Uma Estratégia Hierárquica para Sistemas de Múltiplos Classificadores Distribuídos

Charles Giovane de Salles, André Luiz Brun e Luiz Antonio Rodrigues

[1]Programa de Pós-Graduação em Ciência da Computação (PPGComp)
Universidade Estadual do Oeste do Paraná (Unioeste)
Cascavel – PR – Brasil

charles_giovane@hotmail.com, andre.brun@unioeste.br

Abstract. *The natural distribution of data as well as privacy and security issues justify the need for efficient systems for handling distributed data. On the other hand, data scarcity is a challenge for machine learning that can be mitigated with a distributed approach. In this work, a hierarchical virtual topology based on hypercubes is used to organize the sharing and grouping of distributed training results, increase the accuracy of the final models, and provide a fault-tolerant solution. Experimental results confirm the effectiveness of the technique, with improved results in all simulated scenarios.*

Resumo. *A distribuição natural dos dados e questões de privacidade e segurança justificam a necessidade de sistemas eficientes para lidar com dados distribuídos. Por outro lado, a escassez de dados é um desafio para o aprendizado de máquina, o que pode ser mitigado com uma abordagem distribuída. Este trabalho utiliza uma topologia virtual hierárquica baseada em hipercubos para organizar a troca e agrupamento de resultados de treinamento distribuídos, aumentando a acurácia dos modelos finais e oferecendo uma solução tolerante a falhas. Resultados experimentais confirmam a eficácia da técnica, com melhoria dos resultados em todos os cenários simulados.*

1. Introdução

O crescimento no volume de dados tanto na pesquisa quanto na indústria, apresenta grandes oportunidades, mas também grandes desafios computacionais. De acordo com [Zaharia et al. 2016], a quantidade de dados gerados diariamente ultrapassou os recursos de uma única máquina e os usuários precisam de novos sistemas para dimensionar os dados para outros nós. Segundo [Chen 2018], os dados já são naturalmente distribuídos, muitas vezes por questões de privacidade ou segurança, pois se uma coleta central das informações for necessária existe a possibilidade de vazamento de informações neste processo. Neste sentido, [Lu et al. 2020] complementam que a coleta de grande quantidade de informação de fontes de dados distribuídas, apresentam várias dificuldades incluindo limitação na conectividade de rede e largura de banda, limitação no consumo de energia e a necessidade de preservar a privacidade dos dados brutos.

Outro fator importante é que um gargalo que geralmente impede o desenvolvimento de sistemas mais inteligentes é a quantidade limitada de dados. Sem dados suficientes, os sistemas encontram dificuldades para tomar decisões inteligentes como os humanos [Guo and Zhang 2016]. Em seu trabalho,

[Peteiro-Barral and Guijarro-Berdiñas 2013] afirmam que, tradicionalmente, um gargalo que impedia o desenvolvimento de sistemas mais robustos era a quantidade limitada de dados disponíveis. Nesta situação, desafios são levantados em relação à escalabilidade e eficiência dos algoritmos de aprendizagem no que diz respeito aos recursos computacionais e a maioria das implementações existentes de algoritmos operam com o conjunto de treinamento inteiramente na memória principal.

Este cenário foi sendo redefinido com o crescimento das tecnologias de detecção e coleta de dados desenvolvidas e adotadas nos últimos anos. Porém, muitos dos algoritmos tradicionais de aprendizado de máquina exigem o carregamento de todo o conjunto de dados em um computador, o que torna essa etapa quase impossível, pois o volume de dados é impeditivo [Guo and Zhang 2016]. Dados tais desafios, uma abordagem para ser utilizada nestes casos é o aprendizado de máquina distribuído. [Tsoumakas and Vlahavas 2009] apresenta algumas vantagens em manter os dados distribuídos ao invés de agrupá-los para um processo de classificação. Primeiro, o custo de armazenamento de um conjunto de dados central é muito maior do que a soma do custo de armazenamento de partes menores do conjunto de dados. Da mesma forma, o custo computacional de mineração de um banco de dados central é muito maior do que a soma do custo de análise de partes menores dos dados. Além disso, a transferência de grandes volumes de dados pela rede pode levar muito tempo e também requer um custo financeiro alto. Por fim, mas não menos importante, os dados podem ser privados ou confidenciais, como os registros médicos e financeiros das pessoas.

Como a demanda por processamento de dados de treinamento dos algoritmos de classificação ultrapassou o poder de computação das máquinas, é necessário distribuir a carga de trabalho de aprendizado realizado por uma máquina de forma local para várias máquinas e transformar o sistema centralizado em um sistema distribuído [Verbraeken et al. 2020]. Segundo [Chen 2018], o objetivo do aprendizado distribuído é produzir um resultado de aprendizado com base em todos os conjuntos de dados. Os resultados do aprendizado local precisam ser combinados de alguma forma, pois na aprendizagem distribuída não existe um classificador global para receber a mensagem de todos os classificadores locais nem a transmissão de dados para um ponto central da rede. Para isso, geralmente, existem dois métodos diferentes que podem ser utilizados. O primeiro consiste em mesclar as previsões de classificadores locais. A segunda estratégia dá-se pela combinação dos modelos de classificação.

Nas áreas da ciência, finanças e medicina, os analistas lidam frequentemente com a tarefa de classificar itens com base em dados históricos ou mensuráveis. Um classificador tem como função principal atribuir a um objeto um determinado rótulo [Schmeing et al. 2022]. Esse processo ocorre ao analisar o conjunto de características de um elemento e assim definir o grupo a que ele pertence. De acordo com autor, um classificador pode não ter um bom desempenho em cenários mais complexos. Nesse sentido, [Kittler et al. 1998] sugere que diferentes classificadores podem oferecer informações complementares sobre um padrão a ser classificado podendo assim melhorar o desempenho do classificador em questão. Ao associar vários classificadores, espera-se obter melhor desempenho [Gunes et al. 2003]. No entanto, ao utilizar um sistema de múltiplos classificadores é necessário definir uma estratégia de unir as opiniões dos classificadores utilizados. Neste sentido, [Kittler et al. 1998] demostra algumas abordagens, nas quais

é possível combinar diferentes classificadores, como a regra do produto, regra da soma, regra do mínimo, regra do máximo, regra da média e voto majoritário.

