FedSA: Arrefecimento Simulado Federado para a Aceleração da Detecção de Intrusão em Ambientes Colaborativos

Helio N. C. Neto, Diogo M. F. Mattos, Natalia C. Fernandes

[1]MídiaCom - PPGEET/TET/UFF
Universidade Federal Fluminense - UFF

Resumo. *Sistemas de detecção de intrusão (IDS) baseados em aprendizado federado treinam um modelo global com a colaboração de participantes que treinam modelos locais de aprendizado de máquina. Desafios de otimização implícitos ao aprendizado federado são relacionados à heterogeneidade e ao desbalanceamento da distribuição dos dados entre participantes. Este artigo propõe a meta-heurística Arrefecimento Simulado Federado (Federated Simulated Annealing — FedSA), que visa selecionar adaptativamente os hiperparâmetros e a seleção de participantes para a agregação de rodadas globais no aprendizado federado. A otimização da seleção de participantes e a definição adaptava de hiperparâmetros do Arrefecimento Simulado Federado reduzem o número de iterações até obter a convergência do modelo, promovendo, assim, uma disseminação rápida de novos conhecimentos extraídos dos modelos locais. A avaliação da proposta em um cenário de sistema de detecção de intrusão federado mostra que o modelo global usando do FedSA converge em apenas 10 iterações globais, enquanto o algoritmo tradicional necessita de 20 iterações para que ambos alcancem acurácia de 99,8% na detecção de ataques.*

Abstract. *Federated learning-based intrusion detection systems (IDS) train a global model with participants collaboration, who train local models based on machine learning. Optimization challenges, implicit in federated learning scenarios, are related to heterogeneity and imbalance in data distribution among participants. This paper proposes the Federated Simulated Annealing (FedSA) aggregation algorithm to select the hyperparameters and the participant's selection for each global aggregation rounds in federated learning. The adapted Federated Simulated Annealing participant's selection and hyperparameters selection reduces the iteration number and speeds up convergence. Thus, promoting rapid dissemination of new learnings extracted from the local models. The proposed assessment in a federated intrusion detection system scenario shows that the global model using FedSA converges in just ten global iterations. In comparison, the traditional algorithm requires twenty iterations for both to achieve 99.8% accuracy in Detecting Attacks.*

1. Introdução

A Internet das Coisas fomenta a conexão de cada vez mais dispositivos às redes de computadores. Cerca de 30 bilhões de dispositivos devem estar conectados à Internet até o final de 2021[1]. Como consequência do aumento do número

Este trabalho foi realizado com recursos da CNPq, CAPES, FAPERJ e RNP.

[1]Disponível em: https://www.paymentscardsandmobile.com/global-iot-growth-surges/ Acessado em 13/01/2021

de dispositivos e das limitações de hardware que dificultam a adoção de medidas avançadas de segurança, os ataques de rede também crescem, muitas vezes ocultados pelo grande volume de tráfego gerados pela diversidade de dispositivos da Internet das Coisas [Viegas et al., 2019]. Dessa forma, abordagens de análise de tráfego de rede baseadas em modelos de aprendizado de máquina centralizados são propostas com alta acurácia na identificação de ameaças na rede ao custo do treinamento com uma base de dados rotulados ampla [Andreoni Lopez et al., 2019, Liu e Lang, 2019, de Souza et al., 2020]. Como contraponto ao aprendizado centralizado, o aprendizado federado [Brendan McMahan et al., 2017] possibilita que diversos participantes treinem um modelo de aprendizado de máquina global de forma distribuída e sem compartilhamento de dados. Assim, sistemas de detecção de intrusão (*Intrusion Detection System* — IDS) baseados em aprendizado federado treinam colaborativamente um modelo global entre diversos participantes sem que cada um exponha os seus dados.

Uma das dificuldades para projetar IDSs federados é que os modelos globais são treinados com dados dos participantes altamente não independentes e não identicamente distribuídos (*Non Independent and Identically Distributed* — Non-IID) [Lim et al., 2020, Cunha Neto et al., 2020], devido à ausência de uma coordenação global que tenha acesso aos dados dos participantes. A dependência dos dados e distribuição heterogênea acarretam desafios para a convergência ótima do modelo treinado. Contudo, no cenário de um IDS federado, a convergência deve ocorrer no menor tempo possível, para que a disseminação de novos padrões de ataques ocorra rapidamente.

Este artigo propõe a meta-heurística para otimização de seleção de usuários e hiperparâmetros chamada Arrefecimento Simulado Federado (*Federated Simulated Annealing* —FedSA). O FedSA é uma variante da meta-heurística Arrefecimento Simulado (*Simulated Annealing* — SA) aplicada ao cenário do aprendizado federado. A meta-heurística proposta visa selecionar adaptativamente, a cada rodada de agregação, hiperparâmetros relacionados à convergência do modelo global, acelerando o compartilhamento das informações mais relevantes. Os hiperparâmetros otimizados são a taxa de aprendizado, o número de atualizações locais e os participantes selecionados para a rodada de agregação. A proposta dispensa a etapa longa de ajuste fino dos hiperparâmetros. A proposta alcança a convergência do modelo global em um número de rodadas de agregação 50% menor que o algoritmo de agregação de média federada (Federated Averaging — FedAVG) tradicional. A proposta atinge 99,8% de acurácia em 10 iterações, sendo a acurácia superior às alcançadas pela FedAVG em todos os cenários testados.

