Pontifícia Universidade Católica
do Rio de Janeiro

Tatiana Escovedo

Aprendizagem Neuroevolutiva e Detecção de Concept Drift em Ambientes Não Estacionários

Tese de Doutorado

Tese apresentada ao programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio como parte dos requisitos parciais para obtenção do título de Doutor em Engenharia Elétrica.

Orientadora: Profa. Marley Maria Bernardes Rebuzzi Vellasco

Co-orientador: Prof. André Vargas Abs da Cruz

Rio de Janeiro

Setembro de 2015

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Tatiana Escovedo

Aprendizagem Neuroevolutiva e Detecção de Concept Drift em Ambientes Não Estacionários

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Profa. Marley Maria Bernardes Rebuzzi Vellasco
Orientadora
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Andre Vargas Abs da Cruz
Co-orientador
UEZO

Profa. Karla Tereza Figueiredo Leite
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Jorge Luís Machado do Amaral
UERJ

Prof. Fabiano Saldanha Gomes de Oliveira
UERJ

Prof. Douglas Mota Dias
UERJ

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 17 de setembro de 2015

Tatiana Escovedo

É mestre em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2007), na área de Engenharia de Software e é bacharel em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2005). Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: Inteligência Computacional, Business Intelligence, Data Warehouse, Sistemas Colaborativos, BPM, Redes Neurais. Atualmente, é Analista de Sistemas da Petrobras e Professora da Pós-Graduação da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

Ficha catalográfica

Escovedo, Tatiana

Aprendizagem neuroevolutiva e detecção de concept drift em ambientes não estacionários / Tatiana Escovedo ; orientadores: Marley Maria Bernardes Rebuzzi Vellasco, André Vargas ABS da Cruz. – 2015.

149 f. : il. (color.) ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2015.

Inclui bibliografia

1. Engenharia elétrica – Teses. 2. Neuroevolução. 3. Concept drift. 4. Detecção de concept drift. 5. Comitê de redes neurais. 6. Classificação. I. Vellasco, Marley Maria Bernardes Rebuzzi. II. Cruz, André Vargas Abs da. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

Agradecimentos

Gostaria de agradecer a todos que contribuíram direta ou indiretamente para a realização desta pesquisa. Primeiramente, aos meus orientadores e amigos professores Marley Vellasco e André Abs da Cruz, por terem acreditado no meu potencial e guiado com carinho os meus passos durante os anos do doutorado.

A minha família, em especial aos meus pais – Cristina e Mauricio -, por terem me proporcionado sempre a melhor educação que eu poderia ter. Ao meu irmão Rafael pela amizade e ao meu marido Rubens, por toda a paciência, apoio e compreensão da minha ausência durante os últimos anos de estudo.

Ao meu "filho adotado" Adriano Koshiyama, que muito contribuiu para a idealização e implementação dos modelos propostos na tese, e compartilhou comigo discussões, experimentos, artigos e pizzas.

Aos meus gerentes da Petrobras, Leonardo Matos e Carlos Alberto Rechelo, e aos meus ex-gerentes Vania Campinho e Flaviano Motta, por terem possibilitado conciliar o doutorado com minhas atividades profissionais, compreendendo minha ausência em diversos momentos. A todos os meus colegas e ex-colegas de trabalho da área de RISCOS/AQR, E&P/EXP/GEO e TIC/CPSW/IST-II que se tornaram grandes amigos e me apoiaram neste caminho.

Aos meus amigos de todas as horas, Arthur Barbosa, Carlos Eduardo Leal, Patricia Macedo, Ava Rozenblat, Ana Paula Morrissy, Ana Beatriz Nascimento, Julia Longo, Gabriella Allegri, Gabriela Patrício, Clara Silveira, Igor Sacramento, Vanissa Vieira, Juliana Frota, Luiza Mallet, João Carrilho, Ana Paula Freitas e a todos os meus grandes amigos bailarinos do Studio de Ballet Bertha Rosanova, que me ajudaram a manter o equilíbrio com a diversão das aulas e ensaios. Aos amigos desde a graduação do N-eto pelo carinho e aos colegas Zair Ramos, Fábio Pimentel, Silas Garrido, Paulo Gonçalves, Eric Praxedes, Eduardo Alvarenga e Fábio Mendonça, por terem contribuído de alguma forma em algum aspecto técnico da implementação da tese. Finalmente, às minhas cachorrinhas Belinha e Pequena, pela companhia nas madrugadas de trabalho.

Resumo

Escovedo, Tatiana; Vellasco, Marley Maria Bernardes Rebuzzi (Orientadora); da Cruz, André Vargas Abs (Co-orientador). **Aprendizagem Neuroevolutiva e Detecção de Concept Drift em Ambientes Não Estacionários.** Rio de Janeiro, 2015. 149p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Os conceitos do mundo real muitas vezes não são estáveis: eles mudam com o tempo. Assim como os conceitos, a distribuição de dados também pode se alterar. Este problema de mudança de conceitos ou distribuição de dados é conhecido como *concept drift* e é um desafio para um modelo na tarefa de aprender a partir de dados. Este trabalho apresenta um novo modelo neuroevolutivo com inspiração quântica, baseado em um comitê de redes neurais do tipo *Multi-Layer Perceptron* (MLP), para a aprendizagem em ambientes não estacionários, denominado NEVE (*Neuro-EVolutionary Ensemble*). Também apresenta um novo mecanismo de detecção de *concept drift*, denominado DetectA (*Detect Abrupt*) com a capacidade de detectar mudanças tanto de forma proativa quanto de forma reativa. O algoritmo evolutivo com inspiração quântica binário-real AEIQ-BR é utilizado no NEVE para gerar automaticamente novos classificadores para o comitê, determinando a topologia mais adequada para a nova rede, selecionando as variáveis de entrada mais apropriadas e determinando todos os pesos da rede neural MLP. O algoritmo AEIQ-R determina os pesos de votação de cada rede neural membro do comitê, sendo possível utilizar votação por combinação linear, votação majoritária ponderada e simples. São implementadas quatro diferentes abordagens do NEVE, que se diferem uma da outra pela forma de detectar e tratar os *drifts* ocorridos. O trabalho também apresenta resultados de experimentos realizados com o método DetectA e com o modelo NEVE em bases de dados reais e artificiais. Os resultados mostram que o detector se mostrou robusto e eficiente para bases de dados de alta dimensionalidade, blocos de tamanho intermediário, bases de dados com qualquer proporção de *drift* e com qualquer balanceamento de classes e que, em geral, os melhores resultados obtidos foram usando algum tipo de detecção. Comparando a acurácia do NEVE com outros modelos consolidados da literatura, verifica-se que o NEVE teve acurácia superior na maioria dos casos. Isto reforça que a abordagem por comitê neuroevolutivo é uma escolha robusta para situações em que as bases de dados estão sujeitas a mudanças repentinas de comportamento.

Palavras-chave

Neuroevolução, Concept Drift, Detecção de Concept Drift, Comitê de Redes Neurais, Classificação; Ambientes Não Estacionários.

Abstract

Escovedo, Tatiana; Vellasco, Marley Maria Bernardes Rebuzzi (Advisor); da Cruz, André Vargas Abs (Co-advisor). **Neuroevolutive Learning and Concept Drift Detection in Non-Stationary Environments.** Rio de Janeiro, 2015. 149p. DSc. Thesis – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Real world concepts are often not stable: they change with time. Just as the concepts, data distribution may change as well. This problem of change in concepts or distribution of data is known as concept drift and is a challenge for a model in the task of learning from data. This work presents a new neuroevolutive model with quantum inspiration called NEVE (Neuro-EVolutionary Ensemble), based on an ensemble of Multi-Layer Perceptron (MLP) neural networks for learning in non-stationary environments. It also presents a new concept drift detection mechanism, called DetectA (DETECT Abrupt) with the ability to detect changes both proactively as reactively. The evolutionary algorithm with binary-real quantum inspiration AEIQ-BR is used in NEVE to automatically generate new classifiers for the ensemble, determining the most appropriate topology for the new network and by selecting the most appropriate input variables and determining all the weights of the neural network. The AEIQ-R algorithm determines the voting weight of each neural network ensemble member, and you can use voting by linear combination and voting by weighted or simple majority. Four different approaches of NEVE are implemented and they differ from one another by the way of detecting and treating occurring drifts. The work also presents results of experiments conducted with the DetectA method and with the NEVE model in real and artificial databases. The results show that the detector has proved efficient and suitable for data bases with high-dimensionality, intermediate sized blocks, any proportion of drifts and with any class balancing. Comparing the accuracy of NEVE with other consolidated models in the literature, it appears that NEVE had higher accuracy in most cases. This reinforces that the neuroevolution ensemble approach is a robust choice to situations in which the databases are subject to sudden changes in behavior.

Keywords

Neuroevolution, Concept Drift, Concept Drift Detection, Neural Networks Ensembles, Classification; Non-Stationary Environments.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

1.1.
Motivação

A capacidade de um classificador aprender a partir de dados incrementais e dinâmicos, extraídos de um ambiente não estacionário (quando a distribuição dos dados se altera ao longo do tempo), representa um desafio para o campo da inteligência computacional. No âmbito das redes neurais o problema se torna ainda mais complexo, pois a maioria dos modelos existentes devem ser retreinados quando um novo bloco de dados se torna disponível, usando todo o conjunto de padrões aprendidos até então. A fim de lidar com este problema de aprendizagem em ambientes não estacionários, um classificador deve, idealmente, ser capaz de [Schlimmer & Granger, 1986]:

- Monitorar e detectar qualquer tipo de mudança na distribuição da base de dados;
- Aprender com novos dados sem a necessidade de apresentar novamente todo o conjunto de dados para o classificador;
- Ajustar os seus próprios parâmetros, a fim de tratar as alterações detectadas nos dados;
- Esquecer o que foi aprendido quando esse conhecimento não for mais útil para a classificação de novos padrões.

Todas estas habilidades buscam, de uma forma ou de outra, tratar um fenômeno chamado de *concept drift* [Tsymbal, 2004; Karnick et al., 2008]. Este fenômeno define conjuntos de dados que sofrem alterações ao longo do tempo como, por exemplo, quando há mudança na relevância das variáveis, ou quando a média e a variância das variáveis sofrem alterações. A maioria dos trabalhos na área de ambientes não estacionários de aprendizagem foi publicada na última década e pode-se observar que, até agora, não existe uma terminologia padrão [Moreno-Torres et al., 2012]. Porém, o termo *concept drift* é o que tem sido mais

usado para caracterizar problemas deste tipo e, por este motivo, optou-se por utilizar este termo ao longo deste trabalho.

Muitas abordagens já foram concebidas a fim de contemplar algumas ou todas as habilidades acima mencionadas. Uma das abordagens mais antigas e também mais simples consiste em utilizar uma janela deslizante (nem sempre contínua) sobre os dados de entrada e treinar o classificador com os dados delimitados por esta janela [Hulten et al., 2001]. Outra abordagem consiste em detectar desvios e, caso ocorram, ajustar o classificador [Carvalho & Cohen 2006]. Uma abordagem mais bem-sucedida e muito utilizada atualmente consiste em utilizar um conjunto de classificadores (comitê de classificadores ou *ensemble*). Este tipo de abordagem utiliza um grupo de diferentes classificadores, a fim de ser capaz de lidar com as alterações no ambiente. Vários modelos diferentes de comitê de classificadores foram propostos na literatura [Kuncheva, 2008; Elwell & Polikar, 2011; Kuncheva, 2004; Gama et al., 2014] e podem ser categorizados em três tipos, sendo que estas categorias não são mutuamente exclusivas, podendo ser combinadas umas com as outras:

- Comitês com Combinação Dinâmica: os classificadores base são treinados apenas uma vez e depois combinados dinamicamente através da modificação da regra de combinação ou dos pesos de votação para responder a mudanças no ambiente, como em [Littlestone, 1988; Blum, 1997; Widmer & Kubat, 1996; Xingquan et al. 2004]. Como os classificadores não são retreinados em nenhum momento e nem são adicionados novos classificadores ao comitê, esta abordagem é pouco adequada para lidar com ambientes não estacionários e tem sido cada vez menos utilizada na literatura para esta classe de problemas [Brzeziński, 2010];

- Comitês com atualização contínua dos seus membros: neste caso é possível retreinar os classificadores em modo *batch* ou atualizá-los *online* usando novos dados. A regra de combinação pode ou não ser modificada ao longo do processo, como em [Breiman, 1999; Fern & Givan, 2003; Oza, 2001; Gama et al., 2004; Connoly et al., 2013];

- Comitês com alterações estruturais nos seus membros: novos classificadores podem ser adicionados e os antigos podem ser desativados ou reativados com base na sua eficiência em dados

recentes, como em [Street & Kim, 2001; Kolter & Maloof, 2003; Bouchachia, 2011; Chen & He, 2011; Elwell & Polikar, 2011].

Os tipos de comitês acima mencionados podem ou não ponderar cada um dos seus membros. A maioria dos modelos que utilizam comitês de classificadores ponderados determinam os pesos para cada classificador usando algum conjunto de heurísticas relacionadas ao desempenho do classificador nos dados recebidos mais recentemente [Karnick et al., 2008]. Embora, a princípio, qualquer classificador possa ser usado para construir os comitês, neste trabalho decidiu-se focar apenas em redes neurais, a fim de melhor explorar as particularidades deste tipo de solução.

