Monitoramento de cargas elétricas residenciais - Aprendizado Incremental

André Felipe Correia de Oliveira
Engenheiro de Controle e Automação
Email: afelip.co@gmail.com

***Resumo*—A aplicação da inteligência computacional e da computação ubíqua são pontos importantes no conceito de redes elétricas inteligentes. É apresentado então, um sistema ciber físico com conexão a Internet sem fio, capaz de extrair características significativas do sinal de corrente da rede elétrica. Essas características, coletadas de equipamentos funcionando de forma isolada, são transformadas então em amostras de entrada de um modelo de classificação treinado com aprendizado incremental. A ideia de um modelo incremental traz uma alternativa aos altos custos de ter que se adquirir os dados de todos os equipamentos de uma residência, para só depois criar um classificador para identificação das cargas. Tenta-se também trazer redução aos altos custos de sistemas tradicionais de monitoramento de cargas elétricas, onde é necessário a instalação de sensores em diferentes pontos. O experimento de aprendizado incremental foca na adição de classes, onde o modelo pode aprender novos equipamentos que são apresentadas com o tempo. Um experimento inicial com cinco classes é apresentado, sendo duas aprendidas de forma incremental. Os resultados estatísticos finais por classe, são apresentados então, com valores de acurácia satisfatórios, acima de 98%, após realização dos treinamentos de atualização do modelo..**

Keywords*—Desagregação de Energia; Aprendizado Incremental; *Learn++*; *Smart Grids

I. Introdução

O presente trabalho apresenta a aplicação de um algoritmo de aprendizado de máquina incremental para o problema de desagregação de energia em um contexto residencial. O objetivo principal é traçar as bases de criação de um sistema de Internet das coisas (IdC), para casas inteligentes, capaz de auxiliar nas decisões de aumento de eficiência energética e redução dos gastos com a energia.

A desagregação da energia é um conceito que diz que a partir da medição de uma fonte central, pode-se dividir o consumo total, entre os diferentes equipamentos elétricos que foram ligados, estudando apenas os dados gerados ao longo do tempo.

Este conceito é tratado em aprendizado de máquina como um problema de classificação. A partir de características extraídas dos sinais da rede elétrica, como tensão e corrente, faz-se a identificação de qual classe de equipamento está ligada. Dentro deste contexto, George W. Hart introduz as ideias de monitoramento de carga não-intrusivo residencial no *Massachusetts Institute of Technology* (MIT) na década de 80, em [1], [2] e [3].

Modelos de aprendizado supervisionado clássicos, exigem um esforço grande na obtenção prévia de amostras rotuladas de forma correta, o que pode gerar grandes custos na criação das bases de dados. Outros modelos, podem considerar que os dados rotulados são entregues de forma gradual, atualizando o "conhecimento" intrínseco ao modelo conforme necessário. Estes são conhecidos como modelos de aprendizado incremental ou aprendizado *on-line*.

O aprendizado incremental pode identificar variações de conceitos dentro dos dados que representam o problema. Dentre as variações de conceitos, podem ser citadas as mudanças nas distribuições das características representantes das amostras, adição de novas classes aos problemas de classificação, entre outros, conforme apresentado em [4].

Baseando-se no monitoramento não intrusivo, apresentado por Hart, foi desenvolvido um sistema ciber físico para a extração de características do sinal de corrente elétrica de um ponto de fonte de energia. Os dados adquiridos são comunicados via internet com um computador onde estará um modelo classificador para detecção de qual equipamento foi ligado. Este é um modelo de aprendizado incremental para aprender novas classes, sendo que os equipamentos (classes) são apresentados em momentos distintos no tempo.

A Figura 1 apresenta o fluxo do sinal no problema de desagregação. As linhas contínuas apresentam o fluxo do sinal para identificação do equipamento. As linhas tracejadas, representam os fluxos de treinamento do modelo. Em sua inicialização e quando é apresentado uma nova classe. Os blocos em marron, representam o sistema embarcado responsável pela extração de características. A identificação dos equipamentos ligados é o objetivo principal do sistema, representado pelo bloco em verde.

Para a implementação do modelo incremental, utilizou-se o algoritmo **Learn++** para aprendizado incremental, apresentado por Robi Polikar et. al em [5], que é baseado na mistura de classificadores "fracos"por meio do procedimento de votação por maioria ponderada.

Na seção II é aprofundado o conceito de desagregação de energia, e como foi introduzido o conceito de monitoramento de carga não intrusivo. Na seção III é discutido o algoritmo criado por Polikar. Na seção IV é apresentado o processo, em sistemas embarcados, de extração de características em tempo real, do sinal de corrente da rede elétrica. Na seção V é apresentado então o experimento de adição de classes, e seus resultados, do aprendizado incremental aplicado a desagregação de energia com o algoritmo *Learn++*.

