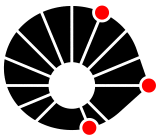
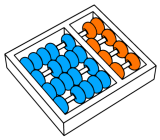




Análise de performance e viabilidade de algoritmos de aprendizado federado utilizando Flower

T. S. Solera *P. S. Nogueira* *A. M. Souza* *J. B. D. Costa*
L. F. Bittencourt

Relatório Técnico - IC-PFG-24-06
Projeto Final de Graduação
2024 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Thiago dos Santos Solera * Pedro Strambeck Nogueira *
Allan Mariano de Souza * Joahannes Bruno Dias da Costa *
Luiz Fernando Bittencourt *

Resumo

O aprendizado federado é uma solução comum para o problema de treinar modelos de inteligência artificial em ambientes onde os dados não podem ser facilmente centralizados, garantindo também a privacidade dos clientes. No entanto, implementar soluções de aprendizado federado em larga escala para aplicações reais pode ser particularmente desafiador. O framework Flower propõe lidar com problemas como a homogeneidade de hardware e linguagem de implementação, ao mesmo tempo em que facilita a execução performática dos algoritmos em testes de maior escalabilidade. Este trabalho visa avaliar a performance e viabilidade dos algoritmos FedProx e FedAvgM, dois algoritmos usados em casos de dados *non-IID*, no contexto de aprendizado federado quando implementados usando o framework Flower, avaliando sua acurácia e escalabilidade, assim como o tempo necessário para alcançar tais resultados, com o objetivo de classificar a viabilidade de uso de cada solução.

1 Introdução

O aprendizado federado tem emergido como uma solução inovadora no campo do aprendizado de máquina, oferecendo uma abordagem descentralizada que preserva a privacidade dos dados. Em vez de centralizar os dados em um único servidor, o aprendizado federado permite que modelos de aprendizado de máquina sejam treinados diretamente nos dispositivos dos usuários, com apenas os parâmetros do modelo sendo compartilhados. Esta metodologia é particularmente valiosa em contextos onde a privacidade dos dados é crucial, como na saúde, finanças e aplicações móveis.

No entanto, a implementação prática do aprendizado federado apresenta diversos desafios. A heterogeneidade dos dados e das capacidades dos dispositivos participantes pode afetar significativamente a eficiência e a eficácia dos modelos treinados. Para enfrentar esses desafios, diferentes algoritmos de aprendizado federado foram desenvolvidos, cada um com abordagens específicas para lidar com a variabilidade dos dados e promover a estabilidade do treinamento. Este trabalho foca na análise de performance e viabilidade de dois algoritmos de aprendizado federado: FedProx e FedAvgM.

A análise de performance e viabilidade desses algoritmos será conduzida utilizando o framework Flower, uma plataforma de código aberto que facilita a experimentação e implementação de aprendizado federado em larga escala. Flower oferece uma infraestrutura

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

flexível e extensível, suportando uma ampla gama de algoritmos de aprendizado federado e permitindo a integração com bibliotecas de aprendizado de máquina.

Além disso, serão empregadas as bibliotecas PyTorch e Hydra. PyTorch é amplamente utilizado para aprendizado profundo, fornecendo ferramentas essenciais para a construção e treinamento de modelos. Hydra simplifica a gestão de configurações complexas, facilitando a realização de experimentos controlados e eficientes.

Este estudo visa fornecer uma compreensão detalhada das capacidades e limitações dos algoritmos FedProx e FedAvgM, avaliando como as métricas do algoritmo afetam sua precisão e sua performance. Na próxima seção, apresentaremos o referencial teórico que embasa este estudo, detalhando os principais conceitos e ferramentas utilizadas na análise de performance e viabilidade dos algoritmos de aprendizado federado.

2 Referencial Teórico

Esta seção apresenta os conceitos fundamentais relacionados ao aprendizado federado e ao framework Flower, além de explicar o papel do PyTorch e da biblioteca Hydra no contexto de aprendizado de máquina distribuído. Também são abordados os conceitos de heterogeneidade de sistema e heterogeneidade de dados, que são críticos no contexto do aprendizado federado. Por fim, os algoritmos FedProx e FedAvgM são descritos.

2.1 Conceitos Chave em Aprendizado Federado

2.1.1 Descentralização

O aprendizado federado descentraliza o treinamento de modelos, permitindo que os dados permaneçam nos dispositivos locais. Isso contrasta com o aprendizado de máquina tradicional, onde os dados são centralizados em um servidor para treinamento.

