acurácia. Caldas *et al.* realizaram uma extensão dos estudos de Konevcny *et al.*, propondo uma compressão com perdas para reduzir os custos de comunicação [Caldas et al., 2018].

Outra forma de poupar a quantidade de bytes que precisa ser transmitido no aprendizado federado é chamada de atualização baseada em importância. Nessa técnica, considera-se o fato de que a maioria dos valores dos parâmetros de um modelo de redes neurais profundas são esparsamente distribuídos e próximos de zero [Strom, 2015]. Dessa forma, Tao *et al.* propõem o algoritmo de gradiente descendente estocástico de borda (*edge Stochastic Gradient Descent* — eSGD), que seleciona apenas uma pequena fração de gradientes importantes para enviar ao servidor do aprendizado federado para atualização de parâmetros durante cada rodada de comunicação [Tao e Li, 2018]. O algoritmo eSGD acompanha os valores de perda em duas iterações de treinamento consecutivas. Se o valor de perda da iteração atual for menor que a iteração anterior, isso implica que os gradientes de treinamento atuais e os parâmetros do modelo são importantes para a minimização da perda de treinamento, portanto, seus respectivos pesos ocultos são atribuídos a um valor positivo. Além disso, o gradiente também é comunicado ao servidor para atualização dos parâmetros. Por outro lado, se a perda aumenta em comparação à iteração anterior, outros parâmetros são selecionados para serem atualizados com base em seus valores de pesos ocultos. Um parâmetro com maior valor de peso oculto tem maior probabilidade de ser selecionado, pois foi rotulado como importante várias vezes durante o treinamento. O peso oculto é sempre um valor real positivo e os parâmetros que possuem pesos ocultos pequenos podem retardar a convergência. Os valores de gradiente de cada rodada são acumulados como valores residuais. Uma vez que os resíduos surgem de diferentes iterações de treinamento, cada atualização do resíduo é ponderada com um fator de desconto usando a técnica de correção de *momentum*. Quando o gradiente residual acumulado atinge um limiar, eles são escolhidos para substituir as coordenadas dos gradientes menos importantes de acordo com os valores de pesos ocultos. Os resultados da simulação mostraram que eSGD com uma taxa de remoção de 50% atinge maior acurácia do que o algoritmo *thresholdSGD* [Strom, 2015], que usa um valor de limiar fixo para determinar quais coordenadas de gradiente devem ser descartadas. Assim, o eSGD reduz o tamanho do gradiente comunicado, mas ainda sofre com a perda de acurácia em comparação com as abordagens SGD padrão. Por sua vez, Wang *et al.* propõem o algoritmo Aprendizado federado com redução de comunicação (*Communication-Mitigated Federated Learning* — CMFL) que carrega apenas atualizações de modelos locais relevantes para reduzir os custos de comunicação, garantindo a convergência global [Wang et al., 2019]. Em cada iteração, a atualização do modelo local de um participante é primeiro comparada com o modelo global para identificar se a atualização é relevante. Os resultados da simulação mostraram que o CMFL requer 3,47 vezes menos rodadas de comunicação para atingir 80% de acurácia para classificação de imagem utilizando o conjunto de dados MNIST e 13,97 vezes menos para previsão de próxima palavra, ambos em comparação com o algoritmo FedAvg, que é usualmente utilizado como parâmetro de comparação.

3.6.2. Heterogeneidade de Dispositivos

No ambiente de aprendizado federado, há uma variação significativa nas características dos dispositivos dos participantes na rede, pois esses dispositivos podem possuir hardware, conectividade de rede e nível de bateria distintos. Essas características de sistema tornam problemas como atrasos significativamente mais prevalentes do que em ambientes típicos de *datacenter*. Para resolver esse problema, soluções foram propostas.

A seleção de participantes refere-se à seleção de dispositivos para participar de cada rodada de comunicação. Normalmente, um conjunto de participantes é selecionado aleatoriamente pelo servidor. O progresso do treinamento do aprendizado federado é limitado pelo tempo de treinamento dos dispositivos participantes mais lentos [Lim et al., 2020], ou seja, retardatários. Novos protocolos de seleção de participantes são, portanto, investigados para resolver o gargalo de treinamento em aprendizado federado. Nishio *et al.* propõem um novo protocolo de aprendizado federado chamado FedCS [Nishio e Yonetani, 2019]. A proposta é um arcabouço MEC em que a operadora do MEC é o servidor de aprendizado federado que coordena o treinamento em uma rede celular que compreende dispositivos móveis participantes que possuem recursos heterogêneos. O servidor primeiro conduz uma etapa de solicitação de recursos para reunir informações como estados de canais sem fio e recursos de computação de um subconjunto de participantes selecionados aleatoriamente. Com base nessas informações, o operador do MEC seleciona o número máximo possível de participantes que pode concluir o treinamento dentro de um prazo pré-estabelecido para a fase de agregação global subsequente. Os resultados da simulação mostraram que, em comparação com o protocolo FedAvg, que contabiliza apenas o prazo de treinamento, o FedCS obtém maior acurácia, pois envolve mais participantes em cada rodada de treinamento ao invés de um número fixo. Da mesma forma, Kang *et al.* consideram sobrecargas de sistemas incorridas a cada dispositivo ao projetar mecanismos de incentivo para encorajar dispositivos com dados de alta qualidade a participarem do processo de aprendizagem [Kang et al., 2019a]. Embora esses métodos se concentrem principalmente na variabilidade dos sistemas para realizar a amostragem ativa, é vantajoso considerar a amostragem ativa de um conjunto de dispositivos pequenos, mas suficientemente representativos, com base na estrutura estatística dos dados.

Outra forma de tentar sobrepor atrasos gerados pela heterogeneidade dos dispositivos é utilizar esquemas de aprendizado federado baseado em uma comunicação assíncrona. Configurações tradicionais de *data centers* são baseadas em esquemas síncronos, nos quais nós trabalhadores esperam uns aos outros para sincronização, e assíncronos, em que os nós trabalhadores executando de forma independente sem sincronização são usados para paralelizar algoritmos de otimização iterativa [Li et al., 2020]. Os esquemas síncronos são simples e garantem um modelo computacional serial equivalente trivial, mas também são mais suscetíveis a atrasos devido à variabilidade dos dispositivos. Os esquemas assíncronos são abordagens usadas para mitigar retardatários em ambientes heterogêneos, particularmente em sistemas de memória compartilhada. No entanto, dependem de suposições de atraso limitado para controlar o grau de desatualização. O algoritmo FedAvg [Brendan McMahan et al., 2017] agrega parâmetros de forma síncrona e é, portanto, suscetível ao efeito retardatário, já que cada rodada de treinamento progride na velocidade do dispositivo mais lento, uma vez que o servidor espera que todos os dispositivos concluam o treinamento local antes que a agregação global possa ocorrer. Além disso, o algoritmo não considera participantes que ingressam quando a rodada de treinamento já está em andamento.