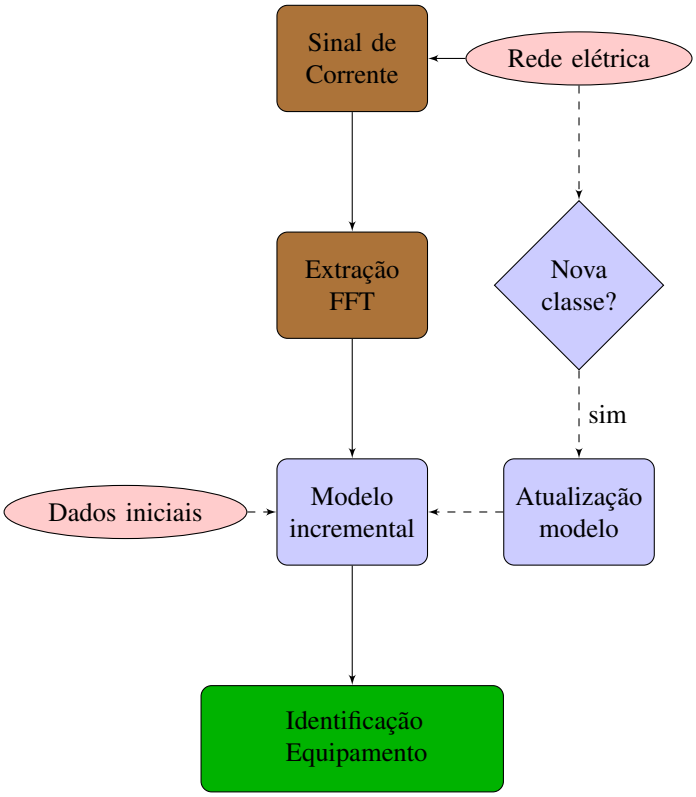

Figura 1. Fluxo do sinal modelo incremental

II. Desagregação de Energia não intrusiva

A instrumentação tradicional, nos sistemas de monitoramento de carga, envolve um sistema complexo de hardware de coleta de dados. Pontos de monitoramento em cada aparelho de interesse e um meio de comunicação conectando cada aparelho a um ponto central de coleta de dados, fornecendo caminhos de dados separados. Em contrapartida, possuem um software simples, que meramente precisa tabular as informações que chegam nesses canais de hardware separados.

O conceito de monitoramento não intrusivo do Hart inverte essa lógica. Em [3], tem-se um único ponto de hardware para medição, e esses dados coletados são enviados para um software, que aplica técnicas computacionais avançadas para detectar qual equipamento está ligado, para assim tabular os dados de consumo conforme identificação.

A medição em um ponto único, coleta dados das formas de onda da corrente e tensão. Em seguida são aplicadas, aos sinais, técnicas de procesamento de sinais para extrair assinaturas que caracterizam as mudanças ocorridas na rede. As assinaturas não intrusivas são aquelas que podem ser medidads observando passivamente a operação normal da rede, por exemplo, uma mudança de passo na potência medida.

No conjunto de assinaturas não intrusivas, existe uma divisão quanto ao estado dos aparelhos. Se a informação sobre o estado do aparelho está continuamente presente no sinal, enquanto o aparelho opera, essa é uma assinatura de ”estado estacionário”. Por sua vez, se a informação obtida aparece brevemente no tempo de transição de estado dos aparelhos, esta é classificada como uma assinatura ”transitória”.

Os trabalhos iniciais de Hart, basearam-se em um plano de assinaturas de duas dimensões ΔP - ΔQ. Eles representam respectivamente as variações de potência ativa e reativa, nas curvas de potência ao longo do tempo. Essas são assinaturas de estado estacionário visto que, a variação continuará presente durante todo o tempo em que o aparelho estará ligado. Este método inicial possui algumas desvantagens, como a dificuldade de se identificar aparelhos com diversidade de estados, que variam sua carga, e também em distinguir aparelhos que funcionam sob condições de P e Q similares [6]. Como solução para o problema de equipamentos com múltiplos estados de funcionamento, Hart propõe modelar as transições de estado dos equipamentos como máquinas de estados finitos (*Finite State Machines* - FSM).

Como solução para aparelhos operando com potências similares, foi proposto em [7] e [8], a obtenção dos harmônicos do sinal de corrente, como assinaturas de estado estacionário complementar as assinaturas de variação de potência.

Os harmônicos são análises matemáticas realizadas com a transformada de Fourier para analisar a distorção de uma forma de onda senoidal. De forma computacional essa transformada é obtida com o uso do algoritmo de transformada rápida de Fourier (*Fast Fourier Transform*- FFT).

O THD (*Total Harmonic Distortion*) é um índice que contabiliza a quantidade de harmônicos em uma onda, demonstrando o quanto uma ela está distorcida. Outro índice que apresenta informações sobre a distorção de uma onda é o fator de crista. Na tese de mestrado [9], é apresentado um estudo que relaciona estes índices com os equipamentos eletro-eletrônicos.

As assinaturas transitórias são introduzidas por Leeb e Shaw em [10], que a partir de 1995 dão continuidade ao trabalho de Hart no MIT. As assinaturas transitórias não são abordadas neste trabalho, pois, toda a extração de características realizadas focou em assinaturas de estado estacionário.

Em [11] e [12] é apresentado um sistema distribuído de monitoramento em tempo real, para coleta de dados em painéis elétricos principais das residências. Esse sistema surge como suporte a uma política que visa compensar os consumidores com preços mais baixos em combustíveis. O bônus advém da disponibilização do controle de alguns aparelhos, para que seja realizado um balanceamento das cargas, principalmente em momentos de alta demanda. Com a informação de quais equipamentos estão ligados, é possível decidir com um mínimo impacto ao consumidor.

Este sistema foi baseado na detecção de borda da potência ativa e uma tabela de correspondência simples para identificar o estado de cada aparelho. A acurácia relatada nesta pesquisa foi de 60% a 90%.