2.1.2 Privacidade dos Dados

Como os dados nunca deixam o dispositivo local, o aprendizado federado melhora a privacidade dos dados, tornando-o particularmente atraente para aplicações sensíveis como saúde, finanças e dados pessoais.

2.1.3 Heterogeneidade

O aprendizado federado lida com a heterogeneidade dos dados e dos dispositivos utilizados para o treino.

- A **heterogeneidade de sistema** refere-se à diversidade nas capacidades computacionais e de rede dos dispositivos participantes em um ambiente de aprendizado federado. Isso inclui variações em termos de poder de processamento, memória, armazenamento, largura de banda de rede entre os diferentes dispositivos e, para dispositivos móveis, disponibilidade de bateria.

- A **heterogeneidade de dados** refere-se às diferenças nas distribuições de dados locais em cada dispositivo participante. Isso pode ser causado por diferentes contextos de uso, preferências do usuário, ou variações geográficas.
  - Distribuição *non-IID (Independent and Identically Distributed)*: Os dados em diferentes dispositivos podem não seguir a mesma distribuição estatística. Por exemplo, em um cenário de saúde, os dados coletados em diferentes regiões podem variar significativamente devido a fatores demográficos e ambientais.
  - Volume de Dados: A quantidade de dados disponíveis em cada dispositivo pode variar, influenciando o impacto de cada dispositivo na atualização do modelo global.
  - Qualidade dos Dados: A qualidade e a natureza dos dados (por exemplo, dados ruidosos versus dados limpos) podem diferir entre os dispositivos.
  - Diversidade Semântica: Os dados podem representar diferentes categorias ou tipos de informações, dificultando a criação de um modelo global que seja igualmente eficaz para todos os dispositivos.

2.1.4 Agregação de Modelos

O servidor central é responsável por agregar os parâmetros dos modelos treinados localmente. Métodos como FedAvg (*Federated Averaging*) são comumente usados para combinar os parâmetros recebidos de maneira eficiente.

2.2 Framework Flower

Flower (FLWR) [3] é um framework de código aberto projetado para facilitar a implementação e experimentação de aprendizado federado. Ele fornece uma infraestrutura flexível que pode ser adaptada a diferentes cenários e requisitos de aprendizado federado, ele suporta uma ampla gama de algoritmos de aprendizado federado e pode ser integrado com várias bibliotecas de aprendizado de máquina como TensorFlow e PyTorch. Este framework é projetado para escalar horizontalmente, possibilitando utilizar um grande número de dispositivos clientes e pode ser usado tanto para simulações em ambiente de desenvolvimento quanto para implementações em produção.

2.3 PyTorch

PyTorch [9] é uma biblioteca de aprendizado de máquina de código aberto desenvolvida pelo *Facebook AI Research (FAIR)*. Ela é amplamente utilizada para aplicações que envolvem aprendizado profundo e aprendizado de máquina em geral. A sub biblioteca **torch.nn** fornece componentes básicos para construir e treinar redes neurais profundas. A sub biblioteca **torch.optim** oferece vários algoritmos de otimização como SGD, Adam, RMSprop, entre outros, que são utilizados para ajustar os parâmetros do modelo durante o treinamento. Por fim, a biblioteca PyTorch facilita a manipulação e carregamento de dados através das subbibliotecas **torch.utils.data** e **torchvision**, que oferecem utilitários para trabalhar com datasets, data loaders e transformações de dados.

2.4 Hydra

Hydra [10] é uma biblioteca de código aberto para configurar projetos complexos de Python, desenvolvida pelo *Facebook AI*. Ela facilita a gestão de configurações de aplicativos, especialmente em cenários onde são necessários múltiplos parâmetros e ajustes. Hydra permite a definição de configurações hierárquicas que podem ser facilmente alteradas ou combinadas e são organizadas em diretórios e arquivos YAML. Ela também permite modificar configurações diretamente da linha de comando, facilitando a experimentação e ajuste fino de parâmetros sem modificar os arquivos de configuração.

2.5 FedProx

FedProx [5] (*Federated Proximal*) é uma extensão do algoritmo FedAvg, projetada para lidar com a heterogeneidade de sistemas e dados em ambientes de aprendizado federado. Ele introduz um termo de proximidade na função de perda local, que visa mitigar os efeitos adversos da variabilidade entre os dispositivos participantes.

