Daniel da Silva Diogo Lara

Uma abordagem para detecção de *Concept Drift* no contexto de detecção de eventos em processos industriais

Brasil

Junho - 2022

Daniel da Silva Diogo Lara

Uma abordagem para detecção de *Concept Drift* no contexto de detecção de eventos em processos industriais

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação do Instituto Federal de Minas Gerais como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Instituto Federal de Minas Gerais

Campus Betim

Programa de Graduação

Orientador: Leandro Freitas de Abreu

Brasil

Junho - 2022

FICHA CATALOGRÁFICA

L318a Lara, Daniel da Silva Diogo
Uma abordagem para detecção de Concept Drift no contexto de detecção de eventos em processos industriais / Daniel da Silva Diogo Lara. – 2022.
49 f. : il.
Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2022.

Orientação: prof. Dr. Leandro Freitas de Abreu

1. Redes neurais (Computação). 2. Variáveis de processo . 3. Concept drift . 4. Processos industriais I. Lara, Daniel da Silva Diogo. II. Título.

CDU: 681.5

Daniel da Silva Diogo Lara

Uma abordagem para detecção de *Concept Drift* no contexto de detecção de eventos em processos industriais

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação do Instituto Federal de Minas Gerais como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Trabalho aprovado. Brasil, 4 de julho de 2022:

Leandro Freitas de Abreu
Professor Dr. Orientador

Reginaldo Vagner Ferreira
Professor Dr. - Convidado

Arthur Hermano Rezende Rosa
Professor Dr. - Convidado

Brasil
Junho - 2022

Este trabalho é dedicado àqueles inquietos que,
apesar das dificuldades, não desanimam! Procurando evoluir, mesmo que lentamente,
sempre!

Agradecimentos

Este trabalho é o resultado da contribuição de cada professor que, ao longo dos anos, pôde compartilhar comigo um pouco do seu conhecimento. Em especial, agradeço ao professor Leandro Freitas pelo tempo e dedicação compartilhados, não somente neste trabalho, mas também em todas as disciplinas cursadas. Também agradeço aos professores Reginaldo Ferreira e Arthur Hermano por aceitarem participar da avaliação deste conteúdo. Além desses, também agradeço ao professor Walmir Caminhas da escola de engenharia da UFMG que contribuiu com muitas ideias em várias reuniões.

Por fim, não menos importante, agradeço também à minha esposa Lorena Lara e aos meus Filhos Daniel Júnior e Antonella Lara pela compreensão e apoio nos momentos de ausência.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Sistemas dinâmicos podem apresentar alterações de estado estacionário de forma inesperada. Algumas destas alterações podem ser ocasionadas por atuação humana como mudança de um *setpoint*, por exemplo. Outras podem ser ocasionadas por falhas ou condições indevidas da planta. O novo estado pode ser válido ou não a depender das razões que ocasionaram a mudança. Apesar disso, é bastante interessante que sistemas dinâmicos críticos sejam, de alguma maneira, avaliados de forma automática com relação ao seu comportamento em estado estacionário.

Sistemas industriais geralmente não são monitorados por um humano a todo tempo. Em sua maioria, cada processo possui um número grande de variáveis dificultando essa monitoração detalhada por uma pessoa. Neste sentido, esse trabalho propõe um método para identificação automática e não supervisionada de *drifts* abruptos. A implementação foi constituída de uma filtragem multidimensional de variáveis de processo no domínio da frequência seguida por uma classificação realizada por meio de uma rede neural. Os resultados foram comparados com outros dois métodos bastante referenciados na literatura (DDM e EDDM). Para avaliação, foram utilizadas oito bases bem referenciadas e conhecidas na literatura para esse fim. Foi possível perceber que, para processos cujas variáveis possuem um comportamento complexo, em algumas bases, o método proposto apresentou um desempenho superior aos comparados.

Palavras-chave: Redes Neurais, Classificador, *drifts*, FFT, Fourier, Variáveis de processo.

Glossário

batch

Batelada. 30

bootstraping

O Bootstraping é uma técnica estatística não paramétrica computacionalmente intensiva de reamostragem que tem como finalidade obter informações de características da distribuição de alguma variável aleatória. 27

data augmentation

Processo de aumentar a quantidade e a diversidade dos dados na base. 40

denoising autoencoders

Autoencoder é uma rede neural artificial de três camadas. A entrada e a saída são as mesmas. Aprende-se a reconstruir a entrada, por exemplo, usando o otimizador adam e a função de perda de erro quadrático médio.. 33

drifts

Veja *drift*. 2, 11, 19, 21, 23, 25, 27, 28, 34, 37, 43, 45, 46

drift

O *concept drift* ou somente *drift* é uma mudança implícita nos dados que leva os modelos a ficarem defasados. 13, 17, 23–25, 27–33, 35, 37–41, 43–47, 49

log-likelihood

Verossimilhança logarítmica. 33

offline

No sentido de processamento em batelada ou em pacotes. 25

online

No sentido de processamento em tempo real. 23, 25, 27, 30, 31, 33, 34, 49

outliers

Ponto fora da curva, dado espúrio. 28, 34, 50

padding

Inserção de valores no início, no final e/ou em pontos intermediários da seqüência a fim de evitar efeitos de borda. 17, 37, 38, 40

setpoints

Veja *setpoint*. 24

setpoint

Setpoint é o valor alvo no qual um controlador tenta manter a variável de processo. 14

surveys

Pesquisa em determinada área. Reunião de vários resultados num mesmo artigo. 27

Siglas

AUC

Area under the ROC Curve. 33, 39, 43, 45

CM

Competence Model-based drift detection. 27

DDM

Drift Detection Method. 11, 19, 21, 25, 27–30, 34, 43, 44, 46, 47, 49

DELM

Dynamic Extreme Learning Machine. 27

EDDM

Early Drift Detection Method. 11, 19, 21, 25, 27–30, 34, 43–47, 49

EDE

Equal Density Estimation . 27

FFT

Fast Fourier Transform. 11, 17, 36, 37

FWDDM

Fuzzy Windowing Drift Detection Method. 27

GMA

Geometric Moving Average . 28

HDDM

Hierarchical Bayesian Estimatimation of Drift-Diffusion. 27

KNN

K-nearest neighbor . 30

LLDD

Learning with Local Drift Detection. 27

LSDD-CDT

Least Squares Density Difference-based Change Detection Test. 27

PCA

Principal component analysis. 49

PCA-CD

PCA-based change detection framework. 27

SCD

Statistical Change Detection. 27

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Sistemas de diferentes naturezas podem apresentar comportamentos em estado estacionário que vão se distanciando de forma muito suave de estados considerados válidos ou conhecidos. Geralmente os estados conhecidos apresentam propriedades estatísticas que não mudam ou, pelo menos não deveriam mudar, ao longo do tempo. Entretanto, sistemas reais não são tão comportados. Na prática, com as variações físicas ou até mesmo por meio de interfaces com outros sistemas, o comportamento em estado estacionário pode se alterar levando a estados indesejados e desconhecidos. Sendo mais específico, sistemas dinâmicos industriais são muito sujeitos a essa natureza não estacionária dos parâmetros.

Além de outras causas de "*drifts*" (desvios de contexto) abrubtos tais como quebras de equipamentos, erro humano ou falhas de comunicação, são esperados também "*drifts*" (desvios) graduais em função do desgaste natural dos equipamentos. A título de exemplo, é comum encontrar alguns modelos usados em controles de processo que necessitam serem retreinados num intervalo frequente para corrigir tais variações. Esse intervalo é muito dependente do processo e da natureza de suas variáveis envolvidas, porém, na maioria das vezes, o momento exato para se determinar quando o sistema migrou de um estado válido para um inválido é dificil de se detectar e depende, muitas vezes, de uma ação reativa do engenheiro de processo após algum efeito negativo já causado pelo estado inválido. Além de prejuízos financeiros, essa atuação reativa pode trazer riscos à planta.

Geralmente, modelos desses sistemas tentam prever ou mesmo inferir comportamentos baseados nos dados históricos e em suas características estatísticas conhecidas. É fato também que essas características estatísticas não são estáticas ao longo do tempo causando consequentemente previsões incorretas. Essa alteração do ambiente de variáveis de um sistema ocasionando a alteração das propriedades estatísticas das variáveis alvo é conhecida na literatura pelo termo em inglês *concept drift* (LU et al., 2020). A correta detecção do ponto em que um *drift* se inicia é de fundamental importância para a mineração de dados a partir de fluxo de dados (G; BEIGY, 2011).

1.1 Motivação

O volume de dados gerado por processos e sistemas tem crescido exponencialmente nos últimos anos. O desenvolvimento de tecnologias de armazenamento cada vez mais baratas, a capacidade de transmissão cada vez mais rápida e a integração *online* dos processos são fatores que contribuem para esse crescimento. Por outro lado, apesar da disponibilidade dessa grande quantidade de dados, é necessário transformá-los em informação. Para isso, modelos são criados com o objetivo de correlacionar conhecimentos

já adquiridos em outras etapas de desenvolvimento com os dados armazenados. Muitas vezes, isso torna-se um desafio quando a natureza dos dados não é estacionária. Esforços da comunidade científica na tentativa de trabalhar com este desafio não são poucos. Quando o *drift* ocorre, o conhecimento estatístico encapsulado nos modelos precisa ser atualizado. Entretanto, a detecção do momento em que os modelos perdem a confiança necessária é um desafio considerável para a área de inteligência computacional. Lu et al. (2016), Lu, Zhang e Lu (2014), Liu et al. (2017), Liu, Zhang e Lu (2017) são exemplos de esforços na pesquisa deste tema.

Alguns sistemas industriais apresentam alterações lentas nas propriedades estatísticas das varíaveis históricas. Nestes casos, não há uma mudança abrupta no conjunto de variáveis de processo e, por esta razão, é possível que os modelos tornem-se inválidos sem que se perceba. Esse problema é conhecido como *concept drift* gradual e/ou incremental e também tem sido foco de pesquisa em muitos problemas práticos (LU et al., 2020). A maior parte das pesquisas tem seu foco em quando o *drift* ocorreu e não em como ou onde. O como e o onde podem ser interessantes também no sentido de entender melhor o comportamento dos sistemas e agregar mais inteligência aos modelos. Isso faz com que a abordagem deixe de ser reativa somente no momento em que aconteceu o *drift* partindo para uma abordagem mais proativa. Ao se entender mais do como e do onde é possível também prever a alteração estatística das variáveis alvo envolvidas.

1.2 Definição do problema

Dada a natureza dinâmica dos processos industriais, a identificação de pontos de operação anômolos é um grande desafio. Esses pontos podem surgir a partir de momentos de falha, inserção de *setpoints* indevidos, existência de alguma característica sazonal do processo ou mesmo o desgaste natural dos equipementos que podem ocorrer de forma bastante lenta. Associada a isso, existe a natureza complexa de monitoração que pode ocorrer por meio de muitas variáveis de processo, sendo umas mais sensíveis a determinados tipos de problema que outras. Um ser humano, mesmo que muito atento e bem treinado, ainda pode não perceber determinadas tendências diante da complexidade das variáveis a serem observadas.