III. Learn ++

Parte das aplicações de reconhecimento de padrões necessitam de conjuntos de treinamento representativos, o que

pode gerar, na prática, um custo alto e um grande consumo de tempo. Por isso é comum em alguns problemas a disponibilização de pequenos conjuntos de treinamento em periodicamente.

Nestes casos são necessários classificadores capazes de aprender novas informações sem comprometer o desempenho de reconhecimento dos dados já treinados anteriormente. Este é o processo de aprendizado incremental. Neste processo pode ser necessário o esquecimento de algum conhecimento adquirido anteriormente, para que o classificador seja capaz de aprender uma nova informação [5].

De forma geral, pode-se considerar um algoritmo de aprendizado incremental aquele que segue os seguintes critérios [5]:

- Capaz de aprender informações adicionais a partir de novos dados;
- Não deve exigir acesso aos dados originais, usados para treinar o classificador existente;
- Deve preservar conhecimentos previamente adquiridos (isto é, não sofrer de um catastrófico esquecimento);
- Capacidade de acomodar novas classes que podem ser introduzidas com novos dados.

O *Learn++* foi proposto então para ser um algoritmo incremental que satisfaça a todos esses critérios, e capaz de superar desvantagens apresentadas em algoritmos incrementais anteriores, como o baseado na lógica nebulosa (*fuzzy*) **ART-MAP**, apresentado por Carpenter et. al. em [13] e [14].

O *Learn++* tem sua inspiração no algoritmo **AdaBoost**, apresentado por Freund et al. em [15] e [16]. Em essência, tanto o *Learn++* quanto o *AdaBoost* geram um conjunto de classificadores "fracos", sendo cada um treinado usando uma distribuição diferente de amostras de treinamento. As saídas desses classificadores são então combinadas usando o esquema de votação majoritária, apresentado por Littlestone e Warmuth em [17], para se obter a regra de classificação final.

O *Learn++* tem como princípio a geração de novos classificadores a cada iteração. A seleção do conjunto de treinamento é baseado em uma distribuição que vai sendo atualizada ao longo das iterações, com o intuito de que os dados que estão sendo classificados erroneamente apareçam mais nos novos treinamentos. O *Learn++* herda todas as propriedades de melhoria do *AdaBoost* [5], apesar da regra de atualização de distribuição do *AdaBoost* ser otimizada para melhorar a precisão do classificador, enquanto a regra de atualização da distribuição do *Learn++* é otimizada para aprendizado incremental de novos dados [5].

As entradas do algoritmo são:

- Os dados de treinamento, divididos em grupos. Processo de batelada;
- Um conjunto de teste, que tenha amostras de todas as classes do treinamento;
- Um classificador "fraco", que será base de todo novo classificador adicionado a mistura;
- Valor de iterações do algoritmo, T_k a ser rodada por *batch*, que representará o total de classificadores criados por *batch*.

A cada iteração $t = 1, 2, \cdots, T_k$ é gerado uma hipótese h que mapeia o espaço de entrada X para o espaço de saída Y. Cada hipótese aprende apenas uma porção do espaço de entrada usando conjuntos diferentes. Estes conjuntos são baseados em uma distribuição D_t para, $t = 1, 2, \cdots, T_k$. A distribuição é obtida normalizando um conjunto de pesos atribuídos a cada instância com base no desempenho de classificação dos classificadores nessa instância.

$$D_t = \frac{d_i}{\sum_{i=1}^{m} d_i}$$

onde d_i é o vetor de pesos, e m é o número total de amostras no *batch* de treinamento. A inicialização da distribuição assim como do vetor de pesos é igualitária, e considera a probabilidade a priori de cada amostra no conjunto. Ao longo das iterações a distribuição D e os valores de d_i são constantemente atualizados, conforme o desempenho dos classificadores.

A cada novo classificador treinado, é calculado o erro da hipótese no conjunto de treinamento, e o erro da mistura final que considera a votação dos classificadores anteriores. Caso esse erro dos classificadores supere um erro considerado grande, pode-se descartar essa hipótese e treinar uma nova. Essa é uma forma de esquecimento para o algoritmo.

Nos passos seguintes, é verificado o erro da composição das hipóteses e é feita a atualização dos valores da distribuição D, para aumentar as chances das amostras não classificadas corretamente de aparecer mais vezes na próxima seleção.

Assim é realizado o treinamento do conjunto de classificadores(*ensemble*). A cada iteração uma nova hipótese é acrescida. O classificador final é o conjunto de classificadores com decisão por votação majoritária. Esse classificador vai aprendendo incrementalmente a cada novo conjunto de dados inseridos no treinamento. Os pesos de cada classificador é definido por seu erro no conjunto de treino.

Dentro do contexto do treinamento do *Learn++*, tem-se um possível problema chamado de "*outvoting*", que é quando tem-se novos classificadores capazes de identificar as novas classes presente nos dados, porém esses classificadores são penalizados, pois dentro da mistura possuem pesos insignificantes.

Uma solução dentro da literatura, foi proposta em [18] pela equipe do Polikar, onde eles discutem mais a fundo o problema do "*outvoting*", e apresentam o algoritmo *Learn++.NC* (*New Classes*). A modificação para este algoritmo não é implementada neste trabalho, sendo o problema do "*outvoting*"resolvido definindo-se número de iterações diferentes para cada treinamento, como será visto na seção V.

