Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Aprendizado combinado

Miguel de Mello Carpi

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Marcelo Finger
Cossupervisora: Drª. Aline Rodrigheri Ioste

São Paulo
2022

Resumo

Miguel de Mello Carpi. **Aprendizado combinado**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Este trabalho estuda a técnica de aprendizado combinado por meio de experimentos empíricos. Os experimentos retratam desafios em aprendizagem distribuída encontrados ao utilizar modelos de redes neurais conhecidos como perceptron multicamada, convolucional e recorrente. Os resultados obtidos com a técnica de aprendizado combinado são comparados com os obtidos pela técnica de aprendizado federado, e chega-se a conclusão que a técnica de aprendizado combinado é válida. No entanto, há experimentos e estudos a serem feitos para entendê-la melhor.

Palavras-chave: aprendizado combinado. aprendizado federado. redes neurais. acurácia. experimentos empíricos.

Abstract

Miguel de Mello Carpi. **Combined learning**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

This work studies the combined learning technique based on empirical experiments. The experiments portray challenges in distributed learning and englobe neural networks models such as multi-layer perceptron, convolutional and recurrent. A comparison between these experiments and the results obtained from the federated learning approach shows that the combined learning approach is valid. However, there are still experiments and studies to be done to understand it better.

Keywords: combined learning. federated learning. neural networks. accuracy. empirical experiments.

Lista de abreviaturas

CL	aprendizado combinado (*combined learning*)
FL	aprendizado federado (*federated learning*)
IID	independentes e identicamente distribuídas

Lista de figuras

Lista de programas

Sumário

Introdução

O aprendizado combinado é uma técnica de aprendizado de máquina distribuída, proposta por Ioste A., 2022, similar à técnica de aprendizado federado. Assim como no aprendizado federado, o aprendizado combinado busca encontrar um modelo robusto a partir da combinação de modelos de mesma arquitetura, mas treinados em conjuntos de dados diferentes. Em ambas as técnicas, os clientes não podem compartilhar os dados entre si, deste modo a obtenção de um modelo melhor ocorre ao combinar os parâmetros aprendidos por cada cliente em seu conjunto de dados particular.

Objetivos

Este trabalho tem os seguintes objetivos: implementar o aprendizado combinado em uma linguagem de programação real, treinar modelos de rede neural (multicamadas, recorrentes e convolucionais) na configuração de aprendizado combinado, comparar a acurácia obtida na configuração de aprendizado combinado com a obtida pelo aprendizado centralizado e aprendizado federado.

Organização do Conteúdo

O capítulo 1 traz uma descrição do aprendizado federado e do aprendizado combinado. Em seguida, no capítulo 2, a metodologia do trabalho é apresentada. Esses dois capítulos compõe o necessário para entender os resultados obtidos.

Já no capítulo 3 encontram-se os resultados obtidos após realizar os experimentos. De forma complementar, mais resultados são apresentados no apêndice A. Essa divisão ocorre, pois os resultados do apêndice complementam os resultados principais e, como são muitos, evitam ofuscar os resultados de maior interesse para a análise.

Por fim, o capítulo 4 apresenta as principais evidências que são observadas nos resultados. Com os resultados obtidos, é possível validar o uso do aprendizado combinado, porém é necessário realizar mais experimentos com formas diferentes de dividir os dados e com outras arquiteturas para entender melhor os casos no qual a técnica se sai melhor.

Capítulo 1

Fundamentação Teórica

1.1 Aprendizado federado e aprendizado combinado

Em Kairouz *et al.*, 2021, os autores definem o aprendizado federado (FL, do inglês *federated learning*) como uma configuração onde há vários clientes (como dispositivos celulares ou organizações) que treinam de forma colaborativa um modelo sob as instruções de um coordenador (servidor central) enquanto mantém os dados descentralizados, isto é, os clientes não trocam dados entre si e também não trocam dados com o coordenador. Como o aprendizado combinado (CL, do inglês *combined learning*) é baseado na mesma configuração, pode-se dizer que o CL é um tipo de aprendizado federado. A Figura 1.1 mostra a arquitetura de um modelo de aprendizado distribuído.

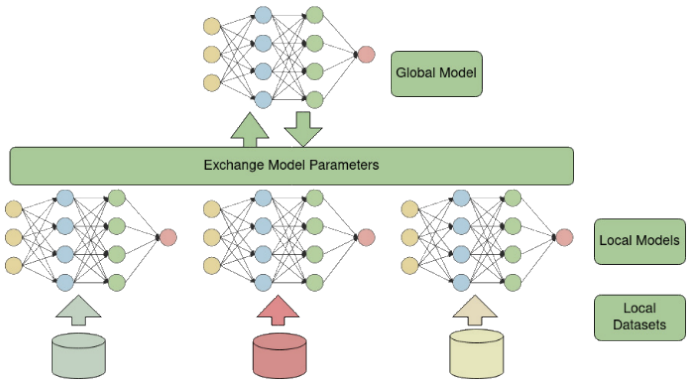

Figura 1.1: *Exemplo de arquitetura distribuída para um problema de classificação.*
Fonte: *Ioste A., 2022*

A diferença entre o aprendizado combinado e o aprendizado federado está na maneira como os parâmetros (ou pesos) dos modelos dos clientes são combinados para gerar o modelo final. No FL é comum utilizar uma função de combinação convexa para tal, como uma média aritmética simples, porém em Ioste A., 2022, visando à obtenção de um modelo melhor, propõe uma função de combinação não convexa.

Os resultados obtidos em Ioste A., 2022 mostram que utilizar uma função de combinação não convexa resulta em modelos melhores em configurações onde os dados não são independentes e identicamente distribuídos entre os clientes. Esse resultado é importante, pois Kairouz *et al.*, 2021 diz que caracterizar as diferenças entre conjuntos de dados de clientes distintos ainda é um problema em aberto na área de aprendizado distribuído.

1.1.1 Visão geral do aprendizado combinado

O treinamento proposto em Ioste A., 2022 tem como objetivo mitigar os custos de comunicação entre os clientes participantes. A maneira de treinar os clientes de forma distribuída no contexto do aprendizado federado, no artigo, assume o seguinte (tradução dos autores deste texto):

1. Os participantes devem completar o treinamento de seus modelos locais
2. Os modelos locais dos participantes devem ser capazes de generalizar bem a partir dos conjuntos de dados locais
3. Os modelos locais gerados pelos participantes podem possuir generalizações distintas
4. Após o treinamento local, cada participante envia os pesos do modelo aprendido para um coordenador central
5. O coordenador combina os pesos recebidos para construir um modelo global
6. O coordenador envia para todos os participantes o modelo global
7. Os participantes podem treinar o modelo global recebido em seus conjuntos de dados locais
8. O processo para obter o modelo global aproxima um modelo treinado com todos os dados disponíveis
9. O coordenador decide quando o processo atinge um ponto estável
10. . Quando o processo termina, todos os participantes recebem uma cópia do modelo combinado, adquirindo a habilidade de processar padrões não vistos em seus respectivos dados locais.

A Figura 1.2 ilustra o processo de, definido um modelo de rede neural que todos os clientes treinarão, como funciona o passo a passo para combinar os modelos. Nele a função *comb*, indica que a função de combinação está sendo aplicada para obter uma nova camada a partir dos pesos obtidos pelos clientes. Esse processo de treino local pelos clientes, envio dos pesos, cálculo dos novos pesos e atualização dos pesos em cada cliente é repetido várias vezes.

1.2 Dados IID e não-IID

Em Kairouz *et al.*, 2021, os autores fazem uma taxonomia de dados não-IID em configurações de aprendizado federado. Dados não-IID são o oposto de dados independentes e identicamente distribuídos (IID) e são bem comuns em configurações de aprendizado

distribuído, como é o caso do CL e do FL, pois as distribuições de dados entre os clientes costuma ser diferente.

Ainda em Kairouz *et al.*, 2021, os autores constatam que é um problema em aberto da área de aprendizado federado caracterizar as diferenças entre conjuntos de dados dos clientes em cenários reais. Em B. McMahan *et al.*, 2017, os autores realizam uma divisão com base na classe para simular uma distribuição de dados assimétrica em um problema de classificação de imagem, e em Hsieh *et al.*, 2020 os autores estudam o problema de dados assimétricos para redes neurais profundas onde um dos estudos trata a classificação de imagens.

Neste trabalho, a divisão de dados não-IID é chamada de arbitrária, no sentido de que ela consiste em distribuir os dados de forma que cada cliente receba apenas os dados de algumas classes, isto é, os clientes não tem dados de algumas classes. A Figura 1.3 ilustra a divisão não-IID com o conjunto de dados MNIST para 5 clientes, nessa divisão cada cliente recebe imagens referentes a apenas dois dígitos e dois clientes diferentes não recebem imagens dos mesmos dígitos. Essa divisão emula um cenário de distribuição assimétrica e é similar ao que acontece ao realizar a divisão feita em B. McMahan *et al.*, 2017 como os autores comentam.

Já a divisão IID consiste em distribuir para cada cliente uma partição do conjunto original com o mesmo número de dados para cada classe. Nesse caso, a probabilidade de selecionar um dado da classe y em um cliente c_i é a mesma para todo i.

1.3 Redes Neurais

Em Ioste A., 2022, a autora sugere que a técnica de aprendizado combinado continue sendo estudada. Desse modo, este trabalho realiza experimentos com redes neurais recorrentes do tipo LSTM (do inglês *Long Short Term Memory*) e com redes neurais convolucionais para classificação de imagens. Esta não é uma lista exaustiva de todas as aplicações possíveis, mas acredita-se que novos cenários são estudados ao avaliar o desempenho do modelo

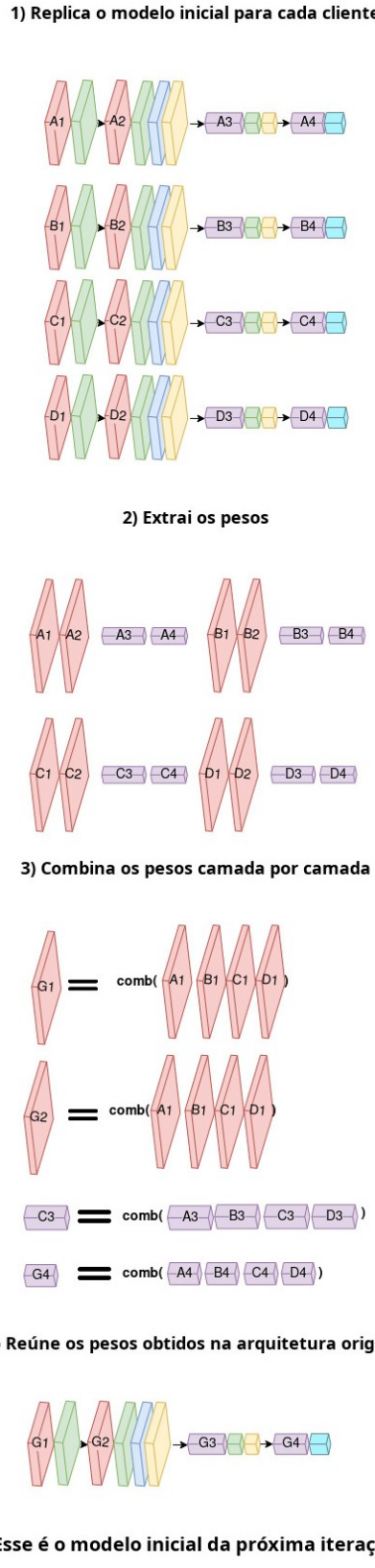

Figura 1.2: *Como os pesos são combinados em uma iteração.*
Fonte: *O Autor*

Figura 1.3: *Exemplo de divisão arbitrária realizada. Cada cliente recebe apenas dois dígitos e dois clientes diferentes não recebem os mesmos dígitos*
Fonte: *O Autor*

Capítulo 2

Metodologia

2.1 Dados

Este trabalho utilizou os seguintes conjuntos de dados: MNIST [1], CIFAR-10 [2] e Wisconsin Breast Cancer [3]. Esses conjuntos de dados são escolhidos por serem conhecidos na área e por serem públicos. Além disso, há dois conjuntos de dados para classificação de imagens (MNIST e CIFAR-10) e um de dados tabulares (Wisconsin Breast Cancer), com isso acredita-se que duas aplicações distintas são estudadas.

2.2 Avaliação do desempenho

A métrica escolhida para medir o desempenho dos algoritmos é a acurácia no conjunto de testes. Cada conjunto de dados foi dividido em três partes: uma para o treino, outra para validação e outra para teste. Os conjuntos de treinamento e validação são relevantes durante o treinamento do modelo. Após o modelo terminar de treinar, isto é, atingir o número total de épocas de treino, a acurácia dele é medida no conjunto de teste.

2.3 Implementação

Para realizar os experimentos descritos em 2.4, é necessário implementar o algoritmo em uma linguagem de programação adequada. No caso optou-se pela escolha da linguagem *Python*[4], devido a familiaridade e a disponibilidade de arcabouços para o desenvolvimento de modelos de redes neurais, como o *Pytorch*[5]. Todo o código do projeto pode ser encontrado no *GitHub*[6] do projeto.

[1] http://yann.lecun.com/exdb/mnist/

[2] https://www.cs.toronto.edu/~kriz/cifar.html

[3] https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)

[4] https://www.python.org

[5] https://pytorch.org/

[6] https://github.com/melloguel/tcc-coln

Como o interesse deste trabalho é explorar configurações nas quais o aprendizado combinado pode ser adequado, e não nas questões de como implementar o aprendizado de forma descentralizada, todos os modelos são treinados na mesma máquina e, por esse motivo, é necessário tomar certos cuidados. Os cuidados envolvem fazer todos os parâmetros, de todos os clientes, caberem na memória disponível e copiar os parâmetros ao invés de passá-los diretamente para evitar que um cliente treine os parâmetros do outro após a combinação.

O programa B.1 ilustra como um cliente, com toda a informação necessária para ele treinar pode ser representado na linguagem escolhida. Já o programa B.2 ilustra como, a partir dos clientes, a combinação dos pesos é realizada. Ao observar os dois programas pode se concluir que os dez pontos apresentados em Ioste A., 2022, e repetidos na seção 1.1.1, são respeitados.

Das funções no programa B.3, a função *combine_camadas* é a mais complexa. Nela ocorre a combinação exponencial dos parâmetros por meio de uma função com forma $a^h = e^{cr_h}$ como apresentada em Ioste A., 2022. Na fórmula, a^h corresponde a uma camada de um cliente h, c é a taxa de aprendizado (*learning rate*) e r_h é o tamanho relativo do conjunto de dados do cliente h em relação ao conjunto de dados inteiro. No programa B.3 c é representado pela variável *conv* e r_h está na lista representada por r. A fórmula em questão aparece na linha 43.