Neste ponto surge um questionamento: como proceder quando uma grande quantidade de dados for utilizada para classificação e reconhecimento de padrões? De acordo com [Mu 2014], a migração e replicação de dados pela rede está sujeita a gargalos ao lidar com um volume de dados muito grande. Desta maneira, seria indesejável movimentar todos os dados por meio da rede até um ponto central para serem utilizados para treinamento ou classificação [Lu et al. 2020]. Além disso, há casos em que uma organização não quer e nem pode disponibilizar as informações para outros nós. Nestes cenários, apenas o modelo de classificação pode ser compartilhado.

Assim, este artigo apresenta uma estratégia hierárquica tolerante a falhas para treinamento de um conjunto de classificadores distribuídos sem que haja necessidade de compartilhamento de dados. Após o treinamento local, os modelos de classificação são compartilhados com os demais nós de forma que todos os elementos presentes na rede tenham o conhecimento sobre os dados representados nos modelos individuais. A comunicação entre os nós da rede segue uma topologia virtual hierárquica baseada em hipercubo denominada vCube [Duarte et al. 2014], que apresenta importantes propriedades logarítmicas. Cada nó no hipercubo se comunica com um conjunto de vizinhos e, em casos de falhas, a topologia se ajusta automaticamente para recompor a árvore de comunicação do sistema.

O restante do artigo está organizado nas seguintes seções. A Seção **??** apresenta os conceitos fundamentais de classificação distribuída e trabalhos relacionados são apresentados na Seção 3. A Seção 2 descreve a solução de classificação distribuída e a lógica de propagação dos modelos locais. Os resultados e discussões são apresentados na Seção 5. A conclusão e trabalhos futuros estão na Seção 6.

2. Classificação Distribuída

Os problemas de classificação consistem em analisar um conjunto de entradas e decidir a qual das N classes disponíveis elas pertencem, baseado no treinamento dos exemplares de cada classe. Uma característica importante sobre o problema de classificação é que ele é discreto, ou seja, cada exemplar pertence a uma classe precisamente e que o conjunto de classe cobre todo o espaço de entradas possíveis [Marsland 2014]. No campo de Aprendizado de Máquina, a utilização de classificadores tem como objetivo encontrar um padrão nas informações analisadas. Desta maneira, o classificador tem como função principal atribuir a um objeto um determinado rótulo segundo [Schmeing et al. 2022].

Para ilustrar o processo de classificação, considere a configuração de um classificador de moedas. Classificá-las apenas pela sua cor seria uma tarefa muito difícil, pois alguns tipos de moedas só possuem como cor uma tonalidade de prata. Quando a moeda é colocada na máquina, algumas informações sobre ela são capturadas, como o diâmetro, peso e forma. Essas características formam o vetor de entrada do classificador, chamado vetor de características. Neste caso, um vetor com três elementos, no qual cada um será representado por um valor, inclusive o formato (neste caso podemos escolher um número para representar determinada forma, por exemplo, 1 = círculo, 2 = hexágono e assim por diante) [Marsland 2014].

Os algoritmos de classificação são diferentes na forma em que eles aprendem a solução. Entretanto, na essência, todos realizam a mesma tarefa que consiste em en-

contrar uma regra que consegue separar diferentes classes [Leban et al. 2006]. Dadas as características usadas como entradas para os classificadores, precisa-se identificar alguns valores dentre aquelas características que serão capazes de dizer qual classe aquela entrada pertence. Uma grande variedade de métodos foi desenvolvida para a aprendizagem preditiva a partir de informações e, para cada método em particular, existem situações em que ele é mais adequado e cenários em que o mesmo algoritmo pode ter um desempenho ruim [Hastie et al. 2009].

Os Sistemas de Múltiplos Classificadores (SMC) visam aumentar a eficiência da classificação dos classificadores "fracos", ou seja, aqueles que apresentam uma taxa de precisão baixa, considerando que classificadores diferentes geralmente cometem erros diferentes em amostras diferentes. Isso significa que a combinação de classificadores gera uma decisão mais precisa [Ko et al. 2008].

A Figura 1 mostra o esquema de funcionamento de um sistema de múltiplos classificadores e suas fases. Na primeira, o conjunto de classificadores é gerado. Na segunda fase, um subconjunto destes classificadores é selecionado. Na última fase, a decisão é feita baseada na predição dos classificadores selecionados [Britto Jr et al. 2014].

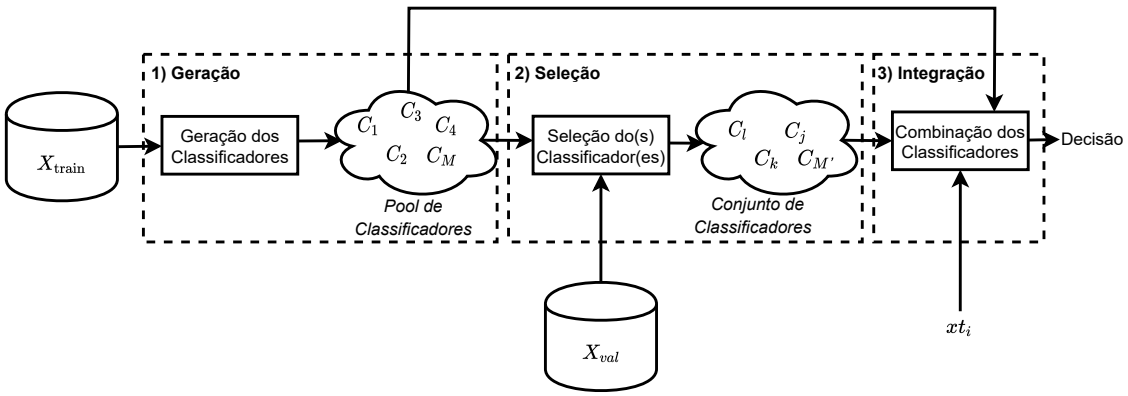


Figura 1. Sistema de Múltiplos Classificadores (adap. de [Britto Jr et al. 2014])