O algoritmo FedAVG seleciona um subconjunto aleatoriamente de participantes para participar da rodada de agregação [Brendan McMahan et al., 2017]. No FedAVG, a taxa de aprendizado pode ser fixa ou variar de acordo com uma taxa de decaimento, enquanto o número de atualizações locais é fixo durante todo o treinamento. Diferente do FedAVG, a proposta FedSA aprende novos valores a cada rodada de agregação. Variando a taxa de aprendizado, o modelo global evita mínimos locais sem reduzir a velocidade do treinamento. Com a escolha adaptativa das atualizações locais, o algoritmo aprende quando aumentar o processamento local nos participantes e procura participantes que contribuem com a convergência do modelo global. Trabalhos anteriores [Preuveneers et al., 2018, Nguyen et al., 2019] propõem o uso de aprendizado fede-

rado para detecção de intrusão utilizando o algoritmo de agregação tradicional FedAVG. Contudo, a proposta de Preuveneers *et al.* demanda alto número de rodadas de agregação e a acurácia obtida é de 97%.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. Os conceitos relativos ao aprendizado federado são apresentados na Seção 3. Na Seção 4, é apresentada a arquitetura e requisitos do sistema de detecção de intrusão federado. A Seção 5 descreve a meta-heurística proposta. O resultado das avaliações é apresentado na Seção 6 e a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

É uma tendência o desenvolvimento de sistemas de detecção de intrusão baseados em aprendizado de máquinas [Andreoni Lopez et al., 2019, Viegas et al., 2019, Liu e Lang, 2019, Guimarães et al., 2020, de Souza et al., 2020]. Contudo, com a quantidade e velocidade com que os ataques se disseminam pela Internet, aprender apenas com dados locais em uma rede monitorada é restritivo. Nesse sentido, alguns trabalhos propõem não apenas aprender com o tráfego monitorado pelo IDS local, mas também por uma colaboração de IDSs por meio de uma federação, estendendo o perímetro de aprendizado [Nguyen et al., 2019, Preuveneers et al., 2018].

Nguyen *et al.* propõem um modelo de detecção de ataques cibernéticos para uma rede de borda utilizando aprendizado federado com seleção aleatória dos participantes de cada rodada [Nguyen et al., 2019]. No trabalho, *gateways* IoT operam como participantes do aprendizado federado e um provedor de serviços de segurança IoT funciona como o servidor de agregação dos modelos treinados de forma colaborativa entre os participantes. Os autores avaliam a proposta em um ambiente real de casas inteligentes e detectaram com sucesso 95,6% dos ataques em aproximadamente 257 ms sem disparar alarmes falsos.

Preuveneers *et al.* propõem o uso da tecnologia de cadeia de blocos para o gerenciamento dos dados compartilhados pelos participantes do IDS compartilhado [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. Apesar das vantagens desse armazenamento, os autores identificaram que o uso da cadeia de blocos gera latência no treinamento.Além disso, o trabalho apresenta uma arquitetura frágil, pois depende que os nós da rede monitorada pelos participantes do treinamento federado enviem todos os seus fluxos para análise.

Outros trabalhos na literatura visam resolver os desafios de otimização do aprendizado federado [Corinzia e Buhmann, 2019, Smith et al., 2017]. Smith *et al.* propõem uma estrutura de otimização para o ambiente federado chamada MOCHA, que permite a personalização do aprendizado federado através do aprendizado de modelos separados, porém relacionados, para cada dispositivo [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. 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 funções-objetivos convexos [Smith et al., 2017].

Outra abordagem visa modelar uma topologia em estrela como uma rede bayesi-

ana 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 computacionalmente custoso realizar generalização em redes federadas grandes.

Wang *et al.* propõem um algoritmo para determinar a troca entre o número de atualizações locais e agregações globais [Wang et al., 2019]. Os autores analisam o limite de convergência do aprendizado federado baseados no gradiente descendente a partir de uma perspectiva teórica, na qual um novo limite de convergência é proposto. Esse limite de convergência incorpora a distribuição dos dados, que normalmente é desbalanceado, entre os participantes e um número arbitrário de atualizações locais entre duas agregações globais. Com isso, consegue-se determinar a frequência ideal de agregações globais, economizando recursos computacionais.

A proposta deste artigo apresenta uma meta-heurística federada para otimização dos hiperparâmetros de aprendizagem de um Sistema de Detecção de Intrusão com treinamento colaborativo. A otimização proposta acelera o processo de escolha de hiperparâmetros que forneçam convergência rápida e com alta acurácia.

3. O Aprendizado Federado

O aprendizado federado envolve o treinamento de um modelo estatístico global utilizando dados de dispositivos remotos. Durante o processo de treinamento, os participantes treinam um modelo local 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.

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. Em cada iteração de comunicação, um subconjunto S^t, $S^t \in N$ dos participantes é selecionado aleatoriamente para fornecer os parâmetros de seus modelos locais para o servidor. Realizar o treinamento com todos os participantes não é uma solução escalável, pois a quantidade de participantes é finita, mas não limitada, o que inviabiliza o treinamento. 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 [Cunha Neto et al., 2020].

O objetivo de cada modelo local é minimizar sua função de perda local $L(w^t)$. A função de perda varia de acordo com o problema, por exemplo, utiliza-se o erro médio quadrático (*Mean Squared Error* - MSE) para problemas de regressão e entropia cruzada para problemas de classificação (*Log Loss*). A função de perda global deve levar em consideração a perda de todos os participantes da rodada de agregação [Lim et al., 2020]. A Figura 1 mostra o processo de treinamento no aprendizado federado. O processo de treinamento pode diferir de acordo com a necessidade de cada cenário, porém o conceito base se mantém.