Apesar de diversos algoritmos já terem sido propostos na literatura para a classificação em cenários de *concept drift* - muitos inclusive utilizando comitês de classificadores - para este tipo de problema, a neuroevolução ainda foi pouco explorada. Neuroevolução é uma forma de aprendizado de máquina que usa algoritmos evolutivos para ajustar os parâmetros que afetam o desempenho das redes neurais artificiais, tais como topologia, taxa de aprendizagem, pesos, entre outros. Neste caso, cada solução armazena uma representação destes parâmetros, que são evoluídos a fim de buscar a rede ótima para o problema. A neuroevolução tende a melhorar o processo de treinamento, uma vez que os algoritmos evolutivos não ficam presos em mínimos locais [Linden, 2012].

Por se tratar de uma arquitetura complexa, é preciso que os modelos neuroevolutivos baseados em comitês de classificadores tenham bom desempenho computacional e apresentem rápida convergência, a fim de serem aptos para aplicação em cenários reais. Esta característica se torna ainda mais relevante em ambientes não estacionários, uma vez que é necessária a atualização do comitê cada vez que novos dados se tornam disponíveis ou que alguma mudança é detectada no comportamento dos dados. Assim, é preciso que esta etapa seja rápida para não comprometer o desempenho global do modelo. Para tal, uma estratégia interessante e ainda pouco explorada na literatura relacionada a modelos neuroevolutivos é a utilização de algoritmos evolutivos com inspiração quântica, a fim de tirar proveito de seu desempenho superior para otimização combinatória e numérica em comparação com seus algoritmos genéticos canônicos homólogos. O algoritmo evolutivo com inspiração quântica para otimização numérica com representação mista (AEIQ-BR) [Pinho, 2010], detalhado posteriormente, tem demonstrado bom desempenho quando usado para treinamento de redes neurais. Aplicado a comitês de redes neurais, o

AEIQ-BR pode também ser usado para determinar os pesos de votação para cada classificador que faz parte do comitê. Assim, cada vez que um novo bloco de dados chega, os pesos do comitê podem ser otimizados, melhorando o seu desempenho na classificação deste novo conjunto de dados.

Os modelos para aprendizagem em ambientes não estacionários podem ou não conter mecanismos de detecção de *drift*. A maioria dos modelos encontrados na literatura assume que as mudanças ocorrem em um contexto escondido externo ao modelo em si e, por isso, o *drift* não pode ser previsto [Gama et al., 2014]. Por este motivo, estes modelos utilizam a abordagem passiva e reativa, ou seja, a partir dos resultados do modelo (no caso de classificação, é feita a comparação do rótulo previsto pelo modelo com o rótulo verdadeiro recebido), verificam que houve o *drift* e reagem a ele só após a ocorrência de erro no modelo. Entretanto, antecipar-se para detectar a ocorrência de *drift* nos dados de entrada antes que eles sejam submetidos para predição (ou seja, antes do recebimento dos rótulos verdadeiros) parece ser uma abordagem mais satisfatória, uma vez que é possível ajustar o modelo previamente, de forma a lidar melhor com o novo cenário e evitar o erro de classificação.

1.2. Objetivos e Contribuições

Dado o exposto anteriormente, o objetivo principal deste trabalho é propor e desenvolver um modelo auto adaptável, flexível, com boa acurácia e adequado para aprendizado em ambientes não estacionários.

A contribuição principal deste trabalho é, portanto, a criação de um novo modelo neuroevolutivo com inspiração quântica, baseado em um comitê de redes neurais do tipo *Multi-Layer Perceptron* (MLP), para a aprendizagem em ambientes não estacionários. Este modelo, denominado NEVE (do acrônimo em inglês *Neuro-EVolutionary Ensemble*), tem as seguintes características:

- Contém mecanismo de detecção de *concept drift* com a capacidade de detectar as mudanças ocorridas. Este método possibilita a reação e o ajuste do modelo sempre que necessário;
- Realiza a geração automática de novos classificadores para o comitê, mais adequados para o tratamento dos novos dados de

entrada. A criação dos classificadores, através do algoritmo evolutivo com inspiração quântica AEIQ-BR, envolve:

- A determinação da topologia mais adequada para a classificação do bloco de dados corrente;
- A seleção das variáveis de entrada mais apropriadas para a rede, dentre as variáveis disponíveis no bloco de dados de entrada;
- A determinação de todos os pesos da rede neural MLP.

- Determina automaticamente os pesos de votação de cada classificador membro do comitê através do algoritmo AEIQ-R, uma versão simplificada do AEIQ-BR.

Como contribuição secundária da pesquisa, foi realizada uma análise de sensibilidade do mecanismo de detecção de *drift* proposto com base na alteração do número de atributos, padrões, taxa de desbalanceamento entre classes, entre outros, a fim de compreender a influência de cada uma destas variáveis no desempenho do método e aprimorá-lo para que possa ser utilizado em modelos de aprendizagem em ambientes não estacionários.

Foram realizados diversos experimentos com bases de dados artificiais e reais a fim de validar o desempenho do mecanismo de detecção de *drift* e do modelo NEVE em problemas de aprendizagem em ambientes não estacionários. Como exemplos de experimentos podem-se destacar:

- Comparação da acurácia do modelo NEVE com outros modelos de aprendizagem em ambientes não estacionários existentes na literatura;
- Verificação da influência da utilização do mecanismo de detecção de *drift* proposto no modelo NEVE quanto à acurácia e ao desempenho do modelo;
- Verificação da influência da utilização do mecanismo de detecção de *drift* proposto em uma rede neural MLP simples quanto à acurácia e ao desempenho do modelo.

1.3. Organização

Este trabalho está estruturado em mais seis capítulos, descritos a seguir:

- O Capítulo 2 apresenta uma revisão da literatura relacionada aos fundamentos de *concept drift*, incluindo definições e terminologia, tipos de *concept drift*, detecção de *drift* e resumo dos principais algoritmos da literatura. Também aborda os modelos evolutivos com inspiração quântica de interesse desta pesquisa: AIEQ-R e AIEQ-BR;
- O Capítulo 3 apresenta o mecanismo de detecção de *drift* proposto;
- O Capítulo 4 apresenta o modelo neuroevolutivo proposto NEVE;
- O Capítulo 5 apresenta os experimentos realizados com o mecanismo de detecção proposto;
- O Capítulo 6 apresenta os experimentos realizados com o modelo NEVE;
- O Capítulo 7 apresenta as conclusões deste trabalho e possibilidades de trabalhos futuros.

2
Revisão da Literatura

Este capítulo apresenta nas duas primeiras seções uma revisão bibliográfica de alguns tópicos importantes relacionados a *concept drift*. Em seguida, na Seção 2.3, são descritos, de forma sucinta, os modelos evolutivos com inspiração quântica de interesse desta pesquisa: AIEQ-R e AIEQ-BR.

2.1.
Concept Drift

Nesta seção será apresentada uma revisão bibliográfica dos tópicos relacionados a *concept drift* de interesse desta tese: definições, tipos de *concept drift* e detecção de *drift*.

2.1.1.
Definições de *Concept Drift*

O termo *concept drift* pode ser definido informalmente como uma mudança na definição dos conceitos (dados) ao longo do tempo e, consequentemente, mudança na sua distribuição. *Concept drift* refere-se a um cenário de aprendizagem onde a relação entre os dados de entrada e a variável alvo muda ao longo do tempo [Gama et al., 2014]. Um ambiente a partir do qual este tipo de dados é obtido é considerado um ambiente não estacionário.

Formalmente, de acordo com [Elwell & Polikar, 2011], considerando a probabilidade a posteriori de uma amostra x pertencer a uma classe y, *concept drift* é qualquer cenário em que esta probabilidade é alterada ao longo do tempo, isto é: $(P_{t+1}(y|x) \neq P_t(y|x))$.

Um exemplo prático de problema de classificação com *concept drift* mencionado em [Kuncheva, 2004] é a detecção e filtro de e-mails *spam*. As descrições das duas classes, "*spam*" e "não *spam*", podem variar ao longo do tempo. Elas são específicas por usuário, e as preferências dos usuários também variam ao longo do tempo. Além disso, as variáveis usadas no tempo *t* para classificar *spam* podem ser irrelevantes em *t+k*. Para dificultar ainda mais, a

variabilidade do ambiente neste cenário é fortemente relacionada ao acaso. Desta forma, o classificador deve lidar com os "*spammers*", que irão continuamente criar novas formas de tentar enganar o classificador para categorizar um *spam* como e-mail legítimo.

2.1.2. Tipos de Concept Drift

Trabalhos da literatura como [Tsymbal, 2004; Gama et al., 2014; Zliobaite, 2009] geralmente classificam os *concept drifts* que podem ocorrer no mundo real em quatro tipos, de acordo com os padrões de configuração das fontes de dados ao longo do tempo. Supondo a existência de duas fontes de dados, os quatro tipos de drift são descritos a seguir:

- **Abrupto:** ocorre a troca brusca de um conceito A por outro B. No tempo t a fonte S_1 é repentinamente substituída por S_2.
- **Gradual:** um conceito A vai sendo trocado pelo outro B aos poucos. Enquanto não ocorre a mudança definitiva do conceito A para o conceito B, observa-se cada vez mais ocorrências de B e menos ocorrências de A. Ambas as fontes S_1 e S_2 estão ativas, mas à medida que o tempo passa, a probabilidade de amostragem da fonte S_1 diminui enquanto a probabilidade de amostra da fonte S_2 aumenta. No início deste *drift*, antes que mais instâncias sejam observadas, uma instância da fonte S_2 pode ser facilmente confundida com ruído aleatório.
- **Incremental:** presença de vários conceitos intermediários A_1, A_2, A_3, ..., etc., entre dois pontos observados, indo gradativamente de um conceito A para outro B. Muitos autores consideram este tipo de *drift* como um subtipo do *drift* gradual. Este tipo de *drift* inclui mais fontes intermediárias entre S_1 e S_2, mas como a diferença entre elas é muito pequena, o *drift* é detectado apenas quando é observado por um longo período de tempo.
- **Contexto Recorrente:** Alguns trabalhos como [Zliobaite, 2009] consideram um tipo particular de *concept drift* o chamado contexto recorrente, que ocorre quando um conceito ativo anteriormente reaparece depois de algum tempo. Este conceito difere-se da noção de sazonalidade comum porque não é periódico e nem há clareza de quando esta fonte pode reaparecer. Neste caso, o

conceito A é trocado pelo B (a fonte S_1 é substituída por S_2) e depois de algum tempo, volta para o conceito A (a fonte volta a ser S_1).

É importante ressaltar que o ruído (ou *outlier*) não é considerado um tipo de *drift*, pois se refere a uma anomalia ou ocorrência isolada de um desvio aleatório. Neste caso, não há nenhuma necessidade de adaptação do modelo, que deve ser robusto ao ruído. A Figura 1 ilustra os tipos de *concept drift* apresentados anteriormente.

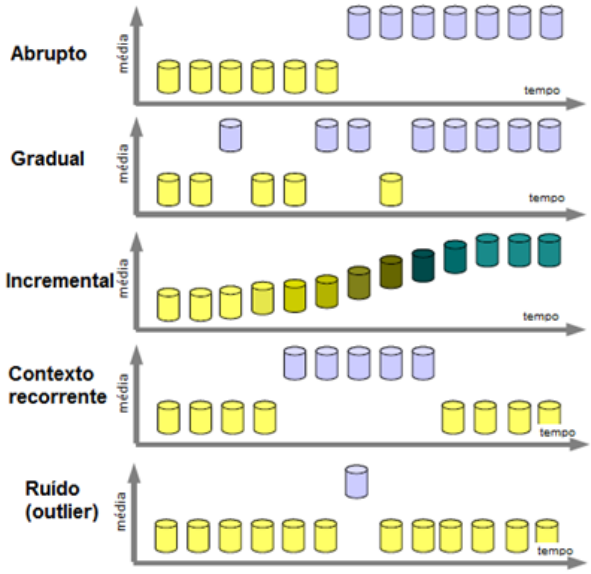


Figura 1. Representação de *drift* abrupto, gradual, incremental, contexto recorrente e ruído, adaptada de [Zliobaite, 2009].

2.1.3. Detecção de *Drift*

O termo “Detecção de Mudança” ou “Detecção de *Drift*” refere-se a técnicas e mecanismos para detecção de *drift*/mudança, através da identificação de pontos de mudança ou pequenos intervalos durante os quais as variações ocorrem, tal que o ambiente mudou suficientemente de forma que os modelos existentes não conseguem mais ser eficazes para predizer o comportamento dos dados correntes [Gama et al., 2014].

Diversos mecanismos de detecção de *drift* já foram propostos na literatura e podem ser utilizados para executar o processo de aprendizagem em conjunto com um classificador base. Tipicamente, o classificador gera a predição de uma classe para cada padrão recebido e, posteriormente, compara sua resposta com o rótulo de classe correto recebido para verificar se o classificador

acertou ou errou cada predição. A seguir, serão apresentados alguns destes métodos de detecção.