O algoritmo *Learn++* aplicado aqui, foi implementado com a linguagem R, baseado em uma implementação em Matlab chamada "*IncrementalLearning*", de autoria do usuário **gditzler** (Gregory Ditzler) do GitHub, e disponibilizada no link https://github.com/gditzler/IncrementalLearning.

IV. Aplicação residencial sem fio

A ideia da criação de um sistema ciber físico, para extração dos harmônicos do sinal de corrente, surge junto as tendências

do momento atual de redes elétricas inteligentes (*Smart Grids*) e Internet das coisas (*Internet of Things* - IoT).

Para aquisição do sinal é utilizado o sensor de corrente por efeito Hall Yhdc modelo SCT-013-000. Ele é um sensor não intrusivo de corrente alternada, fabricado pela empresa *Beijing Yaohuadechang Electronic Co.*. Pelas especificações técnicas, ele tem um limite máximo de medição de 100 A, e uma redução por efeito magnético de 1:2000. Logo a saída máxima nos terminais de saída do sensor é de 50 mA. Apesar do sensor alcançar altas correntes, ele foi instalado em uma tomada de corrente máxima de 15 A. Esta tomada funciona como ponto principal de energia da coleta de amostras dos equipamentos.

A saída do sensor de corrente é conectada ao conversor analógico/digital (*Analog/Digital Conversor* - ADC) do controlador digital de sinal (*Digital Signal Controller* - DSC). Segundo a fabricante, a família C2000 foi desenvolvida para atender a crescente demanda do mercado para soluções de monitoramento em tempo real.

Entre a saída do sensor de corrente e o ADC do F28335 existe um circuito condicionador de sinal para alteração do valor 0V de referência, e o range de variação, já que o ADC aceita apenas tensões entre 0 e 3.3V.

A frequência de amostragem do ADC foi configurada no valor de 12kHz. Um vetor de tamanho 1024 salva os valores registrados pelo ADC. Esse vetor será a entrada para o cálculo da transformada de Fourier do sinal de corrente. A frequência de amostragem e o tamanho do vetor de entrada da FFT está intrinsecamente relacionado aos valores de frequência que representam os harmônicos do sinal da corrente com frequência natural igual a 60 Hz.

Observa-se também o quanto esse vetor representa o sinal real. Como a frequência de amostragem é de 12kHz, pode-se dizer que o tempo total representado pelo vetor de tamanho 1024 é igual a:

$$t = \frac{1}{F_s}N = \frac{1}{12000}1024 = 85,33ms$$

Sabendo que a onda do sinal da rede tem frequência natural de 60 Hz, o período é igual a:

$$T = \frac{1}{60} = 16,66ms$$

Logo, com a razão de t sobre T, tem-se o número aproximado de ondas representadas no vetor, que é aproximadamente:

$$n = \frac{t}{T} = 5,12$$

Esse valor mostra que o vetor de corrente de entrada da FFT representa bem o sinal de corrente de cada equipamento.

A. Extração de características

A família C2000, de controladores digitais, da Texas Instruments já vem com a biblioteca "FPU"(*Float Point Unit*), que possui a implementação do algoritmo de FFT aplicado para um vetor de floats. Este vetor de floats de entrada será o sinal da corrente convertido para seu valor em tensão entre 0 e 3.3V.

Conforme o teorema da amostragem de Nyquist, a maior frequência que é possível representar na transformada discreta de Fourier é metade da frequência de amostragem. Ainda relacionando a FFT com os parâmetros definidos na amostragem, as frequências em que cada amostra foi definida pode ser obtida por:

$$f = \frac{F_s}{N}n$$

com n assumindo valores entre 0 e metade do tamanho do vetor, onde encontra-se a representação da frequência 6kHz. A outra metade dos valores do vetor tem resposta simétrica. Esse vetor de magnitudes é a assinatura estacionária que será enviada para o software de classificação. Essa comunicação ocorre através da Internet, logo antes é necessário a comunicação entre o F28335 e um microcontrolador com periférico de rádio comunicação.

Para comunicação com a Internet, é necessário um dispositivo com capacidade de acesso a rede. O hardware escolhido para a comunicação com a Internet é o ESP32 LoRa desenvolvido pela Heltec. Este dispositivo possui três periféricos de comunicação de rádio frequência, bluetooth, Wi-Fi e LoRa. Ele ainda possui duas unidades de processamento, além dos pinos de entrada/saída GPIOs, periféricos utilizados para comunicação nos protocolos SPI, UART, I2C e CAN. A comunicação entre o F28335 e o ESP32 é feita com o protocolo SPI. O ESP32 recebe do controlador digital de sinal o vetor de magnitudes da FFT a cada 256 milissegundos.

Em seguida é usada uma arquitetura de rede servidor-cliente, onde é feito o envio de dados através de sockets, usando o protocolo de datagrama de usuário (UDP - *User Datagram Protocol*), que permite que aplicações enviem datagramas encapsulados em pacotes IPv4 ou IPv6. A entrega dos pacotes no protocolo UDP não é certa. Pode se dizer que o UDP é um serviço não direcionado a conexão, pois, não há necessidade de manter um relacionamento longo entre cliente e o servidor. O cliente é o ESP32 e o servidor é implementado em Python e salva os dados coletados em arquivos de texto com valores separados por vírgula (CSV - *Comma Separated Values*).