O algoritmo mais usual para agregação de modelos locais é a média federada

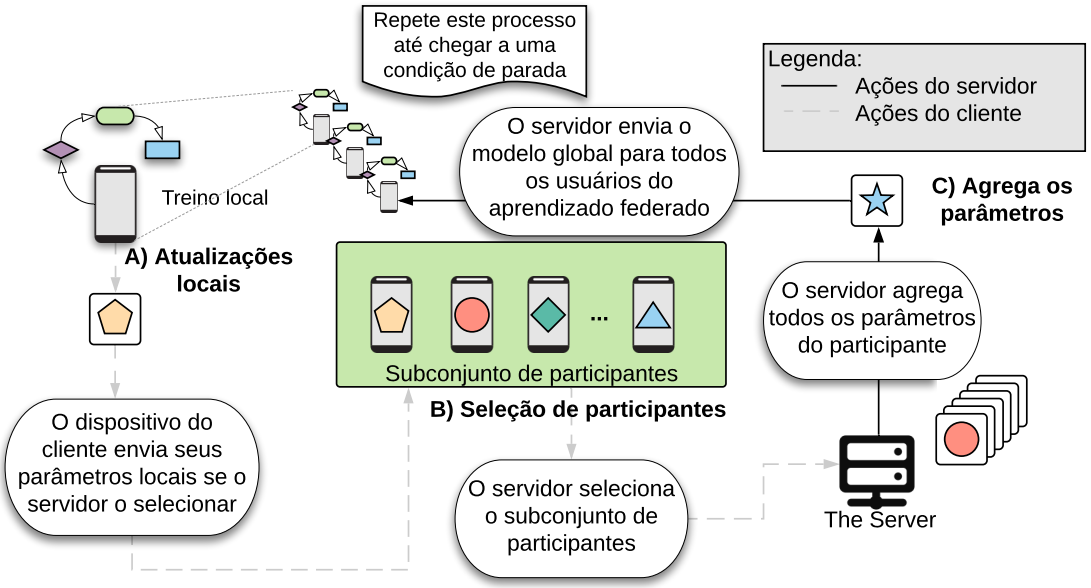


Figura 1. A arquitetura do aprendizado federado. No primeiro passo, o servidor envia o modelo inicial. Posteriormente, cada participante selecionado atualiza o modelo com seus dados locais e, por fim, o servidor agrega os parâmetros recebidos.

(*Federated Averaging* — FedAvg) [Brendan McMahan et al., 2017], um método de treinamento baseado em gradiente descendente estocástico (SGD). O FedAvg seleciona um subconjunto de participantes $S^t \in N$ aleatório. Esses participantes realizam atualizações locais em seus modelos utilizando seus dados locais. Os parâmetros dos modelos locais são enviados ao servidor agregador que realiza a agregação dos parâmetros por meio da média ponderada dos parâmetros. A média é ponderada pela razão entre o tamanho do conjunto de dados do participante em relação ao tamanho do conjunto de dados de todos os participantes envolvidos na iteração de agregação. Contudo, o FedAvg não possui garantias de convergência e pode divergir em cenários práticos quando os dados são heterogêneos [Cunha Neto et al., 2020].

Outra questão de grande relevância é a necessidade da otimização do algoritmo de agregação para garantir sua execução com esforço computacional e tempo limitados. Em aprendizado federado, o treinamento é feito de forma distribuída, mas a agregação é federada. A otimização distribuída assume que os dados são independentes e identicamente distribuídos (*Independent and Identically Distributed* — IID) [Cunha Neto et al., 2020]. Em contraste, na otimização federada [Brendan McMahan et al., 2017], os dados são não independentes e não identicamente distribuídos (*Non Independent and Identically Distributed* — Non-IID), o que significa que as amostras podem ser dependentes entre si e não possuem a mesma distribuição de probabilidade no conjunto de dados.

4. A Arquitetura do Sistema de Detecção de Intrusão Federado

O sistema de detecção de intrusão federado deve manter dois requisitos, i) rápida convergência e ii) privacidade dos dados. No cenário de detecção de intrusão, cada IDS local é participante do aprendizado federado. Os IDS participantes recebem todo o tráfego da rede monitorada, sem necessidade de instalação de agentes nos *hosts* monitorados, e utiliza as estatísticas de tráfego como vetor de características. Apenas alguns participantes

podem possuir dados de novos padrões de ataques. Os IDSs participantes selecionados enviam os parâmetros de seus treinamentos locais para o servidor agregador realizar a agregação global. A frequência de treinamento colaborativo deve ser ajustada pela entidade operadora do sistema federado.

Para realizar a previsão do tráfego de rede, utiliza-se redes neurais artificiais (RNA). As RNAs recebem como entrada vetores de características visando encontrar padrões que permitam estimar uma determinada saída, no caso considerado, a saída esperada é a classificação entre tráfego normal ou ataque. Os conjuntos de dados locais de cada participante são distribuídos de forma desbalanceada e algumas vezes enviesada. Esse enviesamento ocorre devido ao padrão de uso da rede monitorada pelo participante, o que pode gerar problemas de otimização [Silva et al., 2020]. Não há solução definitiva para os problemas de otimização e privacidade [Cunha Neto et al., 2020, Lim et al., 2020], que são os pontos importantes deste trabalho. Além de aprender novos ataques, o modelo global deve aprender os diferentes padrões de tráfego normal para evitar o bloqueio indevido de tráfego.

