Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Rio de Janeiro, 4 de dezembro de 2022

Algoritmos Distribuídos de Aprendizado

Gabriel Arantes Cortines Coelho

PUC RIO

Projeto de Graduação

Algoritmos Distribuídos de Aprendizado

Aluno: Gabriel Arantes Cortines Coelho

Orientador: Rodrigo Caiado De Lamare

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia Elétrica na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Ao entregar este trabalho, gostaria de expressar minha mais sincera e profunda gratidão a minha família, que durante toda a minha vida me forneceu suporte material e emocional, sem os quais não conseguiriam estar redigindo este texto. Em especial a minha mãe Marta Coelho, que por diversas vezes me aconselhou e deu força durante meus trabalhos mais importantes.

Ao professor Rodrigo Caiado De Lamare quero agradecer pelas oportunidades e por toda confiança, me orientando não somente agora neste trabalho, como também durante minhas iniciações científicas. Ao senhor Songcen Xu, agradeço pelo auxílio e disposição a responder dúvidas durante as etapas iniciais deste trabalho.

Por fim, não serei capaz de citar todos os membros do corpo docente e discente da faculdade que me auxiliaram durante minha formação acadêmica, mas gostaria de evidenciar alguns colegas e amigos que foram essenciais para mim. Camila Lima de Sousa, Breno Perlingeiro Corrêa, Christiano Moreira de Sá do Nascimento, Felipe Calliari e Erico de Souza Prado Lopes, muito obrigado por toda ajuda, sorrisos e momentos que compartilhamos.

Resumo

O aumento da quantidade de dispositivos sem fio gerou também o aumento da densidade da faixa de frequências usadas, assim aumentando a probabilidade de interferência entre eles e por ruídos convencionais. Em virtude da necessidade de minimizar ou reduzir estes efeitos, é estudado algoritmos de aprendizado para realização de inferência estatística, desempenhando função de monitoramento e predição de sistemas complexos.

Este projeto é focado no estudo acerca de algoritmos de aprendizado adaptativo capaz de realizar inferência com a finalidade de evitar o efeito de interferência entre os dispositivos e minimizar possíveis erros, em especial utilizando algoritmos baseados no gradiente estocástico e no gradiente próximo em um cenário de aprendizado distribuído.

Esse trabalho foi particionado em três etapas: estudo e simulação de redes distribuídas; teste em cenário real: medição de temperatura; e elaboração do algoritmo próximo e estudo de desempenho no cenário federativo.

Palavras-chave: redes, aprendizado distribuído, LMS, RLS, algoritmo próximo

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Distributed Learning Algorithms

Abstract

The increase in the number of wireless devices also generated an increase in the density of the frequency band used, thus increasing the probability of interference between them and by conventional noise. Due to the need to minimize or reduce these effects, learning algorithms for performing statistical inference are studied, performing the function of monitoring and predicting complex systems.

This project is focused on the study of adaptive learning algorithms capable of performing inference in order to avoid the effect of interference between devices and minimize possible errors, in particular using algorithms based on the stochastic gradient and the close gradient in a learning scenario distributed.

This work was divided into three stages: study and simulation of distributed networks; test in real scenario: temperature measurement; and elaboration of the next algorithm and study of performance in the federative scenario.

Keywords: networks, distributed learning, LMS, RLS, proximal algorithm

Sumário

Lista de Figuras

Lista de Algoritmos

1 Introdução

A necessidade da pesquisa em algoritmos de aprendizado e realização de inferência estatística é justificada pelo desenvolvimento de tecnologias como Internet das Coisas (IoT - Internet of Things), comunicação entre máquinas e sistemas de comunicações sem fio modernos, que aumentam a quantidade de dispositivos sem fio. Com o aumento da densidade da faixa de frequências usada, esses dispositivos aumentam a probabilidade de interferência entre si e por ruídos convencionais.

Técnicas de inferência (detecção, estimação e previsão) e aprendizado são ferramentas de grande importância neste contexto uma vez que permitem aos dispositivos estimar parâmetros, tomar decisões, realizar medidas e calcular localização. Uma abordagem para realizar inferência estatística considera algoritmos de aprendizado adaptativos, que ajustam seus parâmetros por meio de um algoritmo de otimização.

Neste projeto, a adaptação se dará pelo estudo de algoritmos baseados no gradiente estocástico e no gradiente próximo com testes em uma rede de IoT. Nela cada sensor compartilha suas estimativas com um determinado conjunto de sensores que usam essa informação para aprendizado. Essa comunicação pode ser otimizada por algoritmos que encontrem a configuração ou topologia ótima da rede e na realização de múltiplas tarefas de inferência.

2 Estudo e Simulação de Redes Distribuídas

Nesta seção serão apresentados os estudos iniciais acerca de redes de aprendizado distribuídas e as primeiras simulações desenvolvidas para avaliar o desempenho de técnicas de aprendizado tradicionais no cenário distribuído. Sendo assim, é válido primeiramente explicitar o significado de termos chave para compreensão do trabalho, facilitando o entendimento de leitores menos familiarizados com o tema abordado.

As redes do aprendizado são constituídas por elementos unitários chamados de nós, os quais são sensores que podem estar associados também a um processador cada. Cada nó é conectado de modo a ser capaz de comunicar diretamente com um conjunto de nós, sua vizinhança, e indiretamente com os demais, através das ligações da própria vizinhança. Tais conexões não precisam ser necessariamente físicas, neste trabalho por exemplo foi optado por utilizar um protocolo de comunicação via rádio ao invés de fios.

Definindo a estrutura da rede, a tarefa do aprendizado distribuído é realizada com os nós compartilhando informações sobre uma grandeza ou variável de interesse e utilizando algoritmos de aprendizado para modelar seu comportamento, estimando a com maior precisão. Exemplos de aplicações de redes seriam um conjunto de sensores espalhados ao longo de uma cidade, que trabalham para estimar a temperatura média, ou ainda uma rede de sensores espalhados em um terreno, com a finalidade avaliar a qualidade da terra para plantio a partir de dados como Ph e umidade do solo.

a Tipos de Redes de Aprendizado

As redes de aprendizado podem ser classificadas quanto à forma de processamento, sendo estas denominadas centralizadas ou distribuídas. Em uma rede centralizada há um nó central, responsável pelo processamento de todas as informações, enquanto os demais apenas realizam coleta de dados, a Figura 1 exemplifica a topologia de uma rede centralizada. Como o nó central tem acesso e processa toda a informação da rede, esse modelo acaba tendo um melhor desempenho na estimação. Porém, é sujeito a falhas críticas quando há problemas no nó central, além de requerer que o mesmo tenha uma maior capacidade de processamento e consumo de energia no nó central quanto maior for a rede, limitando assim o tamanho da rede.

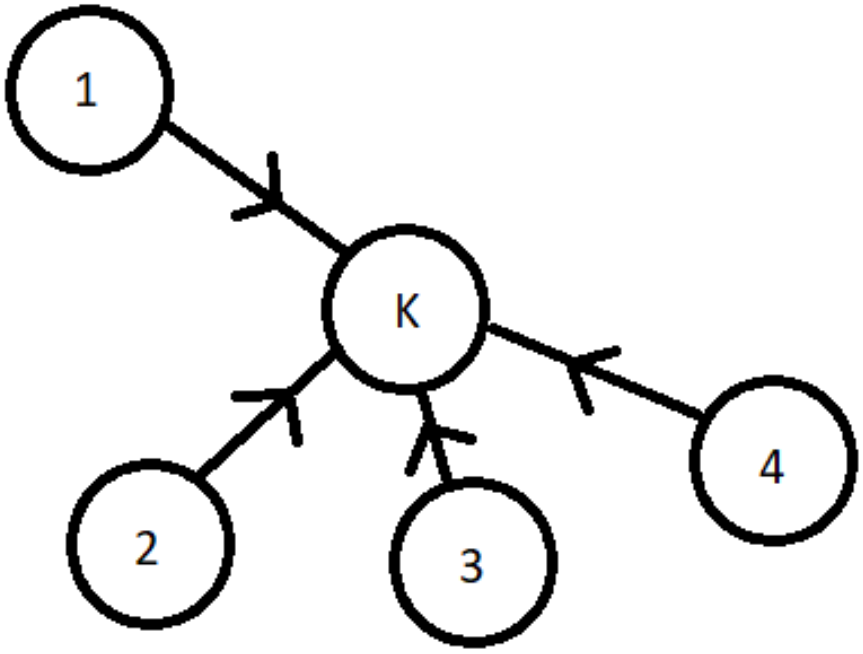

Figura 1: Exemplo de rede centralizada com nó central k com Setas Representando Fluxo de Informação

Em redes com processamento distribuído cada elemento da rede processa os dados próprios e dos nós adjacentes, sua vizinhança conforme a Figura 2, gera uma estimativa que também é compartilhada. Nesse sentido o resultado do processamento se assemelha a um consenso das estimativas de cada nó, sendo a principal vantagem desse tipo de processamento a resiliência da rede, pois seu funcionamento resiste a problemas em nós pontuais.