Desta forma, o problema que este trabalho busca estudar se resume na seguinte pergunta: **como localizar temporalmente, de forma automática, um ponto de desvio (*drift*) de um processo industrial por meio da análise de suas variáveis sensoriadas?** Existem várias abordagens e metodologias que procuram lidar com este problema. Este trabalho procura estudar e entender algumas delas além de propor um novo método.

1.3 Escopo do trabalho

A proposta apresentada procurou endereçar o problema de indentificação de *drifts* abruptos. Apesar de poder ser empregada em desvios graduais, não é o foco principal. Por esta razão, a avaliação dos resultados para *drifts* graduais ficou para os trabalhos futuros.

1.4 Contribuições

Este trabalho apresenta um método não supervisionado para identificação de *drift* de forma automática. O método pode trabalhar tanto com bases rotuladas quanto não rotuladas de forma *online* ou *offline* a depender da natureza do processo. Processos muito rápidos podem inviabilizar seu uso de forma *online*.

1.5 Organização deste Trabalho

O Capítulo 2 apresenta de forma não exaustiva os principais resultados da revisão de literatura que aborda o tema. O Capítulo 3 apresenta e detalha o método proposto. O Capítulo 4 apresenta os principais resultados e os compara com o DDM e EDDM. Por fim, o Capítulo 5 evidencia as principais conclusões e apresenta possíveis trabalhos futuros.

2 Estado da Arte

Este capítulo apresenta um breve compilado do esforço dos pesquisadores da área nos últimos anos na tentativa de apresentar algoritmos e soluções para o problema de identificação automática de *drifts*. Essa é uma área bastante vasta e, por esse motivo, foi realizado um esforço para focar nos principais trabalhos.

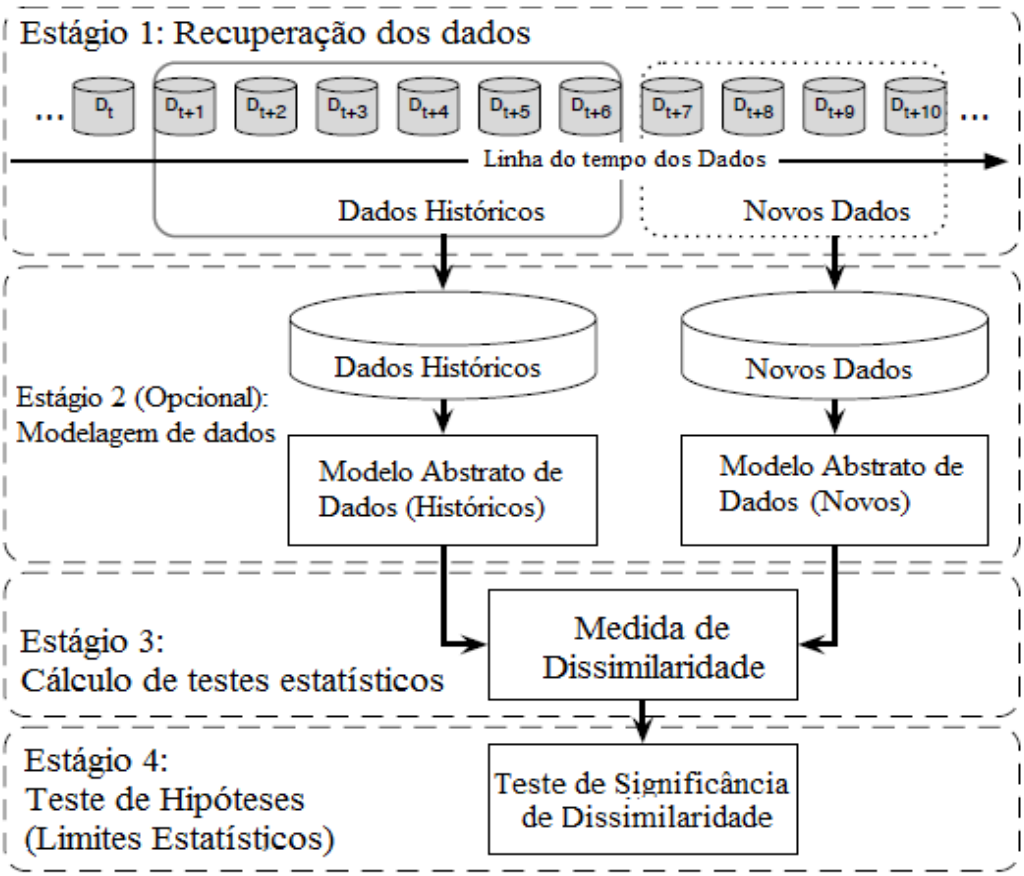
2.1 Abordagens Exploradas na Literatura

Há um grande esforço da comunidade cientifica, nos últimos anos, procurando meios eficazes de tratar o problema de *drift* de contexto em uma grande variedade de cenários. Vale a pena citar seis trabalhos importantes (*surveys*) que resumem de forma clara esses esforços ao longo dos últimos anos, Krawczyk et al. (2017), Ramírez-Gallego et al. (2017), Gama (2012), Silva et al. (2013), Lu et al. (2020) e Gemaque et al. (2020). A Figura 1 apresenta um esquema geral para a avaliação de detecção de *drift*. Basicamente todos os trabalhos de alguma maneira interagem com alguma dessas etapas para a identificação e adaptação ou correção. Entretanto a maior parte destes trabalhos está focada na detecção de *drifts* abruptos. Além disso, salvo para o último citado, praticamente a totalidade deles trabalha com aprendizagem supervisionada. Gemaque et al. (2020) apresentou uma revisão de alguns destes trabalhos que lidam com aprendizagem não supervisionada.

Alguns artigos interessantes se concentraram no estágio 4 (teste de hipóteses), dentre eles, Alippi e Roveri (2008) e Gama et al. (2004) procuraram estimar novas distribuições dos testes estatísticos para adaptar o modelo. Já Bu, Alippi e Zhao (2018) e Dasu et al. (2006) focaram sua abordagem em soluções por meio de *bootstraping*. Lu, Zhang e Lu (2014) concentrou seus trabalhos em testes de permutação enquanto que Blanco et al. (2015) se concentraram na identificação de limites baseados na desigualde de Hoeffding.

Além disso, uma vasta variedade de algoritmos foram propostos ao longo dos anos a fim de endereçar o problema. Cada um com suas especificidades, vantagens e desvantagens a depender dos cenários aplicados. Um dos mais referenciados é o *Drift Detection Method* (DDM) apresentado por Gama et al. (2004) no simpósio brasileiro de inteligênica artificial. Esse foi o primeiro algoritmo que definiu o nível de alerta e o nível de *drift* para a detecção de *concept drift*. A partir deste, inúmeras variações similares surgiram. Ainda em Lu et al. (2020), são apresentadas algumas destas variações que foram LLDD, EDDM, HDDM, FWDDM, DELM dentre outras. Outro importante algoritmo foi o *Information-Theoretic Approach* (ITA) apresentado por Dasu et al. (2006). A ideia por trás do método consiste em usar *bootstraping* para trabalhar, de forma *online*, reparticionando os dados novos e

Figura 1 – Framework apresentado por Lu et al. (2020) para detecção de *drift*. Fonte: Adaptados de Lu et al. (2020).



os históricos buscando situações de *concept drift*. Lu et al. (2020) também apresentam outros algoritmos tais como SCD, CM, PCA-CD, LSDD-CDT, EDE, etc.

Além disso, a maioria dos trabalhos relacionados ao tema busca quando o *drift* ocorreu. Pesquisas no sentido de entender melhor como o *drift* ocorre ainda são escassas. Por outro lado, abordagens que usam aprendizagem não supervisionada também são muito bem vindas. O desenvolvimento de abordagens que contemplem essa deficiência (como o *drift* ocorreu) pode ainda contribuir no sentido de se atuar de forma mais proativa tanto na adaptação quanto no aprendizado no momento em que o evento ocorrer. Outro fator interessante é o fato de que poucos destes trabalhos são dedicados unicamente para processos industriais. Sun et al. (2020) apresentaram uma revisão de alguns destes métodos. A realidade dos processos industriais apresenta desafios que dificultam bastante o bom desempenho de grande parte dos métodos já citados acima. Geralmente, dados industriais são repletos de ruídos e *outliers*. Estudos no sentido de avaliar a robustez destes métodos com relação a ruídos também são escassos. A título de exemplo, o trabalho descrito em Jaramillo-Valbuena, Londoño-Peláez e Cardona (2017) procurou endereçar esse desafio por meio da avaliação do desempenho de 4 métodos conceituados até sua época no que diz respeito à detecção de *drift* com relação a ruídos e *outliers* (avaliou os

resultados de acordo com a acurácia e Yonden's J). Os métodos avaliados foram DDM, EDDM, GMA e *Exponentially Weighted Moving Average Chart Detection*. EDDM foi o que melhor performou. EDDM surgiu com o intuito de resolver uma deficiência do DDM para detectar *drifts* graduais. O DDM e suas variações consistem em treinar um modelo para determinado contexto e, a partir de uma inferência de probabilidade para resultados futuros, determinar se o sistema dinâmico sofreu um desvio. A ideia por trás destes métodos reside no argumento de que, para um modelo bem treinado, a tendência é que novos exemplos sempre estarão sendo previstos com uma probabilidade de acerto elevada. À medida que esses modelos começam a errar muito, significa que o sistema pode ter sofrido um desvio de contexto de forma que o modelo treinado não seja mais válido. A principal diferença entre o DDM e o EDDM reside no fato de que o DDM apenas conta um número absoluto de erros para decidir se o sistema sofreu um *drift* enquanto que o EDDM leva em consideração o número de erros e a distância temporal em que os erros ocorreram. É possível ajustar esses limiares para que se tolere erros mais ou menos esparsos. Entretanto, apesar de métodos baseados no DDM serem tradicionais e essenciais para se entender os métodos mais básicos, o estado da arte tem procurado preencher algumas lacunas que ainda permanecem em técnicas não supervisionadas e online.

2.1.1 Estratégias Supervisionadas

A fim de compreender o básico por trás das técnicas mais clássicas e bem referenciadas na literatura, os algoritmos do DDM e EDDM são apresentados.