5. A Meta-heurística FedSA Proposta

No algoritmo tradicional de aprendizado federado, a cada iteração global, o servidor escolhe aleatoriamente um subconjunto $S^t \in N$ de participantes. Escolher participantes que dispõem dados que ajudam o modelo a interpretar os fluxos de rede normais e de ataque aceleram a convergência do modelo. A taxa de aprendizado, parâmetro do algoritmo de otimização gradiente descendente, assume valor fixo ou decrescente dependendo da abordagem adotada. O otimizador gradiente estocástico é responsável por otimizar os parâmetros de treinamento do modelo de aprendizado de máquina. No caso das redes neurais artificiais, o gradiente descendente estocástico é responsável por otimizar os pesos da rede em relação a uma função de perda [Reis et al., 2020]. A proposta FedSA é responsável por otimizar a seleção de participantes, a taxa de aprendizado e o número de atualizações locais.

A taxa de aprendizado é responsável pela velocidade de convergência do modelo. Se esse valor for muito pequeno, o algoritmo pode percorrer um longo caminho até a convergência e tende a levar a solução a um mínimo local, já que a perturbação nas soluções candidatas não são suficientes para gerar uma mudança que faça o algoritmo sair do mínimo local. No entanto, uma taxa de aprendizado alta pode aumentar a velocidade de aprendizagem, porém pode levar a flutuações em torno do mínimo global, sem que ocorra o refinamento da solução para o valor mínimo. O número de atualizações locais é um hiperparâmetro do aprendizado federado que indica quantas iterações os modelos locais dos participantes executam em seus conjuntos de dados locais.

Assim, é proposto uma meta-heurística baseada no Arrefecimento Simulado (*Simulated Annealing* - SA) para adaptar dinamicamente os hiperparâmetros de taxa de aprendizado, atualizações locais e seleção de participantes, aumentando a probabilidade de rápida convergência ao mínimo global. A meta-heurística proposta, FedSA, escolhe a cada iteração de agregação, o subconjunto de participantes S^t e adapta dinamicamente a taxa de aprendizagem η e o número de atualizações locais τ a cada interação.

O SA é uma meta-heurística de otimização para encontrar o mínimo global em uma função multivariável, com base nos princípios da mecânica estatística. A partir de

uma solução inicial, gerada aleatoriamente, o SA busca uma solução s ótima durante um certo número iterações. A cada iteração, um vizinho aleatório s' é gerado com base na melhor solução. Uma função de perda $f(s)$ é utilizada para avaliar o desempenho da solução. No caso do problema de aprendizado federado, é a função de perda global. As soluções que minimizam a função de perda são sempre aceitas. Caso contrário, a solução só é aceita dentro de uma certa probabilidade. Essa probabilidade depende do parâmetro de simulação da temperatura atual T e da quantidade de degradação de ΔE da função de perda. O ΔE representa a diferença entre o valor da função de perda da solução atual $f(s)$ e da solução vizinha gerada $f(s')$, logo, $\Delta E = f(s') - f(s)$. Portanto, a função de probabilidade de aceitação $P(\Delta E, T)$, é dada pela equação:

$$P(\Delta E, T) = \exp\left(-\frac{\Delta E}{T}\right). \tag{1}$$

De acordo com a Equação 1, quanto menor o valor da temperatura, menor a probabilidade de uma solução pior que a atual ser aceita. Isso ocorre devido ao valor negativo da razão de ΔE e T na função exponencial. $P(\Delta E, T)$ avalia a probabilidade de fazer a transição do estado atual s para um estado candidato s' pior. Conforme a execução da meta-heurística avança, a probabilidade de que uma solução que não minimiza a função de perda seja aceita diminui. Esta probabilidade segue a distribuição de Boltzmann [Van Laarhoven e Aarts, 1987]. O parâmetro α, onde $0 < \alpha < 1$, é uma constante da meta-heurística, responsável por reduzir gradativamente a temperatura simulada, reduzindo, assim, a probabilidade de soluções piores serem aceitas.

A meta-heurística FedSA proposta, baseada no Arrefecimento Simulado, visa procurar a melhor combinação de seleção de participantes, taxa de aprendizado e atualizações locais para cada rodada de agregação global. Para tanto, define-se o conjunto de participantes selecionados, a taxa de aprendizado e o número de atualização local através de tuplas contendo os limites do intervalo para cada hiperparâmetro. A tupla η possui o intervalo entre o menor e o maior valor que a taxa de aprendizado pode assumir durante o treinamento e a tupla τ representa o mesmo para o número de atualizações locais. A tupla μ contém três valores, o menor e o maior índice do conjunto de participantes e o número de participantes que deve ser selecionado a cada iteração. O índice é o identificador único entre os participantes e também indica a ordem que esse participante entrou no treinamento.

O cenário de otimização dos hiperparâmetros do aprendizado federado diverge dos problemas NP-difíceis tradicionais resolvidos com SA. Em problemas NP-difíceis tradicionais, *e.g.*, o problema do caixeiro viajante, a perda de uma determinada solução s é a mesma independente da iteração. Em contrapartida, no cenário de aprendizado federado, uma solução s na iteração i tem perda diferente da mesma solução na iteração $i + 1$. Uma solução s constitui-se de um tupla contendo o número de iterações locais, a taxa de aprendizado e os IDs dos participantes selecionados. Portanto, realizaram-se mudanças na meta-heurística para a otimização dos hiperparâmetros do aprendizado federado. Assim, adicionou-se uma nova etapa ao final de cada iteração, em que é realizada uma agregação global para avaliar a melhor solução. A avaliação da melhor solução permite adicionar aleatoriedade, caso ocorra a degradação da melhor solução, evitando mínimos locais, visto que em um mínimo local a melhor solução permanecerá. O laço interno responsável pela redução da temperatura também foi removido. Na variante pro-