No processo de aprendizado distribuído é necessário definir que tipo de informação será compartilhada entre os nós do sistema uma vez que os dados não serão repassados pela rede. Duas abordagens principais são propostas por [Lu et al. 2020]: a) construir modelos globais a partir de modelos individuais derivados de dados locais; e b) agregar globalmente as saídas de modelos locais.

A primeira abordagem basicamente se resume em compartilhar os modelos de classificação gerados por cada nó da rede de forma individual. No entanto, os algoritmos de aprendizagem levam em conta os atributos locais para formar uma opinião e isto seria uma desvantagem deste método. Além disso, se o tipo de técnica de aprendizagem aplicada em um nó da rede for diferente da empregada em outro, os algoritmos a serem combinados podem ter diferentes representações e, ao combinar os classificadores gerados podem gerar divergências. Desta maneira, seria preciso definir uma representação uniforme na qual classificadores diferentes seriam traduzidos. Porém, não é uma tarefa fácil definir uma representação para encapsular todas as outras representações sem perder uma quantidade relevante de informações. Além disso, uma consequência provável e indesejável dessa tradução seria principalmente a restrição das informações suportadas

pelo classificador [Peteiro-Barral and Guijarro-Berdiñas 2013].

Na segunda abordagem, ao invés de combinar seus modelos, são combinadas as saídas dos classificadores, ou seja, suas opiniões. Desta forma, não há necessidade de traduzir um modelo de classificação, pois sua organização interna não é levada em consideração, já que a informação compartilhada é baseada nas previsões que são as saídas dos classificadores para um determinado conjunto de treinamento. Nesta estratégia, a padronização diz respeito às previsões visto que estas podem ser categóricas ou numéricas. No entanto, a complexidade de definir uma estrutura uniforme para combinar as saídas é muito menor do que a de combinar classificadores [Peteiro-Barral and Guijarro-Berdiñas 2013]. Dentre as estratégias de combinação, destacam-se as regras do produto, da soma, do máximo e do mínimo, da média e do voto majoritário [Kittler et al. 1998].

Ao realizar a predição de uma determinada instância, o classificador determina um certo grau de certeza sobre sua opinião, esta probabilidade pode ser usada na etapa de combinação de vários classificadores. Nas regras da soma e da média, na fase de predição, as probabilidades obtidas pelos modelos individuais para uma mesma instância são somadas ou agregadas de acordo com pesos. Tais estratégias são indicadas quando se quer reduzir a variância do modelo, especialmente em modelos que sofrem com *overfitting*.

Na regra do produto, cada modelo é ajustado aos erros residuais dos modelos anteriores, e as previsões são combinadas multiplicando-se as probabilidades de cada classificador. É eficaz quando se deseja aumentar a capacidade preditiva do modelo, principalmente em modelos que sofrem de *underfitting*. As regras do máximo e mínimo são utilizadas principalmente em métodos como *Random Forest*. As previsões de cada árvore são usadas individualmente e o resultado é determinado pelo máximo ou mínimo das previsões, dependendo do contexto. É útil quando se deseja capturar a variabilidade dos modelos e evitar o *overfitting*, já que a diversidade das árvores pode ajudar a melhorar a robustez do modelo.

Por fim, no método voto majoritário as previsões são combinadas por meio de votação, e a classe mais frequente é selecionada como a previsão final. É eficaz em problemas de classificação binária ou multiclasse, onde a decisão final pode ser determinada pela maioria das previsões individuais, ajudando a reduzir o efeito de ruídos nos dados e melhorar a robustez do modelo.

Diversas topologias podem ser utilizadas para organizar o processo de aprendizado distribuído, incluindo os modelos centralizado, baseado em árvore, servidor de parâmetros (em nuvem) e totalmente distribuído (*peer-to-peer*) [Verbraeken et al. 2020]. Os sistemas de classificação centralizados empregam uma abordagem estritamente hierárquica para agregação, que ocorre em um único local central. Desta maneira, em cada nó é executado um algoritmo de aprendizado sobre os dados locais e, em seguida, o modelo obtido é encaminhado a um nó central para a realização da combinação destes algoritmos, resultando no modelo de classificação final. Após a obtenção do modelo central este deve, de alguma forma, ser compartilhado aos demais nós da rede para que eles possam realizar a classificação localmente, sem a necessidade de enviar os dados de novos registros através da rede até o nó central, aumentando o tráfego na rede e expondo dados que podem ser sigilosos.

A topologia em árvore tem a vantagem de ser fácil de escalar e gerenciar, pois cada nó só precisa se comunicar com seu nó pai e nós filhos. A desvantagem desta topologia é que se um nó pai cair se perde todo o treinamento realizados pelos nós filhos. Cada nó realiza o aprendizado localmente, com base nas informações que possui e o modelo obtido é então enviado ao nó pai, que é responsável por repassar os modelos enviados pelos seus nós filhos e os encaminha ao nível hierárquico superior. A raiz da estrutura é então responsável por combinar os modelos e repassá-lo aos elementos presentes na rede.