2.1.1.1 DDM

O DDM, apresentado por Gama et al. (2004), usa uma distribuição binomial para descrever o comportamento da variável aleatória que entrega o número de erros de classificação. Para cada ponto i na sequência amostrada, a taxa de erro é a probabilidade de classificação incorreta (p_i) com desvio padrão $s_i = \sqrt{p_i(1-p_i)/i}$. A variável i indexa o número da amostra, de forma que para uma base com n amostragens, o último índex a ser considerado no desvio padrão seja n. Os valores p_i e s_i são calculados a partir de uma janela de avaliação comparando os resultados previstos pelos modelos com aqueles rotulados a *priori*. É assumido, de acordo com Mitchell (1997), que (p_i) tende a diminuir de acordo com o aumento de amostras, ou seja, a taxa de erro tende a diminuir com o passar do tempo se a distribuíção for estacionária. Um aumento significante nesta taxa sugere um *drift*. Desta forma, são salvos os valores de p_i e s_i até que atinjam um valor mínimo. São testados os seguintes gatilhos:

- $p_i + s_i \geq p_{min} + 2s_{min}$ para o nível de alerta. A partir neste nível, os novos exemplos são salvos em antecipação para uma possível troca de contexto. Caso realmente ocorra, esses exemplos serão usados para se treinar um novo modelo.

- $p_i + s_i \geq p_{min} + 3s_{min}$ para o nível de desvio. A partir deste ponto o *drift* de conceito é considerado verdadeiro e um novo modelo é treinado com os exemplos armazenados. Os valores de p_{min} e s_{min} são ressetados.

2.1.1.2 EDDM

A proposta do EDDM, apresentado por Baena-Garcıa et al. (2006), é bastante similar à do DDM, salvo para os gatilhos e não mais determinação de p_{min} e s_{min}. No caso do EDDM não são computados valores absolutos de erro para o cálculo de p_i. A ideia básica por trás do algoritmo é considerar a distância (em amostras) entre dois erros de classifcação ao invés do número absoluto de erros. Enquanto o método está aprendendo, as predições serão cada vez melhores e a distância entre dois erros irá aumentar. Desta forma, é calculada a distância média entre dois erros (p'_i) e seu desvio padrão (s'_i). São salvos os valores p'_i e s'_i quando $p'_i + 2s'_i$ atinge seu valor máximo. Desta forma, $p'_{max} + 2s'_{max}$ corresponde com o ponto onde a distribuição de distâncias entre erros é máxima. Esse é o ponto onde o modelo está melhor adaptado para o contexto. De forma análoga ao DDM, são definidos dois gatilhos:

- $(p'_i + 2s'_i)/(p'_{max} + 2s'_{max}) < \alpha$ para o nível de alerta. A partir neste nível, os novos exemplos são salvos em antecipação para uma possível troca de contexto. Caso realmente ocorra, esses exemplos serão usados para se treinar um novo modelo.

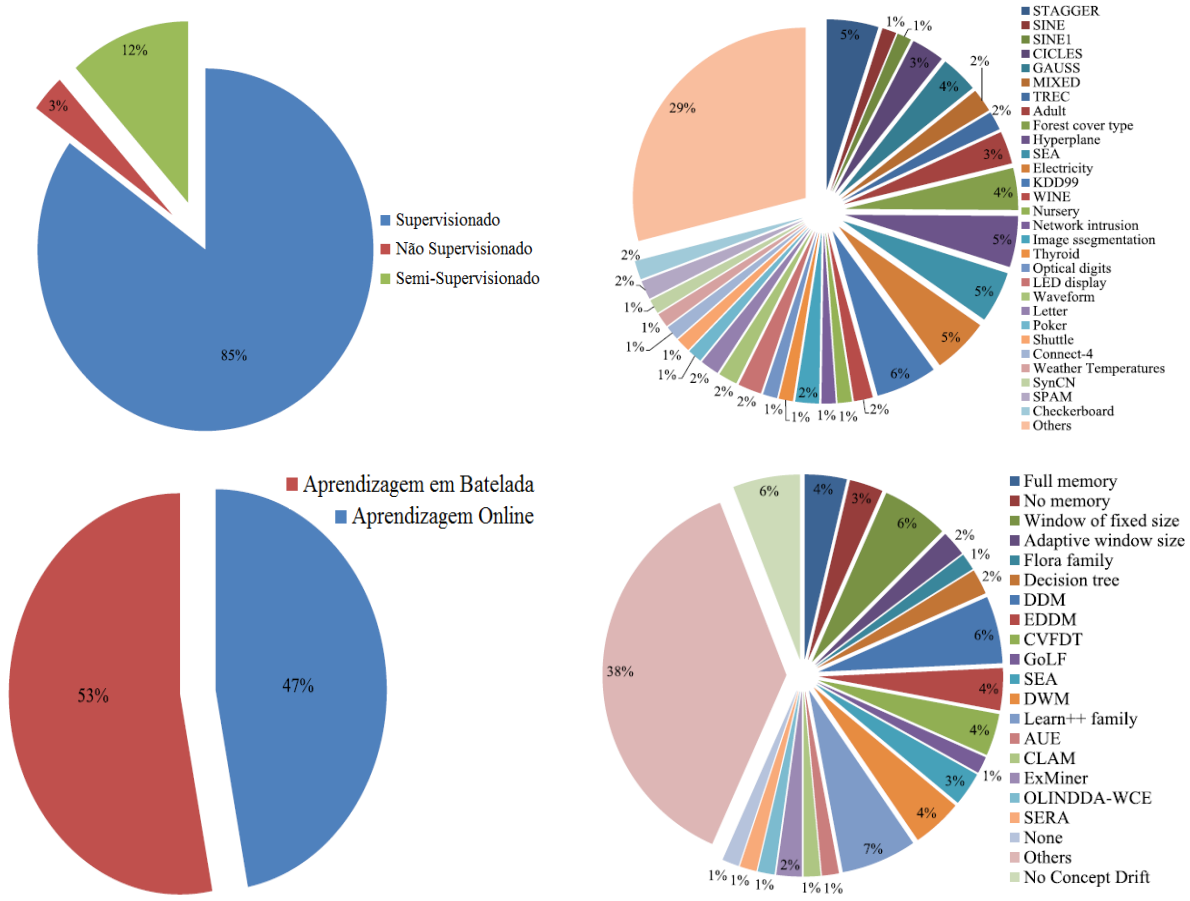
- $(p'_i + 2s'_i)/(p'_{max} + 2s'_{max}) < \beta$ para o nível de desvio. A partir deste ponto o *drift* de conceito é considerado verdadeiro e um novo modelo é treinado com os exemplos armazenados. Os valores de p'_{max} e s'_{max} são ressetados.

2.1.2 Estratégias Não Supervisionadas ou Semi-Supervisionadas

O estado da arte tende a trabalhar com métodos de detecção online e não supervisionados. Esta estratégia é ainda mais interessante para o contexto industrial, uma vez que existe a dificuldade de rotulação dos dados à medida que os processos avançam. Geralmente os métodos online e não supervisionados estão relacionados com alguma técnica de agrupamento (KNN, GUSTAFSON-GK e variações) associada a uma métrica de distância de forma a incluir novos exemplos em grupos que possuem a mais alta verossimilhança para tal amostra. Estratégias de limiarização são usadas para considerar que determinada amostra não pertence a nenhum dos contextos criando novos grupos. Métodos não supervisionados são escassos na literatura. Iwashita e Papa (2018) apresentaram uma análise quantitativa com relação às proporções de publicações relacionadas a *drift*. Foi apresentado que aproximadamente 85% dos artigos publicados estão tratando de métodos supervisionados, 12% semi-supervisionados e somente 3% não supervisionados. Este trabalho também apresentou outras estatísticas interessantes tais como as bases

mais usadas em *papers* que tratam o problema, percentual de abordagens multi-classe ou binária, aprendizagem *online* ou *batch* e os métodos mais interessantes usados com maior frequência em comparações nos artigos. A Figura 2 apresenta alguns dos resultados relacionados pelos autores.

Figura 2 – Análise quantitativa apresentada por Iwashita e Papa (2018) para publicações relacionadas a detecção de *drift*. Fonte: Adaptado de Iwashita e Papa (2018).



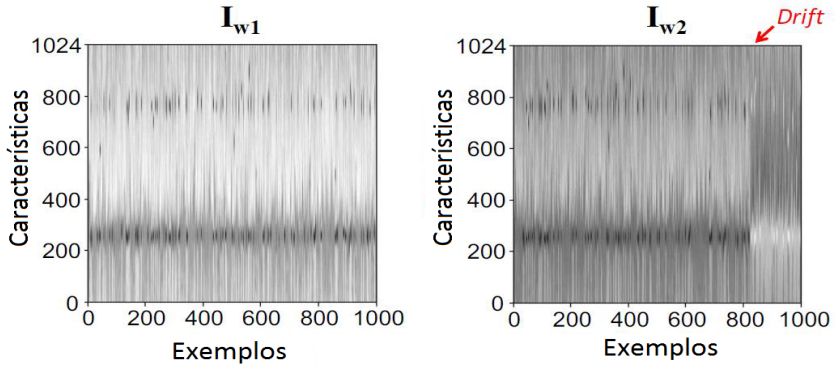
Por outro lado, em corroboração com o apresentado por Gemaque et al. (2020), problemas reais normalmente não possuem rótulos para os dados imediatamente após sua disponibilização. Em muitas situações, é interessante ter a identificação do desvio no momento em que ele ocorre. Neste sentido, métodos não supervisionados ou no máximo semi-supervisionados são os mais interessantes para se trabalhar de forma *online* numa aplicação real.

2.1.2.1 IBDD (*Image-Based Drift Detector*)

IBDD foi apresentado por Souza, Chowdhury e Mueen (2020). Trata-se de uma abordagem de deteção não supervisionada de desvios de contexto baseada em processamento de imagens. Basicamente a ideia consiste em, para cada exemplo, empilhar todos valores normalizados das características (dimensões) em uma coluna da imagem. Vários exemplos

sequenciais formam uma imagem. A resolução da imagem é o **número de características** *versus* **número de exemplos**. A Figura 3 apresenta um exemplo de imagem formada desta forma.

Figura 3 – Método apresentado por Souza, Chowdhury e Mueen (2020) para detecção de *drift*. Fonte: Adaptado de Souza, Chowdhury e Mueen (2020).



A detecção do desvio ocorre por meio de algoritmos de processamento de imagens. É interessante observar que, desta maneira, para este exemplo específico, é possível identificar visualmente o *drift*.

2.1.2.2 IKS (*Incremental Kolmogorov-Smirnov Test*)

Uma proposta interessante apresentada por Reis et al. (2016) traz uma abordagem online não supervisionada para a detecção de *drift* usando IKS(*Incremental Kolmogorov-Smirnov Test*). A sugestão dos autores foi aplicar o teste individualmente em cada atributo. O objetivo era reduzir o uso de testes multivariados e de funções de mapeamento. Isso simplifica o método e traz ganhos em processamento. A detecção do desvio a partir de um único atributo pode ser suficiente para a detecção de um *drift* geral, por outro lado, podem existir desvios ocasionados pela composição de vários atributos que este método não identificaria.