Sprague *et al.* constataram empiricamente que uma abordagem assíncrona é robusta para participantes que ingressam durante a rodada de treinamento em progresso, bem como quando a federação envolve dispositivos participantes com recursos de processamento heterogêneos [Sprague et al., 2019]. No entanto, a convergência do modelo é

significativamente atrasada quando os dados são non-IID e desbalanceados. Como uma melhoria, Xie *et al.* propõem o algoritmo FedAsync, em que cada atualização local recém-recebida é ponderada de forma adaptativa de acordo com seu grau de obsolência, o qual é definido como a diferença entre a época atual e a iteração à qual pertence a atualização recebida [Xie et al., 2019]. Assim, uma atualização desatualizada de um retardatário que está obsoleta, pois deveria ter sido recebida em rodadas de treinamento anteriores, possui menor peso, mas ainda é considerada. Além disso, os autores comprovam a garantia de convergência para um grupo restrito de problemas não convexos. No entanto, os hiperparâmetros do algoritmo FedAsync precisam ser ajustados para garantir a convergência em diferentes configurações. O algoritmo é incapaz de generalizar para se adequar às restrições de computação dinâmica de dispositivos heterogêneos. Dada a incerteza em torno da confiabilidade do aprendizado federado assíncrono, métodos síncronos continuam sendo mais utilizados atualmente [Lim et al., 2020].

Em todos os cenários considerados, na prática, os participantes podem relutar em participar de uma federação sem receber compensação, uma vez que os modelos de treinamento consomem recursos computacionais. Além disso, existe assimetria de informações entre o servidor e os participantes, uma vez que os participantes têm maior conhecimento dos recursos de computação disponíveis e da qualidade dos dados. Portanto, os mecanismos de incentivo devem ser cuidadosamente projetados para incentivar a participação e reduzir os potenciais impactos adversos da assimetria de informação.

Feng *et al.* propõem um esquema de precificação de serviço no qual os participantes atuam como provedores de serviços de treinamento para um proprietário de modelo, *i.e.*, o servidor do aprendizado federado [Feng et al., 2019]. Para superar a ineficiência energética na transferência de atualizações de modelos, uma rede de retransmissão cooperativa é proposta para apoiar a transferência e comercialização de atualizações de modelos. A interação entre os participantes e o proprietário do modelo é modelada como um jogo de Stackelberg [Osborne et al., 2004], no qual o proprietário do modelo é o comprador e os participantes são os vendedores. A proposta do jogo de Stackelberg é que cada participante pode decidir, não cooperativamente, sobre seu próprio preço de maximização de lucro. No sub-jogo de nível inferior, o proprietário do modelo determina o tamanho dos dados de treinamento para maximizar os lucros, considerando a relação côncava crescente entre a acurácia do modelo e o tamanho dos dados de treinamento. No sub-jogo de nível superior, os participantes decidem o preço por unidade de dados para maximizar seus lucros individuais. Os resultados da simulação mostraram que o mecanismo proposto pode garantir a unicidade do equilíbrio de Stackelberg. Atualizações de modelo que contêm informações valiosas têm preços mais elevados no equilíbrio de Stackelberg. De forma semelhante ao trabalho anterior, Sarikaya e Ercetin modelam a interação entre os participantes e o proprietário do modelo como um jogo Stackelberg, que é adequado para representar a interação servidor/participante dos envolvidos no treinamento [Sarikaya e Ercetin, 2020].

Ao contrário das abordagens anteriores, Zhan *et al.* adotam uma abordagem baseada em aprendizado por reforço profundo (*Deep Reinforcement Learning* – DRL) para resolver as formulações de Stackelberg [Zhan et al., 2020]. Na formulação do DRL, o servidor atua como um agente que decide um pagamento em resposta ao nível de participação e histórico de pagamento dos nós participantes, com o objetivo de minimizar

despesas com incentivos. Em seguida, os participantes determinam um nível de participação ideal em resposta à política de pagamento. Esse mecanismo de incentivo baseado em aprendizado por reforço permite que o servidor obtenha uma política ótima em resposta ao seu estado observado, sem necessitar nenhuma informação prévia.

Kang *et al.* apresentam a reputação como uma métrica para medir a confiabilidade dos participantes do aprendizado federado e projetam um esquema de seleção de participantes baseado em reputação [Kang et al., 2019b, Kang et al., 2019a]. Nesse cenário, cada participante tem um valor de reputação derivado de duas fontes, i) opiniões de reputação diretas de interações anteriores com o servidor e ii) opiniões de reputação indiretas de outros editores de tarefas,que são outros servidores de aprendizado federado de outras aplicações. As opiniões indiretas de reputação são armazenadas em uma cadeia de blocos aberta de reputação [Dennis e Owen, 2015] para garantir o gerenciamento seguro da reputação de uma maneira descentralizada. Antes do treinamento do modelo, os participantes escolhem um contrato que melhor se adapta à acurácia de seu conjunto de dados e às condições de seus recursos locais.

3.6.3. Heterogeneidade Estatística dos Dados

Em aprendizado de máquina distribuído, tradicionalmente, o servidor central tem acesso a todo o conjunto de dados de treinamento [Zhan et al., 2020]. Como tal, o servidor pode dividir o conjunto de dados em subconjuntos que seguem distribuições semelhantes. Os subconjuntos são posteriormente enviados aos nós participantes para treinamento distribuído. No entanto, essa abordagem é impraticável para aprendizado federado, uma vez que o conjunto de dados local só pode ser acessado pelo proprietário dos dados. Os desafios surgem ao treinar modelos federados utilizando dados que são altamente distribuídos de forma não idêntica entre os dispositivos, tanto em modelagem de dados heterogêneos quanto em análise comportamental da convergência dos procedimentos de treinamento.

Em aprendizado de máquina, existem trabalhos na literatura que visam modelar a heterogeneidade estatística utilizando métodos como meta-aprendizado e aprendizado multitarefa. O meta-aprendizado consiste de algoritmos de aprendizado automático aplicados a metadados [Vilalta e Drissi, 2002]. O aprendizado multitarefa é uma abordagem de transferência de aprendizado que melhora a generalização usando as informações contidas nos parâmetros de treinamento de tarefas relacionadas como um viés indutivo (*bias*) [Caruana, 1997]. Nesse contexto, uma tarefa pode ser minimizar a função de perda. Essas ideias foram recentemente estendidas ao ambiente federado [Li et al., 2020].

Smith *et al.* propõem uma estrutura de otimização projetada para o ambiente federado chamada MOCHA, que permite a personalização através do aprendizado de modelos separados, porém relacionados, para cada dispositivo, ao mesmo tempo em que potencializa uma representação compartilhada através do aprendizado multitarefa [Smith et al., 2017]. O MOCHA é calibrado com base nas restrições de recursos de um dispositivo participante como, por exemplo, condições da rede e estados da CPU dos dispositivos. No entanto, o MOCHA não pode ser aplicado a modelos de aprendizado profundo não-convexos [Smith et al., 2017].

Frequentemente, há relações internas entre a estrutura dos modelos locais dos participantes. Para capturar essas relações, o aprendizado multitarefa é uma estratégia

natural que melhora o desempenho e aumenta o tamanho da amostra eficaz para cada nó. Esse método tem garantias de convergência teórica comprováveis para os objetivos considerados, mas é limitado em sua capacidade de escalar para redes massivas e é restrito a objetivos convexos [Smith et al., 2017].