O ***Drift Detection Method*** **(DDM)** [Gama et al., 2004] tem como objetivo detectar as sequências de exemplos que seguem uma distribuição estacionária. Para tanto, o algoritmo define um conceito denominado por contexto, isto é, um conjunto de padrões sequenciais onde a distribuição é estacionária. Os padrões de treino são apresentados em sequência: quando um novo padrão se torna disponível, ele é classificado usando o modelo atual. O método controla o rastreamento do erro *online* do algoritmo e define para o contexto atual um nível de *warning* e um nível de *drift (out-of-control)*. A teoria estatística [Mitchell, 1997] garante que enquanto a distribuição seja estacionária, o erro irá diminuir à medida em que novos padrões forem apresentados. Assim, um novo contexto é declarado se em uma sequência de padrões, o erro aumenta alcançando o nível de *warning* no exemplo k_w e o nível de *drift* no exemplo k_d, indicando uma mudança na distribuição dos padrões. O algoritmo então aprende um novo modelo usando apenas os padrões a partir de k_w. Se após o aumento da taxa de erro em k_w for observada uma diminuição, será assumido que foi um alarme falso, sem mudanças no contexto. Na prática, o método escolhe o conjunto de treino mais apropriado para a atual distribuição de classes e é mais adequado para a detecção de *drifts* abruptos.

Outro exemplo similar é o ***Early Drift Detection Method*** **(EDDM)** [Baena-García et al., 2006], um método mais adequado para a detecção de *drifts* graduais. Este método funciona de forma parecida com o DDM com a diferença que o EDDM usa a distância entre erros de classificação (número de exemplos entre 2 erros de classificação) para detectar mudanças.

Já o método ***Exponentially Weighted Moving Average*** **(EWMA)** [Ross et al., 2012] foi originalmente proposto para detectar um aumento na média de uma sequência de variáveis aleatórias, considerando que a média e o desvio padrão dos dados são conhecidos. O método ***EWMA for Concept Drift Detection*** **(ECDD)** [Ross et al., 2012] é uma extensão do EWMA, e monitora a taxa de erro de classificação de um classificador, permitindo que a taxa de falsos positivos seja controlada e mantida constante ao longo do tempo.

O detector ***Paired Learners*** **(PL)** [Bach & Maloof, 2008], por sua vez, é um método baseado em janelas de tempo e usa dois modelos: um estável e outro reativo. O modelo estável prediz baseado em um histórico de dados longo, enquanto o modelo reativo prediz baseado em uma janela de dados recente pequena. A técnica usa o modelo reativo como um indicador de *concept drift* e

usa o modelo estável para fazer predições uma vez que o modelo estável tem melhor acurácia do que o modelo reativo. O método de detecção de *drift* usa as diferenças nas acurácias entre os dois modelos para determinar quando substituir o modelo corrente estável, uma vez que o modelo estável passa a ter pior acurácia do que o modelo reativo quando há ocorrência de *drift*.

De forma similar, o detector ***Statistical Test of Equal Proportions* (STEPD)** [Nishida & Yamauchi, 2007] também considera duas acurácias: a acurácia estimada em relação a todos os *streams*, e a acurácia estimada em relação a uma janela deslizante dos exemplos mais recentes. Um *concept drift* é sinalizado quando um decaimento significativo na acurácia recente é observado.

Estes métodos de detecção de *drift* apresentados, assim como a maioria dos métodos encontrados na literatura, funcionam de maneira reativa, ou seja, após a ocorrência do *drift* e erro do modelo, uma vez que eles dependem dos rótulos de classes dos padrões de entrada. No caso de problemas de classificação, após o recebimento do conjunto completo de dados (padrões e rótulos das classes para os conjuntos de treino e de teste), o detector aplica uma sequência de procedimentos para identificar alguma mudança na distribuição condicional de classes – um *concept drift*. Poucos trabalhos utilizam uma abordagem proativa e um exemplo encontrado é um trabalho recente [Kuncheva & Faithfull, 2014] que aplica análise de componentes principais (PCA) para extração das características antes da detecção de mudança. Os autores discutem e mostram evidências que os componentes com variância mais baixa devem ser guardados como as características extraídas, uma vez que é mais provável que eles sejam afetados por uma mudança. Os autores então escolhem um critério de detecção de mudança baseado na função de log-verossimilhança semiparamétrica que é sensível a mudanças na média e na variância das distribuições multidimensionais.

Uma das contribuições desta tese é a proposta de um novo mecanismo de detecção de *drift,* denominado DetectA (do inglês *Detect Abrupt*). Este método utiliza uma abordagem proativa de detecção e é apresentado no capítulo 3.

2.2. Modelos para Aprendizagem de *Concept Drift*

Os algoritmos para tratar problemas de *concept drift* podem ser categorizados de diversas formas. A Tabela 1, baseada em [Kuncheva, 2008;

Elwell & Polikar, 2011; Kuncheva, 2004], resume as classificações mais comumente utilizadas na literatura, com suas respectivas definições.

Tabela 1. Tipos de Algoritmos

Abordagem Ativa x Passiva	
Ativa	Possui algum mecanismo de detecção de *drift*, atualizando o modelo somente quanto o *drift* for detectado.
Passiva	Assume possível *drift* em andamento e continuamente atualiza o modelo para cada conjunto de dados. Se houve mudança, ela é aprendida, senão, o conhecimento existente é reforçado.
Entrada individual x Entrada em Blocos	
Individual	Aprende uma amostra por vez. Tem melhor plasticidade, mas pior estabilidade; tendem a ser mais sensíveis ao ruído.
Em Blocos	Requer blocos de amostras. Melhor estabilidade, mas podem ser ineficientes se o tamanho do bloco é muito pequeno ou se dados de múltiplos ambientes são apresentados no mesmo bloco. Podem usar algum tipo de *windowing* para controlar o tamanho do bloco.
Classificador específico x Livre	
Específico	Utilizam um classificador específico.
Livre	Podem utilizar qualquer classificador.
Classificador único x Comitê	
Classificador único	Utilizam um único classificador.
Comitê	Combinam múltiplos classificadores em um comitê.

Os algoritmos que utilizam a abordagem passiva (sem detecção de *drift*) atualizam regularmente o modelo assim que novos dados chegam. Entretanto, isto pode ser muito custoso se a quantidade de dados que chegam for excessivamente grande e, para algumas aplicações como categorização de spam, o *feedback* do usuário é necessário para rotular os dados, o que requer tempo e outros recursos. Uma forma de minimizar este problema é detectar as mudanças e adaptar o modelo apenas quando inevitável, usando a abordagem ativa [Tsymbal, 2004]. De forma geral, quando as abordagens ativas detectam um *drift*, alguma ação é tomada, por exemplo, configurando uma janela com os últimos dados e retreinando o classificador, ou adicionando um novo classificador no comitê. Nas abordagens passivas, usa-se uma heurística de esquecimento independente de suspeita ou não de mudança. Por exemplo: em um comitê de classificadores, os pesos dos membros do comitê são atualizados após cada entrada de dados (individual ou em blocos), com base na acurácia recente dos membros do comitê. Sem *concept drift*, a acurácia de classificação será estável e os pesos irão convergir. Caso alguma mudança ocorra, os pesos

irão mudar para refleti-la, sem necessidade de detecção explícita [Kuncheva, 2004].

O método ativo busca apontar quando ocorreu o *drift* e permite ao classificador modificar-se ou continuar aprendendo da mesma forma. Uma desvantagem deste método é o risco de ter um mecanismo imperfeito que pode produzir alarmes falsos, o que é muito comum particularmente em conjuntos de dados com ruído. Já no mecanismo passivo, o aprendiz acredita que o ambiente pode mudar a qualquer momento ou pode estar continuamente em mudança. O algoritmo então continua aprendendo a partir do ambiente construindo e organizando sua base de conhecimento. Se uma mudança aconteceu, ela é aprendida. Se ela não aconteceu, o conhecimento existente é reforçado [Elwell & Polikar, 2011]. Os modelos [Stanley, 2003; Kolter & Maloof, 2005; Kolter & Maloof, 2010] são exemplos de abordagens passivas e os modelos [Gama et al., 2004; Baena-García et al., 2006; Nishida & Yamauchi, 2007; Nishida & Yamauchi, 2007b; Nishida, 2008] são exemplos de abordagens ativas.

Em relação a entrada de dados, vale a pena ressaltar que padrões individuais podem ser convertidos em *batches* ou blocos de dados. O contrário também é possível, mas dados em blocos podem vir em grandes quantidades, tornando o processamento baseado em instância muito demorado [Kuncheva, 2004].

Comparando as abordagens classificador único x comitê, as abordagens baseadas em comitês de classificadores são mais recentes e tendem a ter melhor acurácia, flexibilidade e eficiência do que os que utilizam classificador único [Kuncheva, 2004]. É importante lembrar que em fluxos de dados em massa prefere-se utilizar geralmente modelos simples porque pode não haver tempo para executar e atualizar um comitê. Por outro lado, alguns autores defendem que um comitê simples pode ser mais fácil de usar do que certos classificadores simples adaptativos, como árvores de decisão. Quando tempo não é a preocupação principal, mas é requerida alta acurácia, um comitê seria a solução natural. Por exemplo, no escaneamento de mamografia para detecção de tumores, é aceitável levar alguns minutos por imagem [Kuncheva, 2008].

Há diversos métodos que podem ser usados em comitês para a adaptação ao *concept drift* e diversas soluções foram propostas na literatura. Conforme já mencionado no capítulo 1, as técnicas mais usadas são: atualização da regra de combinação para classificadores fixos; usando classificadores adaptativos como membros do comitê; e atualizando a estrutura do comitê. Este trabalho tem maior foco nos algoritmos da última categoria.

Nestes algoritmos, em caso de mudança no ambiente, os classificadores individuais são reavaliados e segundo algum critério específico, um deles pode ser substituído por um novo classificador treinado com os dados mais recentes. A maioria das metodologias possui algum mecanismo para controle do tamanho do comitê, ou o tamanho máximo é deixado como parâmetro.

A área de aprendizagem em ambientes não estacionários tem recebido nos últimos anos atenção crescente, em parte devido às diversas aplicações práticas, como detecção de *spam*, fraude ou previsão climática, nas quais a distribuição dos dados (ou seus parâmetros) muda ao longo do tempo [Elwell & Polikar, 2011]. A primeira formalização do problema de aprendizagem incremental a partir de dados com ruído foi feita por [Schlimmer & Granger, 1986], que também apresentaram o algoritmo adaptativo STAGGER e introduziram o termo *concept drift*. A partir daí diversos estudos sobre o problema de *concept drift* apareceram, destacando os picos de interesse do tema em 1998, 2004 e de 2007 até atualmente [Zliobaite, 2009].

Após o STAGGER, surgiu um algoritmo para tratamento de *concept drift*, denominado FLORA [Widmer & Kubat, 1996]. Ambos são algoritmos passivos que utilizam um classificador simples e uma janela deslizante para selecionar um bloco de novos padrões para treinar um novo classificador. O tamanho da janela é modificado via heurística de ajuste da janela, baseada em quão rápido o ambiente está se modificando. O algoritmo FLORA tem um mecanismo de esquecimento com a premissa implícita que os padrões que forem retirados da janela não são mais relevantes e a informação trazida por eles pode ser esquecida.

Conforme já mencionado anteriormente, responder a diversos tipos de *concept drift* é uma tarefa difícil para um classificador simples. Por este motivo, diversos sistemas baseados em comitês de classificadores foram propostos recentemente para lidar com aprendizagem em *concept drift*, tais como [Elwell & Polikar, 2011; Fan, 2004; Fan, 2004b; Kolter & Maloof, 2005; Kolter & Maloof, 2005b; Minku & Yao, 2012; Scholz & Klinkenberg, 2005; Scholz & Klinkenberg, 2007; Stanley, 2003; Street & Kim, 2001; Yang et al., 2003]. A seguir, serão apresentados alguns destes exemplos. Primeiramente, será apresentada uma breve descrição textual do modelo, acompanhada por uma figura ilustrando seu funcionamento e em seguida, será apresentado o seu pseudocódigo.

2.2.1.
SEA: Streaming Ensemble Algorithm

O *Streaming Ensemble Algorithm* (SEA) [Street & Kim, 2001] é um algoritmo que recebe dados em blocos, baseado em um comitê de classificadores, que utiliza votação majoritária simples e ajuste de classificadores (para descartar o conhecimento antigo) para garantir que o comitê trate novos ambientes. Os pesos do comitê são obtidos com base na performance dos classificadores e os pesos também são usados como critério de ajuste. O classificador mais fraco é descartado quando o tamanho do comitê excede um limite, não com base na sua acurácia, mas sim com base em um critério de qualidade também proposto pelos autores.