A estratégia com servidor de parâmetros usa um conjunto de nós descentralizados de aprendizado de máquina, onde as informações estão armazenadas (nós folhas) juntamente com um conjunto nós centralizados mestres (em nuvem) que mantém todo o compartilhamento do sistema. Desta maneira, todos os parâmetros do modelo ficam armazenados de forma dividida entre os servidores da nuvem [Verbraeken et al. 2020]. Uma vantagem é que todos os parâmetros do modelo estão em uma memória compartilhada global, o que facilita a inspeção do modelo. No entanto, como a este conjunto de nós mestres acabam se comunicando com toda a rede, é possível haver um gargalo de comunicação, dependendo do tamanho do conjunto de nós folhas. Pois quanto maior for o número de nós folhas, a tendência é que o desempenho se torne parecido com a abordagem centralizada.

Por fim, a topologia *peer-to-peer* demostra uma abordagem contrária a um modelo centralizado, pois cada nó tem sua própria cópia dos parâmetros e os nós trabalhadores se comunicam diretamente uns com os outros. Isso oferece a vantagem de escalabilidade tipicamente maior do que um modelo centralizado e a eliminação de pontos únicos de falha no sistema.

3. Trabalhos Relacionados

Hu et al. (2020) apresentam um modelo de sincronização de parâmetros para aprendizado de máquina distribuído com sistemas heterogêneos na borda. Nesta sincronização dos parâmetros, o tempo de espera é minimizado significativamente. A ideia deste modelo é que os dispositivos de borda mais rápidos continuem o processo de treinamento, enquanto enviam as atualizações dos seus modelos em intervalos estratégicos. Segundo os autores, este modelo supera significativamente os modelos de sincronização de parâmetros comparados.

Em [Hegedűs et al. 2021], uma estratégia de *gossip* foi comparada com uma abordagem centralizada de aprendizado distribuído, na qual o mestre envia o modelo aos trabalhadores e coleta as respostas. Os resultados mostram que a abordagem distribuída teve desempenho comparável ao modelo centralizado, sendo uma solução mais escalável e tolerante a falhas.

Alpcan e Bauckhage (2019) apresentam um *framework* de aprendizagem distribuída para o classificador SVM, onde o problema de classificação é decomposto em múltiplos subproblemas. Em seguida, cada um deles são processados por unidades individuais, como computadores separados ou núcleos de processador, em paralelo e com acesso a apenas um subconjunto dos dados. Assim, uma atualização paralela síncrona converge para a solução aproximada geometricamente.

Wang et al. (2019) propõe um *framework* de aprendizado distribuído assíncrono baseado em uma arquitetura sem servidor chamado *SIREN*, onde as funções são exe-

cutadas na nuvem. Entre as vantagens da ferramenta são o paralelismo e elasticidade. Além disso, autores propõe ainda um escalonador que aprende com o próprio processo de treinamento que busca o menor tempo possível para este treinamento. A partir de testes experimentais os autores chegaram à conclusão que utilizando o SIREN foi possível reduzir o tempo de treinamento em até 44%.

Gupta et al. (2022) abordam a aprendizagem federada utilizando a topologia *peer-to-peer* mostrando que é possível utilizar grafos para construir um algoritmo de comunicação eficiente em uma rede totalmente conectada. O custo da comunicação é calculado como a soma dos pesos das arestas percorridas de um nó até o outro.

Diferente dos trabalhos citados, a solução proposta emprega a propagação e junção dos modelos distribuídos utilizado uma topologia hierárquica autonômica, provendo escalabilidade e tolerância a falhas ao sistema.

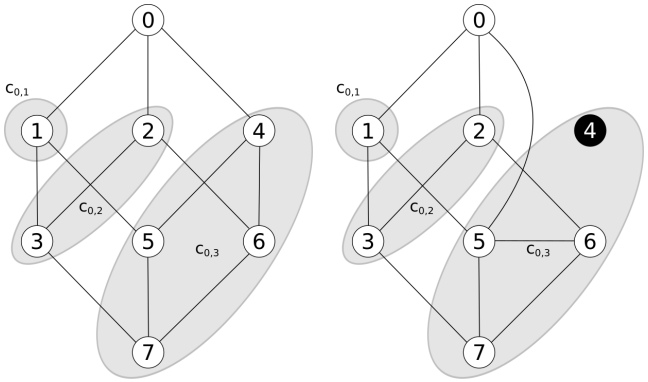
4. Solução Proposta

Neste trabalho, foi utilizada uma topologia virtual baseada em hipercubos, denominada vCube [Duarte et al. 2014], proposto originalmente no contexto de detectores de falhas [Duarte Jr. et al. 2023]. No vCube, cada nó do sistema se comunica com um conjunto de vizinhos preestabelecido, formando um hipercubo completo quando não há nós falhos [Rodrigues et al. 2016, Rodrigues et al. 2018]. A estratégia de comunicação com os vizinhos permite a disseminação das mensagens em uma estrutura autonômica de árvore binominal. Em caso de falhas, a topologia se ajusta para manter a conectividade do sistema, mantendo importantes propriedades (número de vizinhos e distância máxima entre os nós) [Jeanneau et al. 2017, de Araujo et al. 2019].

Os nós do vCube são organizados em *clusters* progressivamente maiores, cada *cluster* com 2^{s-1} nós, sendo $s = 1..\log_2 n$ e n o número de nós do sistema, identificados por $0, 1, .., n-1$. Os membros de cada *cluster* s e a ordem em que são testados por um nó i são dados pela função $c_{i,s} = \{i \oplus 2^{s-1}, c_{i\oplus 2^{s-1},1}, ..., c_{i\oplus 2^{s-1},s-1}\}$, onde $\oplus$ representa a operação XOR (OR exclusivo). A Figura 2(a) mostra uma rede com 8 nós organizados em *clusters* a partir do nó 0. O nó 1 compõe um primeiro *cluster* que, com o grupo formado pelos nós 2 e 3, formaram um agrupamento maior. O mesmo fato é observado para os nós 4, 5, 6 e 7. Em caso de falha de um nó, exemplificado na Figura 2(b) pelo nó 4, a topologia se ajusta para manter a conectividade. No exemplo, o nó 0 se conecta ao nó 5 que é o próximo nó sem falha no *cluster* $c_{0,3} = (4, 5, 6, 7)$.