Eichner *et al.* investigam uma solução pluralista adaptativa, escolhendo entre um modelo global e modelos específicos de dispositivo, para abordar os padrões cíclicos em amostras de dados durante o treinamento federado [Eichner et al., 2019]. Outra abordagem visa modelar uma topologia em estrela como uma rede bayesiana e realiza inferência variacional durante o aprendizado [Corinzia e Buhmann, 2019]. A inferência variacional é uma abordagem que estima distribuições difíceis ou complexas *a posteriori*, i.e., a probabilidade de uma variável dado um evento. Embora esse método lide com funções não-convexas, é caro realizar generalização em redes federadas grandes. Apesar desses avanços recentes, os principais desafios ainda permanecem no desenvolvimento de métodos para modelagem heterogênea que sejam robustos, escaláveis e automatizados em configurações federadas.

Ao modelar dados no ambiente de aprendizado federado, é importante considerar questões além da acurácia. Em particular, resolver ingenuamente uma função de perda agregada, como a vista na Equação 2, pode implicar implicitamente obter vantagem e/ou desvantagem de alguns dos dispositivos, pois o modelo aprendido pode se tornar tendencioso para dispositivos com maiores quantidades de dados [Li et al., 2020]. Para garantir a convergência, Chiu *et al.* propõem o FedProx, que modifica a função de perda global para incluir também um parâmetro ajustável que restringe o quanto as atualizações locais afetam os parâmetros do modelo predominante [Chiu et al., 2020]. O algoritmo FedProx é ajustado de forma adaptativa. Quando a perda de treinamento está aumentando, as atualizações do modelo são ajustadas para afetar menos os parâmetros atuais. Da mesma forma, Huang *et al.* propõem o algoritmo LoAdaBoost FedAvg, no qual os participantes treinam o modelo em seus dados locais e comparam a perda da entropia cruzada com a perda mediana da rodada de treinamento anterior [Huang et al., 2020]. Se a perda da entropia cruzada atual for maior, o modelo é retreinado antes da agregação global para aumentar a eficiência do aprendizado.

3.7. Exemplo de Criação de uma Aplicação de Aprendizado Federado

A seguir é criado um cenário prático de aplicação de aprendizado federado através de simulação. O cenário é um dos mais utilizados para a avaliação de novos algoritmos e métodos em aprendizado federado [Brendan McMahan et al., 2017, Yang et al., 2019]. Na simulação, é utilizado o conjunto de dados clássico MNIST para classificação de imagem. A simulação descreve passo a passo da criação de um ambiente utilizando a linguagem Python [8]. O código-fonte está disponível no GitHub [9]. Em um cenário real, cada participante apresenta seus próprios conjuntos de dados local, isoladamente. É importante lembrar que o objetivo do aprendizado federado é enviar os parâmetros dos modelos locais para o servidor e nunca os dados dos participantes. Para simular os dados dos participantes, também chamados de clientes, são criados fragmentos do conjunto de dados principal.

[8]Disponível em: https://www.python.org/.

[9]Disponível em: https://github.com/helioncneto/FederatedLearning/blob/master/FederatedLearning_python.py.

Para essa simulação, são criados 10 fragmentos do conjunto de dados MNIST [10], um para cada participante. O conjunto de dados consiste em imagens contendo 70.000 dígitos escritos manualmente, em que cada classe é mantida em diretórios separados.

O aprendizado federado é mais adequado para modelos de aprendizado parametrizados como as redes neurais. Técnicas de aprendizado de máquina, como KNN ou similares, que simplesmente armazenam os dados de treinamento durante o aprendizado, não se beneficiam do aprendizado federado. Assim, a aplicação prática cria uma rede perceptron multicamadas (*Multilayer Perceptron* — MLP) com 3 camadas para a tarefa de classificação, utilizado a biblioteca Keras [11]. O primeiro passo é instalar os pacotes necessários para a simulação. A lista de pacotes esté no arquivo requirements.txt. Execute o comando:

```
$ pip install -r requirements.txt
```

e, então, são instaladas as seguintes bibliotecas: *Sci-kit learn*, TensorFlow e numpy. Inicialmente, são importadas as bibliotecas necessárias, como mostrado no Código Fonte 3.1.

Código Fonte 3.1. Importação das bibliotecas necessárias para a execução da aplicação prática.

```
1  import numpy as np
2  import random
3  import cv2
4  import os
5  import tensorflow as tf
6  from imutils import paths
7  from sklearn.preprocessing import LabelBinarizer
8  from sklearn.model_selection import train_test_split
9  from sklearn.metrics import accuracy_score
10 from tensorflow.keras.models import Sequential
11 from tensorflow.keras.layers import Activation
12 from tensorflow.keras.layers import Dense
13 from tensorflow.keras.optimizers import SGD
14 from tensorflow.keras import backend as K
```

Uma função para carregar os dados na memória é criada e são mantidos 30% dos dados para testar o modelo global treinado mais tarde. As imagens são lidas do disco rígido em escala cinza e posteriormente achatada para criar um vetor de características. A etapa de achatamento da imagem é utilizada, pois, é usada uma arquitetura de rede MLP e cada imagem deve ser passada para o modelo em forma de um vetor. Para obter o rótulo de classe de uma imagem, é utilizada a *string* do caminho, pois cada dígito está em um diretório separado. Outro pré-processamento realizado é o dimensionamento das imagens entre 0 e 1, para diminuir o impacto da variação do brilho do *pixel*. Depois disso,

[10] Disponível em http://yann.lecun.com/exdb/mnist/.
[11] Disponível em https://keras.io/. Acessado em 26/09/2020

Código Fonte 3.2. Preprocessamento do conjunto de dados.

```
def load_mnist_bypath(paths, verbose=-1):
    data = list()
    labels = list()
    # loop nas imagens de entrada
    for (i, imgpath) in enumerate(paths):
        # carrega a imagem e extrai os rótulos da classe
        im_gray = cv2.imread(imgpath, cv2.IMREAD_GRAYSCALE)
        image = np.array(im_gray).flatten()
        label = imgpath.split(os.path.sep)[-2]
        # aqui é feita a escala da img para [0, 1] para diminuir o
        # impacto do brilho de cada pixel
        data.append(image/255)
        labels.append(label)
    # retorna uma tupla dos dados e rótulos
    return data, labels

# Declara o caminho do conjunto de dados mnist
img_path = 'mnist/trainingSet/trainingSet'

# Gera a lista de caminhos, utilizando a funcao list_images da
# biblioteca paths
image_paths = list(paths.list_images(img_path))

# Carrega as imagens em arrays
image_list, label_list = load_mnist_bypath(image_paths, verbose=10000)

# Realiza o one-hot encoded para podermos utilizar a
# funcao de perda sparse-categorical-entropy
lb = LabelBinarizer()
label_list = lb.fit_transform(label_list)
# divide os dados em treinamento e conjunto de teste
X_train, X_test, y_train, y_test = train_test_split(image_list,
            label_list, test_size=0.3, random_state=42)
```

é utilizado o objeto `LabelBinarizer` da biblioteca *sci-kit learn* para realizar o *one-hot-encoding* nos rótulos. No *one-hot-encoding*, em vez de ter o rótulo do dígito 1 como número 1, ele terá a forma de um vetor como [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]. Com esse estilo de rotulagem, pode-se usar a função de perda de entropia cruzada dos modelos. Foi utilizado o objeto `train_test_split` da biblioteca *sci-kit learn* para dividir os dados em treino/teste. As etapas de preprocessamento são evidenciadas no Código Fonte 3.2.