Os programas estão no Apêndice B. Eles tiveram algumas partes omitidas em relação ao que foi utilizado nos experimentos, pois aqui o intuito é ilustrar a ideia seguida para implementar o algoritmo e não todos os detalhes necessários, que podem ser encontrados em sua totalidade no *GitHub* do projeto.

2.4 Experimentos

São realizados experimentos para reunir dados de acurácia da rede obtida pelo aprendizado combinado em diferentes configurações. Cada configuração consiste em um conjunto de dados, um número de clientes (2, 5 ou 10), uma divisão desses dados entre os clientes (uniforme ou arbitrária), um modelo de rede neural e uma função de combinação. Espera-se que o modelo obtido a partir da combinação dos pesos na configuração do CL tenha acurácia melhor no conjunto de testes do que os modelos individuais. Além disso, o desempenho do modelo obtido na configuração do CL é comparado com o modelo obtido na configuração do FL mantendo as outras variáveis.

Ao todo são 132 experimentos, onde 60 são com o conjunto MNIST, 48 são com o conjunto CIFAR-10 e 24 são no conjunto Wisconsin Breast Cancer. Esses conjuntos de dados foram escolhidos, pois são conhecidos na área e por serem públicos.

De forma geral, os conjuntos de dados escolhidos tem distribuição praticamente simétrica no número de itens por classe. A discrepância maior é vista no conjunto Wisconsin Breast Cancer e, nesse caso, é aplicado um aumento de dados para equilibrar as duas classes (tumor benigno e tumor maligno). Com os conjuntos de dados balanceados, é necessário dividi-los entre os clientes, e isso é feito de duas maneiras, uma uniforme e outra arbitrária que buscam representar cenários onde os dados dos clientes são IID e não-IID.

As arquiteturas de rede neural escolhidas variam de acordo com o conjunto de dados. As configurações com o conjunto de dados MNIST tem as seguintes arquiteturas: rede neural multicamadas, rede neural convolucional, rede neural recorrente. Já as configurações para o conjunto de dados CIFAR-10 tem as arquiteturas de rede neural convolucional, rede neural recorrente e rede neural multicamadas. Por fim, as configurações para o conjunto Breast Cancer Wisconsin são rede neural multicamadas e rede neural recorrente. Mais detalhes sobre as arquiteturas são apresentados no Capítulo 3.

Capítulo 3

Resultados

Este capítulo apresenta os principais resultados obtidos com os experimentos. Em primeiro lugar são apresentados os resultados para o conjunto de dados MNIST, em seguida os resultados para o conjunto de dados CIFAR-10 e, por último, os resultados para o conjunto de dados Breast Cancer Wisconsin. Em todos os gráficos a linha tracejada verde retrata a acurácia do modelo obtido ao treinar com o conjunto de dados completo para analisar o impacto na acurácia ao treinar o modelo nas configurações do aprendizado combinado e de aprendizado federado. Por último, cabe dizer que a acurácia é medida em um conjunto de dados de teste separado, que possui todas as classes a serem classificadas e, esse mesmo conjunto, é utilizado para testar tanto os modelos centralizados quanto os descentralizados.

3.1 Resultados MNIST

O conjunto de dados MNIST reúne imagens de dígitos (zero a nove), em preto e branco, escritos à mão para classificação. Para testar o algoritmo de aprendizado combinado são escolhidas três arquiteturas de redes neurais diferentes que são multicamadas, convolucional e recorrente. Para a arquitetura de rede neural multicamada são testados três modelos ao variar o número de camadas e parâmetros. Os modelos finais são os seguintes:

- `smlp` com 84.060 parâmetros distribuídos em três camadas lineares;
- `mmlp` com 199.210 parâmetros distribuídos em três camadas lineares;
- `lmlp` com 317.220 parâmetros distribuídos em seis camadas lineares.
- `conv` com 1.199.882 parâmetros distribuídos em duas camadas convolucionais e duas camadas lineares.
- `rnn` com 214.282 parâmetros distribuídos em uma camada LSTM (do inglês *Long Short Term Memory*) e uma camada linear.

3.1.1 Resultados rede neural multicamadas

As figuras 3.1, 3.2, 3.3 apresentam, respectivamente, os resultados para os modelos smlp, mmlp e lmlp. Antes de combinar os pesos, o modelo smlp é treinado por 10 épocas, o lmlp é treinado 20 épocas e o mmlp é treinado por 20 épocas. Os três modelos centralizados obtém acurácia de 97% no conjunto de dados completo após treinarem pela mesma quantidade de épocas que os modelos distribuídos treinam antes de terem seus pesos combinados.

Ao observar o resultado que se obtém de cada modelo, percebe-se que o pior desempenho ocorre na configuração com 10 clientes na divisão arbitrária, e o melhor resultado é na configuração com distribuição uniforme e dois clientes. Esses resultados são razoáveis, já que na configuração com 10 clientes e divisão arbitrária cada cliente possui apenas os dados de uma única classe enquanto que na divisão uniforme os clientes tem conjuntos de dados com distribuições próximas e há um número menor de clientes para combinar.

Além desses resultados, é possível observar que redes neurais com menos parâmetros produzem resultados melhores ao terem seus pesos combinados. Nas figuras 3.1 e 3.2 é possível ver que, embora bem mais baixa em relação à divisão uniforme, a acurácia dos modelos smlp e mmlp com 10 clientes na divisão arbitrária foi mais alta do que a acurácia do modelo lmlp (Figura 3.3) na mesma configuração.

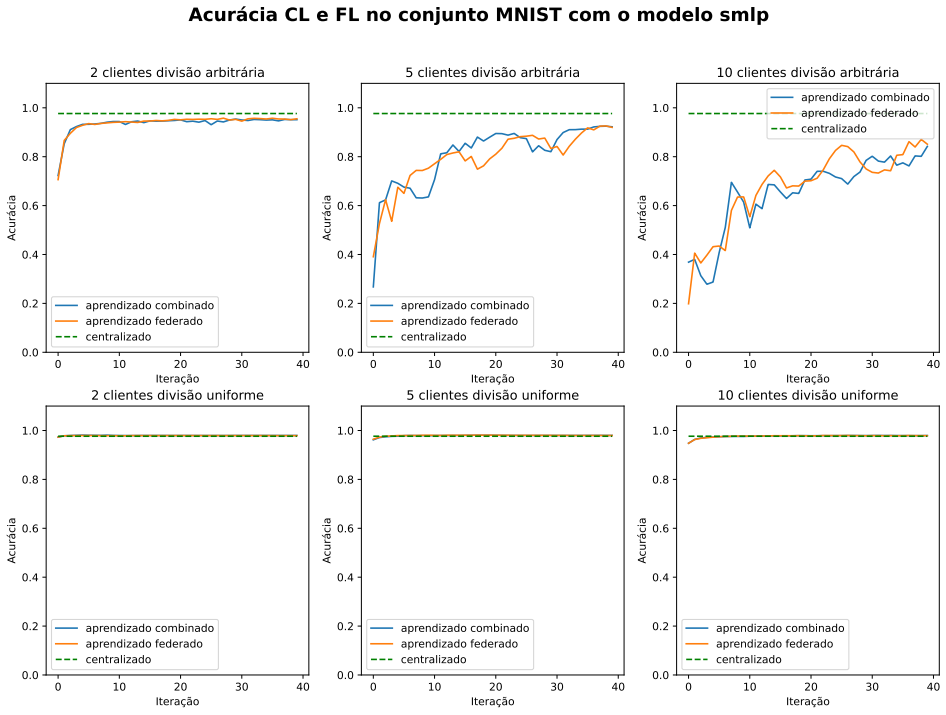

Figura 3.1: *Acurácia das configurações CL e FL no conjunto MNIST com o modelo smlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

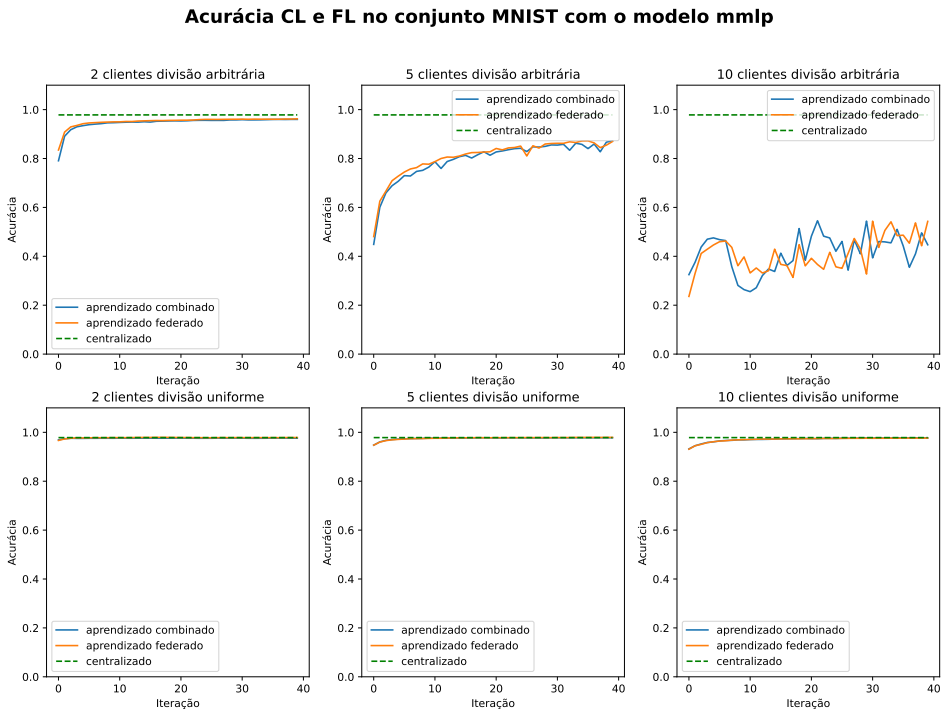

Figura 3.2: *Acurácia das configurações CL e FL no conjunto MNIST com o modelo mmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

3.1.2 Resultados rede neural convolucional

O modelo de rede convolucional centralizado treina 14 épocas no conjunto de dados completo e treina pelo mesmo período antes de cada combinação nas configurações de aprendizado combinado e federado. Na Figura 3.4 observa-se que o modelo de aprendizado combinado nas configurações com dois e dez clientes obtém acurácia melhor do que o modelo de aprendizado federado, embora o resultado tenha sido bem próximo na configuração com cinco clientes. Assim como ocorre com o modelo de múltiplas camadas, o resultado é bem pior quando a divisão dos dados é a arbitrária.

3.1.3 Resultados rede neural recorrente

A Figura 3.5 mostra os resultados para o modelo de rede neural recorrente - rnn. O modelo na configuração centralizada é treinado por duas épocas e atinge acurácia de 96%. Como acontece com os modelos de multicamadas, a acurácia é bem mais baixa na configuração com 10 clientes na divisão arbitrária. Nota-se também que a acurácia do modelo combinado com dois clientes começa mais alta do que a do aprendizado federado, porém cai mais cedo na divisão uniforme com dois clientes.

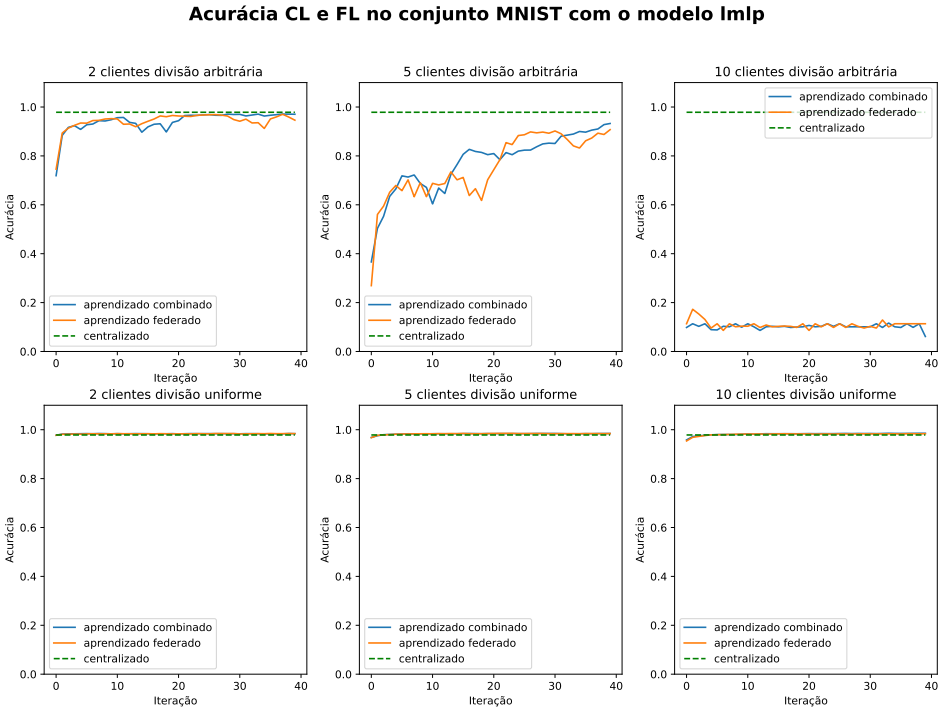

Figura 3.3: *Acurácia das configurações CL e FL no conjunto MNIST com o modelo lmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