- **Termo Proximal**: FedProx adiciona um termo proximal à função de perda local de cada cliente. Este termo penaliza grandes desvios entre os modelos locais e o modelo global, incentivando os modelos locais a permanecerem próximos do modelo global.

$$\mathcal{L}_{prox}(\mathbf{w}) = \mathcal{L}(\mathbf{w}; \mathcal{D}) + \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}_t\|^2$$

  Onde $\mathbf{w}$ são os parâmetros do modelo local, $\mathbf{w}_t$ são os parâmetros do modelo global, $\mathcal{L}(\mathbf{w}; \mathcal{D})$ é a função de perda original, e μ é um hiperparâmetro que controla a força do termo proximal.

O termo proximal ajuda a reduzir o impacto da heterogeneidade dos dados e sistemas, permitindo que os modelos locais sejam treinados de forma mais estável e alinhada com o modelo global, mesmo quando os dados e capacidades dos dispositivos variam significativamente.

FedProx pode ser adaptado a diferentes níveis de heterogeneidade ajustando o valor do hiperparâmetro μ, permitindo um controle mais fino sobre a regularização do treinamento local.

2.6 FedAvgM

FedAvgM [4] (*Federated Averaging with Server Momentum*) é uma variação do algoritmo FedAvg que incorpora a técnica de momentum, utilizada para acelerar a convergência dos modelos de aprendizado de máquina. O uso do momentum ajuda a mitigar os desafios associados à variabilidade nos dados e nas capacidades computacionais dos dispositivos participantes no aprendizado federado.

- **Incorporação de Momentum**: FedAvgM introduz um termo de momentum no processo de atualização dos modelos. O momentum é uma técnica de otimização

que ajuda a acelerar a convergência em direção ao mínimo global ao suavizar as atualizações dos parâmetros do modelo.

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1 - \beta)\Delta \mathbf{w}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{v}_{t+1}$$

Onde $\mathbf{v}_t$ é o termo de momentum, β é o coeficiente de momentum, $\Delta \mathbf{w}_t$ é a atualização dos parâmetros, $\mathbf{w}_t$ são os parâmetros do modelo global, e η é a taxa de aprendizado.

O termo de momentum ajuda a reduzir a oscilação das atualizações dos parâmetros, promovendo uma convergência mais suave e estável do modelo global.

O uso do momentum também contribui para a estabilidade do treinamento, ajudando a evitar que o modelo fique preso em mínimos locais e promovendo uma convergência mais eficiente.

3 Metodologia

Com o objetivo de estudar as dificuldades de se lidar com dados não-IID e heterogeneidade sistêmica, os algoritmos selecionados foram escolhidos entre aqueles presentes nas baselines do framework Flower [8] o que permite a reprodução dos resultados apresentados a seguir de forma mais prática. Como o FedProx adiciona mais uma camada de heterogeneidade, a sistêmica, é possível visualizar como a adição desse fator influencia na acurácia do modelo em relação ao FedAvgM cuja baseline não envolve heterogeneidade entre os dispositivos dos clientes.

3.1 Datasets

- **MNIST** [6]: uma base de dígitos manuscritos consistindo de 10 classes (0 a 9), provendo 60000 exemplos para treinamento e 10000 exemplos para validação. As imagens são normalizadas para um tamanho de 28x28 pixels e são representadas em escala de cinza.

- **Fashion-MNIST** [7]: uma base de diferentes categorias de roupas, semelhante ao MNIST em números de exemplos tanto para treinamento quanto para teste, assim como o total de 10 classes em imagens de dimensões 28x28 representadas em escala de cinza. Este dataset foi utilizado somente para o treinamento e avaliação do FedAvgM.

3.2 Parâmetros

Inicialmente, para o treinamento dos modelos foram utilizados os parâmetros pré-definidos pelas baselines, com foco em comparar a performance entre os dois algoritmos, variando os seguintes itens:

- **Número de clientes total**: foram utilizados 10, 100 e 1000 clientes.

- **Número de clientes em treinamento por rodada**: 10% do número total de clientes, com exceção ao caso em que o número total de clientes é 10, nesse caso foram utilizados 2 clientes para o treinamento.
- **Número de rodadas de treinamento**: os modelos foram treinados com 10, 50 e 100 rodadas.
- **Número de CPUs utilizados para o treinamento**: 2 núcleos.