Para cada atributo, o IKS mantem duas janelas de tamanho W. Uma delas é fixa contendo os dados usados para o treinamento do modelo mais recente. A outra é chamada de janela de detecção. Esta última é deslizante e contém os dados do fluxo de entrada. À medida que cada instância chega, ela é classificada e o teste IKS é executado comparando a distribuição dos dados na janela de referência com a janela de detecção. Se a hipótese nula que as distribuições são idênticas é rejeitada, um desvio é sinalizado. Caso contrário, a janela é deslocada e uma nova instância é obtida a partir do fluxo.

Foram propostas três possíveis reações para o caso de detecção de *drift*:

- Treinar um novo modelo com os dados da janela deslizante para rótulos reais.

- Aplicar uma transformação AB para aqueles atributos responsáveis pelo desvio da janela de referência de forma a fazer com que a média e o desvio padrão sejam idênticos à janela de detecção.
- Implementação de uma árvore de decisão modificada. Quando um desvio é identificado, os rótudos verdadeiros são demandados somente para as instâncias que atingem as folhas com um atributo que disparou o *drift*.

2.1.2.3 NM-DDM (*Nonparametric Multidimensional Drift Detection Method*)

O NM-DDM foi proposto por Mustafa et al. (2017). O método proposto consiste no uso de *denoising autoencoders* e uma detecção de desvio de contexto baseada em *log-likelihood random walk*. A identificação do *drift* reside no cálculo da razão do *log-likelihood* entre dados de duas janelas, uma antes e outra após o desvio de contexto. O cálculo é realizado para cada atributo e se a maior razão entre as duas janelas é maior que um determinado limiar, o *drift* é identificado. Na sequência as amostras da segunda janela são usadas para o treinamento de um novo modelo. Instâncias com baixa confiança são enviadas para o rótulo manual. Instâncias com alto valor de confiança são rotuladas com o valor previsto.

2.1.2.4 *Plover*

Plover foi proposto por Mello et al. (2019) por meio da apresentação de uma análise teórica sobre fundamentos e garantias para detecção não supervisionada de *drift*. Segundo os autores, enquanto métodos supervisionados são suportados pela teoria estatística, métodos não supervisionados dependem de métricas internas que não seguem garantias similares. Medidas internas, tais como distâncias para os centroides, métricas para determinação do quão compacto o *cluster* é ou médias e variâncias são computadas a partir dos dados. Por outro lado, métricas externas dependem de rótulos supervisionados que certamente não podem ser esperadas em métodos puramente não supervisionados.

Plover segue uma abordagem similar a outros métodos baseados em janelas fixas e deslizantes já presentes na literatura. A metodologia sugere o uso de duas janelas para comparação porém com funções mais genéricas para o cálculo da divergência. As funções para o cálculo da divergência entre as janelas devem atender basicamente dois pré-requisitos. A função deve ser selecionada de forma que seja independente dos dados de entrada e os dados devem ser independentes e identicamente distribuídos. Essa função pode ser média, variância, densidade espectral, etc.

2.1.2.5 3D (*Discriminative Drift Detector*)

DDD ou 3D é um método não supervisionado *online* relativamente simples proposto por Gözüaçık et al. (2019) que procura identificar o *drift* por meio da AUC (*area under the*

curve. Os autores propuseram uma janela de tamanho fixo deslizante contendo basicamente dois conjuntos: o antigo e o novo. Um classificador simples é treinado para distinguir entre esses dois conjuntos. O *drift* é detectado de acordo com o desempenho desse classificador medido por meio da AUC. Esse processo é repetido sempre que chegam novos exemplos. A janela contendo os exemplos possui duas partes, uma com dados antigos e outra com dados recentes. O método propõe rotular os antigos com 0 e os novos com 1. Executa-se um classificador nestes dados e o desempenho da AUC é usado para avaliar se houve um *drift*.

2.2 Considerações

Algumas semelhanças e tendências puderam ser identificadas nesta revisão. Ainda existem oportunidades a serem exploradas em trabalhos concentrados unicamente em processos industriais. Não existem tantos métodos bons para identificar desvios graduais ou incrementais. Não existem também tantos estudos relacionados à robustez dos métodos com relação a ruídos e *outliers*. Não foram encontradas bases puramente industriais disponibilizadas para estudo de *drifts*. O estado da arte procura soluções para a detecção de métodos de forma *online* e não supervisionados. O aumento da dimensionalidade dos exemplos também se torna um problema fazendo com que a acurácia dos modelos caia consideravelmente. Alguns métodos procuram abordar técnicas de redução de dimensionalidade tais como PCA, porém essas técnicas trazem a desvantagem de se perder a variável real com maior impacto na mudança de contexto. Os métodos mais famosos e talvez os mais referenciados são o DDM e o EDDM. DDM bastante referenciado em desvios mais abruptos e EDDM nos desvios incrementais e graduais.

3 Método Proposto

Recapitulando o conhecimento obtido nos capítulos anteriores, é fato que métodos baseados em aprendizagem não supervisionada tendem a ser as melhores opções para uso em processos industriais. Normalmente, a rotulagem dos dados demanda atuação de um engenheiro de processo que, além de ser uma mão de obra cara, na maioria das vezes não possui o tempo necessário para essa atividade em função do grande volume de dados gerados. Neste sentido, este trabalho propõe uma identificação automática de *drift* de forma não supervisionada baseada em um agrupamento adequado de dados seguido por uma filtragem no domínio da frequência. A filtragem fornece subsídios para um possível ponto de *concept drift* que deve ser confirmado ou descartado com base nos resutados de uma classificação implementada por meio de uma rede neural.

3.1 Organização dos dados

A proposta apresentada propõe a organização inicial dos dados de forma similar àquela apresentada por Souza, Chowdhury e Mueen (2020). Naquele trabalho, a análise de dados é feita na imagem como um todo. Neste trabalho, isso não faz muito sentido uma vez que as linhas não têm correlação entre si. Em outras palavras, cada análise deve ser feita linha a linha. Portanto, a proposta é que se organize o conjunto de dados de forma similar ao apresentado por Souza, Chowdhury e Mueen (2020) em uma matriz bidimensional, podendo até ser apresentada visualmente como uma imagem, porém cada filtragem será realizada linha a linha considerando apenas correlações temporais de uma mesma variável de processo.

Desta forma, seja $x_i(k)$ uma varíavel do processo $i \in \mathbb{N}$, medida no instante de tempo $k \in \mathbb{N}$, em que $(0 < i \leq n)$ e $(0 < k \leq t)$. A organização dos dados esperada pelo método deve estar na seguinte forma:

$$X = \begin{bmatrix} x_1(1) & x_1(2) & \dots & x_1(t) \\ x_2(1) & x_2(2) & \dots & x_2(t) \\ \vdots & \vdots & \ddots & \vdots \\ x_n(1) & x_n(2) & \dots & x_n(t) \end{bmatrix}$$

Diante disso, o método proposto consiste em, para cada janela de tempo do horizonte de detecção de *drift*, organizar as amostras de todos os sensores de processo de forma sequencial e empilhadas numa estrutura matricial. Organizadas desta maneira, cada linha corresponde aos dados sequenciais de uma variável do processo desde o início ao final da

janela de amostragem. Uma vez na forma matricial, é realizada uma normalização *Z-score*.

3.2 Análise e filtragem

É esperado que um processo funcionando de forma correta possua suas distribuição e variância determinadas para todas as suas variáveis. Quando um desvio ocorre, essa distribuição se altera de forma que as médias e variâncias de algumas ou talvez todas as variáveis se tornem diferentes. Analisando no domínio da frequência, no momento do desvio, ocorre a predominância de harmônicos com frequências mais altas. Caso o processo tenha muitas variáveis que sentiram esse desvio, para cada uma delas, neste momento ocorre essa predominância de um harmônico de frequência mais elevada. Desta forma, a estratégia consiste em realizar um filtro passa altas ideal baseado na FFT de forma similar ao proposto por Christenson, Reddy e Rowlandson (1990).

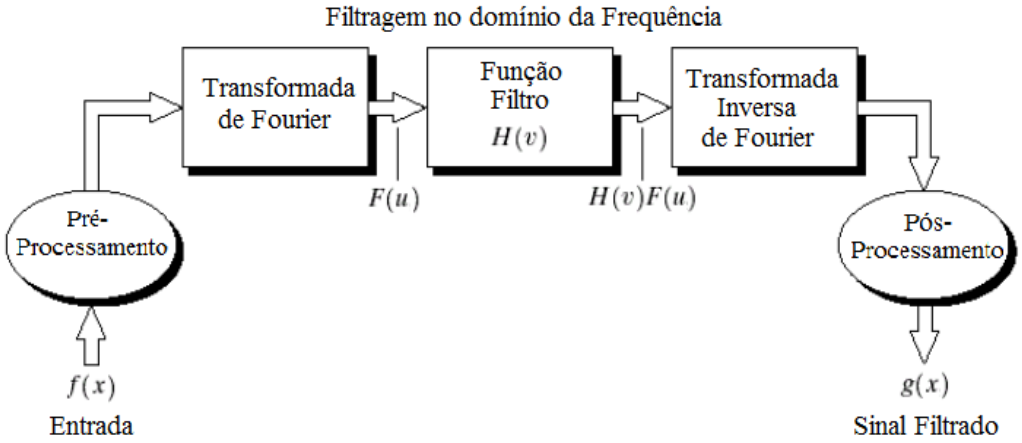
Filtros no domínio da frequência baseados em FFT trazem a vantagem de serem eficientes computacionalmente falando. A Figura 4 apresenta um esquemático para esta estratégia. Basicamente, após uma etapa de pré-filtragem, que neste caso trata-se da normalização *Z-score*[1] dos dados, realiza-se a FFT seguida pela multiplicação por uma função H. Os dados alterados no domínio da frequência retornam ao domínio do tempo por meio da transformada de Fourier inversa, seguidos pelo pós-processamento. Essa filtragem ocorre para cada variável de processo de forma individual no horizonte de detecção do desvio. O pós-processamento ocorre de forma a potencializar o resultado de acordo com o objetivo. Neste caso específico, a etapa de pós-processamento consiste em realizar uma soma dos valores absolutos do resultado da filtragem para cada variável. Maiores detalhes e análise desta etapa serão apresentados na formalização do algoritmo.

3.3 Algoritmo Proposto

Uma vez determinada a organização dos dados dos processos, o algoritmo consiste em realizar a FFT para cada variável de processo no horizonte de detecção de desvio determinado. Alguns parâmetros necessitam ser definidos e analisados de acordo com o problema em questão. O primeiro deles é a frequência de corte do filtro passa altas. Neste caso a frequência é um parâmetro do algoritmo e deve ser definida de acordo com o cenário tratado. Outro desafio é o tratamento das bordas. A resposta de filtros passa altas geralmente apresenta grandes respostas nas bordas das amostras, isto é, no início e final da janela amostrada. Para este desafio, foi implementado um *padding* com os mesmos valores amostrados por um intervalo de $\frac{1}{10}$ do período do horizonte de detecção antes e após a janela de amostragem.