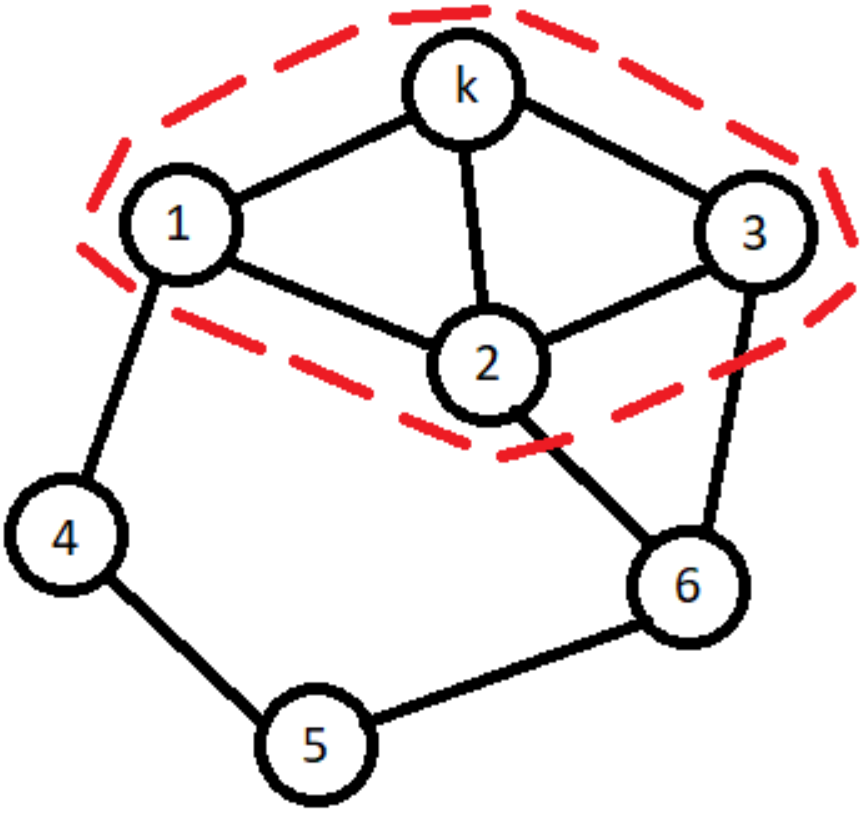

Figura 2: Exemplo de rede distribuída com vizinhança do nó k circulada em vermelho

b Processamento Distribuído

Para a estimação de medidas da rede, é preciso encontrar um vetor de parâmetros **w** ótimo, que relaciona o vetor de regressão **x**, que contém as medidas amostradas, com uma medida escalar y, registrada por cada nó. No caso do processamento distribuído, cada um dos nós da rede utiliza algoritmos, como least mean square (LMS) [1] e recursive least square (RLS) [2] , para calcular o **w**, compartilhando suas estimativas, medição de y e vetor **x**, esse processo se chama difusão. [3]

Dois métodos podem ser empregados para efetuar a difusão, estes são combine then adapt (CTA) e adapt then combine (ATC), seguindo as etapas abaixo.

Algoritmo 1 Método Difusão CTA

Ordem das etapas do método de difusão CTA;

- Os nós compartilham o vetor de parâmetros **w** com sua vizinhança.
- Cada nó gera uma estimativa **v** fazendo uma soma ponderada dos parâmetros **w** dos vizinhos.
- Os nós compartilham a medida y e o vetor **x** com sua vizinhança.
- Cada nó atualiza seu parâmetro **w** a partir da correção da sua estimativa **v** pelo erro obtido ao utilizá-la para relacionar y e **x**.

Algoritmo 2 Método Difusão ATC

Ordem das etapas do método de difusão ATC;

- Os nós compartilham a medida y e o vetor **x** com sua vizinhança.
- Cada nó gera uma estimativa **v** a partir da correção do seus parâmetros **w** pelo erro obtido ao utilizá-lo para relacionar y e **x**.
- Os nós compartilham a estimativa **v** com sua vizinhança.
- Cada nó atualiza seus parâmetros **w** fazendo uma soma ponderada das estimativas **v** dos vizinhos.

A mudança na ordem das etapas de difusão pode, a princípio, não parecer significativa. Porém, na difusão ATC, por os nós começarem se ajustando às medidas da vizinhança, a atualização do parâmetro w de um nó carrega informação das

medidas das vizinhanças de cada nó ligado a ele, enquanto no CTA só se tem da própria vizinhança, o que explica o melhor desempenho do método ATC, em relação ao CTA, mesmo não tendo maior custo computacional.

c Combinadores

O papel dos combinadores é fornecer pesos c, para soma ponderada na etapa de combinação, com o objetivo de agregar as informações presentes em cada nó. Esses pesos c, tem três características fundamentais, todo c deve ser maior ou igual a 0, c de um nó k é igual a 0 quando se está associado a um nó que não é da vizinhança de k e o somatório dos c de cada nó na vizinhança de k deve ser igual a um.

Os pesos c utilizados na combinação são geralmente armazenados em matrizes de combinação com regras específicas de formação, entre elas a uniforme [4] e a metrópoles [5] . Sendo n o número de nós na vizinhança do nó k, pela regra uniforme $c = 1/n$ se o nó pertence a vizinhança de k e 0 caso o contrário, o que geram uma matriz estocástica à esquerda. Já para metrópoles $c = 0$ se o nó não pertence a vizinhança de k, se pertence a vizinhança mas não é o próprio nó k $c = 1/max[n_k, n_m]$, sendo n_k o n do nó k e n_m o n de um nó m qualquer ligado a k, e $c = 1-$ somatório dos demais c, se é o próprio nó k, resultando em uma matriz duplamente estocástica.

Uniforme:

- $c = 0$, se não pertence a vizinhança de k.
- $c = 1/n$, se o nó pertence a vizinhança de k.

Metrópoles:

- $c = 0$, se não pertence a vizinhança de k.
- $c = 1/max[n_k, n_m]$, se o nó pertence a vizinhança de k, sendo n_k o n do nó k e n_m o n de um nó m qualquer ligado a k.
- $c = 1-$ somatório dos demais c, para o próprio nó k.

Justamente pela regra metrópoles gerar matrizes duplamente estocástica, acaba gerando melhores resultados na otimização de parâmetros. Sendo essa regra de combinação, muito utilizada para algoritmos de aprendizado em redes.

d Algoritmo LMS

O algoritmo LMS tem como objetivo encontrar o valor ótimo do parâmetro **w** calculando em cada iteração i, o erro médio quadrático entre a medida y e sua extrapolação, utilizando **w** e vetor de regressão **x**, para ajustar o valor de **w** através da função custo $E[(y - \mathbf{x}^T \cdot \mathbf{w})^2]$. Com a finalidade de minimizá-la temos a atualização de **w** em um instante dada por $\mathbf{w}(i) = \mathbf{w}(i-1) + u \cdot \text{erro} \cdot \mathbf{x}$, onde o erro é definido por $y - \mathbf{x}^T \cdot \mathbf{w}(i-1)$, sendo $\mathbf{w}(i) = 0$ para $i < 0, w(0)$ é uma estimativa inicial e u é um escalar arbitrado chamado de tamanho de passo.

O tamanho de passo é um termo escalar menor que 1, o qual dita a que taxa o parâmetro ajusta a seu erro, de modo geral é necessário rodar diversas simulações para determinar seu melhor valor, para o qual **w** converge no menor número de iterações e o algoritmo acompanha oscilações do erro após a convergência. Para estudar os efeitos do tamanho de passo u, foram rodadas simulações com 60 repetições e 800 iterações, no MATLAB, tomando como sinal de entrada **x**, vetor de números normalmente distribuídos com ruído branco de variância 0,001, além do **w** parâmetro ótimo, vetor de números normalmente distribuídos entre 0 e 1. Com os **x** e **w** utilizados nos testes, gerava-se um valor de interesse y e os algoritmos eram avaliados em quão próximo suas estimativas conseguiam chegar do **w** ótimo conhecido.