Após os testes terem sido realizados utilizando estes parâmetros cada algoritmo foi testado oscilando outras métricas que afetam o modelo, tais execuções estarão mais detalhadas na seção resultados.

3.3 Particionamento dos Dados

3.3.1 FedAvgM

Para construir uma distribuição não-IID é utilizado um coeficiente de concentração α, responsável por controlar a distribuição obtida com *Latent Dirichlet Allocation (LDA)* utilizado no particionamento dos dados.

O parâmetro α afeta a distribuição de modo que quando α tende ao infinito, todos os clientes possuem uma distribuição idêntica, enquanto quando α tende a 0, cada cliente possui exemplares de apenas uma única classe.

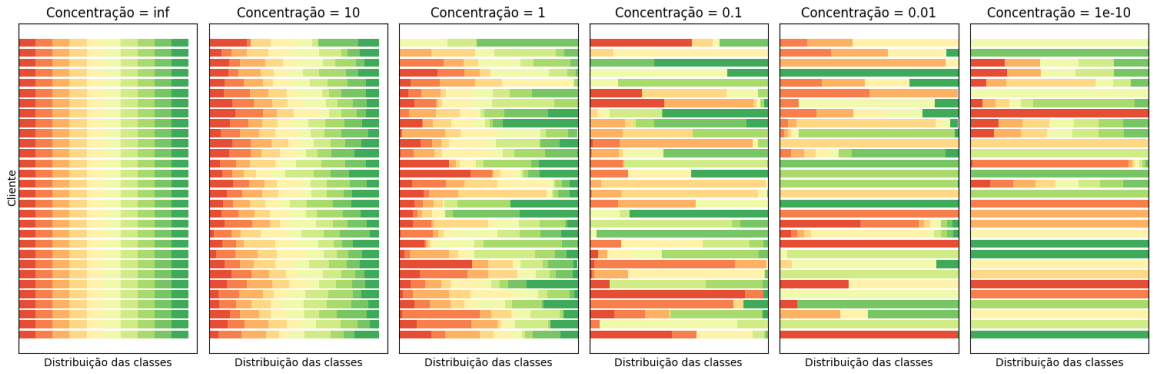


Figura 1: Exemplo de comportamento das distribuições de acordo com o parâmetro α. (Imagem gerada utilizando o código presente no diretório da baseline "FedAvgM" acessível no repositório do framework Flower)

3.3.2 FedProx

O particionamento da base de dados é feito seguindo uma divisão patológica onde cada cliente possui exemplos de duas (das dez) classes de rótulos. O número de exemplos em cada cliente é derivado por amostragem a partir de uma distribuição seguindo a Lei de Potência.

3.4 Especificações das máquinas utilizadas no experimento

Processador	Memória RAM	Algoritmo
Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz	16 GB 2400 MHz DDR4	FedAvgM
Apple M1 Chip 8-core CPU 3.20GHz 4	8 GB 3733 MHz LPDDR4x	FedProx

Tabela 1: Equipamentos usados nos experimentos.

4 Resultados

Nesta seção são apresentados os resultados experimentais obtidos com a execução de ambos os algoritmos. Primeiramente, mostra-se os resultados dos experimentos com parâmetros em comum entre os dois modelos e, em seguida, uma apresentação mais detalhada da execução de cada um dos algoritmos é feita, utilizando os dados colhidos nos testes individuais.

4.1 Experimentos em comum entre os algoritmos

Os dados presentes nessa subseção foram obtidos pela execução dos algoritmos nas respectivavs máquinas apresentadas na seção anterior e o dataset utilizado para o treino e validação de ambos os modelos é o MNIST.

Clientes	Clientes/rodada	Rounds	Algoritmo	Acurácia	Tempo execução(s)
10	2	10	FedAvgM	0,5414	56,65
10	2	10	FedProx	0,4286	13,28
10	2	50	FedAvgM	0,9470	192,61
10	2	50	FedProx	0,6808	74,55
10	2	100	FedAvgM	0,9593	592,61
10	2	100	FedProx	0,7526	147,84
100	10	10	FedAvgM	0,5300	76,42
100	10	10	FedProx	0,6779	11,96
100	10	50	FedAvgM	0,9604	319,46
100	10	50	FedProx	0,8026	60,84
100	10	100	FedAvgM	0,9789	628,48
100	10	100	FedProx	0,8578	116,41
1000	100	10	FedAvgM	0,4593	86,44
1000	100	10	FedProx	0,7459	81,28
1000	100	50	FedAvgM	0,9575	230,08
1000	100	50	FedProx	0,8714	440,21
1000	100	100	FedAvgM	0,9609	652,17
1000	100	100	FedProx	0,8942	848,73

Tabela 2: Resultados dos experimentos em comum entre os algoritmos.