[1] Normalização Z-Score é a transformação das variáveis subtraindo a média e dividindo pelo desvio padrão

Figura 4 – Esquema para filtragem no domínio da frequência por meio de FFT. Fonte: Adaptado de Gonzalez e Woods (2002) - Capítulo 4, Figura 4.5.



Na sequência, é esperado que, no momento do *drift*, ocorra a presença de harmônicos de frequências mais elevadas em algumas ou talvez em todas as variáveis. O resultado desta filtragem por variável de processo, em valor absoluto, é retornado linha a linha em que os valores diferentes de zero denotam a região de *drift*. Desta forma, todas as linhas que representam os resultados são somadas para evidenciar mais claramente o momento do desvio. Por fim, os intervalos em que foram implementados o *padding* são desprezados e o resultado é apresentado como um indicativo de ponto de desvio. O ponto de maior valor de intensidade no resultado da soma dos filtros é tido como o ponto de início da tendência do desvio.

A Figura 5 apresenta um exemplo de resultado para uma base de dados de 10 dimensões e 1000 amostras geradas artificialmente com um desvio inserido após a amostra 700 sem *padding*. Para todas as dimensões, os dados foram gerados de acordo com uma distribuição normal com valores de média para as amostas de 1 a 700 sendo $[5.1; 3; 5; 4; 5.8; 5; 1; 2; 5.5; 7]$. Os valores de médias para as amostras de 701 a 1000 foram $[3; 3.2; 2.8; 3; 6; 5; 1; 2; 5; 8]$. A matriz de covariancia utilizada foi a identidade de dimensão [10 x 10] multiplicada pela constante 5. É possível perceber que, apesar do método ter identificado o *drift* a partir da amostra 700, também identificou *drifts* tanto no início quanto no final da janela de amostragem. Já a Figura 6 apresenta o mesmo exemplo com a implementação do filtro após a aplicação do *padding*. Neste caso, é possível ver que o ponto de intensidade máxima acontece na amostra em que ocorreu o *drift*.

3.3.1 Algoritmo

O algoritmo apresentado toma como entrada uma matriz M já organizada como apresentado nas seções anteriores. Parte do processamento é apresentada em um vetor de intensidades somadas de indicação da existência de desvio para cada amostra. O parâmetro

Figura 5 – Exemplo de filtragem sem *padding*. Nota-se o valor elevado nas bordas não ocasionado pelo *drift*, mas pelo efeito de bordas. Fonte: Autor.

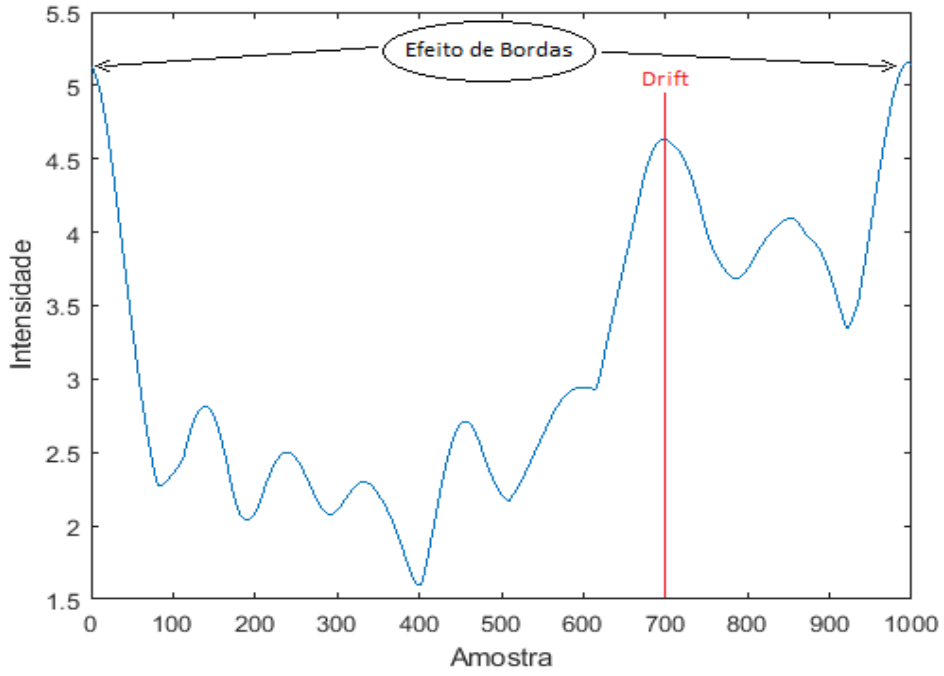
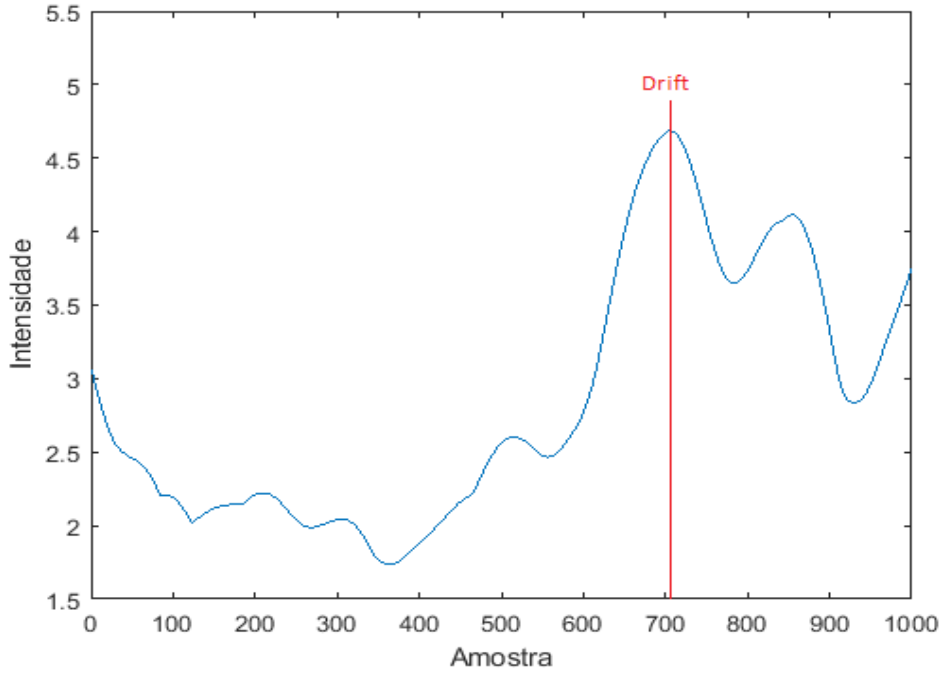


Figura 6 – Exemplo de filtragem com *padding*. Verifica-se que o efeito de bordas foi removido. Fonte: Autor.



de entrada é a frequência de corte para o filtro passa altas. A função H(v) implementa um filtro ideal passa altas (resposta nula para frequências indesejadas no domínio de Fourier) de acordo com o apresentado por Christenson, Reddy e Rowlandson (1990). A

função SOMA retorna a soma acumulada de todos os elementos da mesma coluna de uma matriz qualquer. Na sequência, para cada horizonte analisado, é encontrada a amostra que apresentou a maior resposta em intensidade para a soma das amostras filtradas. Esse pico acontece em uma amostra candidata para o *drift*. Por definição de *drift*, é esperado que todas as amostras antes deste ponto pertençam a uma classe enquanto todas após pertencem a outra classe (por exemplo, antes do momento de falha - classe sem falha - e após o momento de falha - classe com falha).

Neste sentido, caso esse seja um *drift* real, é possível construir uma rede neural capaz de classificar todos os exemplos anteriores à amostra candidata como pertencentes a uma classe e, todos aqueles posteriores, pertencentes a outra. Entretanto, a amostra candidata à separação de classes pode se apresentar próxima a uma das extremidades da janela de identificação. Nesta situação, o treinamento da rede neural seria realizado com exemplos bastante desbalanceados (maioria esmagadora de exemplos pertencentes a uma classe enquanto uma minoria pertencente a outra). Isso prejudica uma avaliação do modelo unicamente por meio da acurácia. Neste sentido, de acordo com o apresentado por Rosset (2004), é feita uma avaliação do modelo por meio da AUC. Essa métrica é capaz de avaliar bem um modelo mesmo para o caso de classes desbalanceadas. Assim, caso a AUC apresente um bom resultado, é entendido que aquela amostra realmente separava duas classes distintas indicando um *drift* real.

3.3.2 *Data Augmentation*

A fim de se obter uma maior evidência do desvio, foi proposta uma técnica para o aumento de dimensionalidade das amostras. É conhecido que, baseado na estratégia de somar as contribuições de cada variável para a identificação do *drift*, um número maior de variáveis causa um maior resultado para a soma no ponto de desvio permitindo uma identificação mais evidente do candidato a ponto de *drift*. Portanto a proposta para aumento de dimensionalidade das amostras foi uma concatenação de acordo com a matriz M.

$$M = \begin{bmatrix} x_1(1) & x_1(2) & \dots & x_1(t) \\ \vdots & \vdots & \ddots & \vdots \\ x_n(1) & x_n(2) & \dots & x_n(t) \\ |x_1(1) - x_1(1)^{1.5}|^2 & |x_1(2) - x_1(2)^{1.5}|^2 & \dots & |x_1(t) - x_1(t)^{1.5}|^2 \\ \vdots & \vdots & \ddots & \vdots \\ |x_n(1) - x_n(1)^{1.5}|^2 & |x_n(2) - x_n(2)^{1.5}|^2 & \dots & |x_n(t) - x_n(t)^{1.5}|^2 \\ x_1(1)^{1.7} & x_1(2)^{1.7} & \dots & x_1(t)^{1.7} \\ \vdots & \vdots & \ddots & \vdots \\ x_n(1)^{1.7} & x_n(2)^{1.7} & \dots & x_n(t)^{1.7} \end{bmatrix}$$

Os valores dos expoentes procuram introduzir uma não linearidade que seja potencializada nos pontos de *drift*.