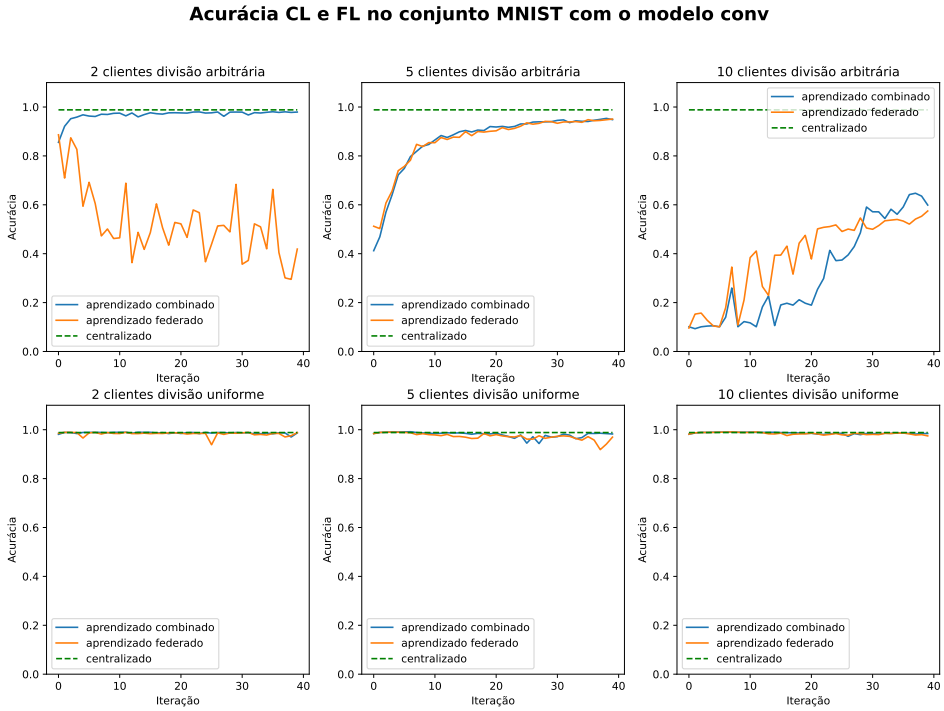

Figura 3.4: *Acurácia das configurações CL e FL no conjunto MNIST com o modelo conv ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

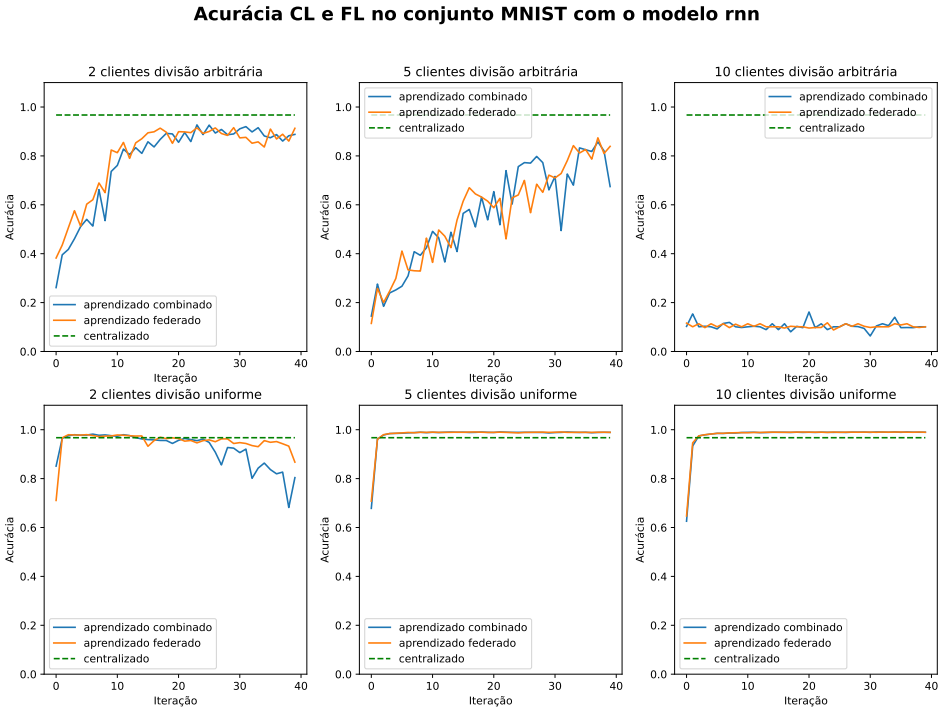

Figura 3.5: *Acurácia das configurações CL e FL no conjunto MNIST com o modelo rnn ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

3.2 Resultados CIFAR-10

CIFAR-10 é um conjunto de dados de imagens coloridas com 10 classes diferentes para classificação. A tarefa de classificação no CIFAR-10 é bem mais difícil do que no conjunto MNIST, logo é uma configuração interessante para estudar o comportamento do CL com um modelo que não atinge uma acurácia elevada. Neste conjunto de dados são utilizados os seguintes modelos de rede neural:

- `smlp` com 3.675.710 parâmetros, distribuídos em quatro camadas lineares;
- `mmlp` com 16.201.010 parâmetros, distribuídos em três camadas lineares;
- `conv` com 666.890 parâmetros, distribuídos em três camadas convolucionais e quatro camadas lineares;
- `rnn` com 60.778 parâmetros e uma *Long Short Term Memory Layer* e duas camadas lineares.

3.2.1 Resultados rede neural multicamadas

As figuras 3.6 e 3.7 apresentam os resultados nas configurações propostas com os modelos smlp e mmlp, respectivamente. A configuração centralizada do modelo smlp atinge acurácia de 42,79% após 16 épocas de treinamento no conjunto de dados completo, já o modelo mmlp obtém acurácia de 45,23% após 20 épocas de treinamento no conjunto de dados completo. Os resultados com esses modelos são interessantes, pois o comportamento é similar ao que ocorre com os modelos multicamadas no conjunto de dados MNIST ao olhar apenas para a divisão arbitrária. Ao olhar para os resultados da divisão uniforme, nota-se que os modelos obtidos pelo CL e pelo FL tem acurácia superior ao do modelo centralizado, além disso nota-se que a acurácia obtida pelo FL é superior a obtida pelo CL, mas somente no smlp (Figura 3.6), pois no modelo mmlp a acurácia obtida pelo CL está bem próxima da obtida pelo FL.

3.2.2 Resultados rede neural convolucional

A rede neural convolucional treina por 30 épocas na configuração centralizada e antes de cada combinação nas configurações de CL e FL. A acurácia que se obtém de forma centralizada é de 67,5% e os modelos descentralizados acalçam acurácia próxima ou superior a essa somente quando a divisão dos dados é uniforme. Na divisão uniforme, ainda, nota-se que tanto a combinação feita pelo CL quanto pelo FL dão resultados bons com poucas iterações, contudo é necessário ressaltar que no CL a acurácia máxima é atingida com menos iterações do que no FL.

3.2.3 Resultados rede neural recorrente

Na Figura 3.9 os resultados para a rede neural recorrente são apresentados. É possível ver que o modelo centralizado, ao ser treinado durante 10 épocas no conjunto de dados completo obtém acurácia de 50,4% e que os modelos obtidos a partir do CL e FL, de novo, não se saem tão bem na divisão arbitrária, mas apresentam resultado similar ao do modelo

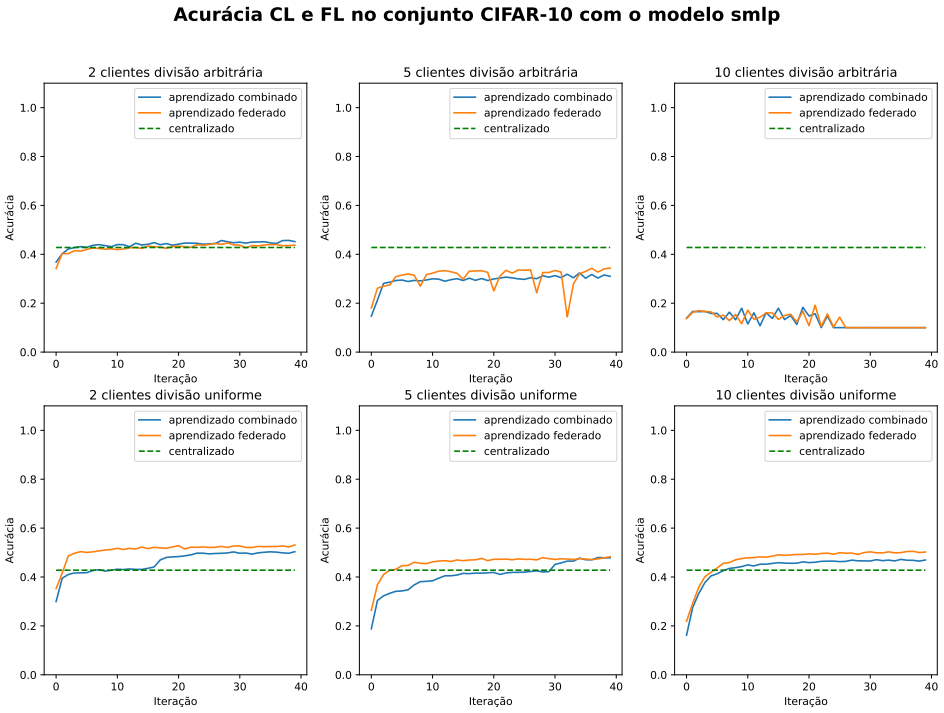

Figura 3.6: *Acurácia das configurações CL e FL no conjunto CIFAR-10 com o modelo smlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

centralizado quando a divisão dos dados é uniforme. Nesse último caso, nota-se que a acurácia ao decorrer das iterações se aproxima melhor do centralizado quando há mais de dois modelos, com dois modelos a acurácia do centralizado é rapidamente alcançada pelo CL que consegue mostrar um resultado superior com menos iterações em relação ao FL.

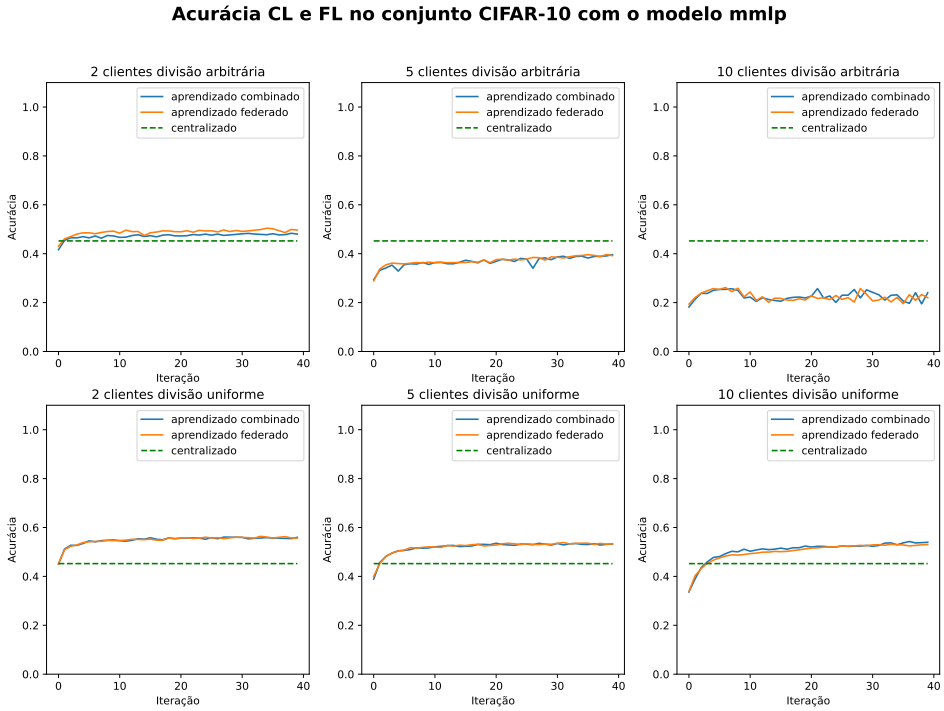

Figura 3.7: *Acurácia das configurações CL e FL no conjunto CIFAR-10 com o modelo mmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

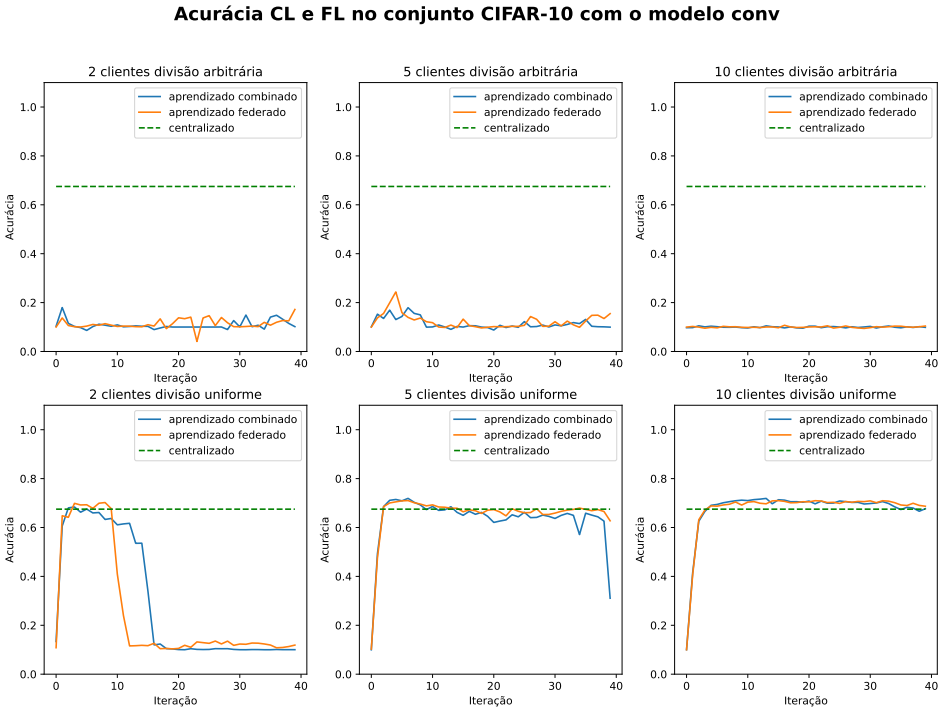

Figura 3.8: *Acurácia das configurações CL e FL no conjunto CIFAR-10 com o modelo conv ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

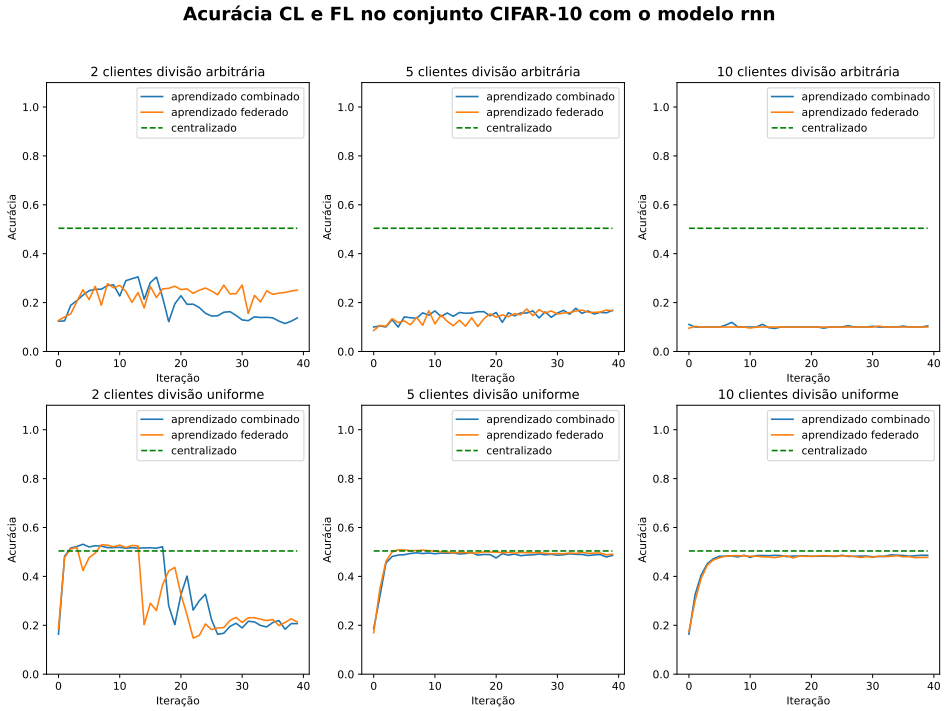

Figura 3.9: *Acurácia das configurações CL e FL no conjunto CIFAR-10 com o modelo rnn ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

3.3 Resultados Breast Cancer Wisconsin

O Breast Cancer Wisconsin é um conjunto de dados que descrevem uma amostra de tumor. A tarefa é classificar, a partir das características da amostra, o tumor em benigno ou maligno. Uma rede neural com três camadas lineares consegue atingir acurácia de 100% no conjunto de dados completo. Esse conjunto de dados é escolhido, pois possui dados tabulares e não imagens como nos conjuntos anteriores (MNIST e CIFAR-10). Outra diferença deste conjunto é que ele possui só duas classes, deste modo a divisão arbitrária retrata apenas a configuração em que um cliente possui somente dados de tumores benignos e o outro só possui dados de tumores malignos.