Algoritmo 1: Meta-heurística Arrefecimento Simulado Federado

Entrada: η, τ, μ, $T_{inicial}$, $epocas$, α
Saída: $melhor\_solucão$

$T \leftarrow T_{inicial}$
$melhor\_solucão \leftarrow$ **INICIAR**(η, τ, μ)
$melhor\_perda \leftarrow$**AGREGAR**$(melhor\_solucão)$
para $i < epocas$ **faça**
  $solucão\_atual \leftarrow$ **GERA_SOLUÇÃO_VIZINHA**$(melhor\_solucão)$
  $perda\_atual \leftarrow$**AGREGAR**$(solucão\_atual)$
  $\Delta E \leftarrow perda\_atual - melhor\_perda$
  $p = \exp(-\frac{\Delta E}{T})$
  se $melhor\_perda < perda\_atual$ **então**
    $melhor\_solucão \leftarrow solucão\_atual$
    $melhor\_perda \leftarrow perda\_atual$
  fim
  senão se ***unif***$(0, 1) < p$ **então**
    $melhor\_solucão \leftarrow solucão\_atual$
    $melhor\_perda \leftarrow perda\_atual$
    $T \leftarrow$ **RESFRIAMENTO**(T, α)
  fim
  $teste\_melhor\_perda \leftarrow$ **AGREGAR**$(melhor\_solucão)$
  se $teste\_melhor\_perda < melhor\_perda$ **então**
    $melhor\_solucão \leftarrow$ **INICIAR**(η, τ, μ)
  fim
fim
retorna $melhor\_solucão$

posta, a temperatura é reduzida sempre que uma solução pior é selecionada, diminuindo, assim, a probabilidade da seleção de soluções piores. Logo, sempre que uma solução pior é aceita, a probabilidade $P(\Delta E, T)$ diminui. Reduzir a temperatura no fim do laço interno reduz a probabilidade da aceitação de soluções piores mesmo na ausência de aceitação de soluções piores. No treinamento de modelos de aprendizado de máquina, normalmente as primeiras iterações reduzem a função de perda, logo, sempre serão as melhores soluções. Da forma tradicional, no cenário de otimização de hiperparâmetros e seleção de participantes, a meta-heurística SA acaba se tornando gulosa.

Além da variante proposta, desenvolveu-se as funções `INICIAR` e `GERA_SOLUÇÃO_VIZINHA` relacionadas à modelagem do problema de otimização de hiperparâmetros. A função `INICIAR` é responsável por gerar uma solução aleatória seguindo uma distribuição uniforme, em que a função de densidade de probabilidade é $\frac{1}{b-a}$, sendo a e b os limites de menor e maior valor respectivamente. A função `AGREGAR` representa uma rodada de agregação global e retorna a perda do modelo global. Já a função `GERA_SOLUÇÃO_VIZINHA` é responsável por gerar uma solução vizinha da melhor solução atual. A função deve gerar soluções de vizinhança para valores discretos (seleção de participantes e atualizações locais) e valores contínuos (taxa de aprendizado). Para gerar um vizinho de um valor discreto, a função seleciona o valor subsequente. O

valor subsequente pode ser na direção positiva, simplesmente adicionando $+1$ ao valor atual, ou na direção negativa, reduzindo -1 ao valor atual. Caso o valor da melhor solução seja um valor limite (valor máximo e mínimo que esse hiperparâmetro pode assumir) e a direção da iteração corrente faça a solução vizinha extrapolar esse valor limite, então a direção é invertida. No caso de seleção de participantes, diversos vizinhos são selecionados. Então, um valor vizinho pode ser selecionado mais de uma vez. No entanto, um participante não pode ser selecionado mais de uma vez. Nesse caso, a função pode realizar duas ações: a primeira é mudar a direção e a segunda é escolher aleatoriamente outro participante que ainda não foi selecionado. A função sempre escolherá mudar a direção, mas, caso o ID de participante da direção inversa também já tenha sido selecionado, é feita a seleção aleatória. A Figura 2 ilustra as exceções descritas.

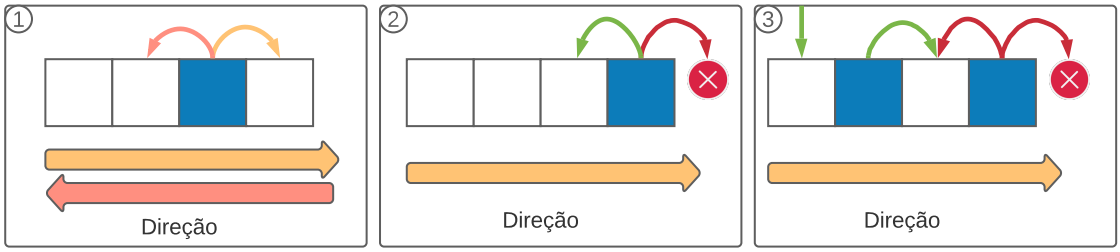


Figura 2. Representação gráfica da estrutura de vizinhança de valores discretos. Cada quadrado representa um valor em uma lista de possíveis valores. A direção da direita representa $+1$ e esquerda -1. As setas vermelhas no cenário 2 e 3 indicam movimentos que a função não pode realizar e as verdes são movimentos possíveis. No cenário 1, a nova solução assumira um valor vizinho da solução atual dependendo da direção. A direção é positiva no cenário 2, mas o valor da solução atual é a última da lista, então a direção é invertida. Já no cenário 3, o último valor não pode ir na direção da iteração, no caso positiva, pois é o último valor da lista, tampouco para a direção negativa, pois esse valor já foi selecionado. Um novo valor aleatório é escolhido.