Neste algoritmo, os dados são lidos sequencialmente em blocos. O algoritmo cria inicialmente um comitê de classificadores vazio, de tamanho fixo. Enquanto houver dados de entrada disponíveis, o algoritmo lê um bloco de dados D_i e cria um classificador específico C_i para este bloco. O classificador C_i é então avaliado usando D_i e, em seguida, todos os classificadores do comitê são avaliados usando D_i. Se o comitê não estiver cheio, C_i é inserido nele. Caso contrário, verifica-se se C_i é melhor do que outro classificador do comitê e em caso afirmativo, este classificador é substituído por C_i no comitê. Ou seja, o novo classificador C_i só é adicionado no comitê se realmente puder contribuir com a sua decisão final. A classificação final do comitê é determinada usando votação majoritária ponderada. A Figura 2 ilustra o funcionamento do algoritmo SEA.

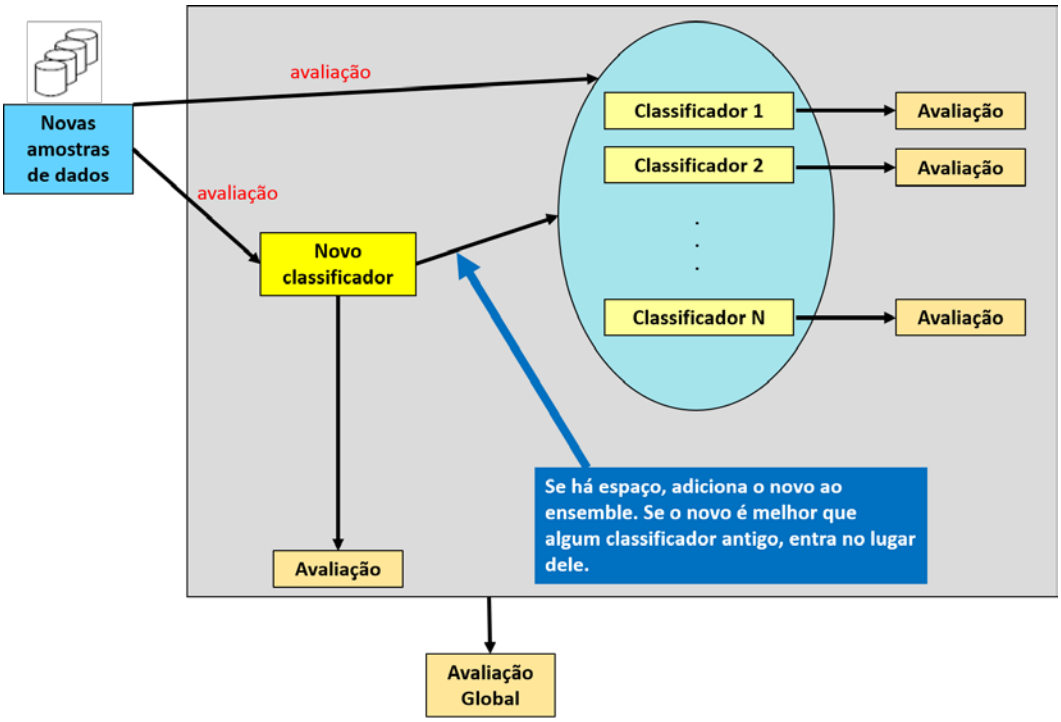


Figura 2. Visão geral do funcionamento do algoritmo SEA.

O pseudocódigo simplificado do SEA é apresentado a seguir:

```
Enquanto dados de entrada estiverem disponíveis
   Ler di padrões, criando o conjunto de treino Di
   Construir classificador Ci usando Di
   Avaliar o classificador Ci com Di
   Avaliar todos os classificadores no comitê E em Di
   Se E não está cheio
         Inserir Ci em E
   Senão se qualidade(Ci) > qualidade (Ej) para algum j
         Substituir Ej por Ci
   Fim-senão
Fim-enquanto
```

Figura 3. Pseudocódigo simplificado do algoritmo SEA.

O algoritmo SEA tem boa performance em ambientes com mudanças graduais, mas não responde bem a mudanças bruscas. Se um *drift* abrupto ocorre, as saídas incorretas dos classificadores dos conceitos antigos interferem nas saídas dos classificadores dos novos conceitos.

2.2.2.
DWM: Dynamic Weighted Majority

O algoritmo *Dynamic Weighted Majority* (DWM) [Kolter & Maloof, 2007] é um comitê de classificadores incremental que utiliza votação majoritária ponderada e esquemas de ajuste do comitê diferentes: o comitê é atualizado periodicamente somente quando necessário adicionando um classificador ou ajustando um classificador quando seus pesos caem abaixo de um limite de performance. O ajuste é baseado no erro e não há limite de tamanho do comitê.

O algoritmo adiciona e remove classificadores do comitê com base na performance global do algoritmo: Se o algoritmo global erra, é adicionado um novo classificador com peso 1. Se um classificador erra, seu peso é reduzido. Se um classificador tem um histórico de muitos erros, sinalizado por baixo peso, ele é removido do comitê.

Na inicialização do algoritmo, é criado um comitê com um único classificador de peso 1. A seguir, o comitê recebe um (ou mais) dos padrões do fluxo de dados e apresenta a cada um dos classificadores. Se um classificador erra, seu peso é decrementado usando uma constante multiplicativa β. A classificação global do comitê é calculada com base nas classificações individuais: independente do resultado individual, a resposta de cada classificador e seu peso são considerados para computar a soma ponderada para cada possível classe. A classe com o maior peso é a resposta do comitê.

A seguir, os pesos dos classificadores são normalizados e aqueles com desempenho ruim historicamente e com peso menor que um limiar θ são eliminados do comitê. Como os pesos são constantemente decrementados, a normalização garante que os pesos estejam numa escala uniforme de no máximo 1, evitando que os classificadores mais novos dominem as classificações. Se o comitê global erra, é criado um novo classificador de peso 1 que é adicionado ao comitê. Todos os classificadores são treinados com o novo padrão e a classificação global é recalculada.

Ou seja, o algoritmo mantém um conjunto de especialistas E, de tamanho m, cada um com um peso w_i, i = {1,..., m} e recebe a cada iteração n exemplos de treino (vetor de características e classe), existindo c possíveis classes. Há um parâmetro p que permite o tratamento de ruído, que define o período no qual o DWM não irá atualizar os pesos dos classificadores nem irá remover ou criar classificadores. A Figura 4 ilustra o funcionamento do algoritmo DWM.

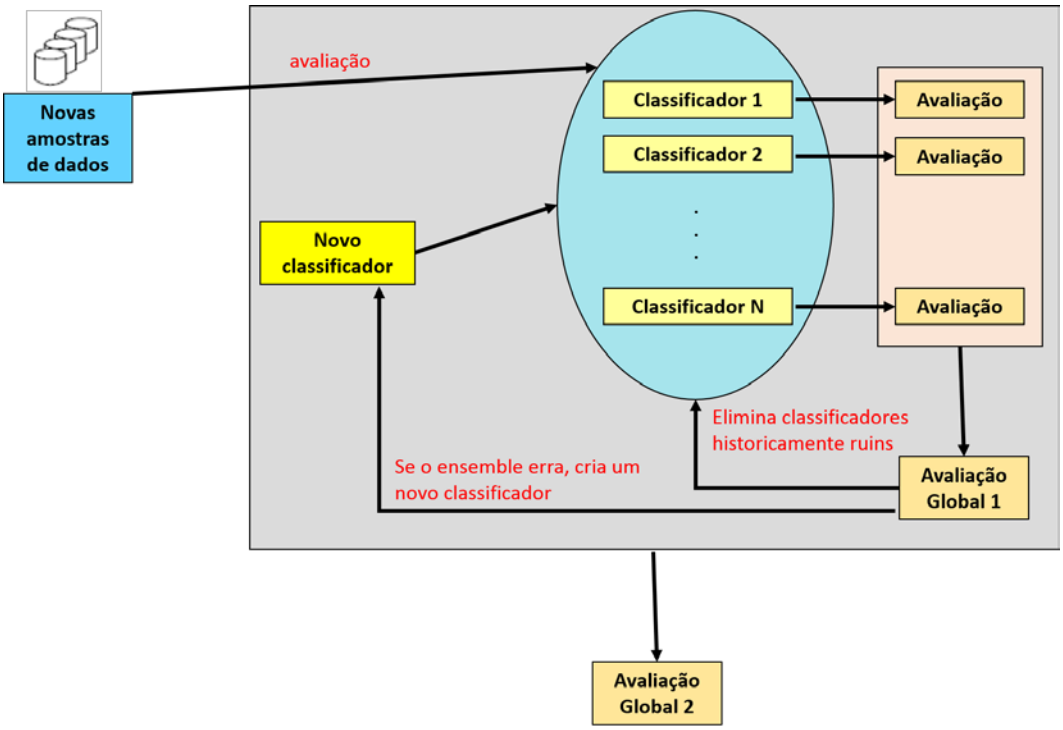


Figura 4. Visão geral do funcionamento do algoritmo DWM.

O pseudocódigo simplificado do DWM é apresentado a seguir:

```
Enquanto dados de entrada estiverem disponíveis
 Ler padrão de entrada Di
 Para cada classificador Ck de E
       Avaliar Ck usando Di
       Se avaliação(Ck) = erro
             Decrementar peso pk de Ck usando B
       Fim-se
 Fim-para
 Decisão = votação majoritária ponderada usando Pk
 Normalizar Pk
 Para cada classificador Ck de E
       Se Ck tem desempenho historicamente ruim
             Eliminar Ck de E
       Fim-se
 Fim-para
 Se Decisão = erro
       Construir classificador Ci usando Di
       Inserir Ci no comitê E
 Fim-se
 Para cada classificador Ck de E
       Treinar Ck com Di
 Fim-para
 Nova_Decisão = votação majoritária ponderada usando Pk
Fim-enquanto
```

Figura 5. Pseudocódigo simplificado do algoritmo DWM.

2.2.3. Learn++.NSE

O algoritmo Learn++.NSE [Elwell & Polikar, 2011] é um algoritmo incremental que recebe blocos de dados de entrada e é baseado em um comitê de classificadores que usa votação majoritária ponderada. Os pesos são dinamicamente atualizados em função dos ajustes dos erros dos classificadores no ambiente atual e nos anteriores. O algoritmo usa apenas os dados atuais para treinamento, ou seja, assume que todos os dados já vistos (relevantes ou não para o aprendizado) não estão acessíveis ou não é possível armazená-los. Assim, toda a informação relevante sobre os dados anteriores deve estar armazenada nos parâmetros dos classificadores previamente gerados.

O Learn++.NSE utiliza mecanismo passivo de detecção de *drift*, assumindo que, a cada passo, pode ou não ter ocorrido mudança e, caso tenha ocorrido, a taxa não é conhecida e assume-se que não é constante. Dependendo da natureza da mudança, o algoritmo retém, constrói ou temporariamente descarta conhecimento, para que os novos dados possam ser categorizados. Este algoritmo pode ser utilizado em diversos ambientes não estacionários, incluindo os que apresentam *drift* rápido ou lento, gradual ou abrupto, cíclico ou de taxa variável. É um dos poucos algoritmos encontrados na literatura que suporta adição de conceitos (novas classes) ou eliminação de uma classe existente.

A base de conhecimento é representada pelo comitê de classificadores, que é inicializado através da criação de um único classificador no primeiro bloco de dados disponível. Uma vez que o conhecimento prévio esteja disponível, o comitê de classificadores atual é avaliado pelos dados correntes: o algoritmo identifica quais padrões novos não foram reconhecidos pela base de conhecimento existente e esta é atualizada adicionando um novo classificador treinado nos dados correntes. A seguir, cada classificador do comitê (incluindo o recém-criado) é avaliado com estes dados de treinamento. Identificando padrões desconhecidos, a penalidade de erro na sua classificação é considerada no cálculo do erro, ou seja, mais crédito é dado aos classificadores capazes de identificar previamente amostras desconhecidas e os classificadores que erram na classificação de amostras conhecidas são penalizados. O erro do classificador é ponderado considerando o tempo: a competência recente é levada mais em conta na categorização do conhecimento. Os pesos de votação são determinados: se o conhecimento de um classificador não é compatível com

o ambiente atual, ele recebe pouco ou nenhum peso e é temporariamente removido da base de conhecimento. Ele não é descartado: se passar a ser relevante novamente receberá pesos de votação mais altos. A decisão final é obtida com a votação majoritária ponderada dos membros atuais do comitê de classificadores.

Vale a pena ressaltar que o algoritmo Learn++.NSE adiciona continuamente novos classificadores. Para tratar esta proliferação, pode-se estabelecer um número limite, removendo os classificadores excedentes com base na sua idade ou erro. Esta estratégia, entretanto, não é recomendada pelos autores: segundo os mesmos, os benefícios de guardar o classificador superam os baixos custos computacionais e de armazenamento. A Figura 6 ilustra o funcionamento do algoritmo Learn++.NSE.