O próximo passo é criar uma função que gere os dados locais dos clientes de forma non-IID, como mostrado no Código Fonte 3.3, utilizando o conjunto de dados de treinamento. É criada uma lista de nomes de clientes usando um prefixo que é passado como parâmetro da função. As listas de dados e seus respectivos rótulos são compactados

Código Fonte 3.3. Função de geração dos dados locais dos clientes.

```
def create_clients(image_list, label_list, num_clients=10,
                   initial='clients'):

    # cria a lista de nomes de clientes
    client_names = ['{}_{}'.format(initial, i + 1)
                    for i in range(num_clients)]

    # embaralha os dados
    data = list(zip(image_list, label_list))
    random.shuffle(data)

    # fragmenta os dados e divide para cada cliente
    size = len(data) # num_clients

    shards = [data[i:i + size]
            for i in range(0, size * num_clients, size)]

    # Verifica se o numero de fragmento é igual ao de clientes
    assert(len(shards) == len(client_names))

    return {client_names[i]: shards[i] for i in range(len(client_names))}

# Cria os clientes
clients = create_clients(X_train, y_train, num_clients=100,
                   initial='client')
```

e depois embaralhados. Por fim, essa função cria fragmentos desses dados baseado no número desejado de clientes, `num_clients`. Então, é retornado um dicionário contendo o nome de cada cliente como chave e seu conjunto de dados como valor. Essa função, então, é aplicada ao conjunto de dados de treinamento.

Em seguida, cada conjuntos de dados dos clientes é convertido para um conjunto de dados do TensorFlow, e, posteriormente, transformado em lote. Para simplificar essa etapa de conversão de dados e evitar a repetição, é realizado o encapsulamento desse procedimento em uma função chamada `batch_data`, como mostrado no Código Fonte 3.4. É importante lembrar que cada conjunto de dados dos clientes deve se tornar uma lista de tuplas (dados, rótulo) de `create_clients`. Nessa função, primeiro, é feita a separação da tupla em listas. Em seguida, é criado o conjunto de dados TensorFlow, realizado o embaralhamento dos dados e o agrupamento a partir dessas listas. Durante a aplicação da função, é separado o conjunto de teste e treinamento de cada cliente.

O Código Fonte 3.5 mostra a criação do modelo global MLP de 3 camadas que serve de modelo para iniciar a tarefa de classificação. Na simulação, são utilizados os módulos Keras que foram importados anteriormente. Para construir um novo modelo, é criada uma classe MLP e o seu construtor inicia o modelo. Como é usado o conjunto de

Código Fonte 3.4. Função de encapsulamento para a adequação do conjunto de dados à biblioteca TensorFlow.

```
1  def batch_data(data_shard, b=32):
2      # Separa cada fragmento em listas de dados e rótulo
3      data, label = zip(*data_shard)
4      dataset = tf.data.Dataset.from_tensor_slices((list(data),
5                  list(label)))
6      return dataset.shuffle(len(label)).batch(b)
7
8
9  # Processa e agrupa os dados de treinamento para cada cliente
10 clients_batched = dict()
11 for (client_name, data) in clients.items():
12     clients_batched[client_name] = batch_data(data)
13
14 # Processa e agrupa conjunto de teste
15 test_batched = tf.data.Dataset.from_tensor_slices((X_test, y_test))
16 .batch(len(y_test))
```

dados MNIST, o tamanho do parâmetro de entrada é $28 * 28 * 1 = 784$, pois cada imagem possui o formato 28x28 *pixels*, enquanto o número de classes é 10. Após, são definidos um otimizador, uma função de perda e métricas para avaliar o modelo posteriormente. É utilizado o otimizador SGD. A função de perda é a `categorical_crossentropy`.

Finalmente, a métrica utilizada para avaliação é a acurácia. A variável `comms_round` controla o número de agregações globais que são executadas durante o treinamento. Portanto, em vez de diminuir a taxa de aprendizado em relação ao número de épocas locais, é diminuído em relação ao número de agregações globais [Zhao et al., 2018]. Essa escolha de hiper-parâmetros depende da implementação.

Até esse ponto, foram apresentados os passos da implementação de uma aplicação de aprendizado profundo padrão, com exceção do particionamento de dados e da criação dos clientes. A partir de então, é implementado o algoritmo *FedAvg*, mostrado no Codigo Fonte 3.6, que é o ponto principal desta seção.

Os dados estão particionados horizontalmente [Yang et al., 2019]. Assim, é feita a média dos parâmetros dos componentes, que representa um peso de acordo a proporção de amostras de dados fornecidos por cada cliente participante. Essa parte é realizada de acordo com as Equações 2 e 1. Esse procedimento foi encapsulado em três funções:

1. *weight_scalling_factor* - Calcula a proporção dos dados de treinamento local de um cliente com os dados gerais de treinamento mantidos por todos os clientes. Primeiro, é obtido o tamanho do lote do cliente (linha 3), que é utilizado para calcular o número de pontos de dados (linha 5). Em seguida, é obtido o tamanho geral dos dados de treinamento globa (linha 9). Por fim, é calculado o fator de escala de um determinado cliente como uma fração (linha 11). Essa, certamente, não pode ser a abordagem em uma aplicação do mundo real, pois esse escalar deve ser calculado

pelo servidor. Nesse caso, cada cliente deverá indicar o número de pontos de dados com os quais treinou enquanto atualiza o servidor com novos parâmetros após cada etapa de treinamento local.

2. *scale_model_weights*: Dimensiona cada um dos pesos do modelo local com base no valor de seu fator de escala calculado em weight_scalling_factor.

3. *sum_scaled_weights*: Soma todos os pesos já escalados dos clientes.

São criadas duas funções, uma para avaliar o modelo global (`test_model`) e outra para verificar a perda dos modelos locais(`check_local_loss`), mostradas no Código Fonte 3.7. A lógica de treinamento tem dois laços principais, evidenciados no Código Fonte 3.8. O laços externo é para a iteração global, enquanto o interno é para iterar através do treinamento local de cada cliente. No entanto, há um terceiro laço implícito, que são as épocas locais, que é controlado pelo argumento *epochs* do método `model.fit`.