Para gerar soluções para valores contínuos o processo possui um conceito parecido. Contudo, valores contínuos possuem infinitas vizinhanças. A estrutura de vizinhança para valores contínuos segue a equação:

$$\eta \leftarrow \eta_\beta \pm \epsilon \times unif\left(\eta_a, \eta_b\right), \tag{2}$$

em que, η é o hiperparâmetro contínuo, η_β é o melhor valor até o momento, η_a e η_b é o menor e maior valor que esse hiperparâmetro pode assumir respectivamente. Então, é gerado um valor seguindo uma distribuição uniforme contínua entre os valores mínimo e máximo desse hiperparâmetro. Esse valor será multiplicado por uma constante ϵ, onde $0 < \epsilon < 1$. A constante ϵ dita o passo para o valor vizinho: quanto maior essa constante, maior o passo. O produto entre o valor da distribuição uniforme e do ϵ é somado ou subtraído, dependendo da direção da iteração.

O Algoritmo 1 mostra o funcionamento do esquema proposto para seleção de hiperparâmetros. A primeira seleção é gerada de forma aleatória, ou seja, os usuários são selecionados aleatoriamente, assim como a taxa de aprendizagem e a quantidade de atualizações locais. As seleções seguintes são soluções vizinhas da melhor solução calculada em cada época do algoritmo. Os outros parâmetros do algoritmo são temperatura inicial ($T_{inicial}$), o número máximo de épocas ($epocas$) e um valor de resfriamento (α).

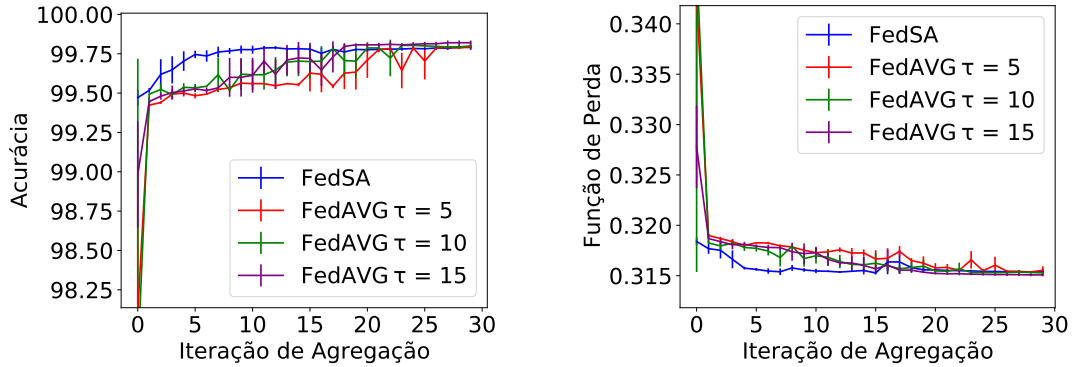
As variáveis que compõem a tupla da solução ($melhor\_solucão$) do algoritmo são: i) O subconjunto de participantes selecionados S_t, ii) a taxa de aprendizagem η e iii) o número de atualizações locais τ.

Na linha 5 do algoritmo, a função GERA_SOLUÇÃO_VIZINHA é responsável por gerar uma solução ($solucão\_atual$) com valores vizinhos à melhor solução já encontrada até o momento ($melhor\_solucão$). Sempre que uma nova solução vizinha é gerada, é realizada uma agregação global (AGREGAR) utilizando os hiperparâmetros dessa solução, como mostrado na linha 6. A perda dessa agregação, calculada pela função de perda global, é armazenada na variável $perda\_atual$. A seguir, na linha 7, o ΔE da solução atual é calculado, demonstrando se a solução atual ($solucão\_atual$) é uma melhoria com relação à melhor solução já encontrada até o momento ($melhor\_solucão$). Na linha 8, é calculada a função de aceitação, que gera um valor p que é a probabilidade de aceitação de uma solução ruim seja escolhida para generalização do algoritmo, ou seja, para impedir que a solução do algoritmo fique presa em mínimos locais. Para tanto, se a solução atual for melhor que a anterior (linha 9), ela é selecionada. Se não for melhor, então verifica-se se o valor gerado pela função de aceitação, p, é maior que um número entre 0 e 1, gerado aleatoriamente seguindo uma distribuição uniforme (linha 13). Caso seja maior, então essa solução sabidamente pior é definida como a melhor. A função RESFRIAMENTO, na linha 16, multiplica a temperatura pelo valor da variável α, logo, $T = T \times \alpha$. Ocorrendo o aceite ou não, é feita uma agregação global com a melhor solução até o momento (linha 18). Essa nova agregação é feita para validar a melhor solução. Caso a melhor solução deixe de ser a melhor, uma solução totalmente aleatória é gerada com a função INICIAR (linha 20), assim o algoritmo não fica preso em uma única região.

6. Avaliação da Proposta e Resultados Experimentais

Para a avaliação da proposta, foi desenvolvido um simulador que gera os participantes e o conjunto de dados desses participantes. Para a distribuição dos fragmentos do conjunto de dados aos participantes, cada participante recebe um fragmento aleatório do conjunto de dados. Como o conjunto de dados utilizado possui mais amostras normais do que ataque, é provável que a maioria dos participantes receba apenas amostras de tráfego normal. Separou-se, também, 30% do conjunto de dados para validação do modelo global. Este cenário é importante para avaliações de aprendizado federado, porque, no ambiente real, cada usuário tem um perfil de uso diferente — cada um gera dados de forma diferente, alguns participantes sofreram ataques, outros não. O cenário simulado conta com 100 IDSs participantes, porém apenas 30% dos participantes são selecionados a cada iteração. Para a escolha do número de participantes levou-se em consideração uma quantidade que evitasse o sobreajuste do modelo global. Para os testes, utilizou-se o modelo *Multilayer Perceptron* (MLP) com duas camadas escondidas. A função de ativação utilizada entre as camadas foi a *Rectified Linear Units* (ReLU), enquanto na camada de saída utilizou-se a função Softmax.