Na forma distribuída do LMS [6] , a atualização do parâmetro **w** passa por modificações para incorporar a etapa de combinação dos dados de nós adjacentes. Considerando a utilização de um método difusão ATC e regra de combinação metrópoles, sendo $\mathbf{v}_m$ estimativa, $\mathbf{w}_m$ parâmetro e $\mathbf{x}_m$ vetor de regressão do nó m na vizinhança de k, se segue duas etapas:

Algoritmo 3 Atualização do Parâmetro w, LMS

Os vetores $\mathbf{v}_k$, $\mathbf{w}_k$ e $\mathbf{x}_m$ tem lx1 dimensões, sendo l o filter tap escolhido;

- $\mathbf{v}_k(i) = \mathbf{w}_k(i-1) + \text{u} \cdot \sum(erroemm \cdot \mathbf{x}_m)$
- $\mathbf{w}_k(i) = \sum(c_m \cdot \mathbf{v}_k)$

A Figura 3 é um gráfico do erro médio quadrático (mean square error - MSE) do algoritmo em uma simulação com tamanho de passo muito grande, u = 0,1. Como esperado de um passo grande, o algoritmo converge rapidamente, mas justamente

por esse tamanho de passo sobre compensar o erro não consegue fazer ajustes finos, acarretando em oscilações. Já na Figura 4 a simulação teve tamanho de passo pequeno, u = 0,005. Pode-se notar uma suave curva de convergência, porém com esse passo a conversão demora iterações demais. Após diversas simulações foi constatado o melhor valor para o tamanho de passo, u = 0,06, que resulta na Figura 5, onde ocorre uma conversão rápida seguida de estabilidade.

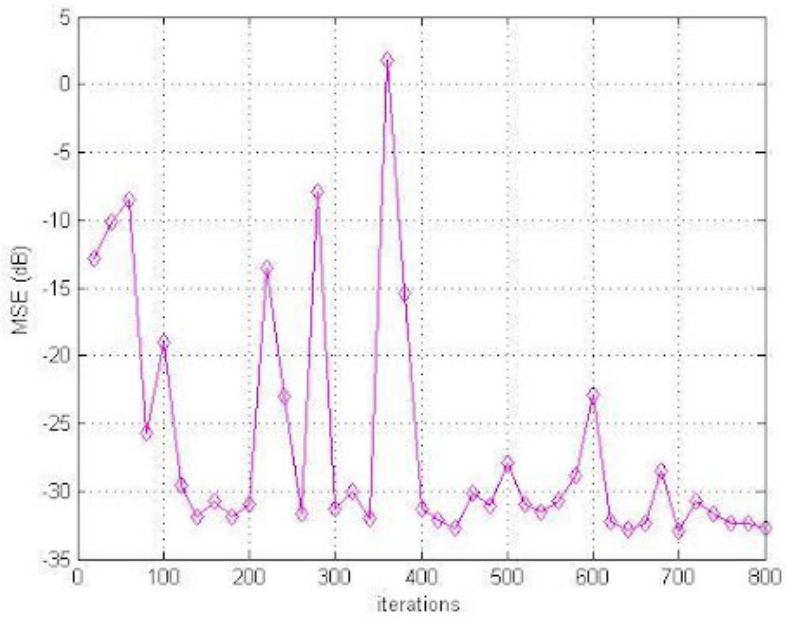

Figura 3: Gráfico MSE ao longo das iterações do LMS com u = 0,1

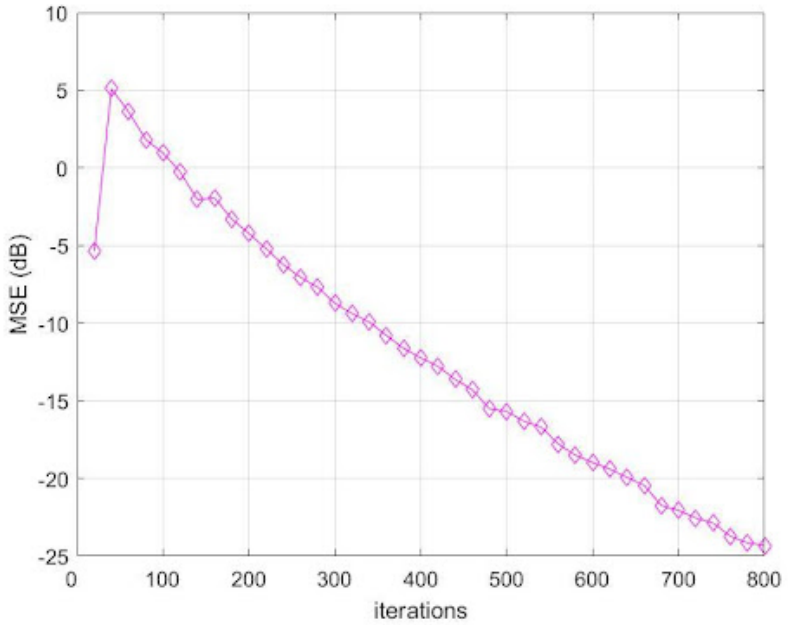

Figura 4: Gráfico MSE ao longo das iterações do LMS com u = 0,005

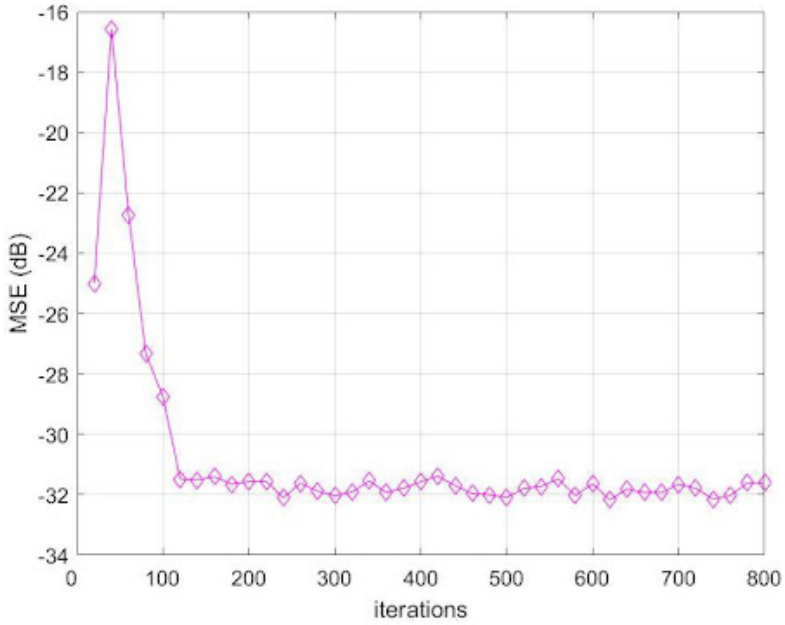

Figura 5: Gráfico MSE ao longo das iterações do LMS com u = 0,06

e Algoritmo RLS

No algoritmo RLS, propõe se que para a melhor estimativa do parâmetro **w** o vetor formado pelos y medidas de cada nó, pertence ao espaço gerado pelas colunas da matriz **X**, cujo as linhas são os vetores de regressão **x** de cada nó, desse modo é possível por meio de uma regressão encontrar o valor ótimo de **w**. Utilizando uma regressão rígida com parâmetro de regularização q, para dar mais peso as bases do espaço gerado com maior desvio padrão, e empregando uma técnica recursiva temporal utilizando fator de esquecimento b, para dar melhor capacidade ao algoritmo de se adaptar a padrões de dados mais recentes, pode-se calcular o ganho de Kalman que é multiplicado ao erro para atualizar a estimativa de **w**. Na inicialização no

RLS distribuído, temos a matriz **Q**, proveniente da multiplicação do parâmetro de regularização com uma matriz identidade cujo, ambas as dimensões iguais ao número de nós, e dois vetores auxiliares **z** e **r**, com número de linhas igual ao de nós da rede. Obtendo a seguinte atualização do **w** para o nó k nas iterações i:

Algoritmo 4 Atualização do Parâmetro w, RLS

A matriz $\mathbf{Q}$ tem dimensões lxl dimensões, sendo l o filter tap escolhido;
Os vetores $\mathbf{z}_k$, $\mathbf{r}_k$,$\mathbf{x}_m$, $\mathbf{v}_k$ e $\mathbf{w}_k$ tem dimensões lx1 dimensões;

- $\mathbf{z}_k(i) = \mathbf{Q} \cdot \mathbf{x}_m$
- $\mathbf{r}_k(i) = \mathbf{z}_k(i)$ / [b + $\mathbf{x}_m^T \cdot \mathbf{z}_k(i)$]
- $\mathbf{v}_k(i) = \mathbf{w}_k(i-1) + \mathbf{r}_k(i) \cdot \sum(y_m(i)$ - $\mathbf{w}_k(i-1)^T \cdot \mathbf{x}_m)$
- $\mathbf{w}_k(i) = \sum(c_m \cdot \mathbf{v}_k)$

f Comparação entre RLS e LMS

Com a finalidade de avaliar a performance entre os algoritmos LMS e RLS empregados em uma rede com 20 nós, difusão ATC e combinador metrópoles, foi utilizado como base um código em MATLAB para simular LMS e RLS distribuído. Após a otimização dos parâmetros de cada algoritmo, por meio de diversas simulações com 60 repetições e 500 iterações, foram encontrados: tamanho de passo u = 0,06 para o LMS; e parâmetro de regularização q = 0,01 e fator de esquecimento b = 0,685 para o RLS. Como mostra a Figura 6, o RLS conseguiu alcançar a conversão com a metade das iterações necessárias para o LMS.