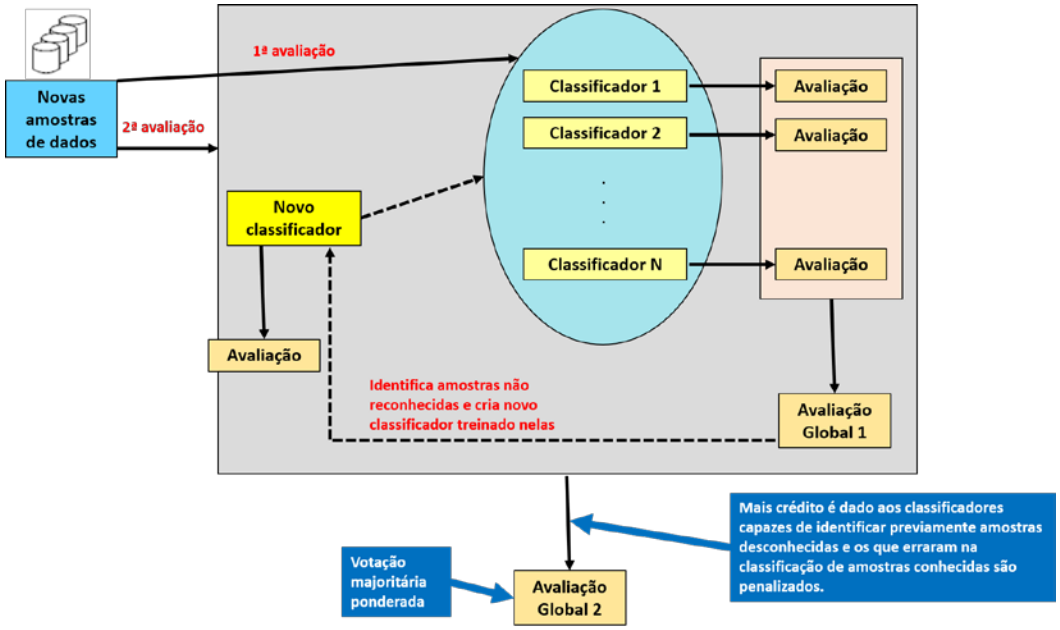


Figura 6. Visão geral do funcionamento do algoritmo Learn++.NSE.

O pseudocódigo simplificado do Learn++.NSE é apresentado a seguir:

```
Enquanto dados de entrada estiverem disponíveis
   Ler bloco de dados Di
   Construir classificador Ci usando Di
   Inserir Ci no comitê E
   Para cada classificador Ck de E
         Avaliar Ck usando Di
   Fim-para
   Para cada classificador Ck de E
         Determinar peso de votação pk para Di
   Fim-para
   Decisão = votação majoritária ponderada usando Pk
Fim-enquanto
```

Figura 7. Pseudocódigo simplificado do algoritmo Learn++.NSE.

2.2.4. DDD: Diversity for Dealing with Drifts

O algoritmo *Diversity for Dealing with Drifts* (DDD) [Minku & Yao, 2012] é uma abordagem *online* baseada em comitê de classificadores que recebe entradas individuais, ou seja, um padrão por vez. O algoritmo opera em 2 modos: antes da detecção de *drift* e depois da detecção de *drift*. O método de detecção usado detecta mudanças o mais cedo possível e foi projetado para ser robusto a alarmes falsos.

Antes do *drift* ser detectado, o modelo é composto de dois comitês: um com baixa diversidade (hnl) e outro com alta diversidade (hnh). Ambos os comitês são treinados com os padrões de entrada, mas apenas o comitê de baixa diversidade é usado para predições do sistema (pois é mais provável que o comitê de alta diversidade tenha menos acurácia no novo conceito). O DDD assume que, se não há convergência nas distribuições para um conceito estável, novas detecções de *drift* irão ocorrer, ativando este modo após uma detecção de *drift*. O DDD permite então o uso do comitê de alta diversidade na forma de um comitê antigo de alta diversidade.

O método de detecção de *drift* monitora o comitê de baixa diversidade utilizando qualquer método de detecção existente na literatura. Após a detecção de um *drift*, são criados novos comitês de baixa e alta diversidade e os comitês de baixa e alta diversidade anteriores à detecção de *drift* são mantidos e passam a ser chamados comitê antigo de baixa diversidade e comitê antigo de alta diversidade. O comitê antigo de alta diversidade começa a aprender com o de baixa diversidade de forma a aprimorar sua convergência para o novo conceito. Manter os comitês antigos permite uma melhor exploração de diversidade, o uso

da informação aprendida a partir do conceito antigo para ajudar no aprendizado do novo conceito e também ajuda na robustez a falsos alarmes.

Tanto os comitês antigos quanto os novos realizam o aprendizado e as predições do sistema são determinadas através da votação majoritária ponderada das saídas dos comitês: antigo de alta diversidade, novo de baixa diversidade e antigo de baixa diversidade. O comitê novo de alta diversidade não é usado porque é provável que tenha baixa acurácia no novo conceito. Os pesos são proporcionais a acurácia prequencial desde a última detecção de *drift* até o passo de tempo anterior. O peso do comitê antigo de baixa diversidade é multiplicado por uma constante W que mede o balanceamento entre robustez a alarmes falsos e acurácia na presença de *drifts*, e todos os pesos são normalizados.

Durante o modo após a detecção de *drift*, o comitê novo de baixa diversidade é monitorado pelo mecanismo de detecção. Se duas detecções consecutivas ocorrem e se entre elas não há retorno ao modo anterior à detecção de *drift*, o comitê antigo de baixa diversidade após a segunda detecção pode ser tanto igual ao comitê antigo de alta diversidade aprendendo com baixa diversidade após a primeira detecção de *drift* ou ao comitê novo de baixa diversidade após a primeira detecção de *drift*, dependendo de qual dos dois tem maior acurácia. Isto porque assim que ocorre a primeira detecção, o comitê novo de baixa diversidade pode não ter acurácia suficiente para tornar-se o comitê antigo de baixa diversidade. Esta estratégia também ajuda o algoritmo a ser mais robusto a alarmes falsos.

Todos os quatro comitês são mantidos no sistema até que algumas condições sejam satisfeitas, então, o sistema retorna ao modo anterior à detecção de *drift*. Ao retornar a este modo, tanto o comitê antigo de alta diversidade quanto o comitê novo de baixa diversidade podem se tornar o comitê de baixa diversidade usado no modo anterior à detecção, dependendo de qual tiver maior acurácia.

A Figura 8 ilustra o funcionamento do algoritmo DDD.

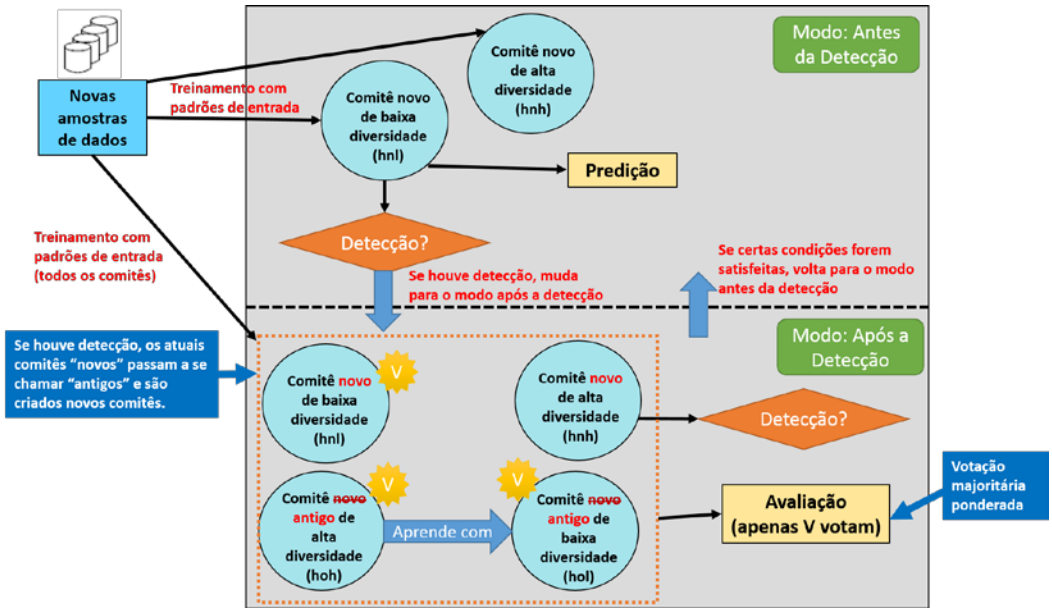


Figura 8. Visão geral do funcionamento do algoritmo DDD.

O pseudocódigo simplificado do DDD é apresentado a seguir:

```
Parâmetros:
   W: constante multiplicativa para o peso do comitê antigo de
baixa diversidade
   EnsembleLearning: algoritmo de aprendizagem online do comitê
   pl e ph: parâmetros para aprendizagem do comitê,
respectivamente, baixa diversidade e alta diversidade
   DeteccaoDrift: método de detecção de drift
   pd: parâmetros para o método de detecção de drift
   D: stream de dados

   Inicialização:
   modo = antes de drift
   hnl = comitê novo de baixa diversidade
   hnh = comitê novo de alta diversidade
   hol = hoh = null (comitês antigos de baixa e alta diversidade)
   accol = accoh = accnl = accnh = 0 (acurácias)
   stdol = stdoh = stdnl = stdnh = 0 (desvio padrão)
```

```
   Enquanto dados de entrada estiverem disponíveis
      Ler padrão de entrada Di
      Se modo == antes de drift
            decisão = predição(hnl, d)
      Senão
            sumacc = accnl + accol * W + accoh
            wnl = accnl/sumacc
            wol = accol * W/sumacc
            woh = accoh/sumacc
            decisão = votação majoritária ponderada usando (hnl,
hol, hoh, wnl, wol, woh)
            Atualizar(accnl, stdnl, hnl, d)
            Atualizar(accol, stdol, hol, d)
            Atualizar(accoh, stdoh, hoh, d)
      Fim-se
      drift = DeteccaoDrift(hnl, d, pd)
      Se drift
            Se modo == antes de drift || (modo == depois de drift
&& accnl > accoh)
                  hol = hnl
            Senão
                  hol = hoh
            Fim-se
            hoh = hnh
            hnl = novo comitê
            hnh = novo comitê
            accol = accoh = accnl = accnh = 0
            stdol = stdoh = stdnl = stdnh = 0
            modo = depois de drift
      Fim-se
      Se modo == depois de drift
            Se accnl > accoh && accnl > accol
                  modo = antes de drift
            Senão
                  Se accoh − stdoh > accnl + stdnl && accoh -
stdoh > accol + stdol
                        hnl ← hoh
                        accnl ← accoh
                        modo = antes de drift
                  Fim-se
            Fim-se
      Fim-se
      EnsembleLearning(hnl, d, pl)
      EnsembleLearning(hnh, d, ph)
      Se modo == depois de drift
            EnsembleLearning(hol, d, pl)
            EnsembleLearning(hoh, d, pl)
      Fim-se
      Se modo == antes de drift
            Saida = predição de hnl
      Senão
            Saida = predição de hnl, hol, hoh,wnl,wol,woh
      Fim-se
   Fim-enquanto
```

Figura 9. Pseudocódigo simplificado do algoritmo DDD.

Esta seção apresentou uma revisão bibliográfica de tópicos relacionados a *concept drift*. A próxima seção apresentará os modelos evolutivos com inspiração quântica de interesse desta tese.

2.2.5. RCD: Recurring Concept Drifts

O algoritmo *Recurring Concept Drifts* (RCD) [Gonçalves Júnior, 2013], baseado no algoritmo FLORA, é uma abordagem alternativa para lidar com *drifts* recorrentes e trabalha com a ideia de contexto. O RCD cria um novo classificador para cada contexto encontrado e armazena uma amostra dos dados usados para construí-lo. Quando um novo *drift* ocorre, o algoritmo compara o novo contexto com os antigos, utilizando um teste estatístico não paramétrico multivariado para verificar se ambos os contextos provêm da mesma distribuição. Em caso afirmativo, o classificador correspondente é reutilizado. Caso contrário, um novo classificador é gerado e armazenado.

O teste estatístico utilizado para detecção de drift é o kNN (*k nearest neighbors*) [Hastie et al., 2005], que classifica objetos com base em exemplos de treinamento que estão mais próximos no espaço de características. Para comparar duas amostras de dados, o kNN guarda padrões de ambas as amostras, adicionando um atributo para indicar a amostra de origem do padrão. Em seguida, os k padrões mais próximos de cada padrão são calculados. Para cada padrão, verifica-se quantos destes k padrões vêm da mesma amostra. Se ambas as amostras vêm da mesma distribuição, os k padrões mais próximos podem, em média, ser igualmente divididos entre as duas amostras.

O algoritmo RCD funciona da seguinte forma: um classificador é construído junto com um detector de *drift* (por exemplo, o DDM, apresentado na subseção anterior). O novo classificador (c_a) e um *buffer* vazio (b_a) são criados e armazenados nas suas respectivas listas. O detector de drift usa c_a para identificar se um *drift* está começando a ocorrer. Enquanto nenhum *drift* é detectado, b_a é preenchido com exemplos usados no treinamento de c_a.