Como base para comparação com a meta-heurística FedSA proposta, utilizou-se o algoritmo de agregação FedAVG, o qual é o algoritmo de aprendizado federado mais utilizado. No cenário dos testes, o FedAVG também seleciona trinta participantes, porém de forma aleatória. O FedAVG utilizou a taxa de aprendizado de $\eta = 0.1$ e com um decaimento a cada iteração de $\gamma = \frac{0.1}{i}$, onde i é o número da iteração. Atualmente, não há um estudo formal para o valor da taxa de aprendizado, regularmente é feita uma etapa



(a) Acurácia da proposta FedSA comparada com o FedAVG.

(b) Perda da função entropia cruzada.

Figura 3. Acurácia e perda referentes ao conjunto de validação, mostrando que o FedSA é mais rápido que o FedAVG, pois converge com menor número de iterações globais.

de ajuste fino, ou a escolha empírica. A cada iteração a taxa de aprendizado atualiza para:

$$\eta^t \leftarrow \eta^{t-1} \times \frac{1}{1+\gamma i}. \tag{3}$$

Para fins de comparação, 30 agregações globais foram realizadas em todos os testes.

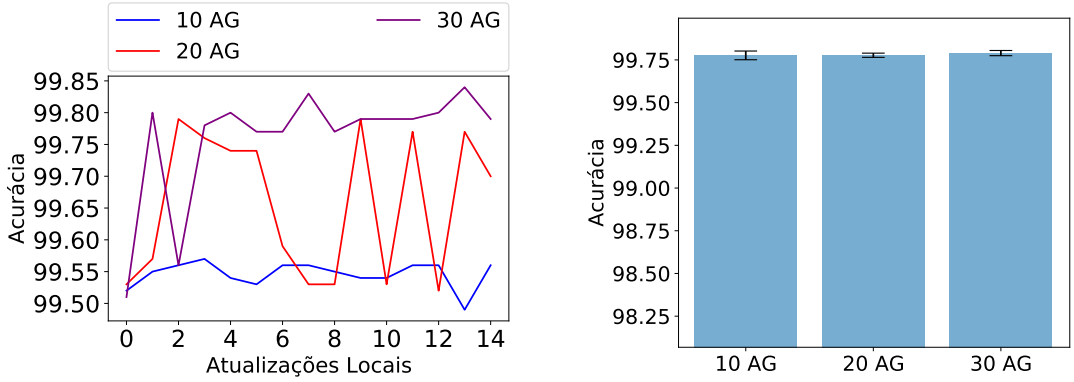
Para a avaliação, foi utilizado um computador equipado com processador Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz, 128GB de RAM. Para criar um ambiente federado, foi utilizada a biblioteca TensorFlow[2], que permite o uso de modelos de aprendizado de máquina da biblioteca Keras[3]. A meta-heurística FedSA proposta foi implementado usando a linguagem Python. O código-fonte usado pode ser encontrado no GitHub[4].

Para gerar os tráfegos em cada rede local, utilizou-se o conjunto de dados CICDDoS2019 [Sharafaldin et al., 2019]. Esse conjunto de dados possui tráfego normal e tráfego de ataques de negação de serviço distribuídos (*Distributed Denial of Service* - DDoS) baseado em reflexão e exploração. Ambos os ataques de reflexão e exploração são ataques em que a identidade do invasor permanece oculta por meio da utilização de componentes legítimos de terceiros. Os pacotes são enviados aos servidores refletores por invasores com o endereço IP de origem como o IP da vítima para sobrecarregar a vítima com pacotes de resposta.

Para transformar os pacotes de rede em fluxos de rede bidirecionais utilizou-se o CICFlowMeter[5]. A ferramenta CICFlowMeter gera 80 características diferentes como duração do fluxo, número de pacotes, número de bytes, tamanho médio dos pacotes, entre outras. As características IP de origem, IP de destino, porta de origem, porta de destino e protocolo de transporte são removidas para que o modelo aprenda com relação às estatísticas dos fluxos de rede. Remover essas características evita o enviesamento.

Primeiramente, avaliou-se o impacto do número de atualizações locais ao longo das iterações globais sobre o FedAVG, comparado à proposta FedSA, que modifica automaticamente a quantidade de atualizações locais. Esse resultado é mostrado na Figura 3.

[2]Disponível em https://www.tensorflow.org/. Acessado em Abril de 2021
[3]Disponível em https://keras.io/. Acessado em Abril de 2021
[4]https://github.com/helioncneto/FederatedLearning/blob/master/Simu_FedSA_IDS.py
[5]Disponível em https://www.unb.ca/cic/research/applications.html#CICFlowMeter. Acessado em 14/03/2021



(a) Acurácia do modelo global com o algoritmo de agregação FedAVG utilizando diferentes valores de atualizações locais no treinamento.

(b) Acurácia do modelo global utilizando a proposta FedSA em três diferentes cenários.

Figura 4. Acurácia do modelo global com diferentes valores de atualizações locais τ, considerando cenários com 10, 20 ou 30 agregações globais.

É possível perceber que a proposta converge em 10 iterações globais, enquanto o melhor resultado do FedAVG foi em 15 iterações globais, utilizando 15 atualizações locais. Os cenários do FedAVG para 5 ou 10 atualizações locais convergem depois de 20 iterações globais. Para garantir a relevância estatística, os resultados são apresentados como médias de rodadas de experimento associadas a um intervalo de confiança de 95%.