3.3.3 Rede neural proposta

A rede neural proposta para a avaliação da separabilidade foi implementada por meio do pacote *Deep Learning Toolbox* do Matlab. Os dados de entradas são concatenados com a classe suposta para cada amostra. A matriz de exemplos e a classe são passadas para uma rede neural profunda com 70 camadas completamente conectadas seguidas por um nível de normalização. A função de ativação usada nos neurônios foi a tangente hiperbólica. O nível de saída usou uma camada para classificação que computa a entropia cruzada e a classificação ponderada por meio de classes mutuamente exclusivas (no Matlab definida como *classificationLayer*). O algoritmo de otimização utilizado foi o Adam. Os demais parâmetros como número de épocas e critérios de parada para o treinamento foram deixados para que o Matlab otimizasse automaticamente por meio de heurísticas próprias. Foram usadas 80% das amostras para treinamento, 10% para validação e 10% para testes. O Algoritmo 3.1 apresenta em pseudo-código o explicado acima.

3.4 Considerações

O algoritmo proposto procurou unir sugestões apresentadas por Souza, Chowdhury e Mueen (2020) para a representação de dados e Gözüaçık et al. (2019) para a classificação. A parte original consistiu no filtro no domínio da frequência para encontrar o *drift* propriamente dito. Foi utilizado *padding* objetivando resolver o desafio de aplicação de filtros nas bordas das janelas. Essa técnica apresentou resultados bastante satisfatórios. A normalização dos dados também foi uma etapa importante no sentido de trazer todas as variáveis para uma mesma ordem de grandeza. Uma das características do algoritmo proposto é sua tendência a obter melhores resultados com bases de maiores dimensões. Neste sentido, foi implementada uma etapa para realizar *data augmentation*, que pode ser habilidada ou não a depender das dimensões do problema tratado. O treinamento da rede neural foi bastante simples. Procurou utilizar ao máximo técnicas já prontas e implementadas no MATLAB, uma vez que esse não era o foco do trabalho. Por outro lado, um melhor ajuste desta rede pode trazer melhores resultados.

Algoritmo 3.1: Identifica *drift*

input : Matriz M de tamanho $n_{variaveis} \times n_{amostras}$
output : *Array* contendo os índices de ocorrência de *drift*
parameter : A frequência de corte para o filtro
parameter : Limiar para detecção de *drift*
parameter : Tamanho da janela para horizonte de detecção

```
realizaDataAugmentation (M);
foreach janela j de M do
    M̄ ← ObtemParticionamentoM(M,j);
    JanelaPaddingInicial ← 1/10 das amostras iniciais de cada linha de M̄ em ordem invertida;
    JanelaPaddingFinal ← 1/10 das amostras finais de cada linha de M̄ em ordem invertida;
    M̄← Concatena (JanelaPaddingInicial,M̄,JanelaPaddingFinal);
    M̄ ← Normaliza (M̄);
    foreach linha l de M̄ do
        AFFT[l] ← FFT(M[l])
    foreach linha l de AFFT do
        AFFT[l] ← AFFT[l]*H(v)
    foreach linha l de AFFT do
        RES[l] ← IFFT(AFFT[l])
    foreach coluna c de RES do
        Intensidade[c] ← SOMA(RES[c])
    AmostraCandidata ← encontraMaximo(intensidade);
    foreach amostra a de Intensidade do
        if a < AmostraCandidata then
            classe[a] ← 0;
        else
            classe[a] ← 1;
    treinaRedeNeural(amostras,classe);
    AUC←calculaAUC(Ypred,Ytest);
    if AUC > limiar then
        Drift[j]←AmostraCandidata ;
Retorna Drift;
```

4 Resultados

Objetivando avaliar o desempenho do método proposto, foram realizados experimentos utilizando bases sintéticas comuns utilizadas em outros trabalhos semelhantes e disponíveis em Lobo (2020).

Também foram avaliados outros dois algoritmos muito citados na literatura para as mesmas bases a fim de realizar uma comparação. Os algoritmos usados para comparação foram o DDM e o EDDM apresentados por Gama et al. (2004) e Baena-Garcıa et al. (2006) respectivamente. As próximas seções explicam os resultados.

4.1 Bases utilizadas

Todas as bases utilizadas possuem 40.000 amostras com *drifts* abruptos nas amostras 10.000, 20.000 e 30.000. As bases são rotuladas contendo duas classes com número de atributos variando de 2 a 4. Foram avaliados os arquivos presentes na url <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/5OWRGB> listados na Tabela 1. Para o caso específico do algoritmo proposto, as classes também foram consideradas como dados de entrada. Apesar de estarem em posições fixas determinadas, é aceitável que os *drifts* sejam encontrados em regiões próximas àquelas listadas.

Tabela 1 – Bases utilizadas na avaliação.

Nome	Atributos
mixed_0101_abrupto.csv	4
mixed_1010_abrupto.csv	4
rt_2563789698568873_abrupto.csv	2
rt_8873985678962563_abrupto.csv	2
sine_0123_abrupto.csv	2
sine_3210_abrupto.csv	2
stagger_0120_abrupto.csv	3
stagger_2102_abrupto.csv	3

4.2 Parâmetros utilizados no algoritmo proposto

O algoritmo proposto possui como parâmetros de entrada a matriz M contendo as variáveis amostradas no tempo, a frequência de corte do filtro, o limiar da AUC para identificação do *drift* e o tamanho da janela de detecção. A Tabela 2 apresenta os valores utilizados.

Tabela 2 – Parâmetros utilizados no algoritmo proposto.

Base	Frequência de corte	Limiar AUC	Janela
mixed_0101_abrupto.csv	1.98π rad/amostra	0.65	700 amostras
mixed_1010_abrupto.csv	1.98π rad/amostra	0.65	700 amostras
rt_2563789698568873_abrupto.csv	1.98π rad/amostra	0.65	700 amostras
rt_8873985678962563_abrupto.csv	1.98π rad/amostra	0.6	700 amostras
sine_0123_abrupto.csv	1.99π rad/amostra	0.65	7000 amostras
sine_3210_abrupto.csv	1.98π rad/amostra	0.65	700 amostras
stagger_0120_abrupto.csv	1.995π rad/amostra	0.65	8000 amostras
stagger_2102_abrupto.csv	1.98π rad/amostra	0.65	700 amostras

4.3 Parâmetros utilizados no DDM

A fim de avaliar a eficiência do DDM para as bases propostas, foi adaptada uma implementação a partir daquela disponível em <https://github.com/pranab/beymani/tree/master/resource>. Entretanto, esta implementação pressupõe que já exista um modelo treinado gerando resultados de acertos e erros quando comparados com a base rotulada. Esse modelo não existia e portando foi implementado por meio da mesma rede neural já definida acima na implementação do algoritmo proposto por este trabalho. Desta forma, treinou-se aquela mesma rede para cada contexto da base tornando possível a integração com a implementação do DDM disponível. Além disso, essa implementação é mais flexível que a proposta original no que diz respeito à identificação do momento do *drift*. Foi inserido um novo parâmetro de limiar para a comparação da soma dos desvios padrões com a probabilidade de classificação incorreta. Desta forma, a implementação utilizou dois parâmetros fixos para todas as bases de acordo com a Tabela 3.

Tabela 3 – Parâmetros utilizados no algoritmo DDM.

Limiar	Tamanho da Janela
6	300 amostras

4.4 Parâmetros utilizados no EDDM

Analogamente ao caso anterior, a implementação do EDDM foi adaptada a partir daquela disponível em <https://github.com/pranab/beymani/tree/master/resource>. Também foi utilizada aquela mesma rede neural para gerar os modelos e os resultados de acertos e erros. Como o algoritmo original pressupõe um α no treinamento e, nesta adaptação, já se obteve o modelo treinado com os respectivos resultados de erros e acertos, os parâmetros utilizados foram β e a janela de detecção. Em outras palavras, o modelo já veio treinado pela rede neural proposta. Entretanto, houve a necessidade de alterar alguns destes parâmetros de acordo com a base utilizada. A Tabela 4 apresenta os detalhes.

Tabela 4 – Parâmetros utilizados no algoritmo EDDM.

Base	β	Tamanho da Janela
mixed_0101_abrupto.csv	0.35	300
mixed_1010_abrupto.csv	0.35	300
rt_2563789698568873_abrupto.csv	1	1000
rt_8873985678962563_abrupto.csv	1	1000
sine_0123_abrupto.csv	0.7	1000
sine_3210_abrupto.csv	0.7	1000
stagger_0120_abrupto.csv	0.7	10010
stagger_2102_abrupto.csv	0.7	10010

4.5 Identificação dos *drifts* e comparação dos resultados

Essa seção apresenta a comparação dos resultados da identificação dos pontos de *drift* para cada um dos algoritmos implementados. Cada base avaliada possui caracterísicas próprias tornando a tarefa de classficação fácil para algumas e não tão fácil para outras. Isso fez com que o desempenho do ponto de identificação dos desvios variasse de algoritmo para algoritmo. A Tabela 5 apresenta os resultados por base e os pontos de *drift* encontrados por cada algoritmo. Os valores em negrito representam os melhores resultados para cada base.

Tabela 5 – Comparação dos resultados por algoritmo - Identificação de *drifts*.

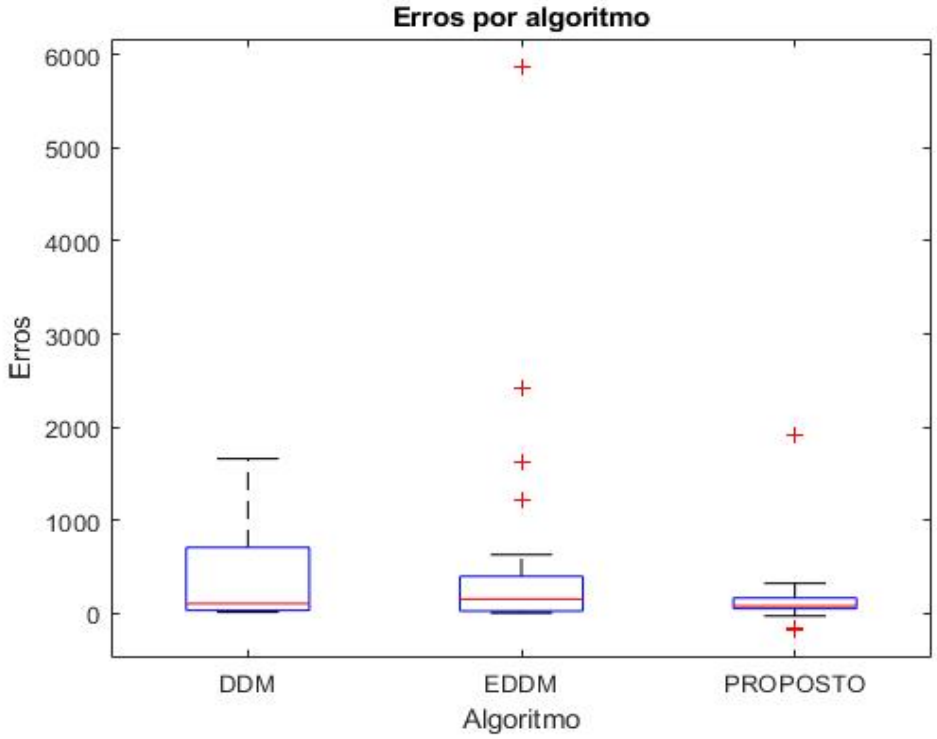
Base	DDM	EDDM	Proposto
mixed_0101_abrupto.csv	[10228 21913 **30032**]	[10116 20093 30192]	[**10079 20049** 30076]
mixed_1010_abrupto.csv	[11660 21054 30839]	[10169 20136 30165]	[**10074 20052 30075**]
rt_2563789698568873_abrupto.csv	[11430 20805 31022]	[11215 20307 31618]	[**10221 20125 30046**]
rt_8873985678962563_abrupto.csv	[10610 19043 30592]	[15869 20628 32420]	[**10044 20070 29821**]
sine_0123_abrupto.csv	[10032 20123 30529]	[10041 **20073 30299**]	[**9969** 20142 31923]
sine_3210_abrupto.csv	[10467 20496 30071]	[10340 20451 **30041**]	[**10134 20076** 29845]
stagger_0120_abrupto.csv	[10031 20014 30015]	[**10000 20004 30000**]	[10186 20320 30218]
stagger_2102_abrupto.csv	[10011 20023 30082]	[**10000 20000 30000**]	[10221 20088 30119]