4.1.1 Acurácia

- **FedAvgM**: Consistentemente, o FedAvgM apresenta uma maior acurácia em todos os cenários, especialmente conforme aumentam o número de rounds. Este comportamento pode ser atribuído ao foco na heterogeneidade dos dados, permitindo um melhor ajuste do modelo aos dados distribuídos.

- **FedProx**: Embora o FedProx tenha uma acurácia inicial inferior ao FedAvgM, ele mostra uma melhoria significativa à medida que aumentam os rounds e o número de clientes. O FedProx é projetado para lidar com a heterogeneidade dos dispositivos e dados, o que pode explicar sua acurácia inferior ao FedAvgM, tendo em vista que a implementação do FedAvgM usada no experimento lida somente com a heterogeneidade de dados.

4.1.2 Tempo de execução

- **FedAvgM**: Apresenta tempos de execução maiores em comparação com o FedProx em cenários com o número de clientes menor. Isso pode ser devido ao fato de o algoritmo FedProx lidar com o treinamento em dispositivos com diferentes configurações e, à medida que o número de clientes aumenta, a quantidade de dispositivos que atrasam o treinamento do modelo também aumenta.

- **FedProx**: O FedProx geralmente apresenta tempos de execução menores nos cenários iniciais (menos rounds e clientes), mas esse tempo aumenta significativamente com mais rounds e clientes. Isso pode ser devido ao esforço adicional para manejar a heterogeneidade dos dispositivos, visto que com uma maior quantidade de clientes a quantidade de dispositivos que causam gargalo no treinamento do modelo também aumenta.

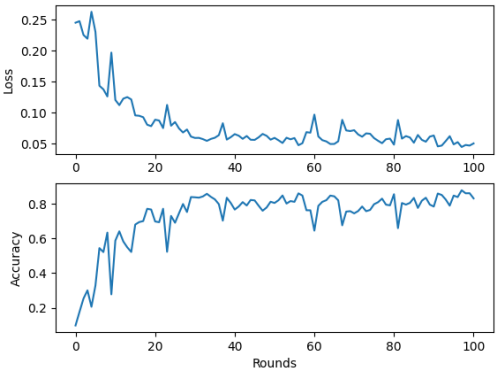
Vale ressaltar que, embora ambos os algoritmos tenham sido executados utilizando o mesmo número de núcleos do CPU, a execução em diferentes máquinas pode ter influenciado no tempo de treino. Essa variação no ambiente de execução pode introduzir discrepâncias que tornam injusta a comparação dos valores absolutos de tempo de execução, potencialmente causando uma aparente desvantagem para um dos algoritmos testados.

4.2 FedProx

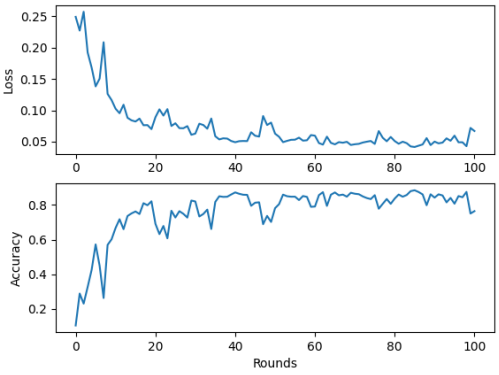
Nesta subseção são apresentados os resultados colhidos executando o algoritmo FedProx utilizando os parâmetros predefinido pela baseline do flower (1000 clientes, 10 clientes por round e 100 rounds), comparando as métricas obtidas localmente com os próprios resultados informados pelo framework [8] e com os resultados informados pelo trabalho original [5] de apresentação do algoritmo. Em seguida, é apresentado o resultado de execuções do algoritmo utilizando um número maior de clientes para verificar a usabilidade do framework em simulações que exigem maior capacidade computacional.