A Figura 3 ilustra de que forma ocorre a comunicação entre os nós de acordo com a topologia de hipercubo adotada. Inicialmente (Etapa A) cada modelo é treinado localmente apenas com os dados contido em seu nó, desconhecendo os demais classificadores. Em um segundo momento (Etapa B), tem início o compartilhamento dos modelos entre os nós. Conforme definido pelo vCube, cada nó envia seu modelo a apenas um outro nó. No exemplo, os nós 0 e 1 trocam modelos, assim como os pares $(2, 3)$, $(4, 5)$ e $(6, 7)$.

Na Etapa C, os pares formados no passo anterior são combinados de forma que agora os modelos são compartilhados entre todos os nós presentes em cada conjunto composto por quatro elementos: *clusters* $(0, 1, 2, 3)$ e $(4, 5, 6, 7)$. No passo final (Etapa D), todos os elementos fazem parte do mesmo grupo, o que implica que todos os nós da rede já conhecem todos os oito modelos treinados individualmente.



(a) Sem falhas (b) Nó 4 falho

Figura 2. vCube de três dimensões ($n = 8$ nós).

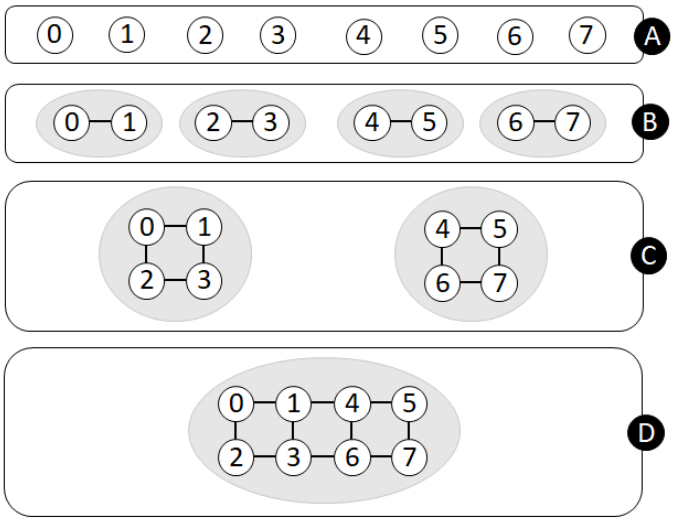


Figura 3. Agrupamento dos nós (modelos) em cada etapa do método proposto.

5. Resultados e Discussões

Esta seção apresenta os resultados dos experimentos realizados envolvendo uma classificação de dados global, na qual todos os dados são utilizados para o treinamento e teste do modelo, e um cenário no qual os dados são distribuídos entre os nós da rede de forma hierárquica. Um cenário de falhas de nó também é simulado.

Para avaliar o desempenho do aprendizado distribuído, foi utilizado o conjunto de dados MotionSense [Malekzadeh et al. 2019] composto por dados de séries temporais gerados por acelerômetros e sensores de giroscópio (altitude, gravidade, aceleração do usuário e taxa de rotação). No total, são 12 atributos e os dados foram coletados para 6 atividades realizadas por 24 participantes de diferentes gêneros, idades, pesos e alturas, totalizando 1.412.865 registros. Cada participante realizou três tentativas para cada uma das atividades “descer escada” (131.856), “subir escada” (157.285) e “caminhar” (344.288), e duas tentativas para cada atividade “correr” (134.231), “sentar” (338.778) e “levantar” (306.427). Todas as tentativas foram realizadas no mesmo ambiente e condições.

Para a etapa de treinamento, foi utilizado o algoritmo Perceptron de Múltiplas

Camadas, que é um modelo robusto e muito utilizado na prática, além de ser capaz de lidar com problemas binários e multiclasse. Como estratégia de combinação dos modelos gerados foram avaliadas as abordagens propostas por [Kittler et al. 1998], que são a regra do produto, regra da soma e a regra do voto majoritário.

A plataforma WEKA [Eibe et al. 2016] foi escolhida para executar os algoritmos porque possui uma biblioteca em Java que fornece uma gama de algoritmos de aprendizado, que pode facilmente ser utilizado e aplicado a qualquer conjunto de dados. Além disso, a biblioteca fornece ferramentas para pré-processar um conjunto de dados, analisar o classificador resultante e seu desempenho. A configuração da máquina de testes segue as seguintes especificações: Sistema Operacional Microsoft Windows 11 de 64 bits, JAVA versão 8, 12GB de RAM, processador Intel Core i7 8ª geração e armazenamento de 1TB de disco.

5.1. Classificação Global

Primeiramente, como parâmetro de comparação, foi realizada uma classificação global, utilizando 70% dos dados para treinamento (990.004 registros) e 30% dos dados para realizar os testes (423.861 registros). As classes foram distribuídas proporcionalmente entre as classes. Nesta primeira abordagem, o classificador levou 20 horas e 13 minutos para treinar o modelo e obteve acurácia de 82,98%, que pode ser considerado um bom resultado, pois como o problema possui 6 classes, isso significa que, para acertar uma classe aleatoriamente, a probabilidade é de apenas 16%.