A Figura 7 detalha graficamente o apresentado na Tabela 5. É apresentado um gráfico com os valores máximos, mínimos e médios dos erros por algoritmo. O limites das caixas mostras os valores máximos e mínimos. A linhas vermelhas evidenciam os valores médios. O quão mais próximo de zero estiver a caixa, menos erros aquele algoritmo cometeu. Quanto menor for a dimensão vertical da caixa, menor é o desvio padrão daquele algoritmo.

4.6 Considerações

As bases `stagger_0120_abrupto` e `stagger_2102_abrupto` apresentaram muito bons resultados no treinamento da rede neural proposta. Tanto a acurácia quanto a AUC ficaram em 1. Isso significa que o modelo foi capaz de acertar 100% das predições. Desta forma, para o EDDM que leva em consideração a distância entre dois erros consecutivos,

Figura 7 – Comparação dos erros de localização do *drift* por algoritmo.



foi impossível calcular a distância média de erros para qualquer janela menor que o contexo analisado pois não houve erros. Isso fez com que o algoritmo fosse incapaz de identificar os *drifts* para qualquer janela de avaliação menor que o próprio contexto (10000 amostras).

Além disso, pôde-se observar que tanto o DDM quanto o EDDM são bastante dependentes da qualidade do modelo treinado. Modelos que erram muito causam um considerável atraso na identificação do *drift*. Isso ocorre porque se tem uma dificuldade para identificar se o número grande de erros está presente em decorrência dos erros médios do modelo ou se realmente é porque o modelo já não é mais válido. Isso explica porque para as duas bases citadas, cujo modelo ficou bastante preciso, a identificação do *drift* ocorreu com um atraso relativamente pequeno. Já o método proposto apresentou dificuldades para encontrar o *drift* principalmente nas bases `sine_0123_abrupto`, `rt_8873985678962563_abrupto` e `stagger_0120_abrupto`. Isso ocorreu muito em função de sua dificuldade em identificar uma resposta alta do filtro no domínio da frequência para poucas dimensões. Essas respostas são somadas em todas as dimensões da entrada com o intuito de identificar mais facilmente as bordas e potencializar os resultados, entretanto, como a entrada para estas bases possui apenas duas dimensões, a resposta não foi num nível adequado. Apesar disso, o método proposto traz a vantagem de ser não supervisionado e ainda capaz de identificar de forma mais rápida a maioria dos *drifts* cujo treinamento da rede neural não apresentou resultados excelentes.

Vale ressaltar algumas observações interessantes. O DDM foi o método mais robusto

com relação à variação de parâmetros. Para todas as bases, obteve resultado satisfatório com os mesmos parâmetros. Já o EDDM foi o mais sensível com relação aos parâmetros. Com relação ao tempo de execução, tanto o DDM quanto o EDDM se mostraram bastante rápidos após a obtenção dos resultados de avaliação dos modelos. Isso era esperado tendo em vista a natureza pouco complexa das operações matemáticas envolvidas no processamento. Tanto o DDM quanto o EDDM são bastante simples, porém, para casos reais, são bastante difíceis de serem implementados diante da dificuldade de se conseguir um bom modelo dependente de bases rotuladas. Nem sempre é possível ter um modelo treinado.

Já o método proposto por este trabalho é completamente independente de bases rotuladas podendo ser executado praticamente online. Além disso, por se tratar de um método baseado em filtros no domínio da frequência, não necessariamente encontra o *drift* após o seu acontecimento real. É possível identificar a tendência e, dependendo dos limiares, alarmar o quanto antes numa região bem próxima do ponto de desvio. Exatamente por isso, naquelas bases de maiores dimensões, um pouco mais complexas, este método foi capaz de encontrar o desvio mais rapidamente. Determinar o tamanho da janela para se obter resultados aceitáveis foi outra atividade bastante complexa para todos os métodos. Esse tamanho depende bastante da natureza dos dados e da posição do desvio. Para aqueles casos em que o limite da janela coincide ou se aproxima bastante do início do *drift*, pode causar a geração de um conjunto de dados desbalanceados para a rede neural. Isso pode levar a um subconjunto de testes contendo todos os exemplos de uma mesma classe e dificultando uma conclusão definitiva se o modelo treinado realmente foi bom ou não. Esse foi um desafio abordado empiricamente.

5 Conclusões

Esse capítulo apresenta as principais conclusões obtidas a partir do estudo e proposta de métodos automáticos para a detecção de *drift*. Foi possível perceber que essa é uma área de pesquisa bastante aquecida com aplicações variando desde áreas industriais, passando pelo mercado de ações até a identificação de mudanças de comportamento de consumidares em plataformas *online* de compras. Também foi possível concluir que não existem tantos trabalhos focados na identificação automática de desvios em processos industriais. Apesar de não ser a maioria dos trabalhos publicados, existe uma tendência de se explorar métodos que trabalham com aprendizagem não supervisionada muito em função da dificuldade de se obter bases reais de qualidade e rotuladas. Ainda, aqueles métodos que funcionam de forma *online* também trazem um diferencial positivo.

O método proposto trouxe a vantagem de ser não supervisionado podendo ser empregado de forma *online* para casos em que o processo é lento. Além disso, outra vantagem identificada foi o fato de que, para processos bem monitorados com um número grande de variáveis, o resultado tende a ser melhor. Isso é um diferencial tendo em vista que a alta dimensionalidade das variáveis tende a ser um problema para muitos métodos apresentados na literatura. Alguns chegam a usar técnicas para a redução de dimensão como PCA.

O capítulo de resultados apresentou a comparação com os métodos DDM e EDDM por meio do uso de 8 bases bem conhecidas na literatura e disponibilizadas para este fim. Por ser baseado em filtros no domínio da frequência, o método proposto traz a vantagem de identificar ou até mesmo antecipar tendências de desvio. Isso auxilia num tempo de resposta menor, não esperando necessáriamente que o *drift* aconteça para alarmar. Os métodos DDM e EDDM se mostraram bastante dependentes do modelo treinado na etapa anterior, de forma que, para modelos ruins, a taxa de erro pode se confundir com o próprio erro causado pelo *drift*. Foi observado na prática que, para modelos bons, os dois métodos têm respostas excelentes. Por fim, algumas possiblidades identificadas para se obter melhores ajustes para o método proposto serão listadas na seção de trabalhos futuros.

5.1 Trabalhos futuros

Muito em função do escopo limitado e a falta de tempo, algumas abordagens interessantes não foram exploradas. Como trabalhos futuros, existe a possibilidade de avaliar o método com bases de dados reais de processos industriais, inserir a possibilidade de se ter uma sobreposição entre as janelas analisadas de forma a minimizar a existência

de bases desbalanceadas para o treinamento da rede, implementar um ajuste fino mais detalhado para a rede neural utilizada, trabalhar com bases graduais e incrementais e avaliar o desempenho com relação a ruídos e *outliers*.

Outra abordagem futura deve ser uma avalição do uso da coerência, ou espectro de coerência cruzada, no domínio da frequência. Essa medida traz um conceito similar ao que seria a correlação no domínio do tempo. Dependendo do resultado desta avaliação, pode ser interessante usar essa métrica ao invés de simplesmente somar os resultados dos filtros de todas as variáveis no domínio do tempo.

APÊNDICE A – Código Fonte

```
clear all
close all
clc

% Geracao da base artificial para teste do algoritmo somente
mu1 = [5.1 3 5 4 5.8 5 1 2 5.5 7];
mu2 = [3 3.2 2.8 3 6 5 1 2 5 8];
Sigma1 = 5*eye(10);
Sigma2 = 5*eye(10);

R1 = mvnrnd(mu1,Sigma1,300);
R2 = mvnrnd(mu2,Sigma2,400);

janela = 700;

corte_Frequencia = floor(janela/100);

drift_pos = zeros (1);
drift_auc = zeros (1);
% Criacao da imagem baseado dos exemplos

Im = [R1;R2];

% ALtere aqui para usar a base artificial gerada acima

Files=dir('./Stagger DataSet/*.csv');
diary Log_Saida
for f=1:length(Files)

    Nome = strcat('Stagger DataSet/',Files(f).name)
    T = readtable(Nome);

    Im_array = table2array(T);

    Im_array_q = abs((Im_array - Im_array.*1.5)).^2;
    Im_array_q2 = ((Im_array+0.4).*0.3).^1.7;

    Im_array = [Im_array,Im_array_q,Im_array_q2];

    Result=split(Im_array,janela);