A simulação proposta começa com a construção do modelo global com 784 entradas e 10 classes. Em seguida, é feita a inicialização dos pesos do modelo global que consiste, neste exemplo, como uma sequência de zeros. Em seguida, é embaralhada a ordem do dicionário de clientes para garantir a aleatoriedade. A partir daí, começa a iteração por meio do treinamento local dos clientes. A quantidade de épocas locais é controlada pela variável `local_epochs`. Após o treinamento, os novos pesos são escalonados e

Código Fonte 3.5. Criação da classe que implementa a rede neural *multilayer perceptron*.

```
class MLP:
    @staticmethod
    def build(shape, classes):
        model = Sequential()
        model.add(Dense(200, input_shape=(shape,)))
        model.add(Activation("relu"))
        model.add(Dense(200))
        model.add(Activation("relu"))
        model.add(Dense(classes))
        model.add(Activation("softmax"))
        return model

lr = 0.01
comms_round = 100
loss = 'categorical_crossentropy'
metrics = ['accuracy']
optimizer = SGD(lr=lr, decay=lr / comms_round, momentum=0.9)
```

Código Fonte 3.6. Funções de implementação do algoritmo de média federada (*Federated Averaging*).

```
def weight_scalling_factor(clients_trn_data, client_name, participants):
    # calcula o tamanho do batch
    bs = list(clients_trn_data[client_name])[0][0].shape[0]
    # primeiro calcula o total de  dados de treinamento entre clientes
    global_count = sum([tf.data.experimental.cardinality
    (clients_trn_data[client_name]).numpy()
    for client_name in participants]) * bs
    # obtem o número total de pontos de dados mantidos por um cliente
    local_count = tf.data.experimental.cardinality
    (clients_trn_data[client_name]).numpy() * bs
    return local_count / global_count


def scale_model_weights(weight, scalar):
    '''Escala os pesos do modelo'''
    weight_final = []
    steps = len(weight)
    for i in range(steps):
        weight_final.append(scalar * weight[i])
    return weight_final


def sum_scaled_weights(scaled_weight_list):
    avg_grad = list()
    # obtem a média dos gradientes dos cliente
    for grad_list_tuple in zip(*scaled_weight_list):
        layer_mean = tf.math.reduce_sum(grad_list_tuple, axis=0)
        avg_grad.append(layer_mean)
    return avg_grad
```

anexados à lista `scaled_local_weight_list`. Essa etapa representa o treinamento local dos participantes. Voltando ao laço externo, é feita a soma de todos os pesos locais treinados e escalados e é realizada atualização do modelo global. Isso encerra uma época de treinamento global.

Os resultados mostram que, após 100 iterações de comunicação, o modelo global obtém 96,5% de acurácia. Existem diversos arcabouços de aprendizado federado que implementam o algoritmo FedAvg. O TensorFlow possui o módulo *TensorFlow Federated* (TFF) [12], que além de implementar o FedAvg, permite que o usuário crie seu próprio algoritmo customizado. Outro arcabouço que implementa o aprendizado federado é o PyTorch [13]. Existe também um arcabouço de conjunto de dados chamado LEAF, que contém conjuntos de dados particionados para aprendizado federado, facilitando assim a

[12] Disponível em https://www.tensorflow.org/federated. Acessado em 26/09/2020

[13] Disponível em https://pytorch.org/mobile/home/. Acessado em 26/09/2020

Código Fonte 3.7. Funções para a medição da acurácia e perda do modelo e verificação local da perda.

```
def test_model(X_test, Y_test, model, comm_round):
    cce = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
    # logits = model.predict(X_test, batch_size=100)
    logits = model.predict(X_test)
    loss = cce(Y_test, logits)
    acc = accuracy_score(tf.argmax(logits, axis=1),
    tf.argmax(Y_test, axis=1))
    print('Rodada de Comunicação: {} | global_acc: {:.3\%}
    | global_loss: {}'.format(comm_round, acc, loss))
    return acc, loss


def check_local_loss(client, model):
    # Verificar perda local
    cce_l = tf.keras.losses.CategoricalCrossentropy
    (from_logits=True)
    client_x = np.array([i[0] for i in clients[client]])
    client_y = np.array([i[1] for i in clients[client]])
    logits_l = model.predict(client_x)
    loss_l = cce_l(client_y, logits_l)
    acc_l = accuracy_score(tf.argmax(logits_l, axis=1),
                    \ tf.argmax(client_y, axis=1))
    print('Local accuracy: {}. Local loss: {}'.format(acc_l, loss_l))
    return acc_l, loss_l
```

avaliação de modelos globais.

3.8. Considerações Finais e Perspectivas Futuras

O aprendizado federado consiste em uma área de pesquisa que está em ascendência em função do aumento da capacidade de processamento de dispositivos móveis e da necessidade de se garantir a privacidade de dados pessoais de usuários. A técnica de aprendizado federado se baseia em desenvolver um modelo de aprendizagem colaborativa em que participantes, muitas vezes nós móveis, executam parte da tarefa de aprendizado de maneira local e contribuem para um modelo global. Contudo, o desenvolvimento de modelos de aprendizado colaborativos apresentam diversos desafios, entre eles a heterogeneidade de dispositivos participantes e a heterogeneidade estatística do dados. Esse trabalho analisou as principais arquiteturas de referência de aprendizado federado e destacou as dificuldades que essas arquiteturas encontram ao tratar dados e dispositivos heterogêneos. Quanto a heterogeneidade de dispositivos, o trabalho destacou o compromisso entre arquiteturas síncronas, limitadas pelo dispositivo com menor capacidade sujeito a ser retardatário, e arquiteturas assíncronas, que permitem a entrega de resultadas mesmo após a rodada ao custo de uma menor participação no modelo global. Quanto a hetero-

Código Fonte 3.8. Execução da aplicação de aprendizado federado implementada.

```
smlp_global = MLP()
global_model = smlp_global.build(784, 10)

for comm_round in range(comms_round):

    # obtem os pesos do modelo global
    #servirá como os pesos iniciais para todos os modelos locais
    global_weights = global_model.get_weights()

    # Cria a lista para coletar os pesos do modelo local após o
    # escalonamento
    scaled_local_weight_list = list()
    client_names = list(clients_batched.keys())
    random.shuffle(client_names)
    client_select = client_names[0:55]

    for client in client_select:
        smlp_local = MLP()
        local_model = smlp_local.build(784, 10)
        local_model.compile(loss=loss, optimizer=optimizer,
        metrics=metrics)

        # Definir o peso do modelo local para o peso do modelo global
        local_model.set_weights(global_weights)
        local_model.fit(clients_batched[client], epochs=1, verbose=0)

        # Dimensiona os pesos do modelo e adicionar à lista
        scaling_factor = weight_scalling_factor
        (clients_batched, client, client_select)
        scaled_weights = scale_model_weights(local_model.get_weights(),
                \ scaling_factor)
        scaled_local_weight_list.append(scaled_weights)

        acc_l, loss_l = check_local_loss(client, local_model)

        K.clear_session()

    # Obtém a média de todo o modelo local, simplesmente
    # pegamos a soma dos pesos escalados
    average_weights = sum_scaled_weights(scaled_local_weight_list)

    # Atualização do modelo global
    global_model.set_weights(average_weights)
```

geneidade de dados, o trabalho destacou o impacto de dados que não são independentes e não são identicamente distribuídos entre clientes. Dados heterogêneos tendem a criar enviesamento nos modelos de aprendizado global.