5.2. Classificação Distribuída

Na abordagem com classificadores distribuídos, foi criado um cenário com oito nós que receberam os mesmos 70% dos registros de treinamento usados na classificação global divididos de forma igualitária, isto é, todos os nós receberam uma quantidade de registros similar e proporcional de cada classe. Neste caso, o conjunto de dados possui aproximadamente 990 mil registros, logo cada nó recebeu em torno de 123 mil registros mantendo a proporção de classes. É importante observar que, dada a divisão dos registros entre os nós, é natural que a acurácia individual seja menor.

No treinamento inicial, cada nó da rede utilizou apenas os dados disponíveis localmente para o treinamento para gerar o modelo de predição. A Figura 4 apresenta o desempenho em relação à acurácia de cada nó. A média de acurácia é de 46,37%. Neste cenário, foi possível perceber que a maioria dos nós (5 no total) tiveram uma acurácia muito parecida, em torno de 43%. Por outro lado, três nós apresentaram resultados diferentes da média. O nó 5, por exemplo, apresentou uma acurácia um pouco abaixo dos demais com o valor de 38,74% de acerto. Já os nós 6 e 7 apresentaram uma acurácia acima da média, com 56,62% e 54,89% de acurácia, respectivamente.

Para o treinamento individual, cada nó consumiu cerca de 2h30m (mínimo 2h29 e máximo 2h50), o que era esperado considerando que todos tinham a mesma quantidade de registros. Após o treinamento dos nós de forma individual, o modelo de classificação é compartilhado por cada nó por meio da vizinhança definida pela topologia virtual do vCube. Na primeira rodada, cada nó troca mensagens com seu vizinho mais próximo, compartilhando seu modelo de classificação. Assim, simulando a primeira rodada de troca de mensagens, são formados 4 grupos (*clusters*) cada um contendo dois nós com

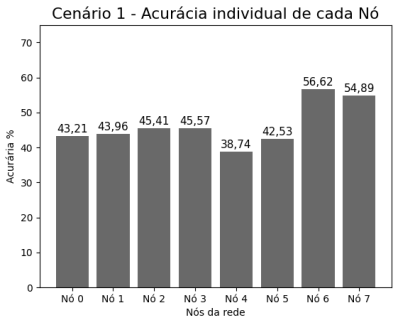
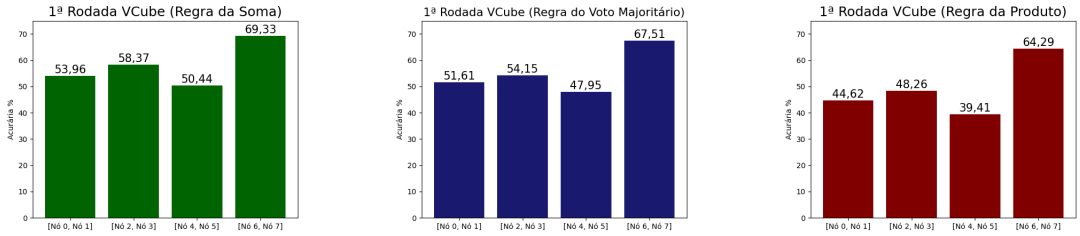


Figura 4. Acurácia individual de cada nó.

seus modelos de classificação da seguinte maneira: Grupo 1 - composto pelos nós 0 e 1; Grupo 2 - composto pelos nós 2 e 3; Grupo 3 - nós 4 e 5; e Grupo 4 - nós 6 e 7.

Percebe-se que já na primeira rodada de execução do método, houve um aumento da média da acurácia na abordagem que utiliza a regra da soma e na regra do voto majoritário (Figura 5). Utilizando a regra do produto, o aumento da acurácia é pequeno e em alguns casos existe uma queda no percentual de acertos.



(a) Regra da soma (b) Regra do voto majoritário (c) Regra do produto

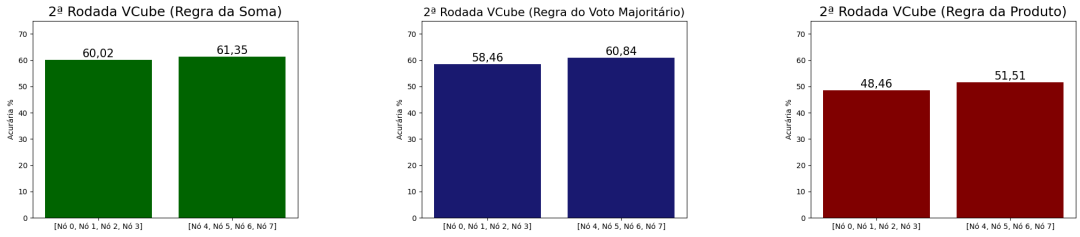
Figura 5. Acurácia na primeira rodada do método distribuído.

É possível perceber que o Grupo 4, formado pelo nós 6 e 7, obteve um desempenho maior em relação aos outros grupos. Isso se deve ao fato de que esses nós obtiveram as maiores acurácias no treinamento individual.

Na segunda rodada de execução do método, foram formados 2 grupos (*clusters*), cada um contendo a metade dos nós da rede da seguinte forma: Grupo 1 - composto pelos nós (0,1,2,3) e Grupo 2 - (4,5,6,7). Ao combinar os classificadores de cada grupo foi possível perceber que na regra da soma a acurácia do Grupo 1 foi de 60,02%, obtendo um valor da acurácia um pouco acima do grupo formado pelo nó 2 e nó 3 (na rodada anterior), obtendo pouco ganho. Já na regra do voto majoritário o ganho foi maior, pois o Grupo obteve uma acurácia de 58,46%, enquanto a acurácia do nó 2 e nó 3 foi de 54,15% na rodada anterior, ou seja, obteve um ganho significativo. Na regra do produto, a acurácia do Grupo 1 foi de 48,46%. Apesar de haver uma melhora em relação à rodada anterior, é possível notar que essa abordagem possui um desempenho muito inferior à regra do produto e do voto majoritário.