Existe um limite de 1024 bytes no tamanho dos datagramas que podem ser enviados pela biblioteca de socket WiFi do ESP32. Logo, não há o envio de todo o vetor de magnitude para o servidor UDP. Também não é garantido que todo o vetor de magnitude enviado possui o mesmo tamanho. Há ainda perdas no valor de marcação do tempo dos dados, obtidas através do protocolo NTP. A solução destes problemas estão entre os próximos passos de desenvolvimento do sistema.

Devido a essa limitação de 1024 bytes, o aprendizado incremental irá considerar somente os 200 primeiros valores de magnitude da FFT. Esses 200 primeiros valores trazem informações da magnitude até a frequência v. A Figura 2 apresenta o espectrograma construido a partir de amostras do

sinal de corrente de um ventilador doméstico, obtidas ao longo do tempo, e salvas em um arquivo CSV.

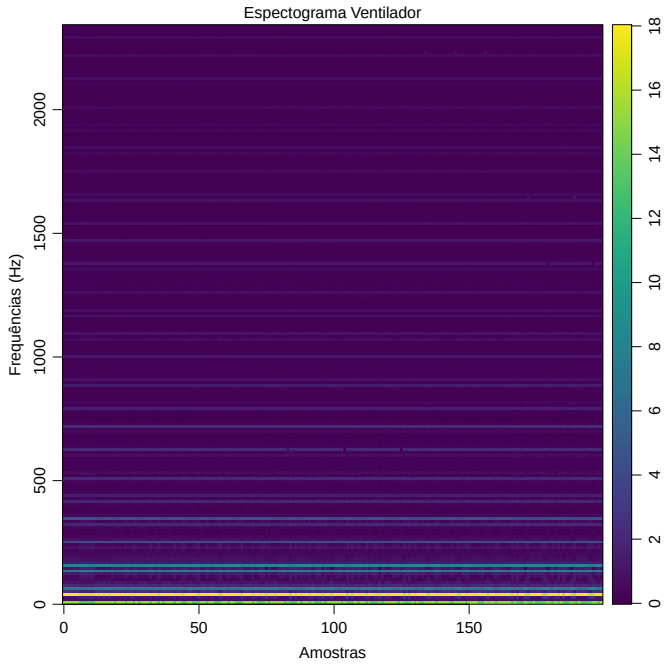

Figura 2. Espectograma do sinal de corrente do ventilador

O eixo x representa amostras no tempo. Mais precisamente, são amostras do vetor de magnitude da transformada de Fourier a cada 256 milissegundos. A Figura 2 demonstra o conceito de assinaturas de estado estacionário, onde tem-se características que se mantém enquanto o equipamento está ligado. Como conseguimos a representação até 2343,75 Hz, temos informações até o trigésimo nono harmônico da frequência natural de 60 Hz. As 200 amostras da Figura 2 representam um tempo total de aproximadamente 51 segundos.

B. Equipamentos

O espectrograma do ventilador, apresentado na Figura 2, representa as características extraídas para uma classe. Para a aplicação do aprendizado incremental, foram coletados mais 4 sinais distintos. O sinal de novos três equipamentos e também foi coletado o sinal quando nenhuma carga está ligada. Os quatro equipamentos são um ventilador modelo Turbo Silencio da marca Arno, um televisor modelo led tv monitor da marca Philips, um carregador de celular Moto G5 Turbo da marca Motorola e um MacBook Pro Retina de 15 polegadas.

A Tabela I apresenta a quantidade de amostras de cada classe após serem divididos em 80% dados de treino e 20% dados de validação. Os dados de treinamento são posteriormente divididos em *batches* que são treinados de forma sequêncial, de modo que a novo treinamento é adicionado uma nova classe. O processo de treinamento incremental é apresentado na seção seguinte.

Apesar de apresentar um conjunto com classes desbalanceadas, como nosso classificador base usado é um algoritmo

Tabela I
CLASSES EQUIPAMENTOS

Classe	Equipamento	Treinamento	Teste
Class 1	TV	1636	410
Class 2	MacBook	800	200
Class 3	Ventilador	592	149
Class 4	Carregador	585	147
Class 5	Vazio	378	95

Tabela II
CLASSES EQUIPAMENTOS

Treinamento	Classes	Iterações
1	1, 2, 3	4
2	1, 2, 3, 4	6
3	1, 2, 3, 4, 5	10

de arvore de decisão, esse desbalanceamento não apresentou problema. As árvores de decisão tem algoritmos baseado na criação de regras para a divisão dos dados, o que não depende exatamente do número de dados, mas depende sim do quanto as características representam o problema.

V. APRENDIZADO INCREMENTAL

Após a coleta dos dados é aplicado então o aprendizado incremental para aprendizado de novas classes com o algoritmo *Learn++*. Nesse sentido foi necessário dividir o conjunto de treinamento em grupos menores, e cada treinamento ocorre com um número de classes diferentes.

Dentro do conceito de aprendizado incremental, um método importante é a identificação de "*outliers*". Os "*outliers*"são os dados que apresentam variações de conceitos. No problema tratado aqui, a identificação de "*outliers*"se daria por meio da identificação de que um novo sinal não visto, não pertence a nenhuma das classes já treinadas. Apesar de ser um objetivo tornar o treinamento do aprendizado incremental autônomo, esta é uma parte que ainda não esta implementada neste trabalho. O objetivo central do experimento foi verificar se o algoritmo *Learn++* conseguia adquirir novo conhecimento sem perder o conhecimento anterior adquirido.