- `smlp` com 8.786 parâmetros distribuídos em 3 camadas lineares;
- `mmlp` com 14.866 parâmetros distribuídos em 5 camadas lineares;
- `rnn` com 214.274 parâmetros e uma *Long Short Term Memory Layer* e uma camada linear.

3.3.1 Resultados rede neural multicamadas

As figuras 3.11, 3.12, 3.13 apresentam os resultado dos modelos, respectivamente, smlp, mmlp e lmlp na divisão de dados uniforme, e a Figura 3.10 contém os resultados na divisão arbitrária dos modelos lmlp, mmlp e smlp. Os modelos centralizados são treinados por 80 épocas, o mesmo número de vezes que os modelos distribuídos são treinados antes de terem seus pesos combinados. O modelo smlp alcança 98,70% de acurácia, já o modelo lmlp atinge 99,35% de acurácia e o modelo mmlp alcança 97,40% de acurácia em suas configurações centralizadas.

Ao analisar os resultados na Figura 3.10 nota-se que em todos os modelos multicamadas ao menos um entre o aprendizado federado e o aprendizado combinado manteve-se com acurácia em 50%. Em dois dos três modelos (mmlp e smlp), justamente os com menos parâmetros observa-se que ao menos um dos aprendizados distribuídos obteve resultados acima de 50%, porém a acurácia flutua bastante com o número de iterações. No modelo maior, o lmlp, o aprendizado federado apresenta um resultado ótimo logo na primeira iteração e logo depois a acurácia cai para 50%.

Em contrapartida, ao analisar os resultados na configuração com divisão de dados uniforme (figuras 3.11, 3.12, 3.13), observa-se um resultado similar ao obtido com redes neurais multicamadas em configurações com divisão de dados uniforme. Assim como nos outros, tanto o aprendizado federado quanto o aprendizado combinado conseguem atingir resultados próximos ou superiores aos resultados dos modelos centralizados, porém com dois clientes nota-se que os modelos maiores, mmlp e lmlp, apresentam uma queda brusca na acurácia após seis iterações.

A queda brusca na acurácia pode ocorrer devido a quantidade de parâmetros no modelo de rede neural e ao número de épocas de treinamento antes da combinação de parâmetros. Nos modelos com mais parâmetros é pouco provável que os ótimos locais de cada cliente estejam próximos, o que deixa a tarefa de encontrar o modelo centralizado mais difícil. É possível também que se os clientes treinem por muitas épocas antes de terem seus

parâmetros combinados, o modelo de cada um deles se ajuste demais e faça com que o modelo combinado não atenda nem um nem outro cliente, veja a Figura 3.13. Nela também é possível notar que aumentar o número de clientes torna esse fenômeno mais raro, já que o fenômeno não é observado nos experimentos com mais de dois clientes.

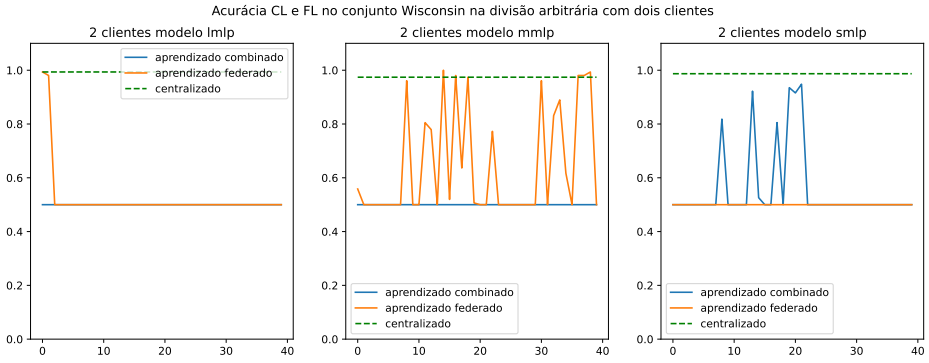

Figura 3.10: *Acurácia CL e FL no conjunto Wisconsin Breast Cancer divisão arbitrária modelos smlp, mmlp e smlp com dois clientes.*
Fonte: O Autor

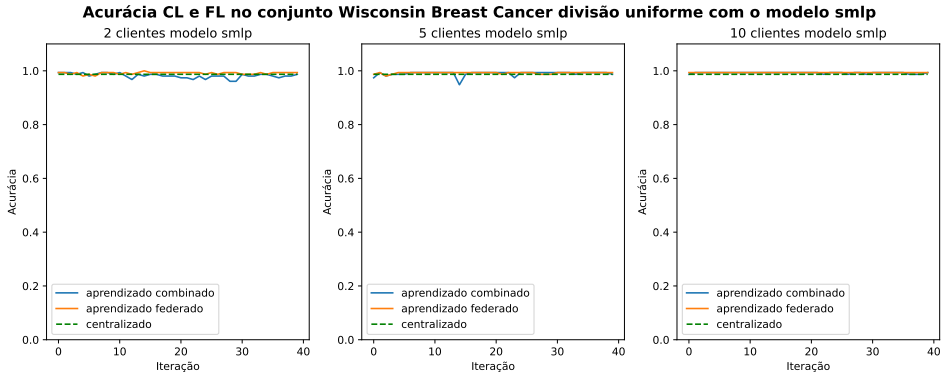

Figura 3.11: *Acurácia CL e FL no conjunto Wisconsin Breast Cancer divisão uniforme modelo smlp número de clientes variado.*
Fonte: O Autor

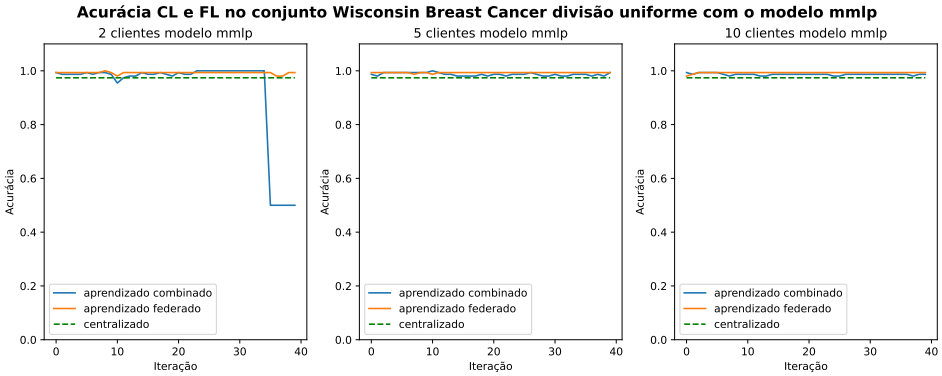

Figura 3.12: *Acurácia CL e FL no conjunto Wisconsin Breast Cancer divisão uniforme modelo mmlp número de clientes variado.*
Fonte: O Autor

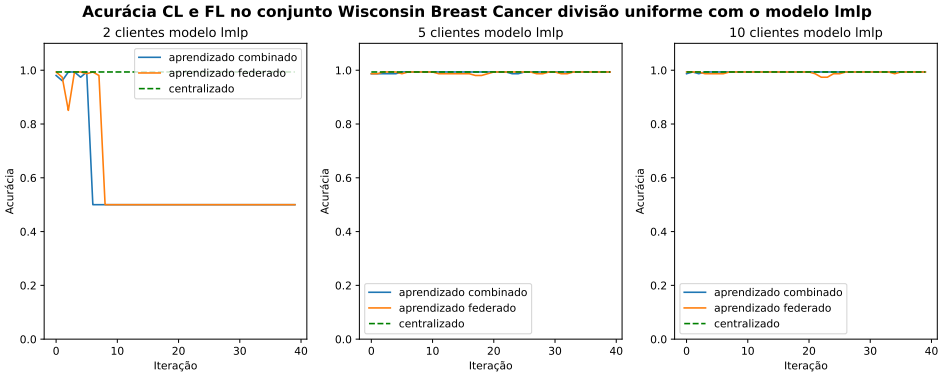

Figura 3.13: *Acurácia CL e FL no conjunto Wisconsin Breast Cancer divisão uniforme modelo lmlp número de clientes variado.*
Fonte: O Autor

Capítulo 4

Discussão e conclusão

4.1 Discussão

O mesmo modelo de rede neural, com o mesmo número de clientes e no mesmo conjunto de dados tem um desempenho melhor quando a divisão dos dados é uniforme. Isso pode ser observado em todos os gráficos do capítulo 3. Esse resultado pode ser explicado pelo motivo de que os parâmetros aprendidos pelas redes neurais são próximos e as funções de combinação, tanto do aprendizado federado quanto do aprendizado combinado, não geram um resultado muito diferente do da entrada. Isso pode ser observado ao ver a acurácia obtida por cada cliente na divisão uniforme, veja as figuras no apêndice A.

Ao observar os resultados na divisão arbitrária, nota-se que além do desempenho ser pior do que na divisão uniforme como dito acima, o desempenho dos modelos distribuídos piora à medida que o número de clientes aumenta. A explicação para esse fenômeno é que as redes treinadas pelos clientes se especializam em poucas classes e a funções de combinar os pesos do aprendizado combinado e do aprendizado federado não são capazes de obter um modelo melhor do que o centralizado. Porém, é possível ver que a acurácia obtida pelos modelos distribuídos é maior do que a obtida por cada cliente para alguns modelos de rede neural em alguns conjuntos de dados.

A análise dos resultados do conjunto MNIST sugere que modelos com uma quantidade menor de parâmetros têm um desempenho melhor nas configurações de aprendizado distribuído, isso pode ser visto ao comparar as figuras A.2 e A.4 no caso com 10 clientes na divisão arbitrária. Com a análise dos resultados do conjunto CIFAR-10, observa-se que um modelo centralizado com uma acurácia melhor pode ter um desempenho pior em sua versão distribuída, que é o caso do modelo conv em relação aos modelos smlp e mmlp. Já com o conjunto Wisconsin Breast Cancer pode-se observar que treinar por muitas épocas antes de combinar os parâmetros de cada cliente deixa o modelo combinado menos estável ou faz com que ele não aprenda - figuras A.24 e A.26.

4.2 Conclusão

Com os resultados que são apresentados neste trabalho, é possível validar o uso do aprendizado combinado. Isso se deve ao fato de que o modelo obtido pelo aprendizado combinado obtém resultados com acurácia similar a obtida pelo aprendizado federado e, além disso, os resultados no Apêndice A mostram que o modelo de aprendizado combinado tem acurácia superior a dos clientes individuais em vários casos.

A partir dos experimentos propostos e resultados obtidos, pode-se dizer que há indícios que em configurações onde os clientes tem conhecimento similar, como nos experimentos com divisão uniforme, o modelo combinado consegue obter acurácia melhor, enquanto que se o conhecimento é mais particular, como nos experimentos com divisão arbitrária, o resultado não costuma ser melhor do que o dos clientes individuais. Além disso nota-se que o aprendizado combinado encontra parâmetros bons com poucas iterações o que é uma característica positiva, já que ajuda a economizar em recursos computacionais importantes como GPU (do inglês *Graphics Processing Unit*) e rede. ele se dá muito bem com modelos com poucos parâmetros e em redes com poucas camadas.

A respeito da análise dos resultados do capítulo 3, é possível levantar algumas questões que devem ser analisadas mais a fundo. A primeira é que um modelo combinado não mostra um bom desempenho quando o modelo de rede neural possui muitos parâmetros ou muitas camadas, isso pode ser observado com os resultados de redes multicamadas nos conjuntos MNIST e CIFAR-10 (3.1, 3.3 e 3.6, 3.7). A segunda é que o CL não obtém um bom desempenho quando a divisão dos dados é muito assimétrica, que é o caso da divisão arbitrária. Em terceiro, o número de épocas de treinamento antes da combinação afeta bem o desempenho, e isto é visto ao analisar as redes multicamadas do conjunto Wisconsin Breast Cancer (figuras 3.11, 3.12, 3.13).

Uma hipótese para o comportamento mencionado acima é a variância na distribuição dos pesos. Embora os clientes comecem o treinamento do mesmo ponto, as diferenças nos conjuntos de dados e no algoritmo de treinamento fazem com que eles convirjam para mínimos locais diferentes. Ao diminuir o número de parâmetros, diminui-se também o conjunto de mínimos locais possíveis, o que torna mais provável que dois, ou mais, clientes distintos atinjam mínimos locais próximos, e por consequência que o modelo combinado seja melhor.