4.2.1 Stragglers

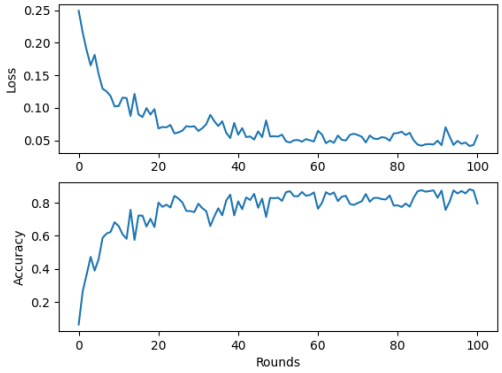
No contexto do aprendizado federado, o termo *stragglers* refere-se a dispositivos participantes no treinamento do modelo cujo desempenho é significativamente mais lento do que o dos outros dispositivos na rede. Sendo assim, os testes executados para fins de comparação entre a execução local do algoritmo e sua baseline, assim como o os resultados apresentados no trabalho original de proposta do algoritmo, são utilizando 0%, 50% e 90% de *stragglers*.



(a) 0% stragglers



(b) 50% stragglers



(c) 90% stragglers

Figura 2: Gráficos de acurácia e perda obtidos na execução local do algoritmo

Os gráficos acima foram obtidos executando o algoritmo utilizando o parâmetro μ igual a 2. Essa escolha foi feita com base na melhor performance do algoritmo ao utilizar valores maiores que zero, tendo como referência os resultados apresentados na página do baseline e no trabalho de apresentação do algoritmo [5], pode-se observar que a convergência do modelo é favorecida quando o termo proximal é maior que zero.

Sendo assim, observa-se que os resultados produzidos pela execução local do algoritmo está em linha com a performance apresentada no baseline e na publicação original do algoritmo. Nota-se que a curva de perda e acurácia com o passar dos rounds segue o padrão

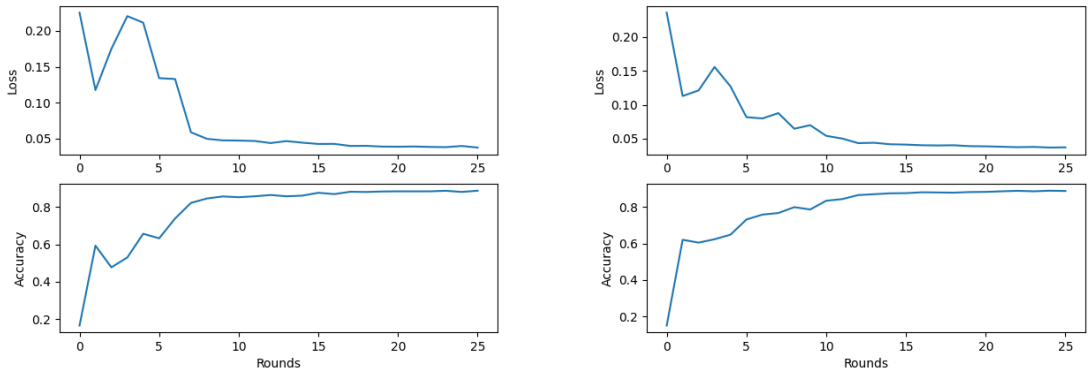
encontrado nas referências citadas anteriormente e, assim como no baseline do modelo, a convergência ocorre por volta do round de número 40.

Em todos os casos a acurácia fica acima de 0.8, porém, após a convergência, pode-se observar uma estagnação na acurácia do modelo que não supera 0.9 independemente da quantidade de rounds utilizadas no treino do algoritmo. Tal comportamento também é visto tanto no baseline quanto na publicação inicial do algoritmo.

4.2.2 Usabilidade

Para verificar a usabilidade do algoritmo e do framework em condições mais próximas da realidade, alguns parâmetros foram modificados nos 2 testes feitos para simular essas condições:

- **Número de clientes**: Em ambos os testes foram utilizados um total de 5000 clientes.
- **Clientes por round**: No primeiro teste foi utilizado 250 clientes por round e no segundo 500 clientes.
- **learning rate**: Para esses testes, foi observado que o parâmetro melhora a performance do modelo quando igual a 0.2 (originalmente é igual a 0.03).



(a) 250 clientes por round

(b) 500 clientes por round

Figura 3: Gráficos de acurácia e perda para os testes de usabilidade.

Clientes por round	Tempo de execução(s)
250	1069,77
500	1801,34

Tabela 3: Tempo de execução dos testes de usabilidade do FedProx.