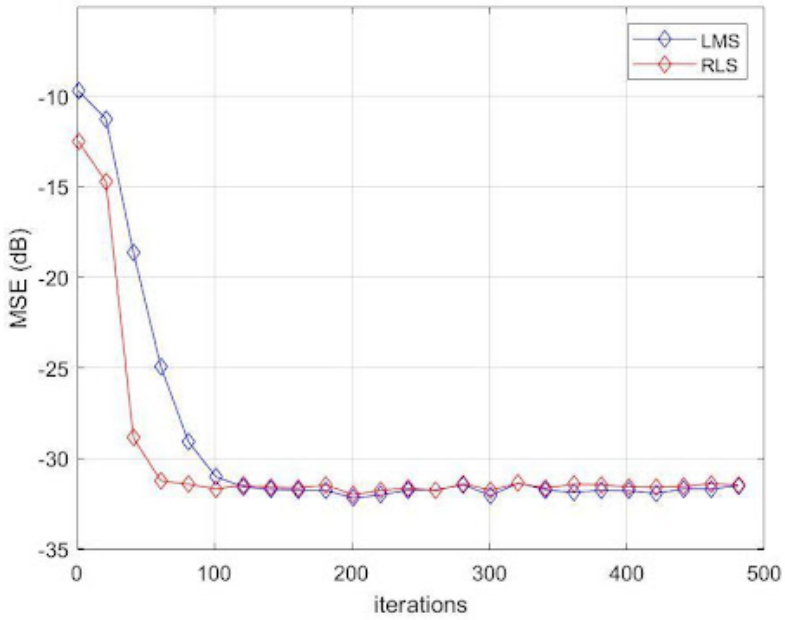

Figura 6: Gráfico MSE ao longo das iterações do LMS e RLS

Quanto a comparação entre os algoritmos LMS e RLS, obtida por meio do código em MATLAB que simula uma rede com 20 nós, difusão ATC e combinador metrópoles modificado pelo aluno, foi constatada uma significativa diminuição no número de iterações necessária para atingir a conversão do parâmetro desconhecido **w**, ao utilizar o RLS. Sendo assim, o algoritmo RLS se mostra mais adequado para redes com processamento distribuído, ainda mais ao considerarmos que cada nó geralmente tem capacidade de processamento limitada.

3 Teste em Cenário Real: Medição de Temperatura

Nesta segunda parte do projeto, foram utilizados os conhecimentos provenientes da pesquisa feita anteriormente sobre algoritmos distribuídos e simulações em MATLAB para construir uma rede real de controladores Arduino, com a tarefa de previsão de temperatura em intervalos de 5 minutos e tratando os dados medidos com um filtro de média móvel, para reduzir o ruído dos sensores. Com estes controladores foram implementados algoritmos de aprendizado least mean square (LMS) e recursive least square (RLS), em nó solitário, onde apenas um controlador realiza toda a tarefa de aprendizado, e em rede, onde vários controladores compartilham a informação para o processo de aprendizado, a fim de comparar o desempenho deles em um cenário real e averiguando as vantagens de uma rede em relação a um nó solitário.

a Nó Solitário com Algoritmo LMS

No caso estudado de previsão de temperatura, há especificidades que valem ser evidenciadas para uma maior entendimento da lógica de funcionamento do algoritmo. Ambos o vetor **x** e o escalar y são termos de um vetor **t**, que armazena os valores de temperatura medidos em intervalos de 5 minutos, sendo y a temperatura no instante i e **x** os demais termos de **t** nos instantes i-1, i-2 e i-5. Tal característica pode acarretar em uma nova interpretação do erro, sendo o erro da iteração anterior a diferença entre o que é medido agora e quanto havia sido previsto, $erro(i-1) = y(i) - \mathbf{w}(i-1) \cdot \mathbf{x}(i-1)$. Outro adendo importante é que pela dinamicidade da previsão de temperatura, não se pode garantir que o **w** ótimo seja um valor constante, ressaltando a importância da capacidade de acompanhamento do erro pelo algoritmo, durante a curva de aprendizado, para avaliação do desempenho.

Para realizar comparações entre os algoritmos em cenário real, foram realizados testes com diversos valores para o escalar u e em horários diferentes do dia, a fim de evitar um possível viés. Como condição inicial dos testes, todos os termos do vetor **w** eram inicializados com zeros, situação em que a rede não possui nenhum conhecimento prévio da grandeza medida, e os termos de **t** assumiram o valor da primeira medida de temperatura, assumindo como partida um estado sem alterações de temperatura anterior ao começo das medições. Conforme explicitado na seção anterior, na busca do tamanho de passo u que otimizasse o algoritmo é necessário cautela, pois mínimas variações em seu valor podem gerar grande impacto na performance, valores grandes de u rapidamente atingiram valores altos de erro, ao exemplo do teste com u = 0,0009, que com apenas 2 iterações o erro médio quadrático MSE ultrapassou 3 casas decimais e apresentava um crescimento exponencial. Para valores muito pequenos de u o problema é o tempo de conversão muito longo, assim como visto nas simulações.

Após diversos processos de medição e estimativas, foram encontrados boas curvas de aprendizado para u = 0,0002 e u = 0,0006, Figura 7 e Figura 8 respectivamente. Nelas o eixo x representa as iterações, apresentando convergência tipicamente em torno de i = 12 com MSE médio de 10^{-3}. Com uma busca detalhada na região entre os dois valores, constatou-se um valor ótimo u = 0,0004, que alcançava o mesmo MSE, mas agora com apenas 7 iterações, Figura 9.