Por fim sugere-se que trabalhos futuros realizem mais experimentos com foco na variação de parâmetros, na variação de épocas de treinamento local, na profundidade da rede neural e na utilização de outras arquiteturas de redes neurais, como de atenção. Com mais testes desse tipo deve ser possível encontrar uma configuração desses parâmetros nos quais o aprendizado combinado tem o melhor desempenho possível. Sugere-se também que novos trabalhos utilizem técnicas de aumento de dados para mitigar problemas na distribuição de dados e melhorar a acurácia de cada cliente.

Apêndice A

Resultados extras

Este apêndice traz a acurácia do aprendizado combinado e do aprendizado federado junto com a acurácia de cada cliente em cada configuração. Esses resultados estão no apêndice e não no texto principal, pois são um complemento dos resultados de maior interesse que são apresentados no Capítulo 3.

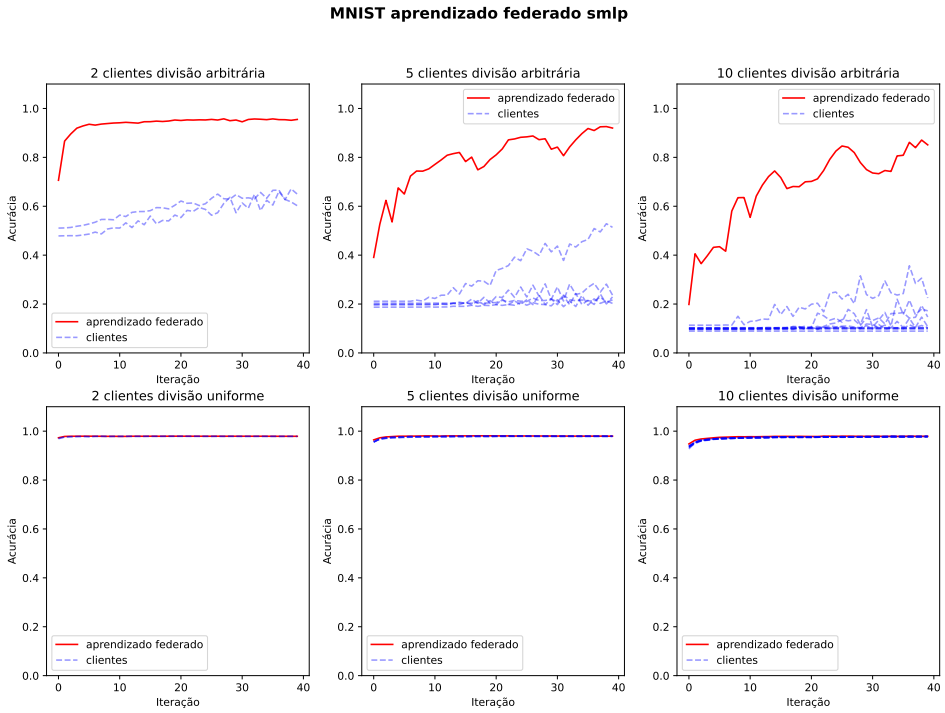

Figura A.1: *Acurácia Fl dos clientes no conjunto MNIST com o modelo smlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

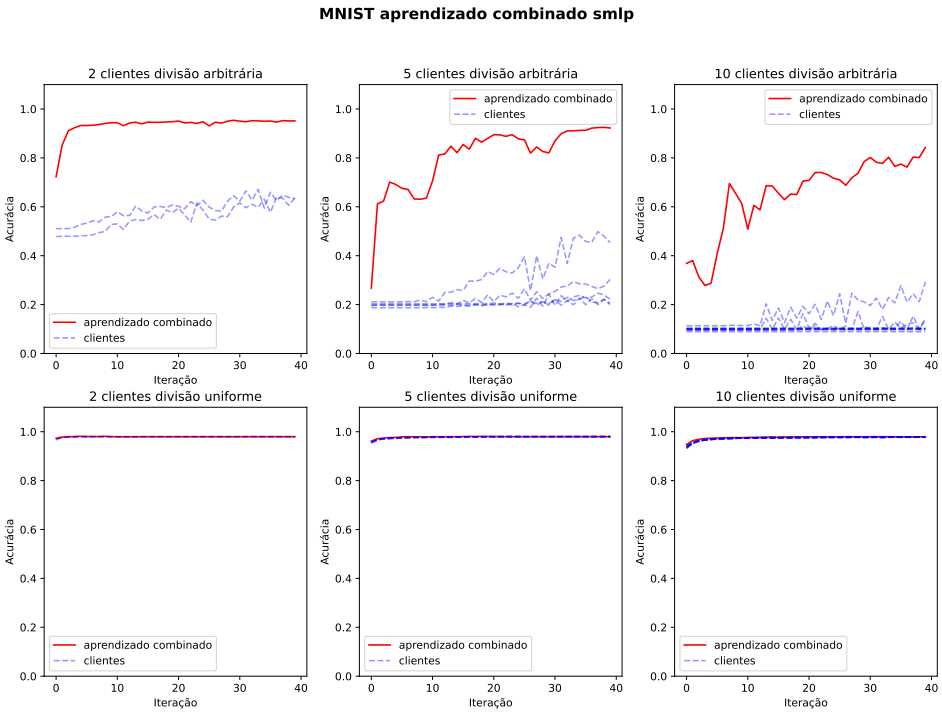

Figura A.2: *Acurácia CL dos clientes no conjunto MNIST com o modelo smlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

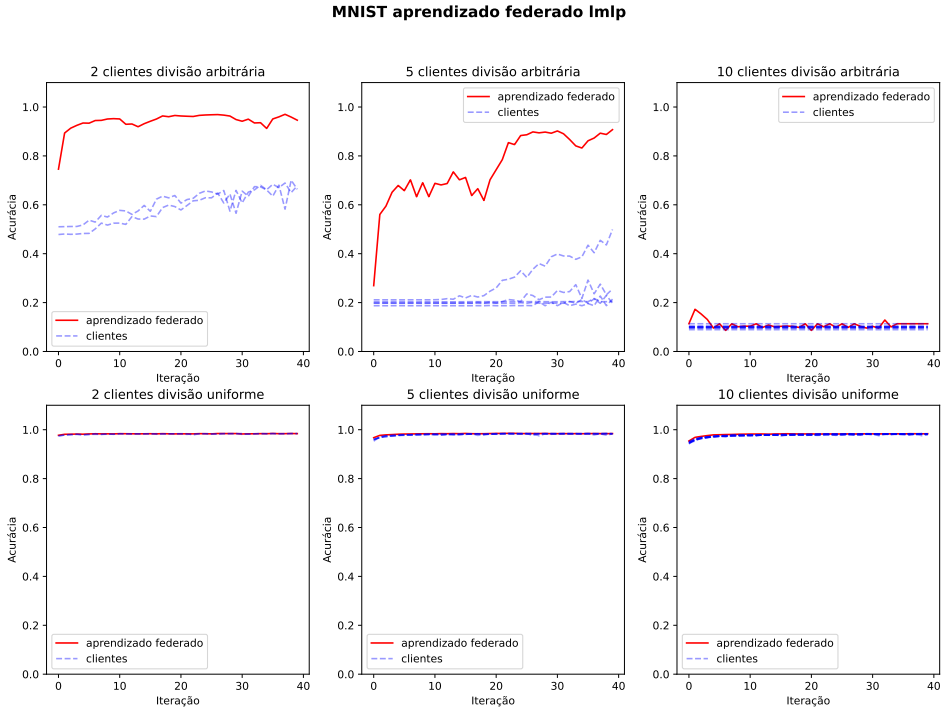

Figura A.3: *Acurácia Fl dos clientes no conjunto MNIST com o modelo lmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

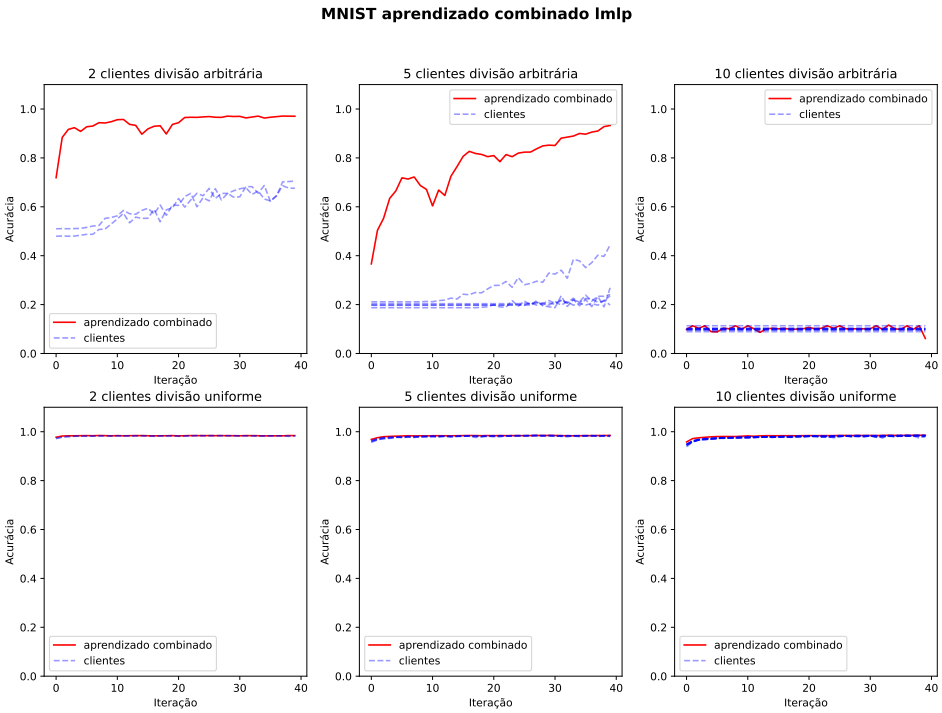

Figura A.4: *Acurácia CL dos clientes no conjunto MNIST com o modelo lmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

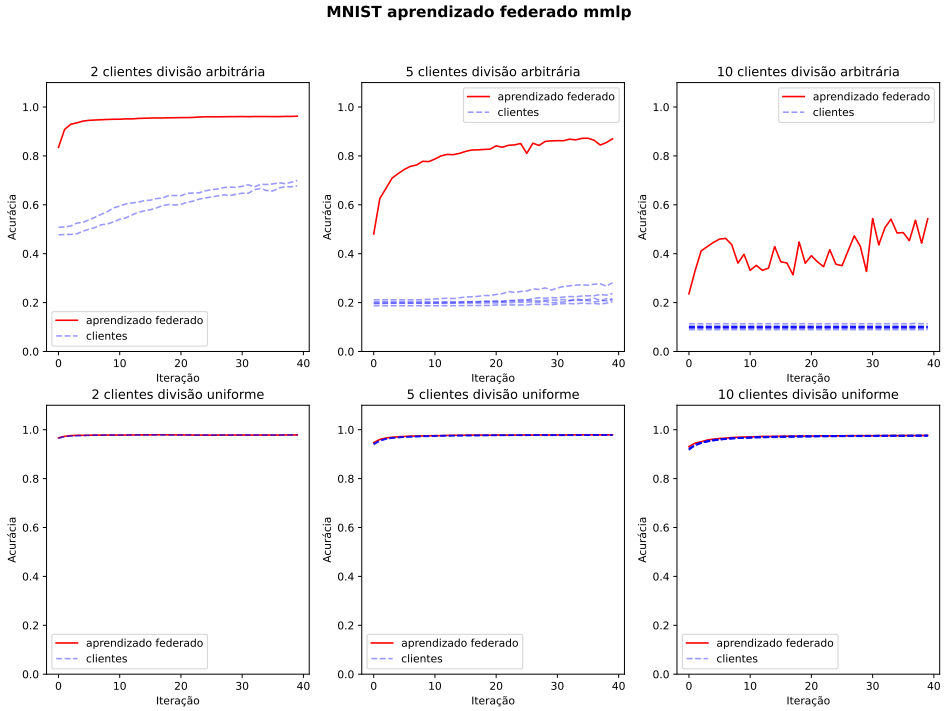

Figura A.5: *Acurácia Fl dos clientes no conjunto MNIST com o modelo mmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

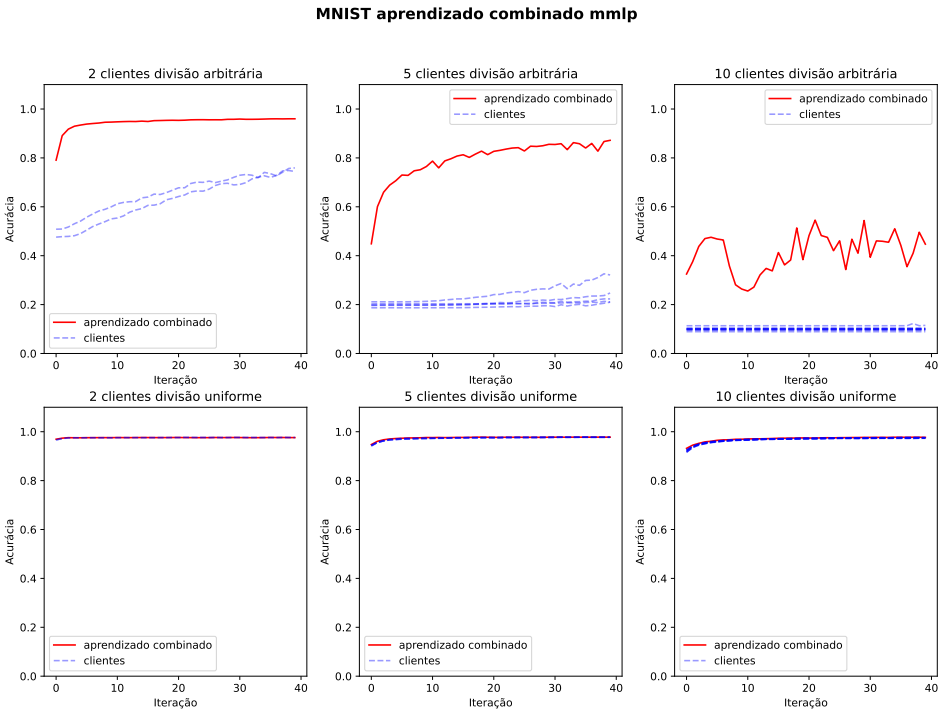

Figura A.6: *Acurácia CL dos clientes no conjunto MNIST com o modelo mmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