Tendo em vista os resultados obtidos, observa-se que o aumento na quantidade de clientes participando do treino do modelo por round e a alteração do valor do parâmetro learning

rate resultaram em uma convergência antecipada do modelo, que, para esses testes, foi por volta do décimo round. Portanto, um número menor de rounds de treino para esses casos é necessário. É notável, também, a suavização nas curvas de acurácia produzidas, que se matém com menos ruído ao longo da execução se comparada as curvas produzidas nos testes anteriores. Isso representa boa adaptabilidade do algoritmo em lidar com um número grande de dispositivos.

Por fim, os tempos de execução para os dois testes estão dentro do esperado, permitindo que o framework seja utilizado de forma viável para simulações de aprendizado federado.

4.3 FedAvgM

Seguem os resultados obtidos com o intuito de avaliar a eficácia do algoritmo FedAvgM em aliviar a dificuldade de se treinar modelos com aprendizado federado usando dados *non-IID* e as discussões pertinentes ao contexto deste trabalho.

4.3.1 Eficácia

A proposta do algoritmo [4] é mitigar a degradação dos resultados que acontece com o aumento da divergência das distribuições quando o treinamento é realizado com o algoritmo FedAvg.

A adição de momento deve acelerar a convergência do modelo, então, para verificar esse comportamento, foram executados testes com ambos os algoritmos aplicando diferentes concentrações para as distribuições.

Os resultados a seguir foram obtidos em execuções com o dataset Fashion-MNIST, para ambos os algoritmos. Os parâmetros de treino consistem em 1 época de treinamento local em cada cliente com *learning rate* de 0.003, um total de 100 clientes com 5% de amostragem para o ajuste do modelo global, com 0.97 de coeficiente de *momentum* para o servidor durante 25 *rounds*. O parâmetro de concentração das distribuições é variado de forma a abranger os valores $\alpha \in \{0.001, 0.01, 0.1, 1, 10, 100, 10000\}$.

Como pode ser observado na Figura 4, a acurácia do FedAvg cai significativamente com o aumento do parâmetro α, enquanto o FedAvgM, apesar de sofrer uma pequena queda, se mantém sempre com um melhor resultado.

A escolha de apenas 25 *rounds* foi feita deliberadamente para ressaltar a aceleração de convergência proporcionada pelo uso do algoritmo FedAvgM, pois como o dataset utilizado no experimento é considerado mais simples, ambos os algoritmos são capazes de alcançar taxas de acurácia elevados para grande parte dos valores de α, mas necessitando de diferentes quantidades de *rounds*. Com essa escolha podemos verificar que mesmo em estágios iniciais do treinamento do modelo, as taxas de acurácia alcançadas com o uso de cada algoritmo chegam a divergir por mais de 50%.

Outra escolha importante de parâmetro é o uso de 100 clientes, pois a influência dos dados *non-IID* pode ser estatisticamente suavizada caso o número de clientes seja baixo o suficiente para que, em média, cada cliente seja escolhido pela amostragem um número semelhante de vezes, o que também é acentuado com a execução de mais *rounds*.

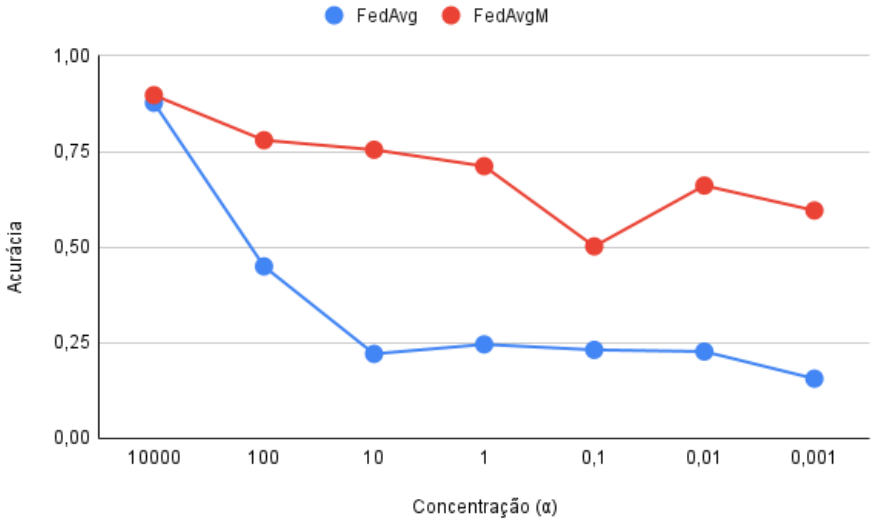


Figura 4: Acurácia dos algoritmos FedAvg e FedAvgM para diferentes valores de α.