Quando a taxa de erro de c_a aumenta, o nível de *warning* do detector de *drift* é atingido, indicando que um *drift* pode estar começando a ocorrer. Um novo classificador (c_n) é então criado juntamente com um novo *buffer* (b_n). Padrões são usados para treinar c_n e são armazenados em b_n, mas b_a é mantido. Se a taxa de erro de c_a diminuir, o detector irá retornar para o nível normal e considerará o *drift* um alarme falso. Nesta situação, c_n e b_n são descartados e b_a

é novamente usado para armazenar padrões. Entretanto, se a taxa de erro de c_a continuar aumentando, o nível *drift* será atingido, indicando que um *drift* foi detectado. O teste estatístico é realizado para comparar b_n com todos os *buffers* armazenados, tentando identificar se este contexto já ocorreu no passado. Se o teste é positivo, significa que é um contexto antigo recorrente: o classificador e o buffer armazenados são considerados os novos c_a e b_a, respectivamente, e os valores antigos de c_n e b_n são descartados. Se o teste é negativo, significa que é um novo contexto: c_n e b_n são armazenados na lista e considerados os novos c_a e b_a. A Figura 10 ilustra o funcionamento do algoritmo RCD.

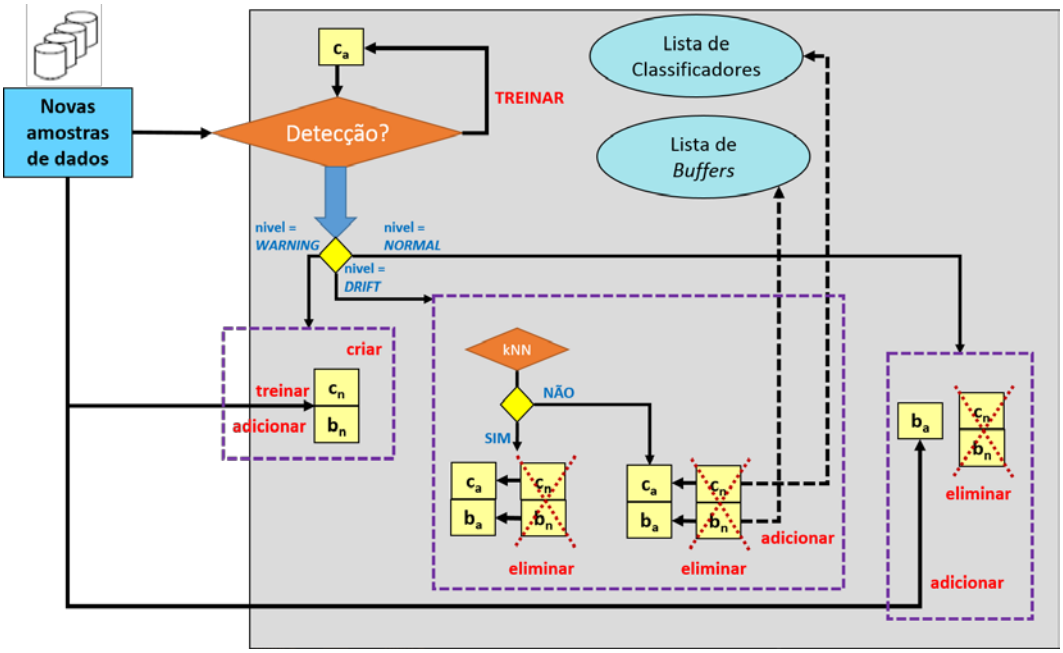


Figura 10. Visão geral do funcionamento do algoritmo RCD.

O pseudocódigo simplificado do RCD é apresentado a seguir:

```
Parâmetros:
ca: classificador atual
ba: buffer atual
cn: novo classificador
bn: novo buffer
S: fluxo de dados
C: lista de classificadores
B: lista de buffers
a: valor de significância

C += criar(ca)
B += criar(ba)
Para cada s de S
   nivel = DDM(ca, s)
   switch(nivel)
```

```
    caso warning
          criar(cn)
          criar(bn)
          bn += s
          treinar(cn, s)
    caso drift
          se kNN(C, B, bn, a)
                ca = cn
                ba = bn
                cn = bn = null
          fim-se
          senão
                C += cn
                B += bn
                ca = cn
                ba = bn
                cn = bn = null
          fim-senão
    caso contário
          cn = bn = null
          ba += s
    fim-switch
    treinar(ca, s)
fim-para
```

Figura 11. Pseudocódigo simplificado do algoritmo RCD.

2.3. Modelos Evolutivos com Inspiração Quântica

Os algoritmos evolutivos clássicos têm sido usados com sucesso para resolver problemas complexos de otimização nas mais diversas áreas, tais como projeto automático de circuitos e equipamentos, planejamento de tarefas, engenharia de *software* e mineração de dados, entre muitas outras [Escovedo et al., 2013].

O fato desta classe de algoritmos não necessitar de formulações matemáticas rigorosas a respeito do problema que se deseja otimizar, além de oferecer um alto grau de paralelismo no processo de busca, são algumas das vantagens da utilização de algoritmos evolutivos. No entanto, alguns problemas são computacionalmente custosos no que diz respeito à avaliação das soluções durante o processo de busca, tornando a otimização por algoritmos evolutivos um processo lento para situações onde se deseja uma resposta rápida do algoritmo (como por exemplo, em problemas de otimização *online*).

A fim de tratar estas questões, surgiram os algoritmos evolutivos com inspiração quântica, que são uma classe de algoritmos de estimação de distribuição que apresentam um melhor desempenho para otimização combinatória e numérica em comparação com seus algoritmos genéticos

canônicos homólogos [Abs da Cruz, 2007; Abs da Cruz et al., 2010]. Estes algoritmos são inspirados em ideias da física quântica, em particular no conceito de superposição de estados, e foram desenvolvidos para problemas de otimização combinatória usando representação binária, como o Algoritmo Evolutivo com Inspiração Quântica (AEIQ-B, ou em inglês, *Quantum-Inspired Evolutionary Algorithm*) [Han & Kim, 2000; Han & Kim, 2002; Han & Kim, 2003; Han & Kim, 2004]. O AEIQ-B utiliza um cromossomo formado por *q-bits* e cada *q-bit* consiste em um par de números (α, β), onde $|\alpha^2| + |\beta^2| = 1$. O valor dado por $|\alpha^2|$ indica a probabilidade de que o *q-bit* terá o valor 0 quando for observado e o valor $|\beta^2|$ indica a probabilidade de que o *q-bit* terá o valor 1 quando for observado. Assim, no AEIQ-B um indivíduo quântico é formado por M *q-bits*, conforme (1):

$$\begin{vmatrix} \alpha_{i1} \\ \beta_{i1} \end{vmatrix} \begin{vmatrix} \alpha_{i2} \\ \beta_{i2} \end{vmatrix} \ldots \begin{vmatrix} \alpha_{iM} \\ \beta_{iM} \end{vmatrix} \quad (1)$$

onde *i = 1, 2, 3, ..., M*.

Posteriormente, os algoritmos evolutivos com inspiração quântica foram estendidos para representação real, para tratar problemas de otimização numérica [Abs da Cruz, 2007]. Nestes problemas, a representação direta é mais adequada, na qual números reais são diretamente codificados em um cromossomo, em vez de converter *strings* binárias em números. Com a representação numérica real, reduz-se a demanda por memória enquanto aumenta-se a precisão [Abs da Cruz et al., 2006]. Assim, foi desenvolvido o Algoritmo Evolutivo com Inspiração Quântica usando Representação Real (AEIQ-R) [Abs da Cruz et al., 2006]; Abs da Cruz, 2007; Abs da Cruz et al., 2010], inspirado no conceito de múltiplos universos da física quântica. Neste cenário, o algoritmo permite realizar o processo de otimização com um menor número de avaliações de soluções, reduzindo substancialmente o custo computacional. As próximas seções apresentarão os modelos AEIQ-R e AEIQ-BR, utilizados neste trabalho por serem mais adequados à neuroevolução.

2.3.1. Algoritmo Evolutivo com Inspiração Quântica e Representação Real (AIEQ-R)

Originalmente proposto em [Abs da Cruz, 2007], o algoritmo foi utilizado inicialmente para a solução de problemas *benchmark* de otimização numérica e para treinamento de redes neurais recorrentes em problemas de aprendizado supervisionado de séries temporais e em aprendizado por reforço em tarefas de

controle. Os resultados obtidos demonstraram a eficiência desse algoritmo na solução destes tipos de problemas.

Da mesma forma que o algoritmo AEIQ-B proposto em [Han & Kim, 2000; Han & Kim, 2002; Han & Kim, 2003; Han & Kim, 2004], o AEIQ-R utiliza uma população de indivíduos quânticos para representar a superposição de estados, os quais são observados para gerar os indivíduos clássicos. Entretanto, ao contrário do AEIQ-B, o AIEQ-R usa a abordagem de funções de onda como inspiração para o algoritmo, a fim de representar a simulação de uma superposição de estados contínuos. A Figura 12 mostra o pseudocódigo do algoritmo AEIQ-R:

```
iniciar
1.   t ← 1
2.   Gerar população quântica Q(t) com N indivíduos com G genes
3.   enquanto (t <= T)
4.       E(t) ← gerar indivíduos clássicos observando indivíduos quânticos
5.       se (t=1) então
6.           C(t) ← E(t)
7.       senão
8.           E(t) ← recombinação entre E(t) e C(t)
9.         avaliar E(t)
10.          C(t) ← K melhores indivíduos de [E(t) ∪ C(t)]
11.      fim se
12.      Q(t + 1) ← Atualiza Q(t) usando os N melhores indivíduos de C(t)
13.      t ← t + 1
14.  fim enquanto
fim
```

Figura 12. Pseudocódigo do algoritmo AEIQ-R [Abs da Cruz, 2007].

O diagrama representado na Figura 13 resume o algoritmo AEIQ-R e mostra a relação entre as partes que formam o modelo completo.

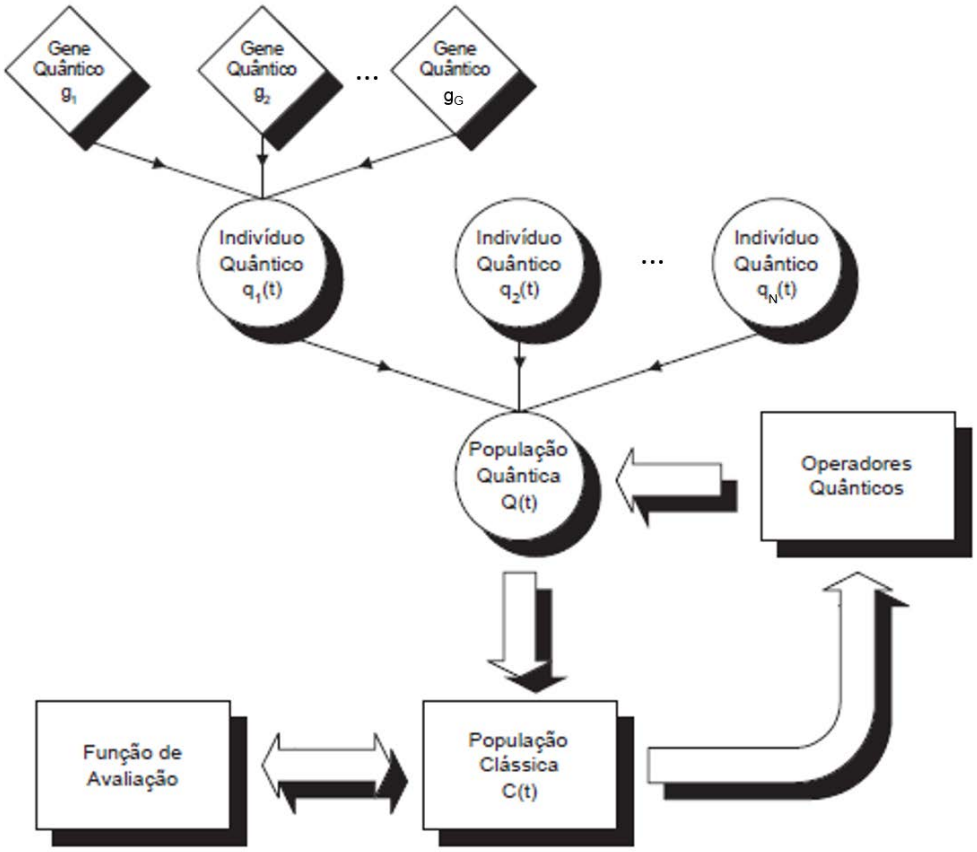


Figura 13. Diagrama completo do Sistema Evolutivo com Inspiração Quântica [Abs da Cruz, 2007].

Este diagrama mostra que a população quântica *Q(t)* é composta de N indivíduos quânticos q_i *(i = 1, 2, 3, ..,N)* que, por sua vez, são compostos de G genes quânticos. Esta população quântica é usada para gerar uma população clássica que é então avaliada e usada para modificar os indivíduos quânticos em um processo iterativo.