A Tabela II apresenta quais as classes presentes em cada um dos treinamentos realizados. Tem-se um total de três treinamentos . No segundo treinamento é adicionado a classe 4, e no terceiro a classe 5.

Em cada treinamento tem-se dois conjuntos de treino contendo as amostras das classes definidas pela Tabela II. Um dos parâmetros do algoritmo é o número de iterações por grupo treinado, e esse número é igual ao total de classificadores criados por grupo.

Como visto anteriormente, o *Learn++* sofre do problema de "*outvoting*". Para evitar esse problema cada treinamento apresenta um número de iterações diferentes que podem ser visto na Tabela II.

Também é necessário um classificador "fraco"para ser a base de todos os classificadores treinados. O modelo escolhido foi a árvore de decisão binária gerada com o algoritmo CART, com profundidade da árvore de tamanho máxima igual a 4.

Tabela III
MATRIZ CONFUSÃO TREINO 1

Classes	Previsão		
	1	2	3
1	410	0	0
2	0	199	1
3	0	0	149
4	147	0	0
5	95	0	0

O conjunto de teste com o qual se calcula o erro de treinamento ao longo das iterações possui todas as 5 classes. Desta forma pode-se ver como é o comportamento do classificador e como seu erro de validação diminui ao longo das iterações e das adições de classe.

O erro do primeiro treinamento ao longo do total de 8 iterações é apresentado na Figura 3. Neste caso, o erro mínimo encontrado ao longo das iterações refere-se ao erro nos dados de teste referente as classes que ainda não foram treinadas.

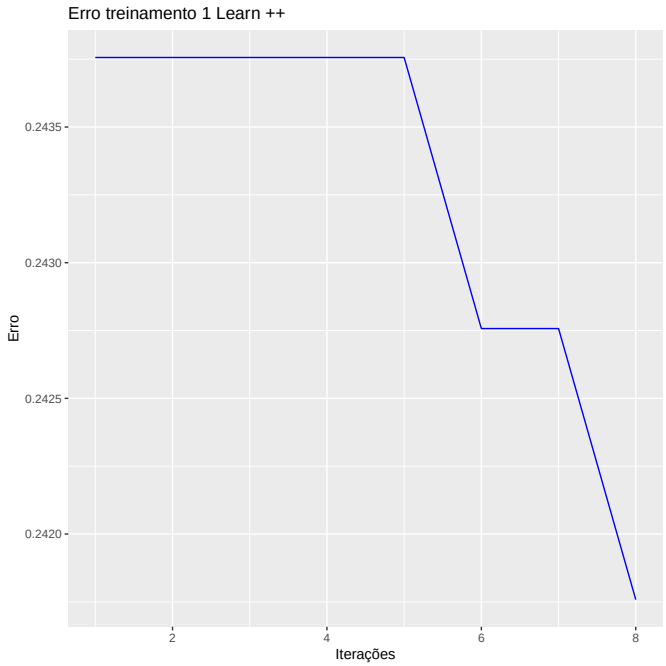

Figura 3. Erro treinamento das três primeiras classes

Uma forma de se comprovar tal afirmação é através da análise da matriz de confusão da previsão realizada com o modelo treinado nos dados de teste. A matriz de confusão do primeiro treinamento é apresentada na Tabela III. Neste caso, os dados de teste das classes ainda não treinadas recebem o rótulo de uma das classes que foram treinadas como é de se esperar.

O próximo treinamento ocorre com a adição de uma nova classe. Conforme Tabela II são realizadas 6 iterações para cada um dos dois conjuntos. Neste caso é criado um total de 12 novos classificadores. A Figura 4 apresenta o erro de treinamento da rede, incluindo o primeiro treinamento que

Tabela IV
MATRIZ CONFUSÃO TREINO 2

Classes	Previsão			
	1	2	3	4
1	410	0	0	0
2	0	199	1	0
3	0	0	149	0
4	0	0	0	147
5	0	0	0	95

são as primeiras 8 iterações. Na figura tem-se um total de 20 iterações, e as 12 últimas se referem ao segundo treinamento.

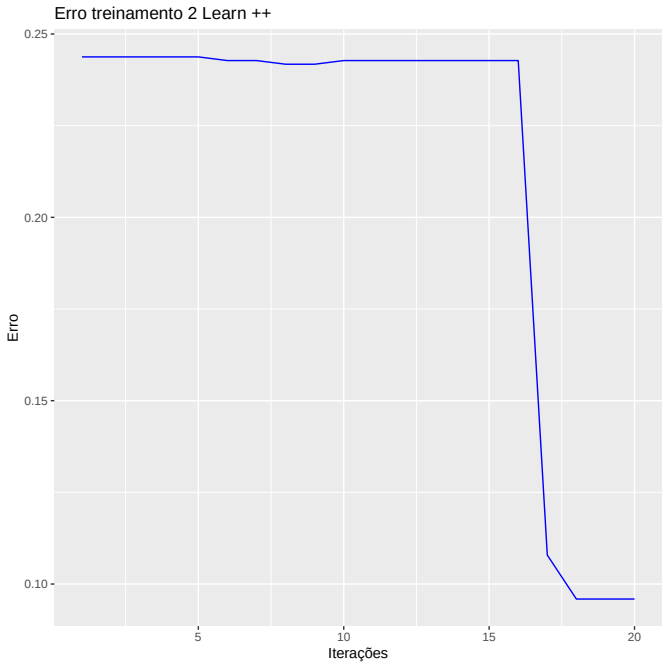

Figura 4. Erro treinamento adição de uma nova classe

Na Figura 4 vê-se que ao final do treinamento o erro convergiu para um novo mínimo, que representa que os dados da classe 4 do conjunto de treinamento agora estão sendo classificados de forma correta.