4.3.2 Usabilidade

Através dos resultados obtidos podemos avaliar a usabilidade, tanto do algoritmo FedAvgM quanto do framework Flower, para um contexto onde dados *non-IID* estão presentes.

O framework auxiliou na execução de experimentos contendo até 1000 clientes com diferentes configurações de treinamento, o que demonstra sua capacidade de fornecer uma estrutura simples e altamente configurável para simulações de larga escala. O tempo de execução está dentro do esperado para o número de *rounds* executados e dos recursos utilizados nas execuções.

Mesmo adicionando altas taxas de heterogeneidade de dados, o algoritmo foi capaz de produzir resultados consistentes com sua proposta, visto que superou consideravelmente a performance do FedAvg quando ambos foram submetidos às mesmas condições de treinamento.

5 Conclusão

Este trabalho apresentou uma análise das dificuldades de lidar com dados não-IID e heterogeneidade sistêmica no contexto do aprendizado federado, utilizando os algoritmos FedAvgM e FedProx. Utilizando o framework Flower, foi possível reproduzir e comparar os resultados dos dois algoritmos, evidenciando suas respectivas performances e tempos de execução.

Os resultados demonstram que o FedAvgM consistentemente alcança maior acurácia em comparação ao FedProx, o que é esperado, visto que o FedAvgM foi testado somente considerando heterogeneidade de dados, em contrapartida ao FedProx que adiciona a isso a heterogeneidade entre os dispositivos utilizados no treino do modelo.

O FedProx demonstrou uma boa adaptabilidade ao lidar com diferentes níveis de stragglers, mantendo a convergência do modelo em níveis semelhantes aos vistos no trabalho original de apresentação do algoritmo. Os testes de usabilidade com o FedProx, variando o número de clientes por rodada e ajustando o learning rate, indicaram que o algoritmo é capaz de se adaptar a cenários com grande número de dispositivos, proporcionando uma convergência mais rápida e curvas de acurácia mais suaves. Isso sugere que o FedProx é altamente adaptável e pode ser utilizado em simulações de aprendizado federado de larga escala.

Por fim, o estudo do FedAvgM utilizando o dataset Fashion-MNIST mostrou que este algoritmo é eficaz em mitigar a degradação de desempenho associada à divergência das distribuições de dados não-IID. A adição de momentum acelera a convergência do modelo, tornando-o uma escolha robusta para cenários com alta heterogeneidade de dados.

Os resultados obtidos confirmam a viabilidade e a eficiência dos algoritmos FedAvgM e FedProx em diferentes contextos de aprendizado federado, destacando a importância de considerar tanto a acurácia quanto o tempo de execução ao escolher o algoritmo mais adequado para uma aplicação específica. O framework Flower provou ser uma ferramenta valiosa para conduzir experimentos e simulações em aprendizado federado, proporcionando uma plataforma flexível e eficiente para pesquisa e desenvolvimento nessa área.

Referências

[1] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu and Sen Zhao (2021), *Advances and Open Problems in Federated Learning* , Foundations and Trends® in Machine Learning: Vol. 14: No. 1–2, pp 1-210. http://dx.doi.org/10.1561/2200000083

[2] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, *Federated Learning: Challenges, Methods, and Future Directions,* in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50-60, (May 2020).

[3] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, Nicholas D. Lane, *Flower: A Friendly Federated Learning Research Framework*. arXiv:2007.14390, (2022).

[4] Tzu-Ming Harry Hsu, Hang Qi, Matthew Brown, *Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification*. arXiv:1909.06335, (2019).

[5] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, Virginia Smith, *Federated Optimization in Heterogeneous Networks*. arXiv:1812.06127v5, (2020).

[6] *THE MNIST DATABASE of handwritten digits*. Disponível em: https://yann.lecun.com/exdb/mnist/. Acesso em 07/2024

[7] *Fashion MNIST dataset, an alternative to MNIST*. Disponível em: https://keras.io/api/datasets/fashion_mnist/. Acesso em 07/2024

[8] *Flower Baselines Documentation*. Disponível em: https://flower.ai/docs/baselines/. Acesso em 07/2024

[9] *PyTorch Documentation*. Disponível em: https://pytorch.org/docs/. Acesso em 07/2024

[10] *Hydra Documentation*. Disponível em: https://hydra.cc/docs/intro/. Acesso em 07/2024