Cada gene quântico é composto por uma função densidade de probabilidade (FDP), a qual representa a superposição de estados e é utilizada para observar o gene clássico. Os indivíduos quânticos podem ser representados por:

$$q_i = [g_{i1} = p_{i1}(x), g_{i2} = p_{i2}(x), \dots, g_{iG} = p_{iG}(x)] \quad (2)$$

Onde *i = 1, 2, 3, ..., N*, *j = 1, 2, 3, ...,G* e as funções p_{ij} representam as funções densidade de probabilidade, usadas pelo AEIQ-R para gerar os valores para os genes dos indivíduos clássicos. Em outras palavras, a função $p_{ij}(x)$ representa a densidade de probabilidade de se observar um determinado valor para o gene quântico quando a superposição do mesmo for colapsada. Uma das funções mais simples que se pode usar como função densidade de

probabilidade é o pulso quadrado, uma função uniforme de geometria simples, que pode ser definida pela equação 3:

$$p_{ij}(x) = \begin{cases} \frac{1}{U_{ij}-L_{ij}} & , se\ L_{ij} \leq x \leq U_{ij} \\ 0\,, & caso\ contrário \end{cases} \quad (3)$$

onde L_{ij} é o limite inferior e U_{ij} o limite superior do intervalo no qual o gene j do i–ésimo indivíduo quântico pode colapsar, ou seja, assumir valores quando observado. A Figura 14 ilustra um exemplo de gene quântico formado por um pulso quadrado. Neste exemplo, os limites L_{ij} e U_{ij} da função $p_{ij}(x)$ estão definidos como −1 e 1 respectivamente.

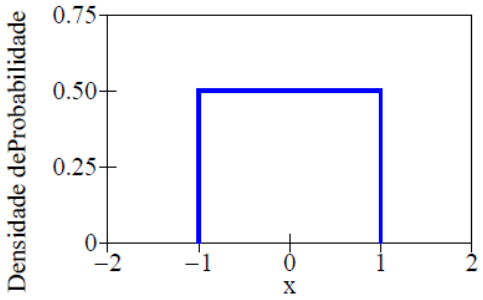


Figura 14. Exemplo de um gene quântico.

Para o caso em que $p_{ij}(x)$ é um pulso quadrado, pode-se representar o gene quântico armazenando-se os valores dos limites inferior e superior para cada gene, ou armazenando a posição do ponto central do pulso quadrado e a largura do mesmo. Por exemplo, considerando um indivíduo quântico q_i formado por dois pulsos quadrados $g_{i1}(x)$ e $g_{i2}(x)$, e supondo que estes dois pulsos quadrados têm uma largura igual a 2 e estão posicionados de tal modo que o seu centro está localizado nas posições −0.5 e 0.5 respectivamente, o cromossomo quântico pode ser representado, caso se esteja usando largura e centro como valores para os genes, por $q_i = \begin{bmatrix} \mu_{i1}=-0.5 \\ \sigma_{i1}=2 \end{bmatrix} \begin{bmatrix} \mu_{i2}=0.5 \\ \sigma_{i2}=2 \end{bmatrix}$, onde μ_{i1} e μ_{i2} indicam o centro e σ_{i1} e σ_{i2} representam a largura dos dois pulsos quadrados respectivamente.

A avaliação dos indivíduos clássicos é usada para ajustar os indivíduos quânticos: μ_{ij} e σ_{ij} são alterados de forma a levar o pulso para a região mais promissora do espaço de busca, aumentando-se a probabilidade de se observar um determinado conjunto de valores para o gene clássico nas proximidades dos indivíduos mais bem-sucedidos da população clássica. No caso do pulso quadrado, pode-se considerar o seguinte processo de atualização dos genes quânticos: modificar a largura dos pulsos de modo que o espaço de busca seja reduzido e modificar a posição dos pulsos de modo que o ponto central dos

mesmos coincida com o valor dos genes de um conjunto de indivíduos da população clássica.

Nesta tese, o AEIQ-R foi utilizado para evoluir os pesos de votação para cada rede neural membro do comitê e, assim, possibilitar a decisão final do comitê. Desta forma, o cromossomo utilizado terá tamanho *n*, onde *n* representa o número de redes do comitê. Cada gene, por sua vez, representará o peso de votação associado a cada rede neural.

Maiores detalhes sobre o AEIQ-R podem ser encontrados em [Abs da Cruz, 2007].

2.3.2. Algoritmo Evolutivo com Inspiração Quântica e Representação Mista (AEIQ-BR)

A motivação principal para a criação de um algoritmo com representação mista é que muitos problemas reais não podem ser resolvidos unicamente por decisões numéricas ou por decisões combinatórias. Mais especificamente na área de redes neurais, o processo de modelagem envolve decisões combinatórias (seleção das variáveis mais relevantes para a camada de entrada, quantos neurônios devem ser utilizados na camada intermediária, etc.) e, simultaneamente, decisões numéricas (valores ótimos dos pesos sinápticos).

Com esta motivação, [Pinho, 2010] propôs a criação de um algoritmo com inspiração quântica e representação binário-real, denominado AEIQ-BR, para otimização simultânea de problemas combinatórios e numéricos, ou seja, de natureza mista. O algoritmo AEIQ-BR foi o primeiro algoritmo evolutivo com inspiração quântica e representação mista proposto na literatura e herdará as principais características de seus precursores, tais como capacidade de otimização global de um problema e representação probabilística do espaço de busca, resultando em alta diversidade populacional em cada indivíduo quântico e necessidade de poucos indivíduos populacionais para explorar o espaço de soluções.

[Botelho, 2014] propôs uma extensão do algoritmo AEIQ-BR quando aplicado à evolução de redes neurais, introduzindo a técnica de regularização de complexidade de decaimento de pesos. A motivação para a utilização do decaimento de pesos é a busca pela minimização do tamanho da rede neural, mantendo-se o seu desempenho. A minimização do tamanho da rede implica na redução do número de seus parâmetros livres. Uma rede com muitos parâmetros tende a aprender também o ruído dos dados de treinamento, o que pode

ocasionar uma resposta incorreta aos padrões nunca vistos. Em uma rede muito complexa, a remoção de alguns pesos pode não aumentar o erro significativamente, indicando que estes pesos têm pouca ou nenhuma influência sobre a rede.

Assim como no AIEQ-R, o algoritmo AEIQ-BR também necessita de uma população de indivíduos que representem a superposição dos possíveis estados que o indivíduo clássico pode assumir ao ser observado. A população quântica Q(t), em um instante t qualquer do processo evolutivo, é formada por um conjunto de N indivíduos quânticos q_i *(i = 1, 2, 3, ..,N)*. Cada indivíduo quântico q_i desta população é formado por *L* genes g_{ij} *(j = 1, 2, 3, ...,L*). A principal diferença entre o AEIQ-R e o AEIQ-BR quanto à representação dos genes é que enquanto no AEIQ-R o indivíduo quântico utilizava genes exclusivamente na representação real, no AEIQ-BR uma parte do indivíduo será representada através de genes quânticos binários (*q-bit*, similar ao AEIQ-B) e outra parte através de genes quânticos reais (*q-real*, similar ao AEIQ-R). Assim, a representação de um indivíduo quântico *i* qualquer num instante de tempo *t* será dada por:

$$q_i = [(q_i)_b\ (q_i)_r] \tag{4}$$

onde o índice *b* representa a parte binária (*q*-bit) e o índice *r* representa a parte real (*q*-real). A parte binária de um indivíduo quântico com *M* genes binários pode ser escrita como:

$$(q_i)_b = \left(g_{i1} = \left|\begin{matrix}\alpha_{i1}\\ \beta_{i1}\end{matrix}\right|, g_{i2} = \left|\begin{matrix}\alpha_{i2}\\ \beta_{i2}\end{matrix}\right|, \ldots, g_{iM} = \left|\begin{matrix}\alpha_{iM}\\ \beta_{iM}\end{matrix}\right|\right)_b \tag{5}$$

A parte real de um indivíduo quântico é similar ao AEIQ-R (2) e o pulso quadrado também é utilizado para representar a FDP. Assim um indivíduo quântico pode ser descrito por:

$$q_i = [(q_i)_b\ (q_i)_r] = \left(\left|\begin{matrix}\alpha_{i1}\\ \beta_{i1}\end{matrix}\right| \left|\begin{matrix}\alpha_{i2}\\ \beta_{i2}\end{matrix}\right| \ldots \left|\begin{matrix}\alpha_{iM}\\ \beta_{iM}\end{matrix}\right|\right)_b \left(\left|\begin{matrix}\mu_{i1}\\ \sigma_{i1}\end{matrix}\right| \left|\begin{matrix}\mu_{i2}\\ \sigma_{i2}\end{matrix}\right| \ldots \left|\begin{matrix}\mu_{iG}\\ \sigma_{iG}\end{matrix}\right|\right)_r \tag{6}$$

Nesta tese, o AEIQ-BR será utilizado para realizar a modelagem completa de uma rede neural artificial MLP, usando o algoritmo quântico para decidir quais variáveis utilizar na camada de entrada, quantos neurônios utilizar na camada intermediária, os pesos da camada de entrada e da camada de intermediária, e, por fim, qual função de ativação utilizar em cada neurônio da camada intermediária e da camada de saída. A Figura 15 ilustra a informação que é codificada em cada um dos genes quânticos, binários ou reais de um cromossomo do AEIQ-BR. Este também será o cromossomo utilizado nos modelos neuroevolutivos que serão apresentados no capítulo 4.

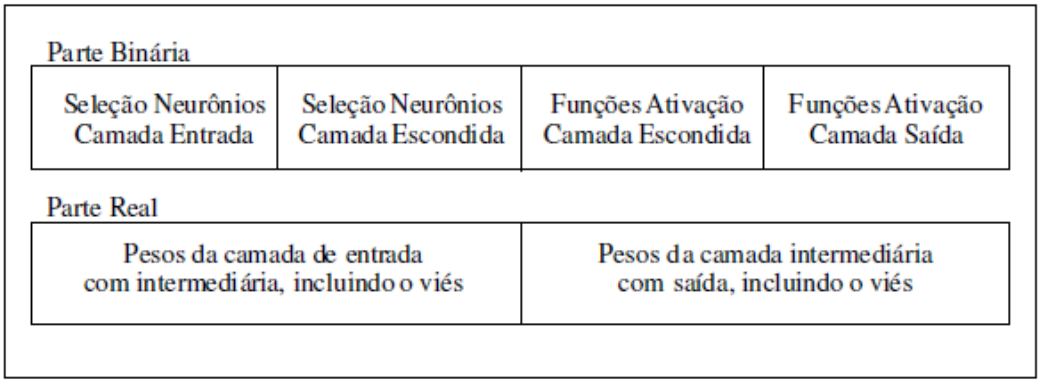


Figura 15. Informação codificada em um cromossomo quântico ou clássico [Pinho, 2010].

A decodificação de um cromossomo clássico é realizada da seguinte forma: para a camada de entrada, códigos na parte binária do cromossomo determinam quais variáveis de entrada serão selecionadas, ou seja, estarão ativas na rede neural, sendo que o código 1 significa presença e o código 0 significa ausência. Na camada intermediária (considerando apenas 1 camada escondida), códigos na parte binária do cromossomo determinam quais neurônios estarão ativos do total definido pelo usuário, sendo que o código 1 significa presença e o código 0 significa ausência. Cada neurônio ativo realizará a soma ponderada dos valores propagados da camada anterior, com os pesos determinados pela parte real do cromossomo. Cada um destes neurônios ativará esta soma ponderada com a função de ativação definida pela parte binária do cromossomo, onde 1 significa tangente hiperbólica e 0 sigmóide logística. Já na camada de saída, todos os neurônios realizarão a soma ponderada dos valores propagados da camada anterior com os pesos também determinados pela parte real do cromossomo, e genes na parte binária do cromossomo indicarão a função de ativação a ser utilizada onde, da mesma forma, o código 1 significa tangente hiperbólica e 0 sigmóide logística [Pinho, 2010].

A predição acontecerá através da atribuição da classe ao padrão de entrada representado pelo neurônio da camada de saída cuja ativação é máxima entre todos os neurônios desta camada. No AEIQ-BR, a evolução de pesos e funções de ativação nos cromossomos quânticos e clássicos está condicionada ao fato do neurônio estar ativo ou não na parte binária correspondente. Ou seja, os genes representantes dos pesos e funções de ativação permanecerão inalteráveis por cruzamentos quânticos e clássicos caso este neurônio esteja inativo. O usuário poderá definir o número mínimo de gerações que um neurônio permanecerá ativo e, portanto, sofrendo atualizações de seus pesos e função de

ativação antes que se permita a sua desativação por um cruzamento clássico [Botelho, 2014].

O processo completo de treinamento da rede neural utilizando o algoritmo AEIQ-BR pode ser ilustrado pela Figura 16. A rede neural criada pelo AEIQ-BR é similar à ilustrada pela Figura 17: o número de neurônios na camada de entrada e da camada escondida são evoluídos pelo AEIQ-BR, sendo o tamanho máximo de neurônios da camada de entrada igual ao número de atributos de entrada do problema (*k*) e o tamanho máximo de neurônios na camada escondida (*nh*) configurado pelo usuário. Desta forma, o número de genes é dado por:

$$num\_genes = (k + 2nh + nc)_b + \big((k+1) \times nh\big) + \big((nh+1) \times nc\big)_r \qquad (7)$$

Neste caso, a função de aptidão usada é a acurácia de classificação dada por:

$$Acurácia = 1 - \frac{1}{n}\sum_{i=1}^{n} \left| C_i - \hat{C}_i \right| \qquad (8)$$

onde C_i é a classe do i-ésimo padrão, ao passo $\hat{C}_i$ é a classe predita pelo indivíduo (MLP). Sempre que $C_i = \hat{C}_i$ então o resultado é zero, caso contrário é igual a um. Cada indivíduo é submetido a esta função de avaliação, de tal forma que os melhores indivíduos são aqueles que possuem maior acurácia. Maiores detalhes sobre o AEIQ-BR podem ser encontrados em [Pinho, 2010].