Pode-se constatar o aprendizado da nova classe analisando também a nova matriz de confusão obtida após o treinamento 2. Ela é apresentada na Tabela IV. Nota-se uma nova coluna ao conjunto de previsão mostrando que agora a classe 4 do conjunto de treinamento está classificada corretamente.

O último treinamento ocorre com a adição da classe 5. Conforme Tabela II são realizadas 10 iterações para cada um dos dois conjuntos. Neste caso é criado um total de 20 novos classificadores. A Figura 5 apresenta o erro de treinamento da rede, incluindo os dois primeiros treinamentos que são as primeiras 20 iterações. Na figura tem-se um total de 40 iterações, e as 20 últimas se referem ao último treinamento.

Na Figura 5 o erro final do treinamento convergiu para 0, representando o aprendizado de todas as classes presentes no conjunto de teste. Como nos treinamentos anteriores, obtém-se a matriz de confusão com o novo modelo para constatar o

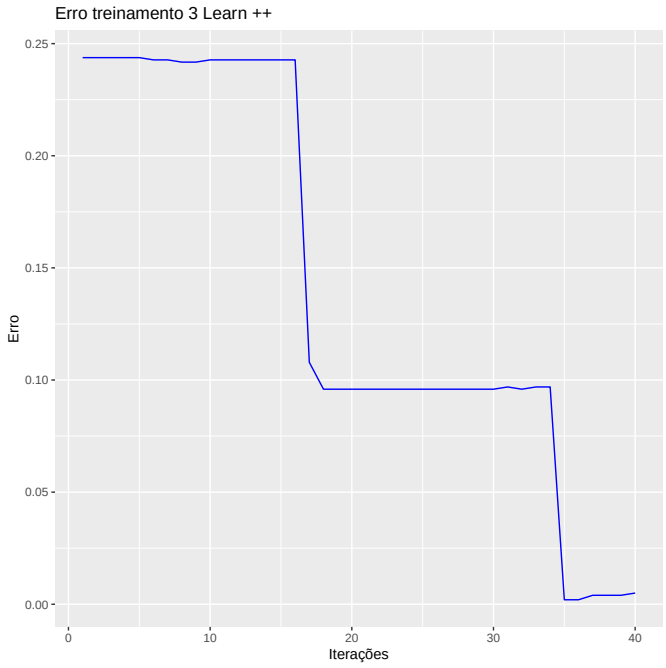

Figura 5. Erro treinamento adição da classe 5

Tabela V
MATRIZ CONFUSÃO TREINO 3

	Previsão				
Classes	1	2	3	4	5
1	410	0	0	0	0
2	0	199	1	0	0
3	0	0	149	0	0
4	4	0	0	143	0
5	0	0	0	0	95

aprendizado da nova classe. A matriz é apresentada na Tabela V. Novamente uma nova coluna é adicionada ao conjunto de previsão mostrando que agora a classe 5 do conjunto de treinamento está classificada corretamente.

A acurácia de acerto relacionada a matriz de confusão da Tabela V é de 99.5%. Esse resultado depende da escolha dos dados de treino e validação, mas em média o resultado do experimento está acima dos 98% de acerto.

A Tabela VI apresenta os resultados para o experimento por classe. Dentre os resultados estão os valores de sensibilidade e especificidade, recall e precisão, e a acurácia da classificação em cada uma das classes.

Os resultados estatísticos apresentados estão diretamente relacionados a matriz de confusão apresentado na Tabela V. Os resultados de recall e precisão estão diretamente relacionados a relevância dos resultados do modelo gerado. Já os resultados de especificidade e sensibilidade estão relacionados a capacidade de identificar corretamente as amostras da classe, e de não identificar ela erroneamente. Neste caso se vê que as classes que apresentam erro na matriz de confusão, os valores de especificidade e sensibilidade tendem a ser menores. A acurácia apresenta o quanto de amostras foram classificadas

Tabela VI
RESULTADOS ESTATÍSTICOS POR CLASSE

	Precisão	Recall	Sensibilidade	Especificidade	Acurácia
Class: 1	100.00	99.03	99.03	100.00	99.52
Class: 2	99.50	100.00	100.00	99.88	99.94
Class: 3	100.00	99.33	99.33	100.00	99.67
Class: 4	97.28	100.00	100.00	99.53	99.77
Class: 5	100.00	100.00	100.00	100.00	100.00

corretamente e mesmo as classes treinadas posteriormente apresentam bons resultados.

VI. CONCLUSÃO

Os resultados alcançados com a aplicação do algoritmo incremental *Learn++* no problema de desagregação de energia satisfazem a necessidade de identificação de equipamentos elétricos ligados de forma isolada em um ponto de energia.

Como parte de um sistema maior para monitoramento constante do consumo de energia em cada equipamento, o algoritmo *Learn++* apresenta uma solução a necessidade limitante dos modelos tradicionais onde é necessário o treinamento de todas as classes em conjunto. Foi comprovada a capacidade do algoritmo de adquirir novos conhecimentos conforme novas classes são apresentadas.