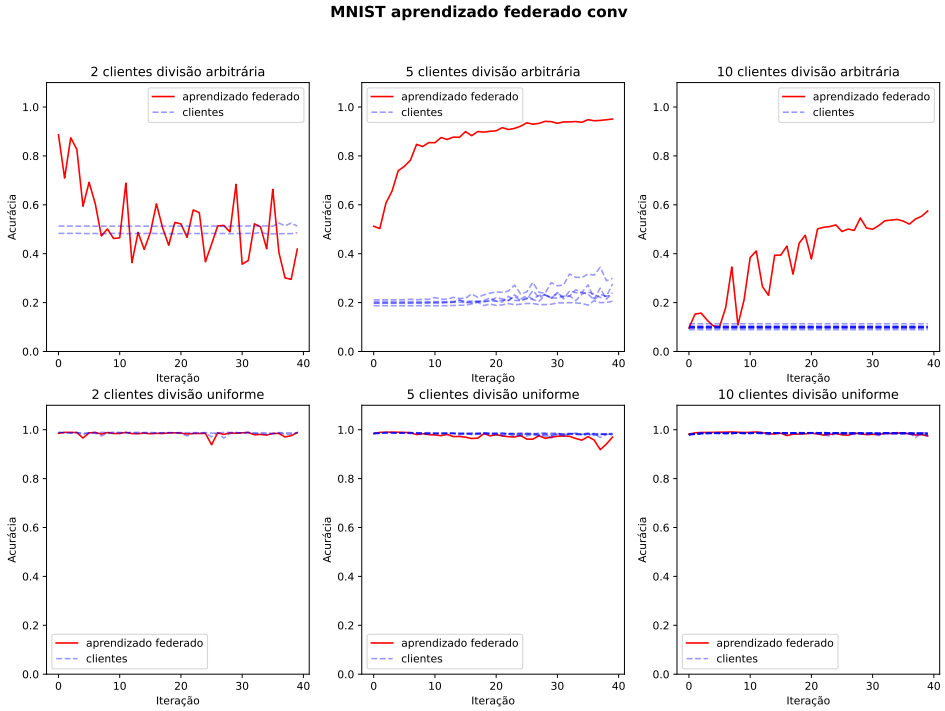

Figura A.7: *Acurácia Fl dos clientes no conjunto MNIST com o modelo conv ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

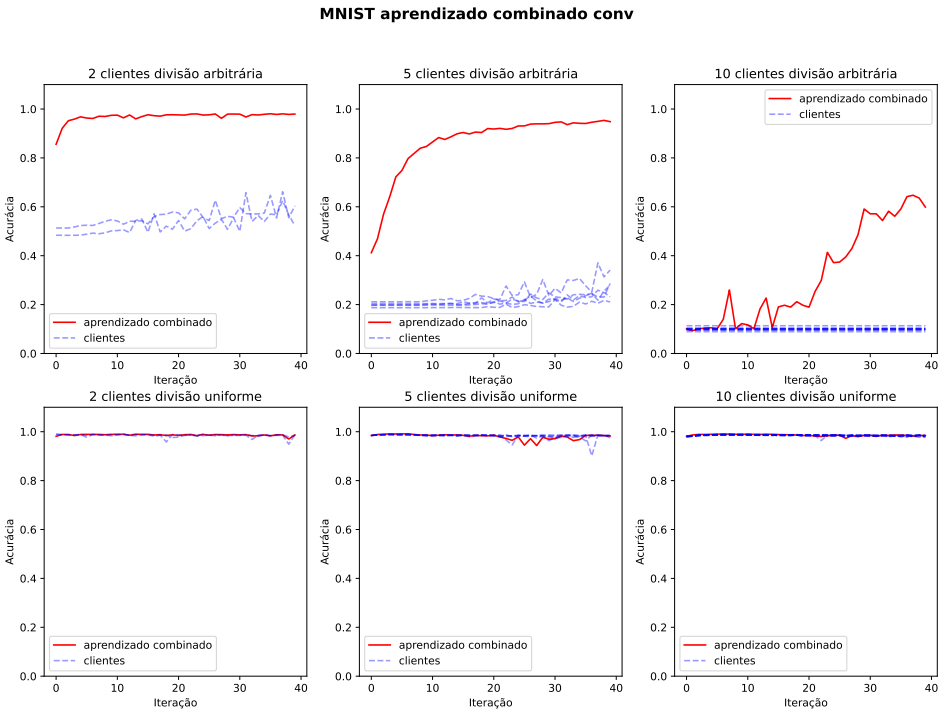

Figura A.8: *Acurácia CL dos clientes no conjunto MNIST com o modelo conv ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

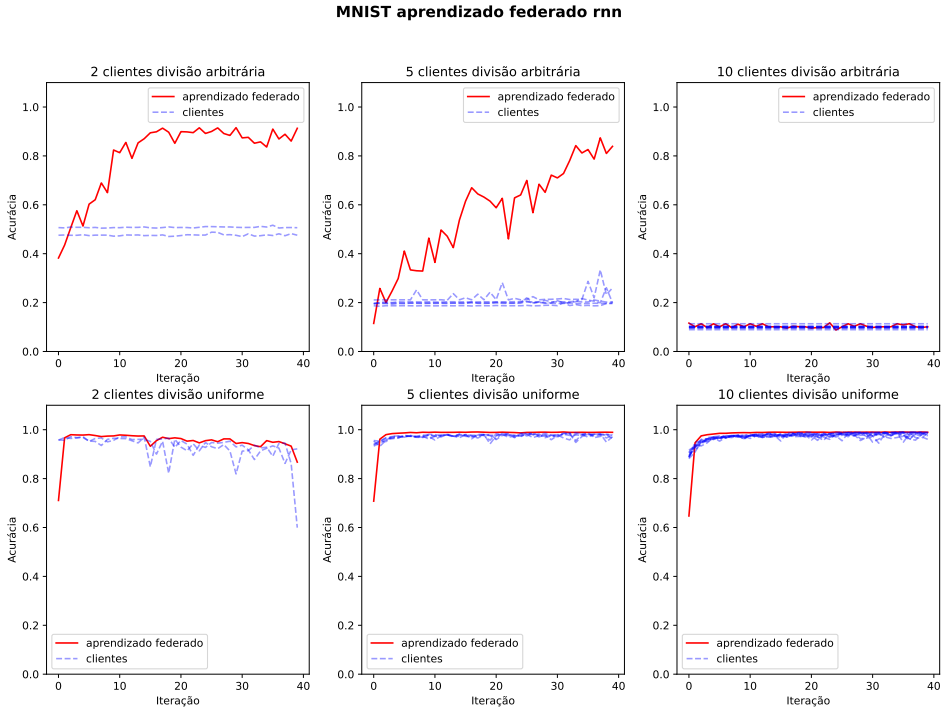

Figura A.9: *Acurácia Fl dos clientes no conjunto MNIST com o modelo rnn ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

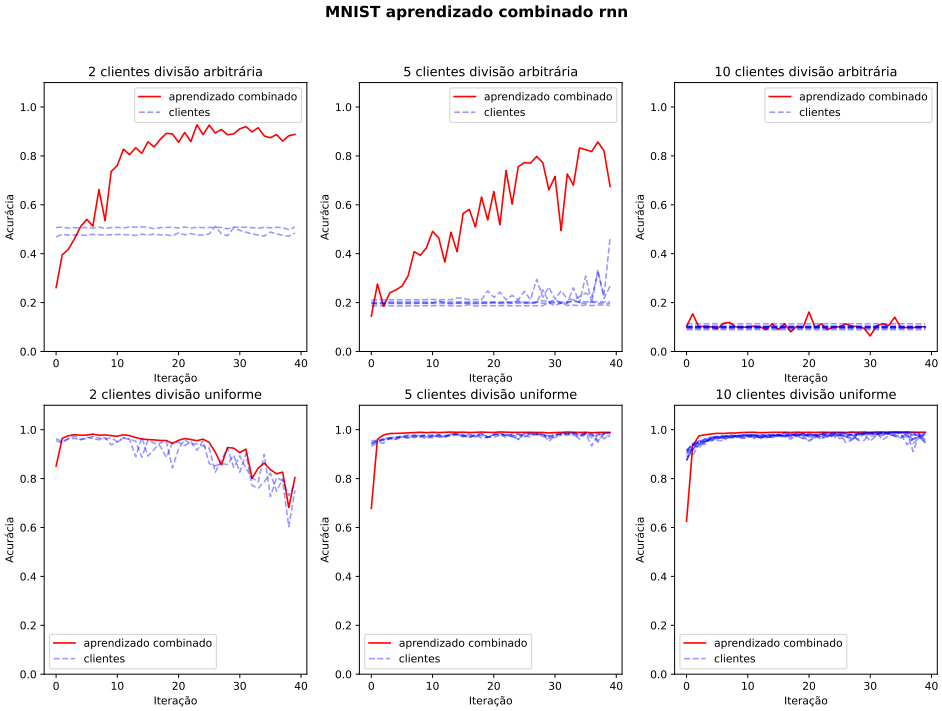

Figura A.10: *Acurácia CL dos clientes no conjunto MNIST com o modelo rnn ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

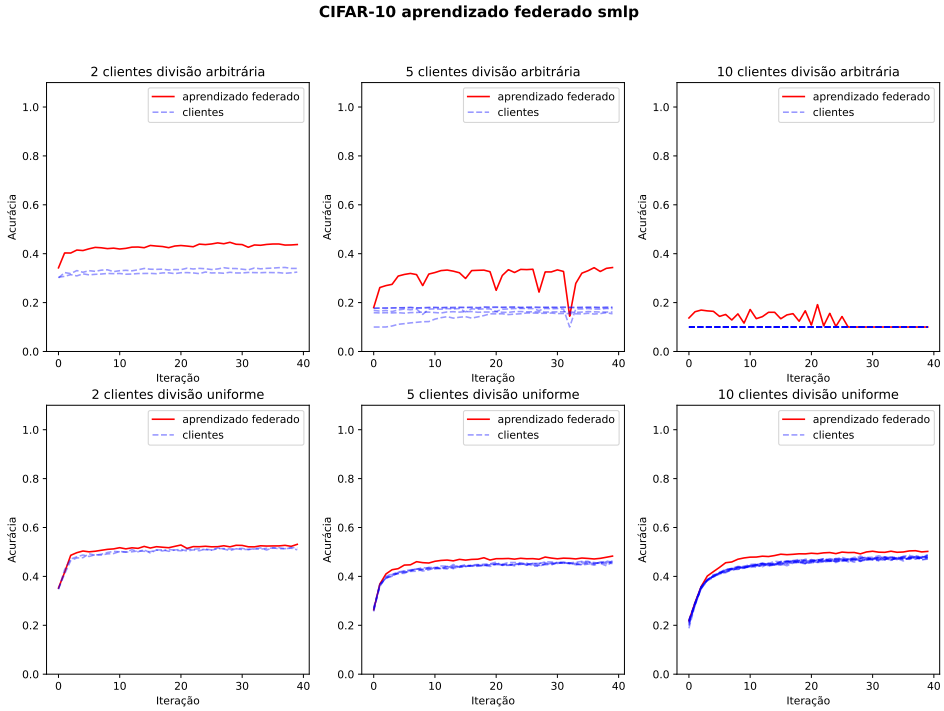

Figura A.11: *Acurácia Fl dos clientes no conjunto CIFAR-10 com o modelo smlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

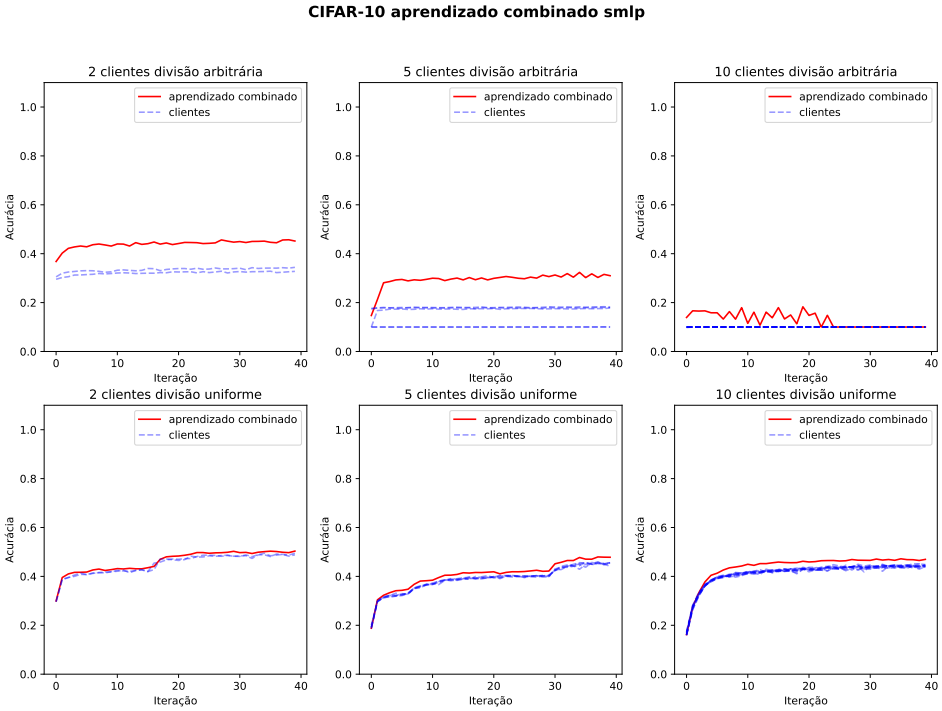

Figura A.12: *Acurácia CL dos clientes no conjunto CIFAR-10 com o modelo smlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

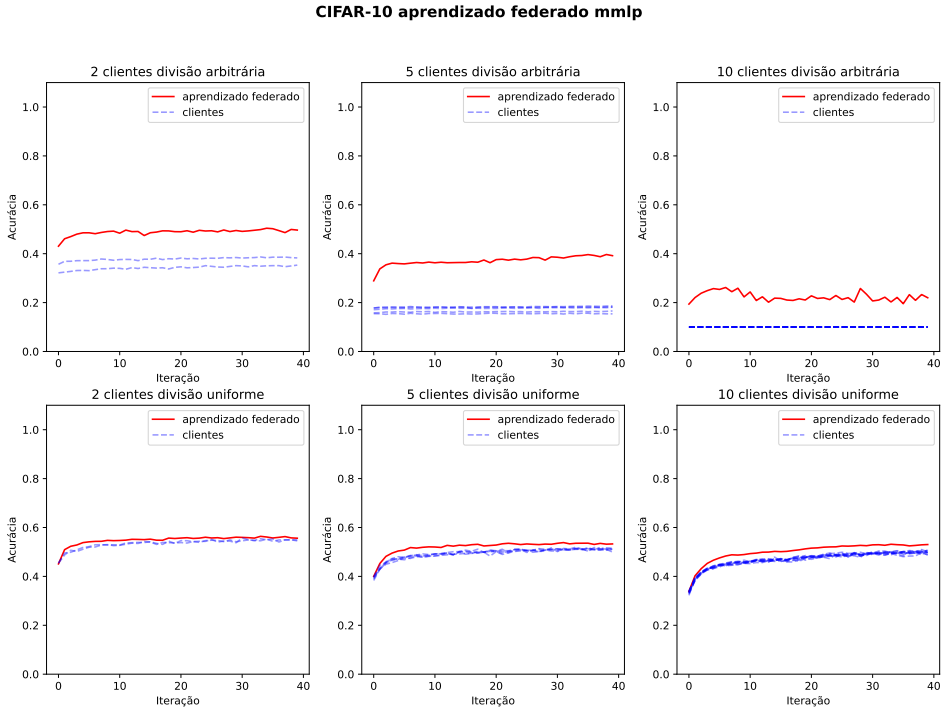

Figura A.13: *Acurácia Fl dos clientes no conjunto CIFAR-10 com o modelo mmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