O trabalho abordou ainda os principais ataques ao aprendizado federado. Ataques aos conjuntos de dados locais tendem a ter influência moderada sobre o modelo global agregado. Contudo, ataques Sybil podem facilmente comprometer o modelo global. Ademais, ataques ao modelo em si, como o envio de parâmetros locais desviados, tendem a ter forte influência sobre o modelo global agregado. Esse minicurso destacou as soluções propostas para os ataques elencados. As principais soluções se baseiam na criptografia homomórfica e na computação multiparte segura. Paralelamente, também foi elicitado o ataque de *free riding*, no qual o atacante egoísta se aproveita do modelo global agregado, porém pouco contribui com recursos locais para o cálculo do modelo. As soluções prevalentes na literatura para este ataque consistem em criar incentivos, como a participação dos clientes em cadeias de blocos, e modelos de reputação. O trabalho elencou ainda diversas aplicações em redes móveis e redes tradicionais que se utilizam do aprendizado federado, demonstrando que, embora recente, já há diversas aplicações que o utilizam. Por fim, foi demonstrada uma simulação de aprendizado federado desenvolvida em Python que se baseia no algoritmo da média federada (*Federated Averaging* – FedAvg), um dos principais algoritmos para o aprendizado federado.

Considerando as principais propostas voltadas para a pesquisa em aprendizado federado, é possível constatar que ainda há diversos desafios a serem superados para que as técnicas de aprendizado federado atinjam a maturidade tecnológica. No entanto, há grande potencial no aprendizado federado para a proteção da privacidade de dados pessoais em ambientes sensíveis, como aplicações móveis ou na federação para extração de conhecimento entre empresas distintas. O aprendizado federado está alinhado com as necessidades atuais de atendimento às leis de proteção de dados pessoais, na medida em que organizações que compartilham dados de usuários para extraírem conhecimentos tendem a ter suas ações mais restritas. Contudo, ao aplicar o aprendizado federado é possível compartilhar modelos de aprendizado sem expor os dados pessoais de usuários e, ainda assim, obter a extração de conhecimento com alta acurácia e baixa perda.

Referências

[Andreoni Lopez et al., 2019] Andreoni Lopez, M., Mattos, D. M., Duarte, O. C. M. e Pujolle, G. (2019). Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data. *Concurrency and Computation: Practice and Experience*, 31(20):e5344.

[Asuncion e Newman, 2007] Asuncion, A. e Newman, D. (2007). Uci machine learning repository.

[Bagdasaryan et al., 2020] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. e Shmatikov, V. (2020). How to backdoor federated learning. Em *Proceedings of Machine Learning Research*, volume 108, p. 2938–2948, Online. PMLR.

[Baldi, 2011] Baldi, P. (2011). Autoencoders, unsupervised learning and deep architectures. Em *Proceedings of the 2011 International Conference on Unsupervised and*

Transfer Learning Workshop - Volume 27, UTLW'11, p. 37–50. JMLR.org.

[Bhagoji et al., 2019] Bhagoji, A. N., Chakraborty, S., Mittal, P. e Calo, S. (2019). Analyzing federated learning through an adversarial lens. Em *International Conference on Machine Learning*, p. 634–643.

[Bogetoft et al., 2009] Bogetoft, P., Christensen, D. L. e Damg, I. (2009). Multiparty Computation Goes Live. *Review Literature And Arts Of The Americas*, p. 1–13.

[Bonawitz et al., 2019] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H. B. et al. (2019). Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

[Bonawitz et al., 2017] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A. e Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. Em *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, p. 1175–1191, New York, NY, USA. Association for Computing Machinery.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Em *Proceedings of COMPSTAT'2010*, p. 177–186, Heidelberg. Physica-Verlag HD.

[Boutaba et al., 2018] Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F. e Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16.

[Bradley et al., 2000] Bradley, P. S., Bennett, K. P. e Demiriz, A. (2000). Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0.

[Brasil, 2018] Brasil (2018). Lei nº 13.709, de 14 de agosto de 2018. Institui a Lei Geral de Proteção de Dados Pessoais (LGPD).

[Brendan McMahan et al., 2017] Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S. e Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. Em *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, volume 54.

[Brisimi et al., 2018] Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C. e Shi, W. (2018). Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, 112:59 – 67.

[Caldas et al., 2018] Caldas, S., Konečny, J., McMahan, H. B. e Talwalkar, A. (2018). Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[Chen et al., 2020] Chen, M., Semiari, O., Saad, W., Liu, X. e Yin, C. (2020). Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks. *IEEE Transactions on Wireless Communications*, 19(1):177–191.

[Chiu et al., 2020] Chiu, T., Shih, Y., Pang, A., Wang, C., Weng, W. e Chou, C. (2020). Semi-supervised distributed learning with non-iid data for aiot service platform. *IEEE Internet of Things Journal*, p. 1–1.

[Chung et al., 2010] Chung, J., Yoon, H.-J. e Gardner, H. J. (2010). Analysis of break in presence during game play using a linear mixed model. *ETRI journal*, 32(5):687–694.

[Corinzia e Buhmann, 2019] Corinzia, L. e Buhmann, J. M. (2019). Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*.

[Cup, 1999] Cup, K. (1999). Available on: http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html.

[De Haan e Ferreira, 2007] De Haan, L. e Ferreira, A. (2007). *Extreme value theory: an introduction*. Springer Science & Business Media.

[de Oliveira et al., 2020a] de Oliveira, N. R., Lúcio Reis, H. A., Fernandes, N. C., Carlos Bastos, A. M., Dianne Medeiros, S. V. e Diogo Mattos, M. F. (2020a). Natural language processing characterization of recurring calls in public security services. Em *2020 International Conference on Computing, Networking and Communications (ICNC)*, p. 1009–1013.

[de Oliveira et al., 2020b] de Oliveira, N. R., Medeiros, D. S. V. e Mattos, D. M. F. (2020b). A sensitive stylistic approach to identify fake news on social networking. *IEEE Signal Processing Letters*, 27:1250–1254.

[Dean et al., 2012] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K. et al. (2012). Large scale distributed deep networks. Em *Advances in neural information processing systems*, p. 1223–1231.

[Dennis e Owen, 2015] Dennis, R. e Owen, G. (2015). Rep on the block: A next generation reputation system based on the blockchain. Em *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, p. 131–138.

[Du et al., 2004] Du, W., Han, Y. S. e Chen, S. (2004). Privacy-preserving multivariate statistical analysis: Linear regression and classification. Em *Proceedings of the 2004 SIAM international conference on data mining*, p. 222–233. SIAM.

[Du e Zhan, 2002] Du, W. e Zhan, Z. (2002). Building decision tree classifier on private data. *Syracuse University - Electrical Engineering and Computer Science*.

[Eichner et al., 2019] Eichner, H., Koren, T., McMahan, H. B., Srebro, N. e Talwar, K. (2019). Semi-cyclic stochastic gradient descent. *arXiv preprint arXiv:1904.10120*.

[Feng et al., 2019] Feng, S., Niyato, D., Wang, P., Kim, D. I. e Liang, Y. (2019). Joint service pricing and cooperative relay communication for federated learning. Em *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, p. 815–820.

[Fredrikson et al., 2015] Fredrikson, M., Jha, S. e Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. Em *CCS '15*, p. 1322–1333, New York, NY, USA. Association for Computing Machinery.

[Fredrikson et al., 2014] Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D. e Ristenpart, T. (2014). Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. Em *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, p. 17–32.

[Fung et al., 2018] Fung, C., Yoon, C. J. e Beschastnikh, I. (2018). Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*.