Já a Figura 4(a) mostra a acurácia do FedAVG em três diferentes cenários. O primeiro cenário utiliza 10 agregações globais, o segundo, 20 e o terceiro, 30. Em cada cenário, variou-se a quantidade de atualizações locais de 1 até 15. Essa etapa é um ajuste fino feito manualmente dos hiperparâmetros do FedAVG. Essa análise avalia se as escolhas do FedSA sobre a quantidade de atualizações locais têm impacto sobre a acurácia da solução, quando comparado ao uso do FedAVG. Assim, a Figura 4(b) apresenta um gráfico em barras com a acurácia da proposta nos mesmo três cenários da avaliação apresentada na Figura 4(a). A proposta obteve os melhores resultados que o algoritmo FedAVG sem a necessidade de realizar de diversos testes de ajuste fino para identificar o melhor conjunto de hiperparâmetros.

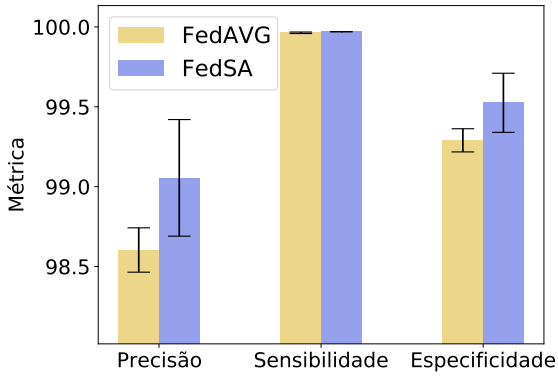


Figura 5. Precisão, Sensibilidade e Especificidade do modelo global no cenário de 10 agregações globais usando o FedSA e FedAVG.

Como o conjunto de dados de validação é um conjunto de dados desbalanceado, há a necessidade da avaliação com outras métricas como precisão, especificidade e sensibilidade, que indicam se o modelo tem boas previsões para ambas as classes. A Figura 5 mostra a barra de precisão, sensibilidade e especificidade da proposta no cenário com apenas 10 agregações globais para o FedSA e FedAVG. Utilizou-se 10 agregações globais, pois foi o necessário para a proposta convergir. Para a comparação com o FedSA, escolheu-se o melhor número de atualizações locais avaliado no teste de ajuste fino do FedAVG. A proposta FedSA obteve melhor precisão, *i.e*, a quantidade de ataques classificados corretamente. Igualmente, a proposta alcançou boa especificidade, *i.e*, a taxa de ataques classificado corretamente.

7. Conclusão

A escolha do subconjunto de participantes para treinamento colaborativo em aprendizado federado é tradicionalmente realizada de forma aleatória. Essa abordagem não garante que os participantes selecionados tenham dados de qualidade para otimizar o modelo global. Da mesma forma, uma taxa de aprendizado global para cada iteração é definida estaticamente ou com um valor de decaimento. Outro hiperparâmetro importante para a convergência do aprendizado federado é a quantidade de atualizações locais, que para o algoritmo de agregação de média federada (*Federated Average* - FedAVG) é um valor fixo. Esse artigo propôs a meta-heurística de Arrefecimento Simulado Federado (*Federated Simulated Annealing* — FedSA) para acelerar a convergência do modelo global de aprendizado sem a necessidade de um processo custoso temporalmente de escolha de hiperparâmetros. A escolha dos usuários, a taxa de aprendizado e a seleção do número de atualizações locais para o treinamento colaborativo em aprendizado federado foram realizadas com o apoio da meta-heurística. Os resultados mostraram que o uso da FedSA para resolver esses problemas obteve 99,8 % de acurácia enquanto a média federada (FedAVG) atingiu 99,24 % de acurácia. A solução proposta atinge acurácia máxima com apenas 10 iterações globais, reduzindo os custos de comunicação, uma preocupação comum do aprendizado federado. Como trabalho futuro, pretende-se avaliar a solução separando os usuários em quadrantes com base em suas perdas locais.

Referências

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.

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.

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

Cunha Neto, H. N., Mattos, D. M. F. e Fernandes, N. C. (2020). Privacidade do usuário em aprendizado colaborativo: Federated learning, da teoria à prática. *Minicursos do Simpósio Brasileiro de Segurança de Informação e de Sistemas Computacionais - SBSeg*, 20:142–195.

de Souza, L. A. C., Antonio F. Rebello, G., Camilo, G. F., Guimarães, L. C. B. e Duarte, O. C. M. B. (2020). Dfedforest: Decentralized federated forest. Em *2020 IEEE International Conference on Blockchain (Blockchain)*, p. 90–97.

Guimarães, L. C. B., Rebello, G. A. F., Fernandes, F. S., Camilo, G. F., de Souza, L. A. C., dos Santos, D. C., de Oliveira, L. G. C. M. e Duarte, O. C. M. B. (2020). Temiant: Threat monitoring and intelligent data analytics of network traffic. Em *2020 4th Conference on Cloud and Internet of Things (CIoT)*, p. 9–16.

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, H. e Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, 9(20):4396.

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.

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, 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.

Sharafaldin, I., Lashkari, A. H., Hakak, S. e Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. Em *2019 International Carnahan Conference on Security Technology (ICCST)*, p. 1–8.

Silva, J. V. V., Lopez, M. A. e Mattos, D. M. F. (2020). Attackers are not stealthy: Statistical analysis of the well-known and infamous kdd network security dataset. Em *2020 4th Conference on Cloud and Internet of Things (CIoT)*, p. 1–8.

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.

Van Laarhoven, P. J. e Aarts, E. H. (1987). Simulated annealing. Em *Simulated annealing: Theory and applications*, p. 7–15. Springer.

Viegas, E., Santin, A., Bessani, A. e Neves, N. (2019). Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, 93:473 – 485.

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.