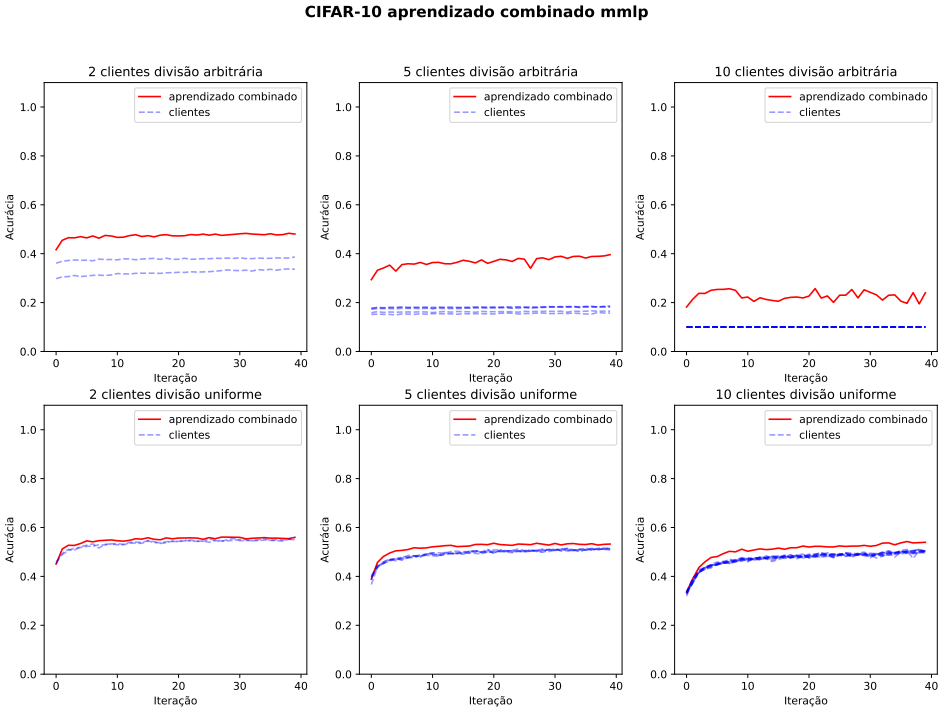

Figura A.14: *Acurácia CL dos clientes no conjunto CIFAR-10 com o modelo mmlp ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

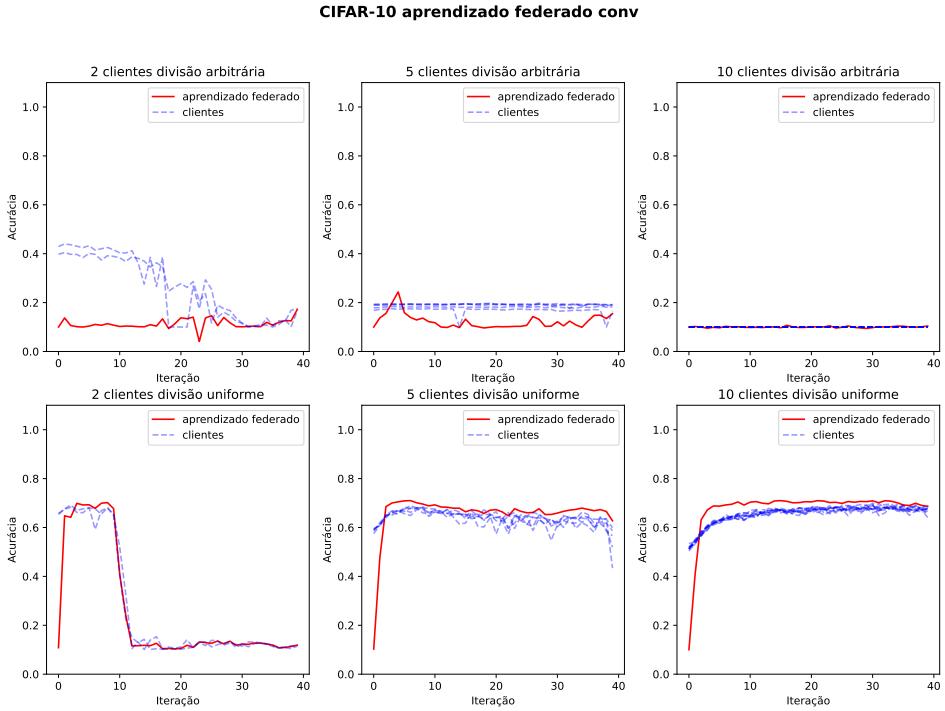

Figura A.15: *Acurácia Fl dos clientes no conjunto CIFAR-10 com o modelo conv ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

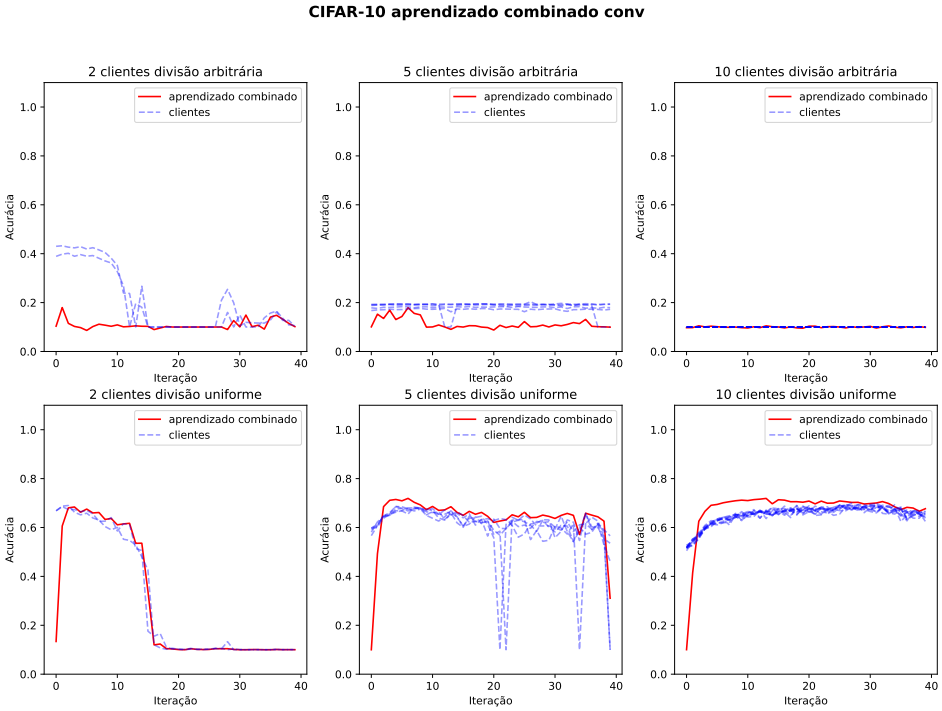

Figura A.16: *Acurácia CL dos clientes no conjunto CIFAR-10 com o modelo conv ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

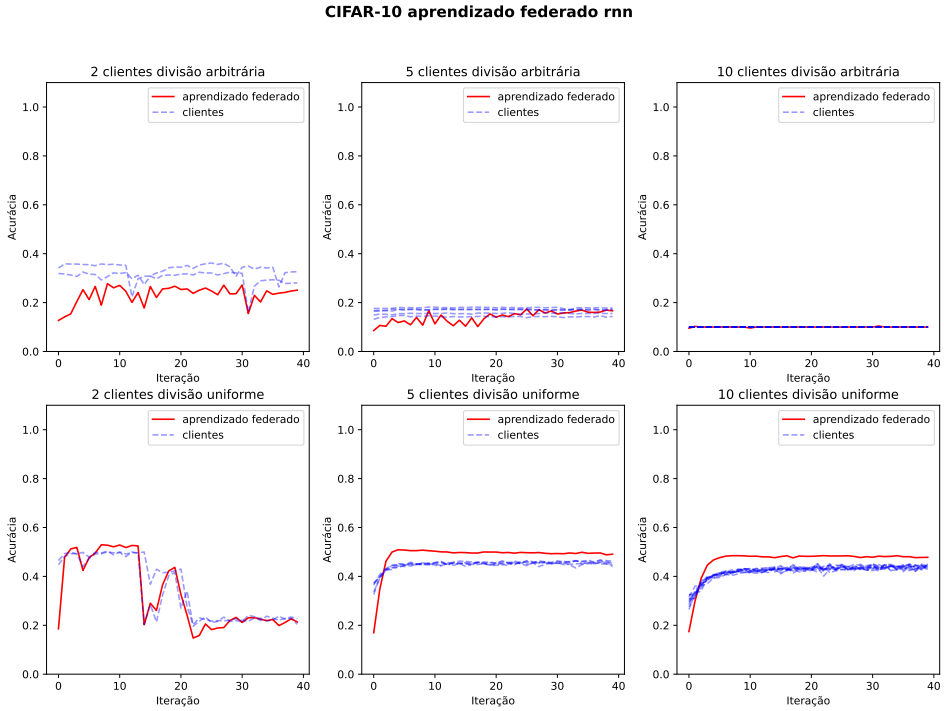

Figura A.17: *Acurácia Fl dos clientes no conjunto CIFAR-10 com o modelo rnn ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

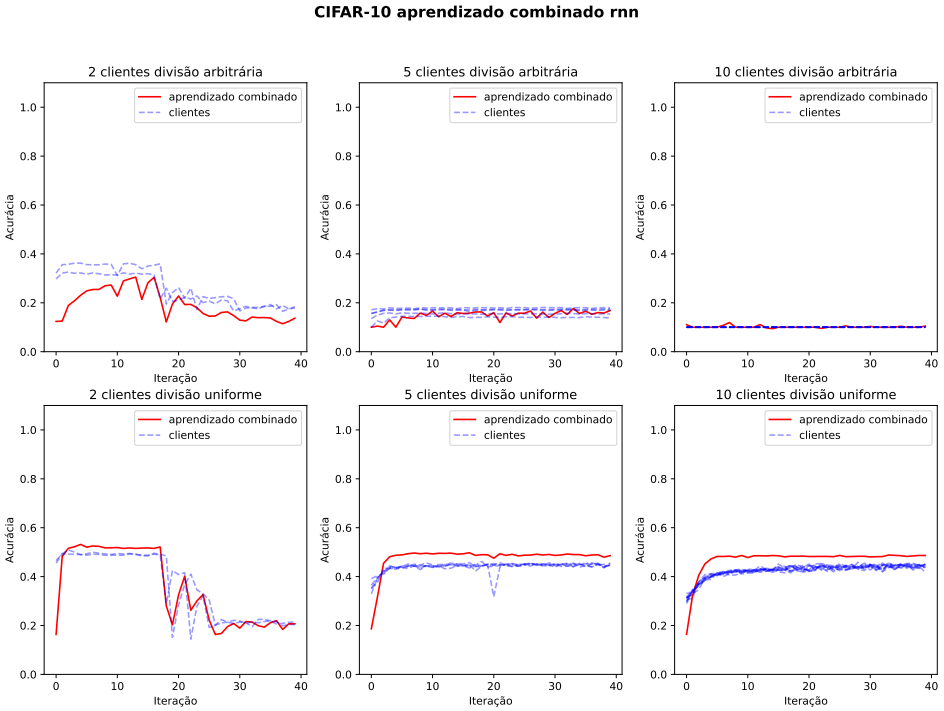

Figura A.18: *Acurácia CL dos clientes no conjunto CIFAR-10 com o modelo rnn ao variar o número de clientes. A cima estão os resultados para a divisão arbitrária, embaixo os resultados para a divisão uniforme.*
Fonte: O Autor

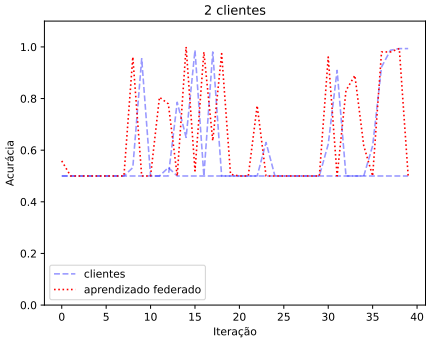

Figura A.19: *Acurácia federado no conjunto Wisconsin Breast Cancer com o modelo mmlp na divisão arbitrária*
Fonte: O Autor

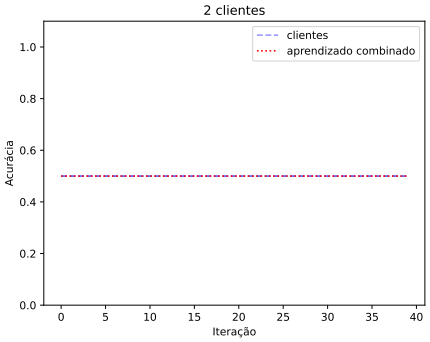

Figura A.20: *Acurácia combinado no conjunto Wisconsin Breast Cancer com o modelo mmlp na divisão arbitrária*
Fonte: O Autor

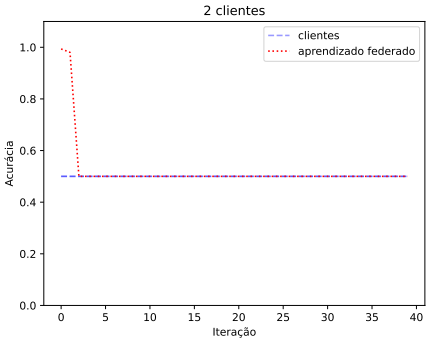

Figura A.21: *Acurácia federado no conjunto Wisconsin Breast Cancer com o modelo lmlp na divisão arbitrária*
Fonte: O Autor