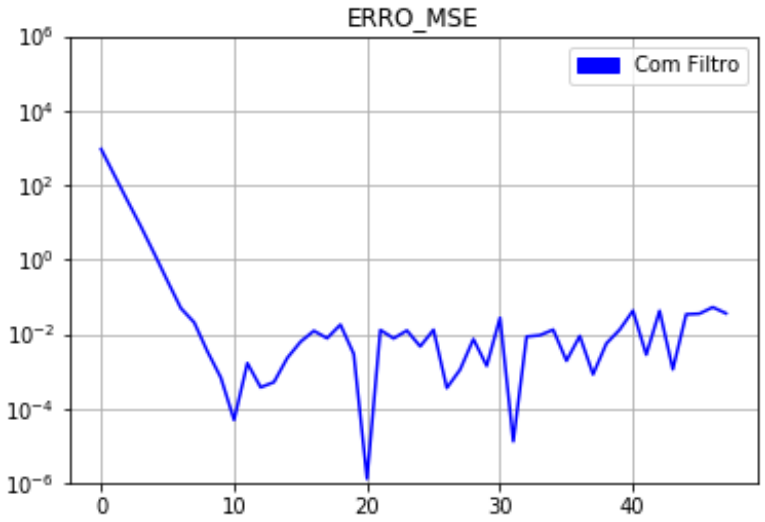

Figura 7: Gráfico MSE Nó Solitário LMS com u = 0,0002

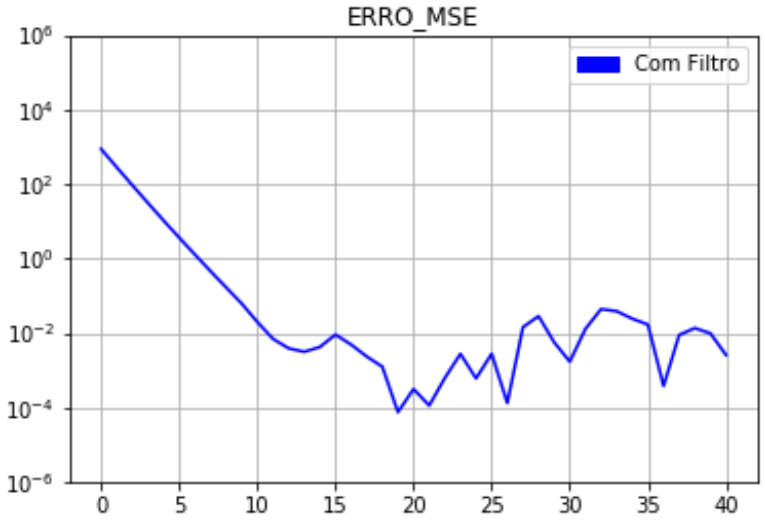

Figura 8: Gráfico MSE Nó Solitário LMS com u = 0,0006

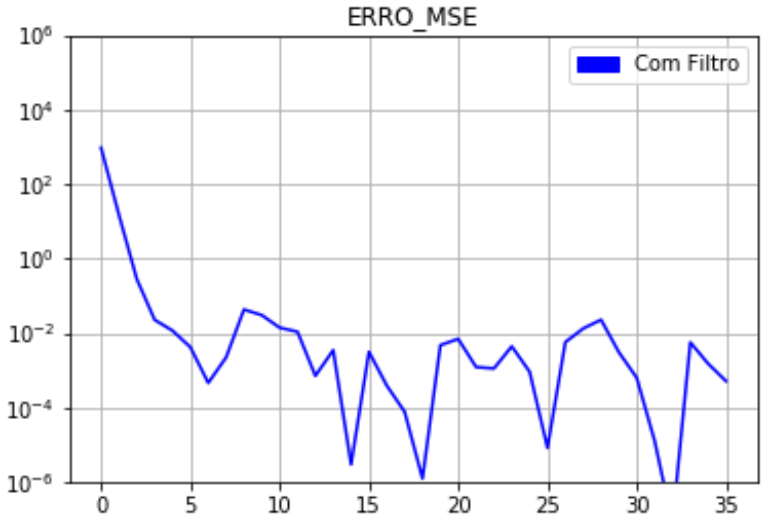

Figura 9: Gráfico MSE Nó Solitário LMS com u = 0,0004

b Nó Solitário com Algoritmo RLS

Nos testes, foram investigados os valores do parâmetro de regularização q e fator de esquecimento b, a fim de otimizar o desempenho do algoritmo, e para validar as comparações foram adotadas as mesmas condições iniciais do LMS. Inicialmente todos os termos vetor **w** foram zerados e os termos de **t** assumiram o valor da primeira medida de temperatura.

O parâmetro q apresentou impacto negativos para valores altos, q = 1 na Figura 10 onde o MSE final é 10^{-2}, apresentando melhor desempenho para valores menores que 0,6, onde foram alcançados os menores valores de MSE. Constatou-se também que o fator b, poderia ser grande a ponto de esquecer os padrões da variação de temperatura após a conversão, vide Figura 11 com b = 10.

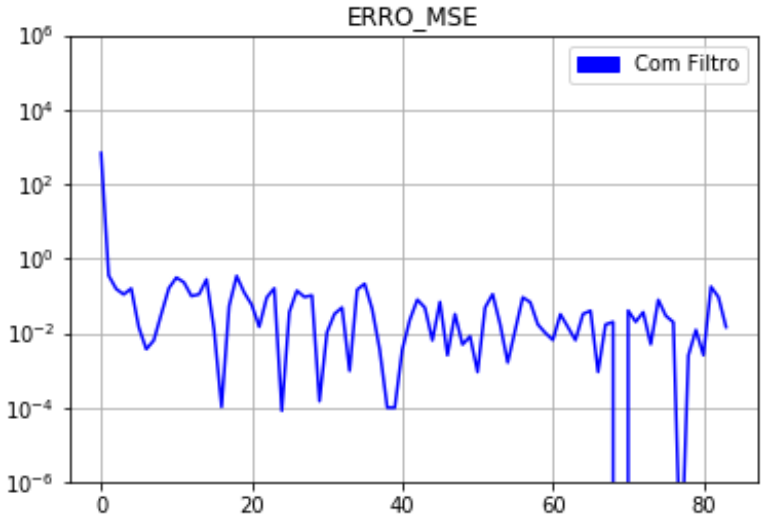

Figura 10: Gráfico MSE Nó Solitário RLS com q = 1 e b = 0,9

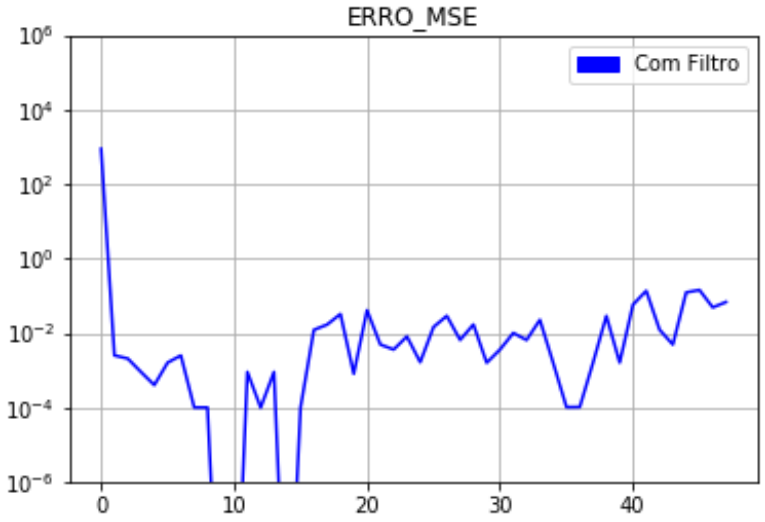

Figura 11: Gráfico MSE Nó Solitário RLS com q = 0,5 e b = 10

O melhor resultado foi encontrado para q = 0,5 e b = 0,9, onde a curva de aprendizado convergia em aproximadamente 4 iterações com MSE de 10^{-3}, Figura 12.

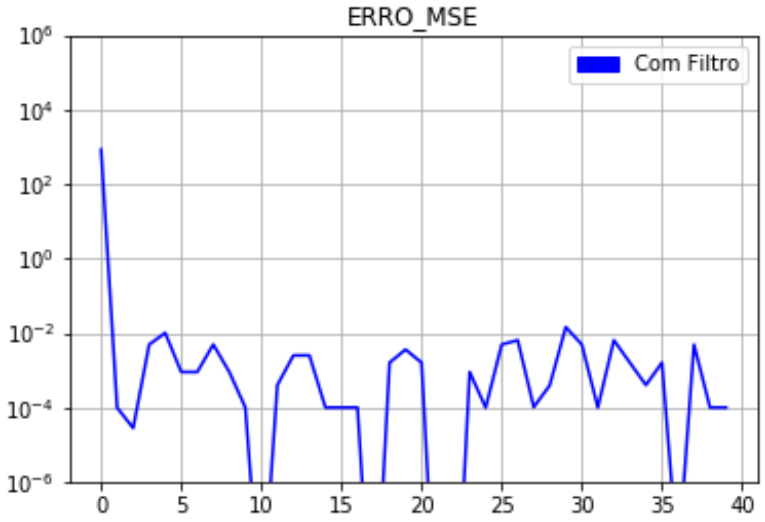

Figura 12: Gráfico MSE Nó Solitário RLS com q = 0,5 e b = 0,9

c Comparação do Nó Solitário com LMS e RLS

Ambos os algoritmos apresentaram um excelente desempenho, mantendo um MSE médio final de 10^{-3} relativamente estável, porém é notável a diferença de velocidade na conversão entre os dois. O algoritmo RLS convergia com aproximadamente metade das iterações necessárias para o LMS, tal resultado se assemelha ao encontrado nas simulações da primeira seção deste projeto.

d Rede Incremental RLS

Tendo em vista o melhor desempenho do algoritmo RLS [7] , foi então criada uma rede de quatro nós para poder averiguar os possíveis benefícios provenientes do processamento distribuído. No modelo de transmissão incremental, cada nó recebe do anterior a última temperatura medida e o **w** encontrado, fazendo sua estimativa de seu **w**, o qual junto com a temperatura medida por ele é passa adiante. A Figura 13 mostra uma rede onde os círculos numerados representam os nós e as setas o fluxo da informação.

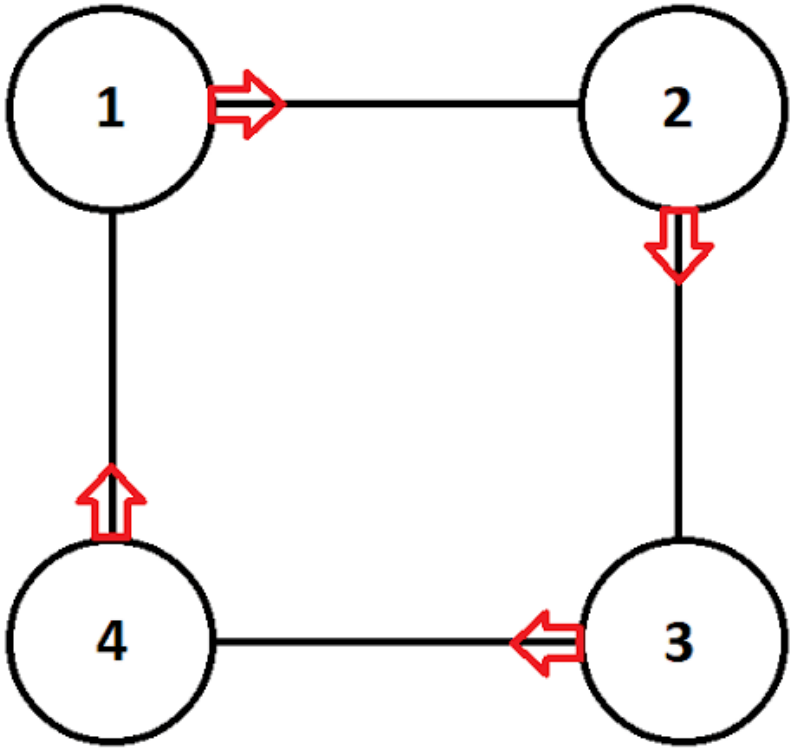

Figura 13: Rede Incremental com 4 Nós

O modelo incremental foi escolhido para testes com rede de controladores físicos por sua fácil modificação para inclusão de mais ou menos nós, sem requerimento de maior poder de processamento individual de cada nó, permitindo seu dimensionamento para os mais diversos cenários de aplicação. Entretanto, existem algumas desvantagens deste método em um cenário real, sobretudo a fragilidade do sistema, devido ao não recebimento da transmissão pelos mais diversos fatores, como interferência ou falha no recebimento, interromperem o funcionamento, ao quebrar o fluxo de informação. Porém, estando ciente desse fato, é possível implementar rotinas de código para aumentar a robustez, como diretivas de monitoramento que identificam possíveis nós problemáticos ou falhas na transmissão e reajustam os padrões de transmissão e topologia da rede, sem necessidade de interrupção do funcionamento.