```

```
42      % Para cada janela, procura pelo drift e avalia por meio da AUC
43
44
45      for k=1:size(Result,3)-1
46
47          Im = Result(:,:,k);
48
49          % Implementa o padding
50
51          Janela_Anterior = flip(Im(1:corte_Frequencia*10,:));
52          Janela_Posterior = flip(Im(end-corte_Frequencia*10:end,:));
53          Im = [Janela_Anterior;Im;Janela_Posterior];
54
55          NIm=normalize(Im,'range'); % normaliza os dados por colunas
56
57          imagem = Im';
58
59
60          [L,C] = size(imagem);
61          Afft = fft(imagem(1,:));
62
63          Afft(1:C-corte_Frequencia)=0; % filtro passa altas
64          for i=2:L
65              Afft = [Afft; fft(imagem(i,:))];
66              Afft(i,1:C-corte_Frequencia)=0;
67          end
68              Res = abs(ifft(Afft(1,:)));
69          for j=2:L
70              Res = [Res; abs(ifft(Afft(j,:)))];
71          end
72
73          Res_Final = Res(:,corte_Frequencia*10:end-corte_Frequencia*20);
74
75          im_final = imagem(:,corte_Frequencia*10:end-corte_Frequencia*20);
76
77          Arr_Filtered = sum(abs(Res_Final));
78          %figure,plot(Arr_Filtered);
79          %ylabel('Intensidade');
80          %xlabel('Amostra');
81          %xlim([0 1000]);
82          %ylim([1.5 5.5]);
83
84          [max_value, ind] = max(Arr_Filtered);
85
86          if (ind == 1 || ind == size(Arr_Filtered,2))
87              ind = floor(size(Arr_Filtered,2)/2);
88          end
```

```
89
90          [Li,Col] = size(im_final);
91          classes = ones(1,Col);
92          classes(1,1:ind) = 0;
93          im_final = [im_final;classes]';
94
95          C1 = arrayfun(@(x) sprintf('Var_%d', x), 1:Li, 'UniformOutput',...
96                                        false);
97          VN = [C1,{'classe'}];
98
99          tbl = array2table(im_final,'VariableNames',VN);
100
101         labelName = "classe";
102
103         tbl = convertvars(tbl,labelName,'categorical');
104
105         tbl = splitvars(tbl);
106
107         classNames = categories(tbl{:,labelName});
108
109         numObservations = size(tbl,1);
110         numObservationsTrain = floor(0.6*numObservations);
111         numObservationsValidation = floor(0.2*numObservations);
112         numObservationsTest = numObservations - numObservationsTrain...
113                               - numObservationsValidation;
114
115         idx = randperm(numObservations);
116         idxTrain = idx(1:numObservationsTrain);
117         idxValidation = idx(numObservationsTrain+1:numObservationsTrain...
118                             +numObservationsValidation);
119         idxTest = idx(numObservationsTrain+numObservationsValidation...
120                       +1:end);
121
122         tblTrain = tbl(idxTrain,:);
123         tblValidation = tbl(idxValidation,:);
124         tblTest = tbl(idxTest,:);
125
126         numFeatures = size(tbl,2) - 1;
127
128         numClasses = numel(classNames);
129
130         if(numClasses == 1)
131             continue;
132         end
133
134         layers = [
135             featureInputLayer(numFeatures)
```

```
136             fullyConnectedLayer(70)
137             batchNormalizationLayer
138             tanhLayer
139             fullyConnectedLayer(numClasses)
140             softmaxLayer
141             classificationLayer];
142
143         miniBatchSize = 4;
144
145         options = trainingOptions('adam', ...
146             'MiniBatchSize',miniBatchSize, ...
147             'Shuffle','every-epoch', ...
148             'ValidationData',tblValidation, ...
149             'Plots','training-progress', ...
150             'Verbose',false,...
151             'plots',"none");
152
153         net = trainNetwork(tblTrain,labelName,layers,options);
154
155         YPred = classify(net,tblTest(:,1:end-1),'MiniBatchSize',...
156                         miniBatchSize);
157
158         YTest = tblTest{:,labelName};
159
160         accuracy = sum(YPred == YTest)/numel(YTest);
161
162         figure
163         confusionchart(YTest,YPred)
164
165         soma = sum(int32(YTest));
166         % se YTeste tem somente uma classe --
167         if(soma == size(YTest,1) || soma == 0)
168             continue;
169         end
170
171         %Taxa de true positivo
172         yp = double(string(YPred));
173         yt = double(string(YTest));
174         total = size(yp);
175         tp = sum(yp(yp==1)& yt(yp==1))/total(1);
176         %Taxa de false positivo
177         fp = sum(xor(yp(yp==1),yt(yp==1)))/total(1);
178
179         [X,Y,T,AUC] = perfcurve(string(yt),yp,"1");
180
181
182         fprintf('pos = %d, max_value = %d, Janela = %d, Acuracy = %f,...
```

```
183                     AUC = %f\n',(k-1)*janela+ind,max_value, k,accuracy,AUC);
184
185           if(round(AUC,3) >= 0.65)
186               drift_auc = [drift_auc,AUC]
187               drift_pos = [drift_pos,ind+(k-1)*janela]
188           end
189       close all;
190       end
191       drift_auc = 0;
192       drift_pos = 0;
193   end
194   diary off
```

Referências

ALIPPI, C.; ROVERI, M. Just-in-time adaptive classifiers - part I: detecting nonstationary changes. *IEEE Trans. Neural Networks*, v. 19, n. 7, p. 1145–1153, 2008. Disponível em: <https://doi.org/10.1109/TNN.2008.2000082>. Citado na página 27.

BAENA-GARCIA, M. et al. Early drift detection method. In: *Fourth international workshop on knowledge discovery from data streams*. [S.l.: s.n.], 2006. v. 6, p. 77–86. Citado 2 vezes nas páginas 30 e 43.

BLANCO, I. I. F. et al. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Trans. Knowl. Data Eng.*, v. 27, n. 3, p. 810–823, 2015. Disponível em: <https://doi.org/10.1109/TKDE.2014.2345382>. Citado na página 27.

BU, L.; ALIPPI, C.; ZHAO, D. A pdf-free change detection test based on density difference estimation. *IEEE Trans. Neural Networks Learn. Syst.*, v. 29, n. 2, p. 324–334, 2018. Disponível em: <https://doi.org/10.1109/TNNLS.2016.2619909>. Citado na página 27.

CHRISTENSON, D. W.; REDDY, B. S.; ROWLANDSON, G. I. Evaluation of fourier transform filter for high-resolution ecg. *Journal of electrocardiology*, Elsevier, v. 22, p. 33–40, 1990. Citado 2 vezes nas páginas 36 e 38.

DASU, T. et al. An information-theoretic approach to detecting changes in multidimensional data streams. *Interfaces*, 01 2006. Citado na página 27.

G, P.; BEIGY, H. New drift detection method for data streams. In: BOUCHACHIA, A. (Ed.). *Adaptive and Intelligent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 88–97. ISBN 978-3-642-23857-4. Citado na página 23.

GAMA, J. A survey on learning from data streams: current and future trends. *Prog. Artif. Intell.*, v. 1, n. 1, p. 45–55, 2012. Disponível em: <https://doi.org/10.1007/s13748-011-0002-6>. Citado na página 27.

GAMA, J. et al. Learning with drift detection. In: BAZZAN, A. L. C.; LABIDI, S. (Ed.). *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*. Springer, 2004. (Lecture Notes in Computer Science, v. 3171), p. 286–295. Disponível em: <https://doi.org/10.1007/978-3-540-28645-5_29>. Citado 3 vezes nas páginas 27, 29 e 43.

GEMAQUE, R. N. et al. An overview of unsupervised drift detection methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 10, n. 6, p. e1381, 2020. Citado 2 vezes nas páginas 27 e 31.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (2nd Edition)*. USA: Prentice-Hall, Inc., 2002. ISBN 013168728X. Citado 2 vezes nas páginas 17 e 37.

GÖZÜAÇIK, Ö. et al. Unsupervised concept drift detection with a discriminative classifier. In: *Proceedings of the 28th ACM international conference on information and knowledge management*. [S.l.: s.n.], 2019. p. 2365–2368. Citado 2 vezes nas páginas 33 e 40.

IWASHITA, A. S.; PAPA, J. P. An overview on concept drift learning. *Ieee Access*, IEEE, v. 7, p. 1532–1547, 2018. Citado 3 vezes nas páginas 17, 30 e 31.

JARAMILLO-VALBUENA, S.; LONDOÑO-PELÁEZ, J. M.; CARDONA, S. A. Performance evaluation of concept drift detection techniques in the presence of noise. *Performance evaluation*, v. 38, n. 39, 2017. Citado na página 28.

KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. *Inf. Fusion*, v. 37, p. 132–156, 2017. Disponível em: <https://doi.org/10.1016/j.inffus.2017.02.004>. Citado na página 27.

LIU, A. et al. Regional concept drift detection and density synchronized drift adaptation. In: SIERRA, C. (Ed.). *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 2017. p. 2280–2286. Disponível em: <https://doi.org/10.24963/ijcai.2017/317>. Citado na página 24.

LIU, A.; ZHANG, G.; LU, J. Fuzzy time windowing for gradual concept drift adaptation. In: *2017 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2017, Naples, Italy, July 9-12, 2017*. IEEE, 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/FUZZ-IEEE.2017.8015596>. Citado na página 24.

LOBO, J. L. *Synthetic datasets for concept drift detection purposes*. Harvard Dataverse, 2020. Disponível em: <https://doi.org/10.7910/DVN/5OWRGB>. Citado na página 43.

LU, J. et al. Learning under concept drift: A review. *CoRR*, abs/2004.05785, 2020. Disponível em: <https://arxiv.org/abs/2004.05785>. Citado 5 vezes nas páginas 17, 23, 24, 27 e 28.

LU, N. et al. A concept drift-tolerant case-base editing technique. *Artif. Intell.*, v. 230, p. 108–133, 2016. Disponível em: <https://doi.org/10.1016/j.artint.2015.09.009>. Citado na página 24.

LU, N.; ZHANG, G.; LU, J. Concept drift detection via competence models. *Artif. Intell.*, v. 209, p. 11–28, 2014. Disponível em: <https://doi.org/10.1016/j.artint.2014.01.001>. Citado 2 vezes nas páginas 24 e 27.

MELLO, R. F. de et al. On learning guarantees to unsupervised concept drift detection on data streams. *Expert Systems with Applications*, Elsevier, v. 117, p. 90–102, 2019. Citado na página 33.

MITCHELL, T. Machine learning. McGraw hill Burr Ridge, 1997. Citado na página 29.

MUSTAFA, A. M. et al. Unsupervised deep embedding for novel class detection over data stream. In: IEEE. *2017 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2017. p. 1830–1839. Citado na página 33.

RAMÍREZ-GALLEGO, S. et al. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, v. 239, p. 39–57, 2017. Disponível em: <https://doi.org/10.1016/j.neucom.2017.01.078>. Citado na página 27.

REIS, D. M. dos et al. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2016. p. 1545–1554. Citado na página 32.

ROSSET, S. Model selection via the auc. In: *Proceedings of the twenty-first international conference on Machine learning*. [S.l.: s.n.], 2004. p. 89. Citado na página 39.

SILVA, J. de A. et al. Data stream clustering: A survey. *ACM Comput. Surv.*, v. 46, n. 1, p. 13:1–13:31, 2013. Disponível em: <https://doi.org/10.1145/2522968.2522981>. Citado na página 27.

SOUZA, V. M.; CHOWDHURY, F. A.; MUEEN, A. Unsupervised drift detection on high-speed data streams. In: IEEE. *2020 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2020. p. 102–111. Citado 5 vezes nas páginas 17, 31, 32, 35 e 40.

SUN, Z. et al. Review of concept drift detection method for industrial process modeling. In: IEEE. *2020 39th Chinese Control Conference (CCC)*. [S.l.], 2020. p. 5754–5759. Citado na página 28.