Com a aplicação de um modelo incremental, o sistema de monitoramento de cargas pode ir aprendendo novos equipamentos conforme a utilização por necessidade dos usuários. Para isso é importante a implementação de detecção de "*outliers*", que faz parte dos trabalhos futuros a serem realizados.

Outro problema identificado ao longo da realização do trabalho é o problema do "*outvoting*". É importante entender de que forma o número de iterações aumentará conforme aumenta-se o número de classes a serem aprendidas.

Como solução para o número de iterações necessárias para a identificação de uma nova classe pode-se usar a validação cruzada, analisando quando o erro nos dados de validação já não se alterarem muito ao longo de novas iterações.

Outra solução é aplicar o metódo *Learn++.NC*, que segue a linha do *Learn++*, porém otimizado para a adição de novas classes. Superando assim o problema do "*outvoting*".

Mais trabalhos futuros envolverão a busca de novas características para a representação do problema, como por exemplo a aplicação da transformada "*wavelet*", e a busca por assinaturas transitórias.

REFERÊNCIAS

[1] G. W. Hart, ≪Prototype nonintrusive appliance load monitor≫, em *MIT Energy Laboratory Technical Report, and Electric Power Research Institute Technical Report*, 1985.

[2] ——, ≪Residential energy monitoring and computerized surveillance via utility power flows≫, *IEEE Technology and Society Magazine*, vol. 8, n.º 2, pp. 12–16, 1989.

[3] ——, ≪Nonintrusive appliance load monitoring≫, *Proceedings of the IEEE*, vol. 80, n.º 12, pp. 1870–1891, 1992.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy e A. Bouchachia, ≪A survey on concept drift adaptation≫, *ACM computing surveys (CSUR)*, vol. 46, n.º 4, p. 44, 2014.

[5] R. Polikar, L. Upda, S. S. Upda e V. Honavar, ≪Learn++: An incremental learning algorithm for supervised neural networks≫, *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, n.º 4, pp. 497–508, 2001.

[6] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford e P. Armstrong, ≪Power signature analysis≫, *IEEE Power and Energy Magazine*, vol. 1, n.º 2, pp. 56–63, 2003, ISSN: 15407977. DOI: 10.1109/MPAE.2003.1192027.

[7] A. Cole e A. Albicki, ≪Nonintrusive identification of electrical loads in a three-phase environment based on harmonic content≫, n.º 7 16, pp. 24–29, 2002. DOI: 10.1109/imtc.2000.846806.

[8] Y. Nakano e H. Murata, ≪Non-Intrusive Electric Appliances Load Monitoring System Using Harmonic Pattern Recognition-Trial Application to Commercial Building≫, *Int. Conf. Electrical Engineering, Hong Kong, ...*, n.º March, pp. 1–5, 2007. URL: http://www.iceecon.org/papers/2007/Oral%7B%5C_%7DPoster%20Papers/07/ICEE-333.PDF.

[9] I. A. Pires, ≪Caracterização de harmônicos causados por equipamentos eletro-eletrônicos residenciais e comerciais no sistema de distribuição de energia elétrica≫, *Ufmg*, p. 173, 2006.

[10] S. B. Leeb, S. R. Shaw e J. L. Kirtley, ≪Transient Event Detection in Spectral Envelope Estimates for Nonintrusive Load Monitoring≫, *IEEE Transactions on Power Delivery*, vol. 10, n.º 3, pp. 1200–1210, 1995, ISSN: 19374208. DOI: 10.1109/61.400897.

[11] D. C. Bergman, D. Jin, J. P. Juen, N. Tanaka, C. A. Gunter e A. Wright, ≪Nonintrusive load-shed verification≫, *IEEE Pervasive Computing*, n.º 1, pp. 49–57, 2010.

[12] D. C. Bergman, D. Jin, J. P. Juen, N. Tanaka, C. A. Gunter e A. K. Wright, ≪Distributed non-intrusive load monitoring≫, *IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT Europe*, pp. 1–8, 2011. DOI: 10.1109/ISGT.2011.5759180.

[13] G. A. Carpenter, S. Grossberg e J. H. Reynolds, ≪ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network≫, *Neural networks*, vol. 4, n.º 5, pp. 565–588, 1991.

[14] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, D. B. Rosen et al., ≪Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps≫, *IEEE Transactions on neural networks*, vol. 3, n.º 5, pp. 698–713, 1992.

[15] Y. Freund e R. E. Schapire, ≪A decision-theoretic generalization of on-line learning and an application to boosting≫, *Journal of computer and system sciences*, vol. 55, n.º 1, pp. 119–139, 1997.

[16] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee et al., ≪Boosting the margin: A new explanation for the effectiveness of voting methods≫, *The annals of statistics*, vol. 26, n.º 5, pp. 1651–1686, 1998.

[17] N. Littlestone e M. K. Warmuth, ≪The weighted majority algorithm≫, *Information and computation*, vol. 108, n.º 2, pp. 212–261, 1994.

[18] M. D. Muhlbaier, A. Topalis e R. Polikar, ≪Learn++.NC: Combining Ensemble of Classifiers With Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes≫, *IEEE transactions on neural networks*, vol. 20, n.º 1, pp. 152–168, 2008.