Ao implementar o algoritmo RLS em uma rede incremental sua lógica é mantida, mas há a necessidade de seguir um protocolo para combinar os vetores **w** recebido e atual do nó. Primeiramente, é calculada a atualização do $\mathbf{w}_p$, **w** do nó presente, e $\mathbf{w}_a$, **w** do nó anterior, por meio do algoritmo RLS, seguindo tanto a série de temperaturas medidas, quanto das recebidas. Tendo $\mathbf{w}_p$ e $\mathbf{w}_a$ atualizados, é combinado o quadrado do erro em suas atualizações nas suas duas séries, gerando a nova estimativa $\mathbf{w}_n$, sendo $\mathbf{w}_n = (erro\_w_a^2)/(erro\_w_p^2 + erro\_w_a^2) \cdot (\mathbf{w}_p - \mathbf{w}_a) + \mathbf{w}_a$.

Foram realizados então, testes do algoritmo RLS na rede incremental adotando condições iniciais semelhantes às do nó solitário, inicialmente todos os termos vetor wa e wp eram zerados e os termos das séries de temperatura assumiram o valor da primeira medida do nó e recebida do antecessor. O algoritmo obteve dessa vez melhor desempenho, para q = 0,01 e b = 0,9 com MSE de 10^{-4} em 4 iterações, a Figura 14 apresenta a melhor curva de aprendizado desse algoritmo.

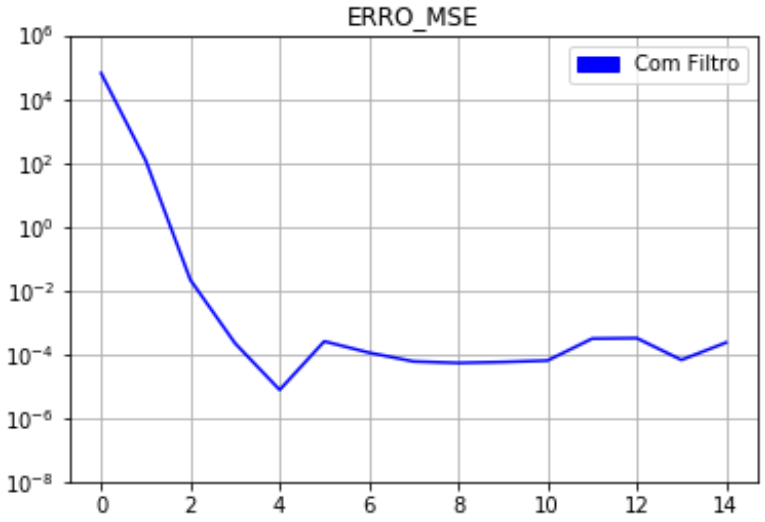

Figura 14: Gráfico MSE Rede Incremental RLS com q = 0,01 e b = 0,9

e Comparação entre Nó Solitário e Rede Incremental RLS

O algoritmo RLS na rede incremental apresentou duas significativas vantagens em relação à implementação no nó solitário, apesar de conversão ocorreu com aproximadamente o mesmo número de iterações, era alcançado um MSE dez vezes menor e as oscilações na curva de aprendizado eram significativamente menores. Sendo assim, é contundente afirmar que no cenário de previsão de temperatura, a rede não só possibilitou um melhor aprendizado do **w**, como também melhor acompanhamento de suas eventuais alterações, resultando em previsões com maior acurácia.

A implementação dos algoritmos LMS e RLS, por meio de controladores Arduino, em um cenário real confere respaldo aos resultados da etapa anterior do projeto, na qual através da simulação em MATLAB havia sido constatado que o RLS apresentava melhor desempenho e atingia convergência em menos iterações que as necessárias para o LMS.

Por fim, é válido reiterar que a previsão de temperatura é uma situação real de implementação. Sendo assim, os métodos desenvolvidos e resultados obtidos podem ser implementados na resolução de diversas outras aplicações, como monitoramento de propriedades químicas do solo e modelagem de sistemas dinâmicos.

4 Elaboração do Algoritmo Próximo e Estudo de Desempenho no Cenário Federativo

Tendo sido validado os conhecimentos obtidos em simulação e no cenário real nas etapas anteriores, estes foram utilizados para criar e estudar implementações de um outro tipo de algoritmo de aprendizado, o algoritmo próximo. Comparando este método de aprendizado à algoritmos de aprendizado tradicionais, least mean square (LMS) e recursive least square (RLS), simulando-os em uma rede distribuída, onde os nós compartilham informação entre si e há uma participação de todos os nós a cada iteração, e para um cenário de aprendizado federativo, onde cada nó participante envia informação para o nó central e apenas uma parcela dos nós atua em cada iteração.

a Algoritmo LMS em Rede Distribuída

O algoritmo LMS [8] tem o objetivo de encontrar o vetor de parâmetros **w** ótimo, que relaciona o vetor de regressão **x** com a medida escalar y, satisfazendo y $=$ **w**$\cdot$**x**, calculando o erro médio quadrático para minimizar sua função custo E[(y - $\mathbf{x}^T \cdot \mathbf{w}$)^2]. A cada iteração i é calculado o erro $=$ y - $\mathbf{x}^T \cdot$ **w**(i-1) e atualiza **w**, **w**(i) $=$ **w**$(i-1) + u \cdot erro \cdot$ **x**, onde u é o tamanho de passo, termo escalar que determina a taxa de ajuste de **w** em relação ao seu erro.

Foram realizadas diversas simulações, a procura do valor do tamanho de passo u, que minimiza o erro, em uma rede de 20 nós. Como condição inicial das simulações todos os termos do vetor **w** foram zeradas e realizadas 300 repetições, tal número de repetições foi escolhido para diminuir a influência de outliers na análise.

Após otimização foi obtido o algoritmo com o valor ótimo do tamanho de passo u $=$ 0,06, convergindo na iteração 141 e erro médio quadrático MSE podendo atingir 10^{-33}. A Figura 15 é sua curva de aprendizado, onde o eixo y é valor do MSE e o eixo x as iterações do processo de aprendizado do algoritmo.

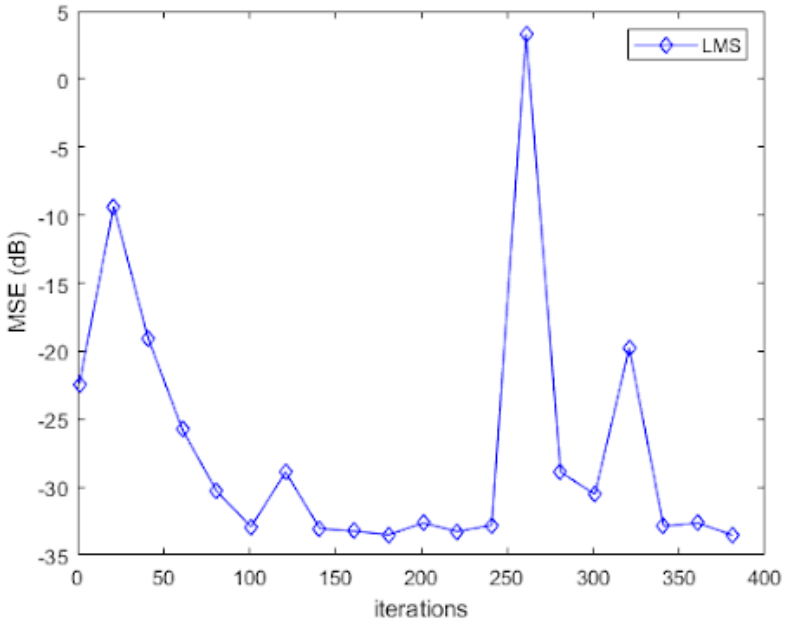

Figura 15: Gráfico MSE Rede Distribuída LMS

O problema desse algoritmo é sua relativa instabilidade, pois conforme visto no gráfico acima, mesmo após sua convergência há grandes picos de erro esporádicos, conforme visto nos instantes 261 e 321. Tais picos, mesmo que decaem rapidamente, podem representar problemas sérios, no caso de uma implementação desse algoritmo em um cenário real onde decisões críticas são tomadas com base nas previsões do algoritmo.

b Algoritmo RLS em Rede Distribuída

Neste algoritmo RLS [9, 10] também busca encontrar o **w** ótimo, utilizando uma regressão rígida com parâmetro de regularização q, para dar mais peso às bases do espaço com maior desvio padrão, e recursão temporal por meio de um fator de esquecimento b, permitindo ao algoritmo melhor capacidade de adaptação aos dados mais recentes. Por fim, calculando o ganho de Kalman e multiplicando pelo erro para atualizar a estimativa de **w**.