[Goldreich et al., 2019] Goldreich, O., Micali, S. e Wigderson, A. (2019). *How to Play Any Mental Game, or a Completeness Theorem for Protocols with Honest Majority*, p. 307–328. Association for Computing Machinery, New York, NY, USA.

[Gupta e Jha, 2015] Gupta, A. e Jha, R. K. (2015). A survey of 5g network: Architecture and emerging technologies. *IEEE Access*, 3:1206–1232.

[Han et al., 2015] Han, S., Mao, H. e Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

[Hard et al., 2018] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C. e Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

[Hardy et al., 2017] Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G. e Thorne, B. (2017). Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*.

[He et al., 2016] He, K., Zhang, X., Ren, S. e Sun, J. (2016). Deep residual learning for image recognition. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[He et al., 2018] He, Y., Zhao, N. e Yin, H. (2018). Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 67(1):44–55.

[Ho et al., 2013] Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J. K., Gibbons, P. B., Gibson, G. A., Ganger, G. e Xing, E. P. (2013). More effective distributed ml via a stale synchronous parallel parameter server. Em *Advances in Neural Information Processing Systems 26*, p. 1223–1231. Curran Associates, Inc.

[Huang et al., 2020] Huang, L., Yin, Y., Fu, Z., Zhang, S., Deng, H. e Liu, D. (2020). Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos one*, 15(4):e0230706.

[Kang et al., 2019a] Kang, J., Xiong, Z., Niyato, D., Xie, S. e Zhang, J. (2019a). Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714.

[Kang et al., 2019b] Kang, J., Xiong, Z., Niyato, D., Yu, H., Liang, Y. e Kim, D. I. (2019b). Incentive design for efficient federated learning in mobile networks: A contract theory approach. Em *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, p. 1–5.

[Karr et al., 2009] Karr, A. F., Lin, X., Sanil, A. P. e Reiter, J. P. (2009). Privacy-preserving analysis of vertically partitioned data using secure matrix products. *Journal of Official Statistics*, 25(1):125.

[Kim et al., 2018] Kim, H., Park, J., Bennis, M. e Kim, S.-L. (2018). On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*.

[Konečnỳ et al., 2016] Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T. e Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

[Kwon et al., 2017] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I. e Kim, K. J. (2017). A survey of deep learning-based network anomaly detection. *Cluster Computing*.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y. e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li et al., 2017] Li, P., Li, J., Huang, Z., Li, T., Gao, C.-Z., Yiu, S.-M. e Chen, K. (2017). Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, 74:76 – 85.

[Li et al., 2020] Li, T., Sahu, A. K., Talwalkar, A. e Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

[Li et al., 2018] Li, W., Logenthiran, T., Phan, V. e Woo, W. L. (2018). Implemented iot-based self-learning home management system (shms) for singapore. *IEEE Internet of Things Journal*, 5(3):2212–2219.

[Li et al., 2019] Li, W., Milletarì, F., Xu, D., Rieke, N., Hancox, J., Zhu, W., Baust, M., Cheng, Y., Ourselin, S., Cardoso, M. J. e Feng, A. (2019). Privacy-preserving federated brain tumour segmentation. Em *Machine Learning in Medical Imaging*, p. 133–141, Cham. Springer International Publishing.

[Liang e Chawathe, 2004] Liang, G. e Chawathe, S. S. (2004). Privacy-preserving inter-database operations. Em *International Conference on Intelligence and Security Informatics*, p. 66–82. Springer.

[Lim et al., 2020] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y. C., Yang, Q., Niyato, D. e Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*.

[Liu et al., 2019] Liu, Y., Kang, Y., Zhang, X., Li, L., Cheng, Y., Chen, T., Hong, M. e Yang, Q. (2019). A communication efficient vertical federated learning framework. *arXiv preprint arXiv:1912.11187*.

[Lobato et al., 2018] Lobato, A. G. P., Lopez, M. A., Sanz, I. J., Cardenas, A. A., Duarte, O. C. M. B. e Pujolle, G. (2018). An adaptive real-time architecture for zero-day threat detection. Em *2018 IEEE International Conference on Communications (ICC)*, p. 1–6.

[Long et al., 2015] Long, M., Cao, Y., Wang, J. e Jordan, M. (2015). Learning transferable features with deep adaptation networks. Em *International conference on machine learning*, p. 97–105. PMLR.

[Mattos et al., 2019] Mattos, D. M. F., Velloso, P. B. e Duarte, O. C. M. B. (2019). An agile and effective network function virtualization infrastructure for the internet of things. *Journal of Internet Services and Applications*, 10(1):6.

[Medeiros et al., 2019] Medeiros, D. S. V., Cunha Neto, H. N., Andreoni Lopez, M., Magalhães, L. C. S., Silva, E. F., Vieira, A. B., Fernandes, N. C. e Mattos, D. M. F. (2019). Análise de dados em redes sem fio de grande porte: Processamento em fluxo em tempo real, tendências e desafios. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2019:142–195.

[Micali et al., 1987] Micali, S., Goldreich, O. e Wigderson, A. (1987). How to play any mental game. Em *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*, p. 218–229.

[Mikolov et al., 2011] Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J. e Khudanpur, S. (2011). Extensions of recurrent neural network language model. Em *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, p. 5528–5531. IEEE.

[Neely, 2010] Neely, M. J. (2010). Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211.

[Nguyen et al., 2018] Nguyen, K. K., Hoang, D. T., Niyato, D., Wang, P., Nguyen, D. e Dutkiewicz, E. (2018). Cyberattack detection in mobile cloud computing: A deep learning approach. Em *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, p. 1–6.

[Nguyen et al., 2019] Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N. e Sadeghi, A. (2019). DÏot: A federated self-learning anomaly detection system for iot. Em *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, p. 756–767.

[Nishio e Yonetani, 2019] Nishio, T. e Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. Em *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, p. 1–7.

[Nock et al., 2018] Nock, R., Hardy, S., Henecka, W., Ivey-Law, H., Patrini, G., Smith, G. e Thorne, B. (2018). Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*.

[Osborne et al., 2004] Osborne, M. J. et al. (2004). *An introduction to game theory*, volume 3. Oxford university press New York.

[Pan e Yang, 2010] Pan, S. J. e Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

[Parlamento Europeu e Conselho da União Européia, 2016] Parlamento Europeu e Conselho da União Européia (2016). Regulamento (ue) 2016/679. `https://eur-lex.europa.eu/legal-content/PT/TXT/PDF/?uri=CELEX:32016R0679&from=PT`.

[Phong et al., 2018] Phong, L. T., Aono, Y., Hayashi, T., Wang, L. e Moriai, S. (2018). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345.

[Preuveneers et al., 2018] Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W. e Ilie-Zudor, E. (2018). Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, 8(12):2663.

[Reis et al., 2020] Reis, L. H. A., Murillo Piedrahita, A., Rueda, S., Fernandes, N. C., Medeiros, D. S. V., de Amorim, M. D. e Mattos, D. M. F. (2020). Unsupervised and incremental learning orchestration for cyber-physical security. *Transactions on Emerging Telecommunications Technologies*, 31(7):e4011.

[Ren et al., 2019] Ren, J., Wang, H., Hou, T., Zheng, S. e Tang, C. (2019). Federated learning-based computation offloading optimization in edge computing-supported internet of things. *IEEE Access*, 7:69194–69201.