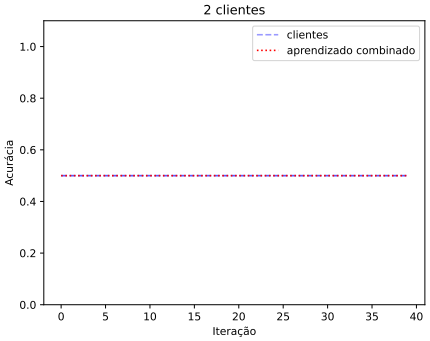

Figura A.22: *Acurácia combinado no conjunto Wisconsin Breast Cancer com o modelo lmlp na divisão arbitrária*
Fonte: O Autor

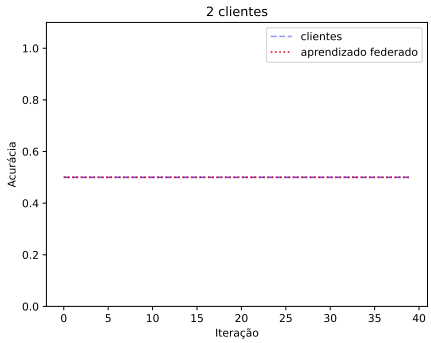

Figura A.23: *Acurácia federado no conjunto Wisconsin Breast Cancer com o modelo smlp na divisão arbitrária*
Fonte: O Autor

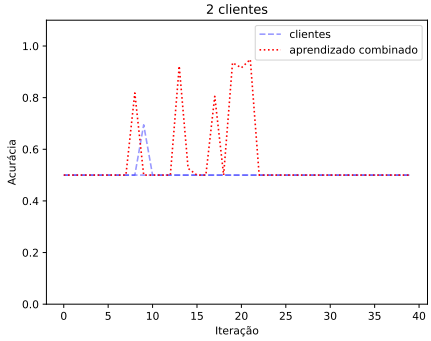

Figura A.24: *Acurácia combinado no conjunto Wisconsin Breast Cancer com o modelo smlp na divisão arbitrária*
Fonte: O Autor

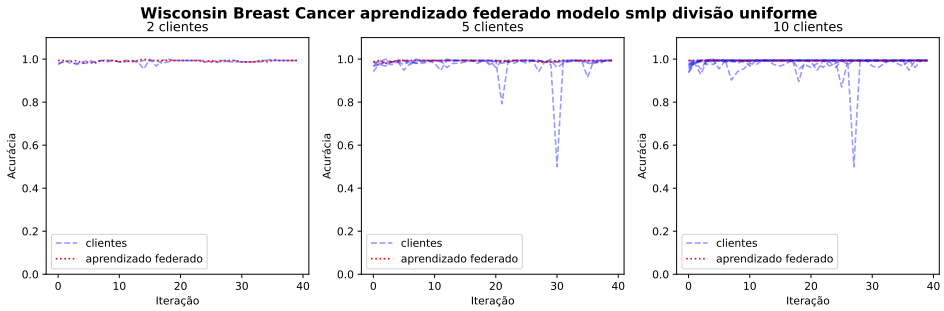

Figura A.25: *Acurácia federado no conjunto Wisconsin Breast Cancer com o modelo smlp na divisão uniforme número de clientes variado*
Fonte: O Autor

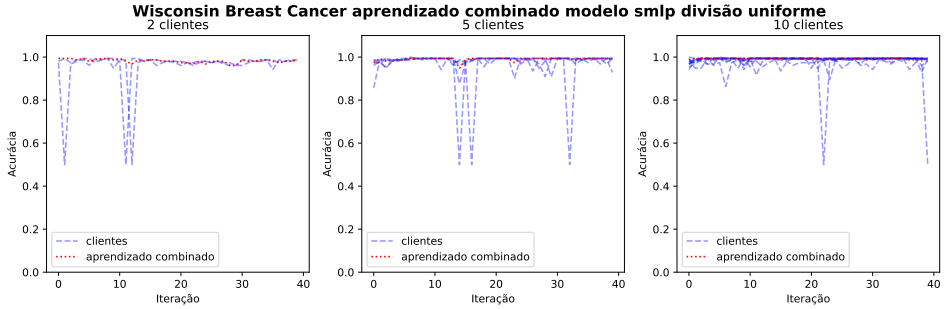

Figura A.26: *Acurácia combinado no conjunto Wisconsin Breast Cancer com o modelo smlp na divisão uniforme número de clientes variado*
Fonte: O Autor

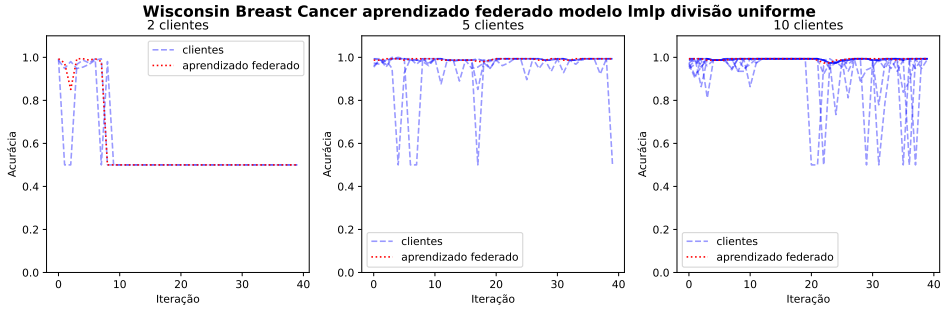

Figura A.27: *Acurácia federado no conjunto Wisconsin Breast Cancer com o modelo lmlp na divisão uniforme número de clientes variado*
Fonte: O Autor

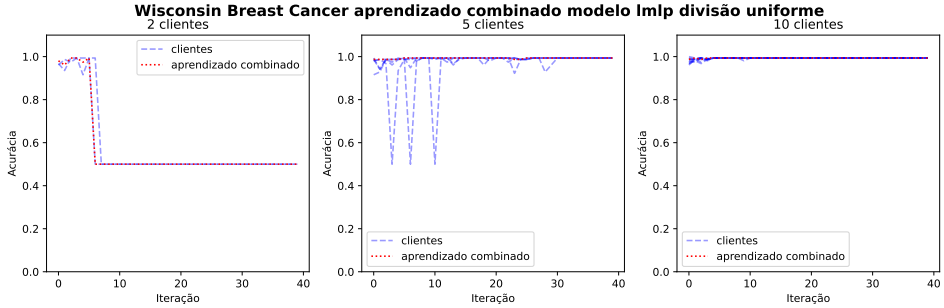

Figura A.28: *Acurácia combinado no conjunto Wisconsin Breast Cancer com o modelo lmlp na divisão uniforme número de clientes variado*
Fonte: O Autor

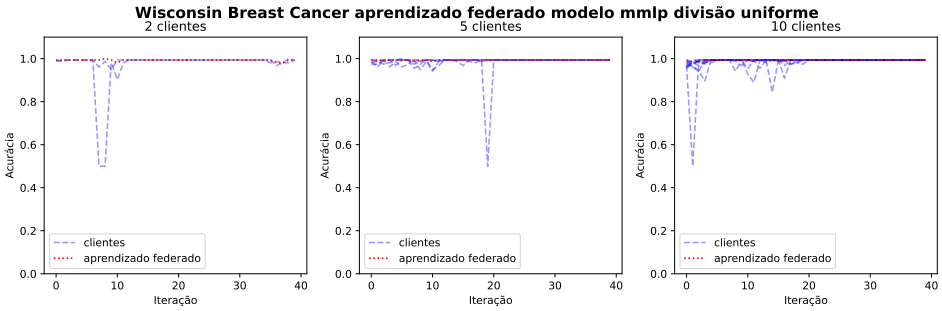

Figura A.29: *Acurácia federado no conjunto Wisconsin Breast Cancer com o modelo mmlp na divisão uniforme número de clientes variado*
Fonte: O Autor

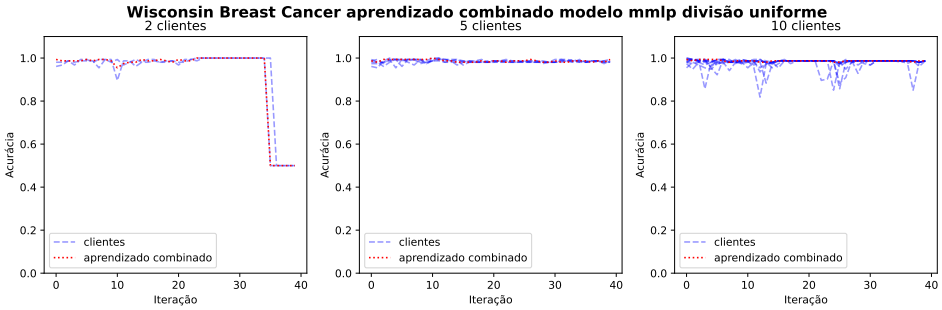

Figura A.30: *Acurácia combinado no conjunto Wisconsin Breast Cancer com o modelo mmlp na divisão uniforme número de clientes variado*
Fonte: O Autor

Apêndice B

Programas

Este apêndice traz uma ilustração dos programas escritos para implementar o algoritmo do aprendizado combinado. Eles são discutidos na seção 2.3. A versão completa, com detalhes de implementação específicos para o arcabouço escolhido, *Pytorch*, está disponível no *GitHub*[1] do projeto.

Programa B.1 Classe Cliente (versão ilustrativa).

```
1   class Cliente:
2      def __init__(self, rede_neural, conjunto_treino, conjunto_teste, r):
3
4         self.rede_neural = rede_neural
5         self.conjunto_treino = conjunto_treino
6         self.conjunto_teste = conjunto_teste
7         self.r = r
8
9
10     def get_camadas(self):
11        # Retorna os parâmetros camada por camada da rede_neural do cliente.
12        return self.rede_neural.camadas
13
14     def set_camadas(self, camadas_combinadas):
15        # Atualiza as camadas da rede_neural do cliente.
16        self.rede_neural.atualiza_camadas(camadas_combinadas)
17
18     def treine(self):
19        # Treine o cliente em seu conjunto de dados de treinamento.
20        self.rede_neural.treine(self.conjunto_treino)
21
22     def teste(self, conjunto_teste):
23        # Avalia o desempenho do cliente no conjunto_teste.
24        return self.rede_neural.teste(self.conjunto_teste)
```

[1] https://github.com/melloguel/tcc-coln

Programa B.2 Programa para treinamento distribuído.

```
1   def combine(clientes, conv, steps):
2      '''
3      Executa [steps] times.
4
5      clientes: lista de clientes que participarão do aprendizado combinado
6      conv: parâmetro usado para combinar os pesos de uma camada
7      steps: número de combinações a serem realizadas
8
9      Passos do algoritmo de aprendizado combinado
10     1) Treinar o modelo de cada cliente em seu respectivo conjunto de dados
11     2) Calcular os pesos finais a partir dos pesos apreendidos por cada
           cliente
12     3) Atualizar os pesos do modelo de cada cliente para serem iguais aos
           pesos calculados
13     '''
14
15     # 0) Énecessário garantir que cada modelo inicie no mesmo ponto
16     # Escolhe aleatoriamente um cliente para ter um ponto de partida igual
17     camadas = choice(clientes).get_camadas()
18     for cliente in clientes:
19        model.set_camadas(camadas)
20
21     for _ in range(steps):
22        # 1) Treine cada cliente
23        camadas = []
24        for cliente in clientes:
25           cliente.treine()
26           camadas.append(model.get_camadas())
27
28        # 2) Combine os pesos
29        r = [cliente.r for cliente in clientes]
30        camadas_combinadas = combine_cl(camadas, conv, r)
31
32        # 3) Atualize os pesos de cada cliente com os novos
33        for cliente in clientes:
34           model.set_camadas(camadas_combinadas)
```

Programa B.3 Função de combinação do aprendizado combinado.

```
1   def combine_cl(camadas, conv, r):
2       '''
3       Combina os pesos em cada camada para computar os novos pesos
4       camadas: lista com as camadas de cada modelo
5       conv: parâmetro para combinar os pesos
6       '''
7       # Agrupe por camada ao invés de por cliente
8       camadas = [[camada[i] for camada in camadas] for i in range(len(camadas[0])
            )]
9       camadas_combinadas = []
10      for camada in camadas:
11          camadac = combine_camadas(camada, conv, r)
12          camadas_combinadas.append(camadac)
13      return camadas_combinadas
14
15  def combine_camadas(camadas, conv, r):
16      '''
17      Combina camadas em uma nova camada
18      camadas: lista de camadas
19      conv: parâmetro para combinar os pesos
20      '''
21      def weighted_sum(x_values, w_values):
22          return sum(map(prod, zip(x_values, w_values)))
23
24      def mult_l2_diff(tensors, sum_result):
25          n = len(tensors)
26          ret = 0.0
27          for i in range(n):
28              for j in range(i+1, n):
29                  ret += (tensors[i] - tensors[j]).pow(2)
30
31          ret = ret.sum() if sum_result else ret
32          ret = ret.sqrt()
33          return ret
34
35      def mult_euclidean_distance(tensors):
36          return mult_l2_diff(tensors, True)
37
38      def mult_abs_diff(tensors):
39          return mult_l2_diff(tensors, False)
40
41      threshold = mult_euclidean_distance(camadas) / prod(camadas[0].shape)
42      n4_1 = mult_abs_diff(camadas)
43      n4_sum = weighted_sum(camadas, [e**(conv * rh) for rh in r])
44
45      le_threshold = n4_1 <= threshold
46      gt_threshold = n4_1 > threshold
47
48      newcamada = weighted_sum([le_threshold, gt_threshold], [n4_sum + n4_1,
            n4_sum])
49
50      return newcamada
```

Referências

[Hsieh *et al.* 2020] Kevin Hsieh, Amar Phanishayee, Onur Mutlu e Phillip B. Gibbons. "The non-iid data quagmire of decentralized machine learning". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020 (citado na pg. 5).

[Ioste A. 2022] Finger M. Ioste A. "Establishing the parameters of a decentralized neural machine learning model". *A ser publicado* (2022) (citado nas pgs. 1, 3–5, 10).

[Kairouz *et al.* 2021] Peter Kairouz *et al.* "Advances and open problems in federated learning". *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210 (citado nas pgs. 3–5).

[B. McMahan *et al.* 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson e Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282 (citado na pg. 5).