Houve uma exploração do espaço de parâmetros de q e b, a fim de otimizar o desempenho do algoritmo. Novamente as simulações foram realizadas numa rede com 20 nós, havendo 300 repetições e iniciando todos os termos vetor **w** com 0. O melhor desempenho do algoritmo foi obtido para q = 0,01 e b = 0,685, tendo sua convergência na iteração 141 e MSE de 10^{-32}. A Figura 16 mostra como é suave a adaptação do algoritmo RLS, praticamente não havendo oscilações antes ou após a convergência, explicitando a precisão do algoritmo.

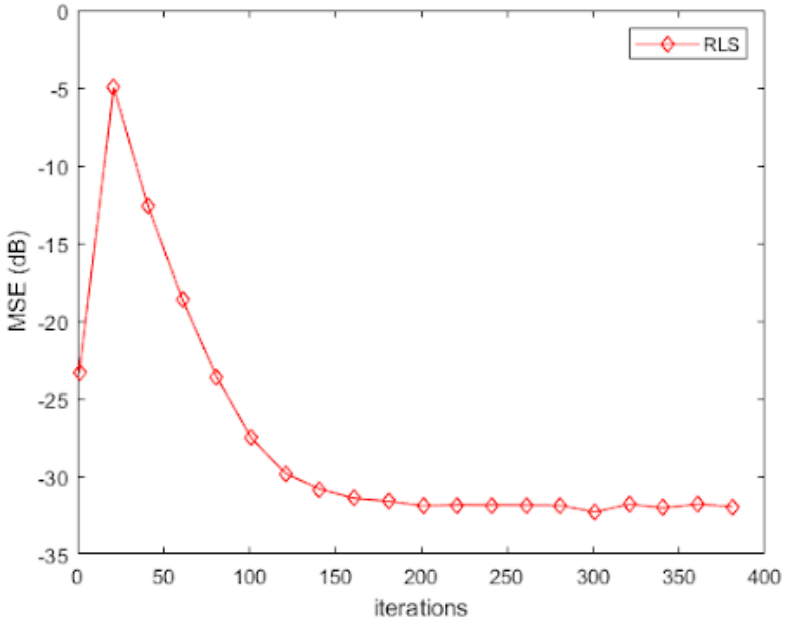

Figura 16: Gráfico MSE Rede Distribuída RLS

Podemos ver na Figura 17 as curvas de aprendizado dos algoritmos LMS e RLS, podendo constatar que, apesar de poder atingir um MSE ligeiramente menor, o algoritmo LMS foi consideravelmente mais instável que o RLS.

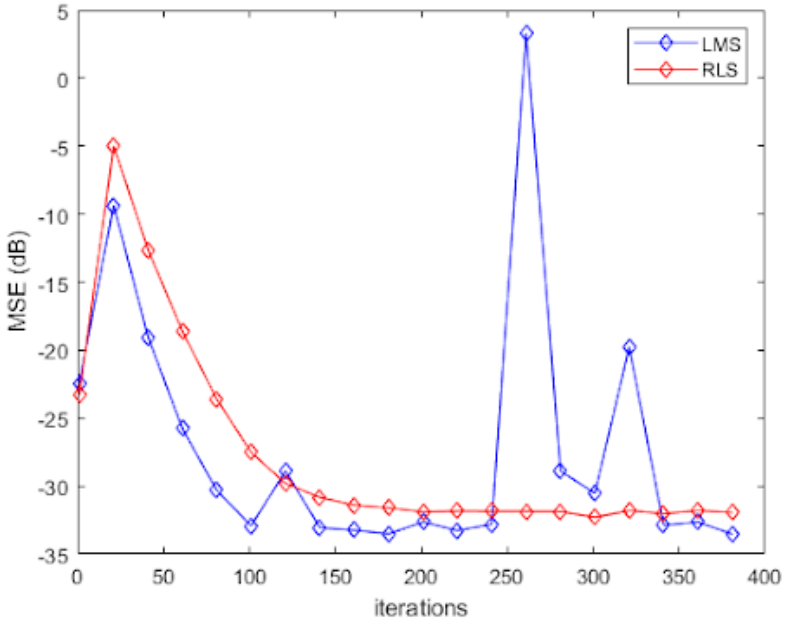

Figura 17: Gráfico MSE Rede Distribuída LMS e RLS

c Algoritmo Próximo em Rede Distribuída

O desenvolvimento desse algoritmo Próximo [11, 12] na forma distribuída foi o principal objetivo dessa etapa da pesquisa, portanto uma explicação mais detalhada é necessária para explicitar sua aplicabilidade e lógica de funcionamento. Assim como os algoritmos citados anteriormente, o algoritmo próximo é utilizado em problemas convexos de otimização, sendo bastante utilizado em casos de minimização suaves, de pequena a grande escala e envolvendo conjuntos de dados com grandes dimensões.

Diferente dos algoritmos tradicionais que focam em operações lineares com gradientes, no algoritmo próximo é avaliada uma função conhecida como operador próximo expressa por $prox[pu]f(p) = argmin(f(\mathbf{x}) + (1/(2pu))||\mathbf{x} - p||^2)$ [13] , sendo pu o tamanho de passo e envolvendo um problema menor de otimização, geralmente tratado pelos algoritmos tradicionais. O processo iterativo do algoritmo é definido por $\mathbf{w}(i+1) = prox[pu]f(\mathbf{w}(i))$, sendo o operador próximo uma função $f(x) : \mathbb{R}_n \to \mathbb{R}$ convexa e i a iteração do algoritmo. Se $f(\mathbf{w})$ tem um mínimo, os termos de $\mathbf{w}(i)$ convergem para os valores que o minimizam essa função.

O algoritmo próximo foi desenvolvido utilizando o LMS para resolver o problema menor de otimização, sendo incluso uma etapa de avaliação do limite, onde a partir do erro registrado das iterações anteriores, limitadas pelo filter tap é determinado um limite máximo para o valor do erro, distinguindo outliers. Nessa etapa utilizamos a constante lambda, para ponderar a dependência do processo entre o erro atual e os anteriores, e a constante cons como fator de esquecimento, com a finalidade de obter o valor de tolerância para o limite.

Após diversas simulações com 300 repetições e iniciando todos os termos do vetor **w** com 0 em uma rede com 20 nós variando os valores dos parâmetros pu, cons, lambda e constante de atualização rho, foram obtidos valores que proporcionam o melhor desempenho do algoritmo. A melhor performance do algoritmo ocorreu para pu = 0,06, cons = 20, lambda = 0,95 e rho = $2 \cdot 10^{-20}$, tendo sua convergência na iteração 81 e MSE de 10^{-33}. A Figura 18 mostra os três algoritmos otimizados, sendo visível a suavidade da curva de aprendizado do algoritmo próximo, não havendo oscilações bruscas em nenhum momento antes ou após convergência, evidenciando a sua precisão.