[Sadeghi et al., 2018] Sadeghi, A., Sheikholeslami, F. e Giannakis, G. B. (2018). Optimal and scalable caching for 5g using reinforcement learning of space-time popularities. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):180–190.

[Samarakoon et al., 2018] Samarakoon, S., Bennis, M., Saad, W. e Debbah, M. (2018). Federated learning for ultra-reliable low-latency v2v communications. Em *2018 IEEE Global Communications Conference (GLOBECOM)*, p. 1–7.

[Saputra et al., 2019] Saputra, Y. M., Hoang, D. T., Nguyen, D. N., Dutkiewicz, E., Mueck, M. D. e Srikanteswara, S. (2019). Energy demand prediction with federated learning for electric vehicle networks. Em *2019 IEEE Global Communications Conference (GLOBECOM)*, p. 1–6.

[Sarikaya e Ercetin, 2020] Sarikaya, Y. e Ercetin, O. (2020). Motivating workers in federated learning: A stackelberg game perspective. *IEEE Networking Letters*, 2(1):23–27.

[Scannapieco et al., 2007] Scannapieco, M., Figotin, I., Bertino, E. e Elmagarmid, A. K. (2007). Privacy preserving schema and data matching. Em *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, p. 653–664.

[Sheth e Larson, 1990] Sheth, A. P. e Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236.

[Smith et al., 2017] Smith, V., Chiang, C.-K., Sanjabi, M. e Talwalkar, A. S. (2017). Federated multi-task learning. Em *Advances in Neural Information Processing Systems*, p. 4424–4434.

[Sprague et al., 2019] Sprague, M. R., Jalalirad, A., Scavuzzo, M., Capota, C., Neun, M., Do, L. e Kopp, M. (2019). Asynchronous federated learning for geospatial applications. Em *ECML PKDD 2018 Workshops*, p. 21–28. Springer International Publishing.

[Strom, 2015] Strom, N. (2015). Scalable distributed dnn training using commodity gpu cloud computing. Em *Sixteenth Annual Conference of the International Speech Communication Association*.

[Suresh et al., 2017] Suresh, A. T., Felix, X. Y., Kumar, S. e McMahan, H. B. (2017). Distributed mean estimation with limited communication. Em *International Conference on Machine Learning*, p. 3329–3337.

[Tao e Li, 2018] Tao, Z. e Li, Q. (2018). esgd: Communication efficient distributed deep learning on the edge. Em {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*.

[Tesauro, 2007] Tesauro, G. (2007). Reinforcement learning in autonomic computing: A manifesto and case studies. *IEEE Internet Computing*, 11(1):22–30.

[Triastcyn e Faltings, 2020] Triastcyn, A. e Faltings, B. (2020). Federated generative privacy. *IEEE Intelligent Systems*, 35(4):50–57.

[Trigeorgis et al., 2016] Trigeorgis, G., Ringeval, F., Brueckner, R., Marchi, E., Nicolaou, M. A., Schuller, B. e Zafeiriou, S. (2016). Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. Em *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 5200–5204.

[Vaidya e Clifton, 2002] Vaidya, J. e Clifton, C. (2002). Privacy preserving association rule mining in vertically partitioned data. Em *KDD '02*, p. 639–644, New York, NY, USA. Association for Computing Machinery.

[Vaidya e Clifton, 2003] Vaidya, J. e Clifton, C. (2003). Privacy-preserving k-means clustering over vertically partitioned data. Em *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, p. 206–215, New York, NY, USA. Association for Computing Machinery.

[Vaidya e Clifton, 2004] Vaidya, J. e Clifton, C. (2004). Privacy preserving naive bayes classifier for vertically partitioned data. Em *Proceedings of the 2004 SIAM international conference on data mining*, p. 522–526. SIAM.

[Vaidya et al., 2008] Vaidya, J., Clifton, C., Kantarcioglu, M. e Patterson, A. S. (2008). Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data*, 2(3).

[Verma et al., 2018] Verma, D., Julier, S. e Cirincione, G. (2018). Federated ai for building ai solutions across multiple agencies. *arXiv preprint arXiv:1809.10036*.

[Vilalta e Drissi, 2002] Vilalta, R. e Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95.

[Wan et al., 2007] Wan, L., Ng, W. K., Han, S. e Lee, V. C. (2007). Privacy-preservation for gradient descent methods. Em *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 775–783.

[Wang et al., 2019] Wang, L., Wang, W. e Li, B. (2019). Cmfl: Mitigating communication overhead for federated learning. Em *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, p. 954–964.

[Wang et al., 2019] Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T. e Chan, K. (2019). Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221.

[Wang et al., 2020] Wang, X., Han, Y., Leung, V. C. M., Niyato, D., Yan, X. e Chen, X. (2020). Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(2):869–904.

[Wang et al., 2019a] Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X. e Chen, M. (2019a). In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165.

[Wang et al., 2019b] Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X. e Chen, M. (2019b). In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165.

[Weng et al., 2019] Weng, J., Weng, J., Zhang, J., Li, M., Zhang, Y. e Luo, W. (2019). Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, p. 1–1.

[Wenliang Du e Atallah, 2001] Wenliang Du e Atallah, M. J. (2001). Privacy-preserving cooperative statistical analysis. Em *Seventeenth Annual Computer Security Applications Conference*, p. 102–110.

[Xie et al., 2019] Xie, C., Koyejo, S. e Gupta, I. (2019). Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

[Yang et al., 2019] Yang, Q., Liu, Y., Chen, T. e Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19.

[Yao, 1982] Yao, A. C. (1982). Protocols for secure computations. Em *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, p. 160–164.

[Yao et al., 2018] Yao, X., Huang, C. e Sun, L. (2018). Two-stream federated learning: Reduce the communication costs. Em *2018 IEEE Visual Communications and Image Processing (VCIP)*, p. 1–4.

[You e Yang, 2014] You, P. e Yang, Z. (2014). Efficient optimal scheduling of charging station with multiple electric vehicles via v2v. Em *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, p. 716–721.

[Yu et al., 2006a] Yu, H., Jiang, X. e Vaidya, J. (2006a). Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. Em *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, p. 603–610, New York, NY, USA. Association for Computing Machinery.

[Yu et al., 2006b] Yu, H., Vaidya, J. e Jiang, X. (2006b). Privacy-preserving svm classification on vertically partitioned data. Em *Advances in Knowledge Discovery and Data Mining*, p. 647–656, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Zhan et al., 2020] Zhan, Y., Li, P., Qu, Z., Zeng, D. e Guo, S. (2020). A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal*, 7(7):6360–6368.

[Zhang et al., 2019] Zhang, J., Chen, B., Yu, S. e Deng, H. (2019). Pefl: A privacy-enhanced federated learning scheme for big data analytics. Em *2019 IEEE Global Communications Conference (GLOBECOM)*, p. 1–6.

[Zhang e Xiao, 2018] Zhang, Y. e Xiao, L. (2018). *Communication-Efficient Distributed Optimization of Self-concordant Empirical Loss*, p. 289–341. Springer International Publishing, Cham.

[Zhao et al., 2018] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D. e Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.