Já em relação ao Grupo 2, percebe-se que nos três cenários de combinação foi o grupo que obteve a melhor acurácia. Isso se deve ao fato de que os nós com os melhores

desempenho individuais (nós 6 e 7) estavam neste grupo. Na regra da soma, a acurácia foi de 61,35%, obtendo um desempenho inferior comparado ao grupo dos nós 6 e 7 da rodada anterior, que foi de 69,33%, ou seja, o desempenho dos nós 4 e 5 afetou negativamente a acurácia do Grupo 2. Na regra do voto majoritário, a acurácia do Grupo 2 foi de 60,84%, tendo também uma queda da acurácia comparado com os grupos formados na rodada anterior. Por fim, na regra do produto, a acurácia do Grupo em questão foi de 51,51%, sendo um desempenho inferior aos resultados obtidos na rodada anterior.



(a) Regra da soma (b) Regra do voto majoritário (c) Regra do produto

Figura 6. Acurácia na segunda rodada do método distribuído.

Na terceira rodada de execução do método distribuído, cada um dos oito nós possui todos os modelos de classificação da rede, sendo possível fazer um modelo de combinação utilizando todos os modelos intermediários. Desta maneira, foi possível perceber que a abordagem que utilizou a regra da soma foi a que obteve a melhor acurácia, com 66,23% de acertos, seguida pela regra do voto majoritário com a acurácia de 64,66% e, com o pior desempenho, a regra do produto. Observa-se que a regra do produto teve um desempenho pior que alguns nós individualmente (Figura 7).

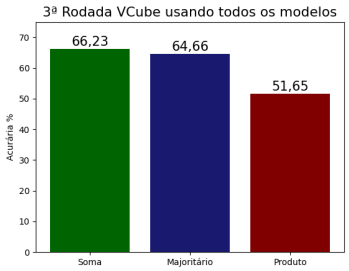


Figura 7. Acurácia da combinação de todos os nós após a rodada 3 do método.

Em relação a quantidade de dados trocados entre os nós da rede, o modelo de classificação gerado em cada nó tem tamanho médio de 45 kilobytes, ou seja, em um cenário com 8 nós no sistema, cada nó pode receber 7 modelos de classificação distintos, gerando uma carga total de 450 kilobytes. Por outro lado, se a opção for movimentar os dados para um determinado nó para que desta maneira fosse realizado o treinamento do modelo de classificação local, seria necessário transferir 113 megabytes de informação para cada nó, ou seja, uma quantidade de dados 250 vezes maior.

5.3. Cenário com falha

Neste cenário, foi realizada uma simulação na qual um nó do sistema não conseguiu enviar o seu modelo de classificação para os outros nós da rede. Assim como nos ca-

sos anteriores, todos os nós têm uma quantidade de registros semelhantes e distribuídos igualmente entre as classes. Para obter uma simulação onde os resultados mostrassem de maneira clara o impacto da falha de um nó da rede, foram realizados dois testes distintos da seguinte maneira: 1) uma falha do envio do modelo do nó com o pior desempenho individual; 2) uma falha do envio do modelo do nó com o melhor desempenho individual.

Apesar de os resultados serem parecidos com o cenário sem falhas, foi possível perceber que quando um nó com desempenho baixo é excluído, a acurácia do modelo combinado aumenta, como pode ser observado na Tabela 1. Quando o nó 5 foi excluído, a acurácia do modelo final foi de 68,03%.

Tabela 1. Acurácia (%) do SMC distribuído com falha do nó 4 (menor desempenho) ao longo das três etapas do processo.

Etapa	Grupos	Soma	Voto Majoritário	Produto
Rodada 1	1. $(0, 1)$	53,96	51,61	44,62
	2. $(2, 3)$	58,37	54,15	48,26
	3. (5)	42,53	42,53	42,53
	4. $(6, 7)$	69,33	67,51	64,29
Rodada 2	1. $(0, 1, 2, 3)$	60,02	58,46	48,46
	2. $(5, 6, 7)$	62,81	61,08	57,49
Rodada 3	1. $(0, 1, 2, 3, 5, 6, 7)$	68,03	67,32	55,11

Da mesma forma, quando um Nó com boa assertividade é excluído, o desempenho do modelo combinado diminui. Na Tabela 2, quando o nó 7 foi excluído, o desempenho do modelo combinado foi de 63,22%.

Tabela 2. Acurácia (%) do SMC distribuído com falha do nó 6 (maior desempenho) ao longo das três etapas do processo.

Etapa	Grupos	Soma	Voto Majoritário	Produto
Rodada 1	1. $(0, 1)$	53,96	51,61	44,62
	2. $(2, 3)$	58,37	54,15	48,26
	3. $(4, 5)$	50,44	47,95	39,40
	4. (7)	54,89	54,89	54,89
Rodada 2	1. $(0, 1, 2, 3)$	60,02	58,46	48,46
	2. $(4, 5, 7)$	55,42	52,98	44,33
Rodada 3	1. $(0, 1, 2, 3, 4, 5, 7)$	63,22	61,77	48,62

6. Conclusão

Este trabalho apresentou uma estratégia hierárquica para a combinação de modelos de predição treinados de forma distribuída, sem que haja necessidade de movimentação de dados entre as partes interessadas. A troca de modelos segue uma estratégia de comunicação baseada em hipercubos, que oferece importantes propriedades logarítmicas em relação ao número de vizinhos e à distância de comunicação entre os nós da rede.