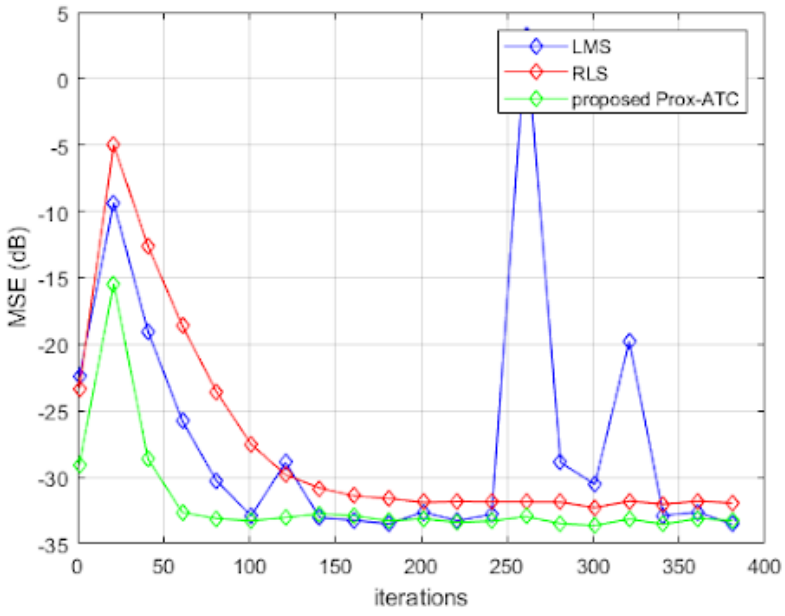

Figura 18: Gráfico MSE Rede Distribuída LMS, RLS e Próximo

É possível observar na Figura 18 que o algoritmo próximo consegue obter MSE comparáveis aos melhores do LMS e ser tão estável quanto o RLS, como também se ajusta aos dados mais rapidamente que ambos os algoritmos tradicionais, convergindo em menos iterações. Com essa análise é possível afirmar a grande eficiência do algoritmo próximo em uma rede distribuída, ultrapassando métodos tradicionais. No próximo tópico será averiguado o desempenho desses algoritmos em um cenário de aprendizado federativo.

d Algoritmos no Cenário de Aprendizado Federativo

Aprendizado federativo é uma técnica de aprendizado distribuído de máquina caracterizado pela colaboração através do compartilhamento de modelos de previsão para o nó central, mas mantendo os dados de treinamento nos próprios nós, garantindo segurança e privacidade dos dados individuais. Nesse modo, o nó central transmite para cada nó os modelos dos demais, o qual então ajusta seus próprios parâmetros. [14, 15]

Por não ser armazenado os dados de treinamento de cada nó, diminui a necessidade de armazenamento de dados nos nós, por isso essa técnica de aprendizado é utilizada em casos que envolvem com grandes números nós. Outra característica desse cenário de aprendizado é que tal técnica não requer participação de todos os nós durante cada iteração, garantindo maior robustez, pois problemas internos ou de comunicação de cada nó não interferem no funcionamento do processo total de aprendizado.

Após diversas rodadas de simulações numa rede com 20 nós, havendo 300 repetições e iniciando todos os termos vetor **w** com 0, a melhor performance do algoritmo ocorreu para pu $=$ 0,06, cons $=$ 20, lambda $=$ 0,95 e rho $= 2 \cdot 10^{-20}$, tendo sua convergência na iteração 61 e MSE de $10^{-30,5}$. A Figura 19 mostra os três algoritmos otimizados, sendo visível a eficiência do aprendizado do algoritmo próximo, tendo erro inicial muito pequeno e sendo estável durante todo o processo de adaptação, não oscilando mesmo após a convergência.

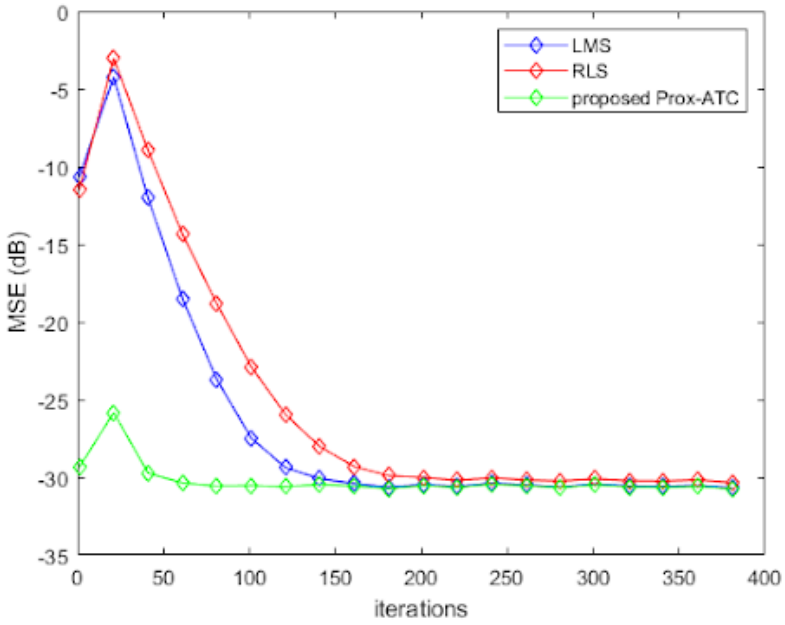

Figura 19: Gráfico MSE Rede Cenário Federativo LMS, RLS e Próximo

Podemos ver na Figura 19 que o algoritmo próximo novamente obteve MSE iguais ou menores que os do LMS e RLS, além disso convergindo com cerca de 120 a menos iterações que os algoritmos tradicionais, demonstrando um processo de aprendizado consideravelmente mais rápido. Assim como no caso cenário distribuído anterior, o algoritmo próximo também teve um melhor desempenho que os algoritmos de aprendizado tradicionais, sendo ideal para aplicações reais, onde é necessário reações rápidas e precisão.

5 Conclusões & Trabalhos Futuros

Ao projetar algoritmos de aprendizado distribuídos para redes, cada parte do processamento é crítica para o desempenho do algoritmo, sempre tendo em mente as especificidades da topologia da rede e o perfil dos dados coletados. A metodologia ATC empregada no processamento distribuído pode fornecer mais informações sobre o parâmetro desconhecido **w** sem aumentar o custo computacional, sendo a que obtive melhor desempenho em todos os algoritmos testados. A regra de combinação metrópoles, muito utilizada por apresentar ótimos resultados pela matriz de combinação duplamente estocástica, proporcionou também excelente nos casos estudados, porém em situações onde o ruído é muito alto outras regras podem obter melhor desempenho.

Na comparação entre os algoritmos do nó solitário e a rede incremental, com algoritmo RLS, foi evidenciada a melhora proporcionada pela rede, obtendo em um número próximo de iterações um MSE médio final dez vezes menor, além de uma curva de aprendizado mais estável. Comprovando assim, os benefícios de algoritmos distribuídos na tarefa de aprendizagem.

Comparando os algoritmos de aprendizado na rede distribuída, foi evidente a eficiência do algoritmo próximo, convergindo em um número menor de iterações que os algoritmos tradicionais e atingindo MSE igual ou inferior aos demais. Além de possuir uma curva de aprendizado extremamente estável, sendo ideal para aplicações em cenário real.

No aprendizado federativo também é visível a estabilidade dos algoritmos, que com exceção do LMS, não apresentaram oscilações abruptas, apesar de atingir um MSE maior do que o da rede distribuída. Neste caso, novamente foi possível observar o melhor desempenho do algoritmo próximo em relação aos demais, apresentando conversão ainda mais rápida e menor MSE que os algoritmos de aprendizado tradicionais.

Tendo, sido cumprido a premissa deste trabalho de estudos e teste dos algoritmos em cenários distribuídos, seja em simulações ou casos reais e validando a importância das redes. Para trabalhos futuros, outros algoritmos de aprendizado poderiam ser avaliados e outro tópico de interesse seria o efeito da topologia das redes no desempenho dos algoritmos.

Referências

[1] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[2] Y. Yu, H. Zhao, R. C. de Lamare, Y. Zakharov, and L. Lu, "Robust distributed diffusion recursive least squares algorithms with side information for adaptive networks," *IEEE Transactions on Signal Processing*, vol. 67, no. 6, pp. 1566–1581, 2019.

[3] F. S. Cattivelli and A. H. Sayed, "Diffusion lms strategies for distributed estimation," *IEEE transactions on signal processing*, vol. 58, no. 3, pp. 1035–1048, 2009.

[4] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani, "Random generation of combinatorial structures from a uniform distribution," *Theoretical computer science*, vol. 43, pp. 169–188, 1986.

[5] L. Herault and R. Horaud, "Figure-ground discrimination: a combinatorial optimization approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 899–914, 1993.

[6] Y. Yu, H. He, T. Yang, X. Wang, and R. C. de Lamare, "Diffusion normalized least mean m-estimate algorithms: Design and performance analysis," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2199–2214, 2020.

[7] J. Jeong and D. Culler, "Incremental network programming for wireless sensors," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.* IEEE, 2004, pp. 25–33.

[8] R. Arablouei, S. Werner, Y.-F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Transactions on Signal Processing*, vol. 62, no. 2, pp. 472–484, 2014.

[9] A. H. Sayed and C. G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *2006 fortieth asilomar conference on signals, systems and computers*. IEEE, 2006, pp. 233–237.

[10] A. Danaee, R. C. de Lamare, and V. H. Nascimento, "Distributed quantization-aware rls learning with bias compensation and coarsely quantized signals," *IEEE Transactions on Signal Processing*, vol. 70, pp. 3441–3455, 2022.

[11] N. G. Polson, J. G. Scott, and B. T. Willard, "Proximal algorithms in statistics and machine learning," *Statistical Science*, vol. 30, no. 4, pp. 559–581, 2015.

[12] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.

[13] M. Hong and T.-H. Chang, "Stochastic proximal gradient consensus over random networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 11, pp. 2933–2948, 2017.

[14] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

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