Os resultados foram obtidos a partir de uma base de dados aberta com aproximadamente 1,4 milhão de registros e seis classes distintas. Uma rede Perceptron de múltiplas camadas foi utilizada como caso de teste. Como comparação, o cenário de classificação centralizada obteve uma acurácia de 82,98%. No modelo distribuído em oito nós simulando conjuntos de dados distintos, a acurácia variou entre 38% e 56%, sendo aprimorada

à medida que os modelos foram agrupados usando diferentes estratégias (regra da soma, do produto e do voto majoritário) e em casos de falhas.

Trabalhos futuros incluem utilizar outras combinações para a quantidade de registros em cada nó distribuído, por exemplo, considerando nós que possuem mais registros que outros. Além disso, pretende-se executar os cenários para outros conjuntos de dados e comparar com outros algoritmos de classificação, como Naïve Bayes, Árvore de Decisão e *Support Vector Machines*. Outra abordagem é considerar a estratégia para redes de topologia arbitrária [Duarte et al. 2011].

Agradecimentos

Este trabalho foi parcialmente financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Referências

Alpcan, T. and Bauckhage, C. (2009). A distributed machine learning framework. In *Proc. of the 48h IEEE Conf. on Decision and Control (CDC)*, pages 2546–2551. IEEE.

Britto Jr, A. S., Sabourin, R., and Oliveira, L. E. (2014). Dynamic selection of classifiers—a comprehensive review. *Pattern recognition*, 47(11):3665–3680.

Chen, S. (2018). A distributed algorithm for machine learning. *AIP Conference Proceedings*, 1955(1):040079.

de Araujo, J. P., Arantes, L., Duarte, E. P., Rodrigues, L. A., and Sens, P. (2019). VCube-PS: A causal broadcast topic-based publish/subscribe system. *JPDC*, 125:18–30.

Duarte, E. P., Bona, L. C. E., and Ruoso, V. K. (2014). VCube: A provably scalable distributed diagnosis algorithm. In *2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pages 17–22.

Duarte, E. P., Weber, A., and Fonseca, K. V. (2011). Distributed diagnosis of dynamic events in partitionable arbitrary topology networks. *IEEE TPDS*, 23(8):1415–1426.

Duarte Jr., E. P., Rodrigues, L. A., Camargo, E. T., and Turchetti, R. C. (2023). The missing piece: a distributed system-level diagnosis model for the implementation of unreliable failure detectors. *Computing*, 105.

Eibe, F., Hall, M. A., and Witten, I. H. (2016). The weka workbench. online appendix for data mining: practical machine learning tools and techniques. In *Morgan Kaufmann*. Morgan Kaufmann Publishers San Francisco, California.

Gunes, V., Ménard, M., Loonis, P., and Petit-Renaud, S. (2003). Combination, cooperation and selection of classifiers: A state of the art. *Int'l Journal of Pattern Recognition and Artificial Intelligence*, 17:1303–1324.

Guo, H. and Zhang, J. (2016). A distributed and scalable machine learning approach for big data. In *IJCAI*, pages 1512–1518.

Gupta, V., Luqman, A., Chattopadhyay, N., Chattopadhyay, A., and Niyato, D. (2022). Travellingfl: Communication efficient peer-to-peer federated learning.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Hegedűs, I., Danner, G., and Jelasity, M. (2021). Decentralized learning works: An empirical comparison of gossip learning and federated learning. *JPDC*, 148:109–124.

Hu, H., Wang, D., and Wu, C. (2020). Distributed machine learning through heterogeneous edge systems. In *AAAI Conf. on Artif. Intelligence*, volume 34, pages 7179–7186.

Jeanneau, D., Rodrigues, L. A., Arantes, L., and Duarte Jr, E. P. (2017). An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection. *Journal of the Brazilian Computer Society*, 23:1–14.

Kittler, J., HATEF, M., DUIN, R., and MATAS, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

Ko, A. H., Sabourin, R., and Britto Jr, A. S. (2008). From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition*, 41(5):1718–1731.

Leban, G., Zupan, B., Vidmar, G., and Bratko, I. (2006). Vizrank: Data visualization guided by machine learning. *Data Mining and Knowledge Discovery*, 13(2):119–136.

Lu, H., Li, M.-J., He, T., Wang, S., Narayanan, V., and Chan, K. S. (2020). Robust coreset construction for distributed machine learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2400–2417.

Malekzadeh, M., Clegg, R. G., Cavallaro, A., and Haddadi, H. (2019). Mobile sensor data anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, pages 49–58, New York, NY, USA. ACM.

Marsland, S. (2014). *MACHINE LEARNING*. Chapman & Hall/CRC, New York.

Mu, Z. (2014). Optimization of inter-network bandwidth resources for large-scale data transmission. *JOURNAL OF NETWORKS*, 9(3):689–694.

Peteiro-Barral, D. and Guijarro-Berdiñas, B. (2013). A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11.

Rodrigues, L. A., Arantes, L., and Duarte, E. P. (2016). An autonomic majority quorum system. In *30th AINA*, pages 524–531. IEEE.

Rodrigues, L. A., Duarte Jr, E. P., de Araujo, J. P., Arantes, L., and Sens, P. (2018). Bundling messages to reduce the cost of tree-based broadcast algorithms. In *LADC*, pages 115–124. IEEE.

Schmeing, E., Brun, A. L., and Silva, R. A. (2022). Dynamic selection of classifiers based on complexity measures. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 82–89.

Tsoumakas, G. and Vlahavas, I. (2009). Distributed data mining. In *Database technologies: Concepts, methodologies, tools, and applications*, pages 157–164. IGI Global.

Verbraeken, J. et al. (2020). A survey on distributed machine learning. *ACM computing surveys*, 53(2):1–33.

Wang, H., Niu, D., and Li, B. (2019). Distributed machine learning with a serverless architecture. In *IEEE INFOCOM*, pages 1288–1296. IEEE.

Zaharia, M. et al. (2016). Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59(11):56 – 65.