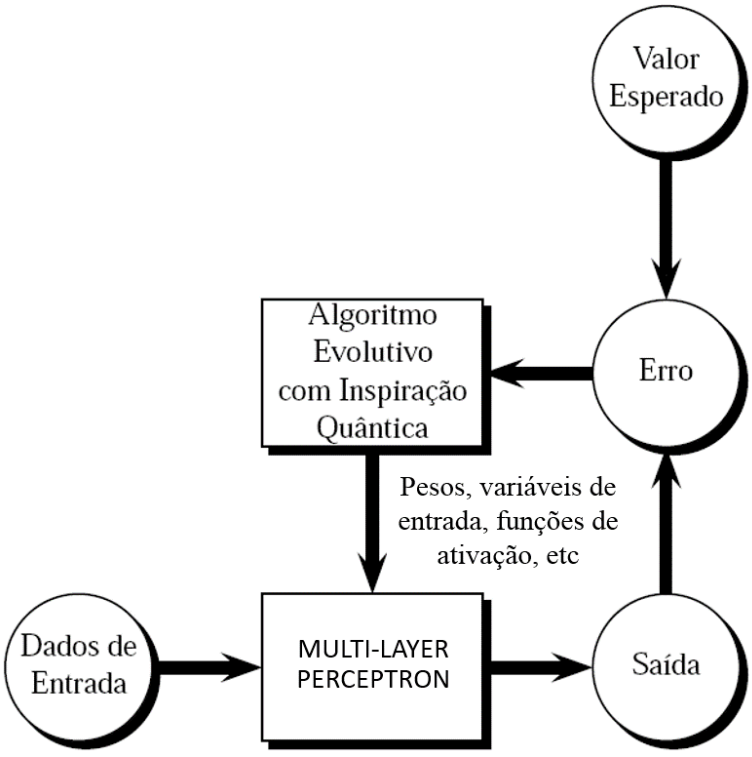


Figura 16. Processo de treinamento da rede neural com o AEIQ-BR.

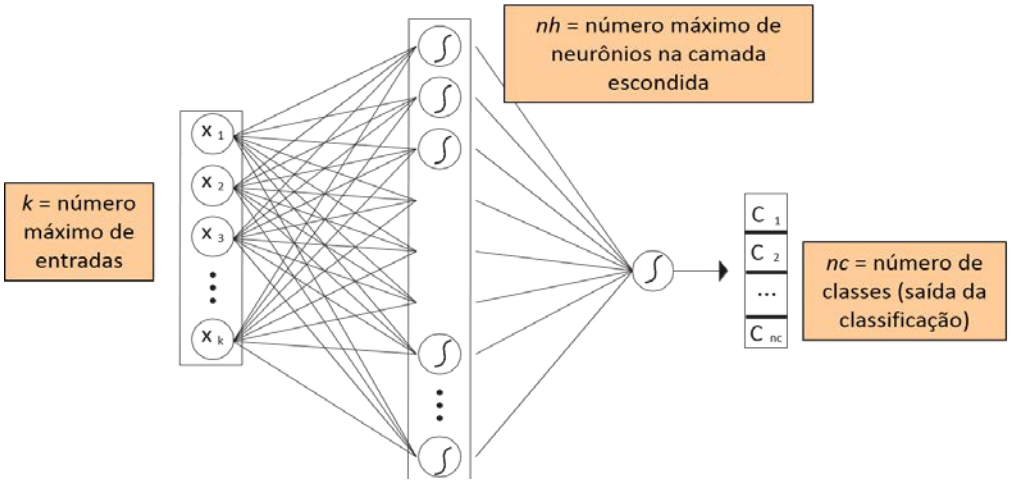


Figura 17. Rede Neural Criada pelo AEIQ-BR.

Esta seção apresentou uma revisão bibliográfica dos modelos evolutivos com inspiração quântica de interesse desta tese. O próximo capítulo apresentará detalhes do modelo de detecção de *drift* proposto.

3
Mecanismo de Detecção DETECTA

Este capítulo apresenta a proposta de um novo mecanismo de detecção de *drift,* denominado DetectA (do inglês *Detect Abrupt*). Resumidamente, este método é composto de 3 etapas: (i) os padrões do conjunto de teste são rotulados através de um método não-supervisionado; (ii) algumas estatísticas são computadas a partir dos conjuntos de treino e de teste, condicionadas aos rótulos de classes determinados na etapa anterior; e (iii) as estatísticas de treino e de teste são comparadas usando um teste de hipótese multivariado. A partir do resultado do teste de hipótese é tomada uma decisão com respeito a ocorrência ou não de *drift*. Caso ocorra um *drift*, medidas são tomadas para contornar a mudança de comportamento e continuar o processo de aprendizagem.

3.1.
Mecanismo de Detecção

Conforme apresentado no Capítulo 2, a literatura define que em um ambiente não estacionário a Função Densidade de Probabilidade (FDP) Conjunta Condicional se altera quando se comparam dois ou mais instantes de tempo. Formalmente, seja $\boldsymbol{X} = [X_1, X_2, \dots, X_J]$ um vetor composto por variáveis aleatórias (ou vetor aleatório) que possui uma FDP conjunta $f^{(t)}_{X_1,\dots,X_J}(x_1, \dots, x_J) = f^{(t)}_{\boldsymbol{X}}(\boldsymbol{x})$ no instante *t* de tempo (*t=1,...,T*). Considere por $f^{(t)}_{\boldsymbol{X}}(\boldsymbol{x}\,|C = k)$ a FDP condicional conjunta do vetor aleatório $\boldsymbol{X}$ dados eventos pertencentes à classe *k* (*k=1,...,K*).

Um ambiente é tido como estacionário quando a condição:

$$f^{(t)}_{\boldsymbol{X}}(\boldsymbol{x}\,|C = k) = f^{(t+\gamma)}_{\boldsymbol{X}}(\boldsymbol{x}\,|C = k) \qquad (9)$$

está assegurada para todo γ. Quando a $f^{(t)}_{\boldsymbol{X}}(\boldsymbol{x}\,|C = k)$ varia com o tempo, ou seja, a condição (9) não é válida, tem-se a ocorrência de um *drift*.

Como exemplo, pode-se considerar uma pesquisa social realizada anualmente no Brasil a partir de 1970 com objetivo de mensurar o Peso e a Altura de indivíduos do sexo Masculino e Feminino. Neste caso, têm-se duas

variáveis aleatórias $X_1 = Peso$ e $X_2 = Altura$, que são funções que associam a cada indivíduo pesquisado um número real referente à sua respectiva medição. Adicionalmente, têm-se duas classes: $C_1 = Feminino$ e $C_2 = Masculino$. As $f_{X_1,X_2}^{(1970)}(x_1, x_2 \mid C_1)$ e $f_{X_1,X_2}^{(1970)}(x_1, x_2 \mid C_2)$ representam as FDP conjuntas condicionais do peso e altura dos indivíduos dado o sexo Masculino e Feminino, respectivamente.

Neste caso, um *drift* ocorre quando em alguma pesquisa anual:

$$f_{X_1,X_2}^{(1970+\gamma_1)}(x_1, x_2 \mid C_1) \neq f_{X_1,X_2}^{(1970+\gamma_2)}(x_1, x_2 \mid C_1) \quad (10)$$

ou

$$f_{X_1,X_2}^{(1970+\gamma_1)}(x_1, x_2 \mid C_1) \neq f_{X_1,X_2}^{(1970+\gamma_2)}(x_1, x_2 \mid C_1) \quad (11)$$

Ou seja, alguma das desigualdades ocorre. Desta maneira, é possível identificar tanto o momento do *drift* quanto em qual dos sexos houve uma variação na distribuição de Pesos e Alturas.

De um ponto de vista teórico, as definições estabelecidas são razoáveis (cabe ressaltar que são bastante similares às usadas em séries temporais). Contudo, raramente se observa a $f_{\boldsymbol{X}}^{(t)}(\boldsymbol{x} \mid C = k)$, sendo possível estima-la a partir de um conjunto de observações $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ rotuladas, de maneira que a FDP condicional conjunta estimada $\hat{f}_{\boldsymbol{X}}^{(t)}(\boldsymbol{x} \mid C = k)$ se aproxime da real, conforme o número de observações cresce. Há alguns procedimentos apresentados na literatura, tais como a estimação de uma $\hat{f}_{\boldsymbol{X}}^{(t)}(\boldsymbol{x} \mid C = k)$ a partir da composição de funções do tipo Kernel [Chen, 2000; Botev et al., 2010], ou ao combinar as distribuições marginais empíricas (ou teóricas) a partir do uso de operadores Cópulas [Cherubini et al., 2004; Trivedi & Zimmer, 2007].

Contudo, ambas as abordagens demandam um número excessivo de amostras, principalmente quando o número de atributos (variáveis aleatórias) é elevado. Para contornar esta limitação, pode-se: (*i*) usar uma das abordagens de estimação acima apresentadas para $\hat{f}_{\boldsymbol{X}}^{(t)}(\boldsymbol{x} \mid C = k)$ e permanecer nessa linha, buscando testes de hipóteses ou limiares para qualificar uma diferença significativa; ou (*ii*) aumentar as hipóteses sobre a FDP conjunta condicional e, assim, inferir a presença de *drifts* a partir dos parâmetros que a compõem.

Esta pesquisa segue a linha exibida em (*ii*). Assim, presume-se que o vetor aleatório $\boldsymbol{X}$ condicionado à classe *k* segue uma Distribuição Normal Multivariada. Para se verificar a validade dessa hipótese, podem ser usados os testes de Mardia [Mardia, 1970] ou BHEP [Baringhaus & Henze, 1988]. Apesar de existirem casos em que tal pressuposto é violado, advoga-se por ele, pois

este viabiliza um conjunto de definições menos estritas e, portanto, facilmente verificáveis na prática. Além disso, este possibilita o delineamento de procedimentos preditivos de *drift*, que colaboram para o ajuste e maior chance de acerto do classificador. A próxima subseção discute com maiores detalhes esses temas.

3.1.1. Definição de um *Drift* sob Novos Pressupostos

Nesta subseção são apresentadas definições sobre diferentes tipos de *drifts*. Neste sentido, será utilizado o pressuposto enunciado anteriormente de que o vetor aleatório $\boldsymbol{X}$ condicionado à classe *k* segue uma Distribuição Normal Multivariada [Johnson & Wichern, 2014], que em termos matemáticos se manifesta:

$$\boldsymbol{X} \mid C_k \sim N_J(\boldsymbol{\mu_{C_k}}, \Sigma_{C_k}), \text{ onde } f_{\boldsymbol{X}}(\boldsymbol{x}|C_k) = \frac{1}{(2\pi)^{J/2}|\boldsymbol{\Sigma_{Ck}}|^{J/2}}\, e^{-\left(x-\boldsymbol{\mu_{C_k}}\right)^T \boldsymbol{\Sigma_{C_k}^{-1}} \left(x-\boldsymbol{\mu_{C_k}}\right)\left(\frac{1}{2}\right)} \qquad (12)$$

onde:

- $\boldsymbol{\mu_{C_k}} = \left[\boldsymbol{\mu_{X_1|C_k}}, \boldsymbol{\mu_{X_2|C_k}}, \ldots, \boldsymbol{\mu_{X_J|C_k}}\right]^{\boldsymbol{T}}$ é o vetor com J coordenadas composto pelas médias condicionais da classe k de cada j-ésima variável aleatória que compõe o vetor aleatório $\boldsymbol{X}$.
- $\boldsymbol{\Sigma_{C_k}} = \begin{bmatrix} \boldsymbol{\sigma^2_{X_1|C_k}} & \boldsymbol{\sigma_{X_1,X_2|C_k}} & \cdots & \boldsymbol{\sigma_{X_1,X_J|C_k}} \\ \boldsymbol{\sigma_{X_2,X_1|C_k}} & \ddots & & \vdots \\ \vdots & & & \\ \boldsymbol{\sigma_{X_J,X_1|C_k}} & \cdots & & \boldsymbol{\sigma^2_{X_J|C_k}} \end{bmatrix}$ é a matriz de variâncias ($\boldsymbol{\sigma^2_{X_j|C_k}}$) e covariâncias ($\boldsymbol{\sigma_{X_l,X_j|C_k}}$) condicionais à classe k do vetor aleatório $\boldsymbol{X}$. Por definição, esta matriz é simétrica e positiva.

por fim, $\left|\Sigma_{C_k}\right|$ é o determinante da matriz de variâncias-covariâncias. Portanto, a Distribuição Normal Multivariada depende de dois conjuntos de parâmetros: o vetor de médias e a matriz de variâncias-covariâncias. A partir do conhecimento de ambos os parâmetros (ou de suas estimações) é possível realizar inferências sobre toda a população (obviamente quando esta segue o modelo Normal Multivariado). A Figura 18 apresenta as curvas de nível para a FDP da Distribuição Normal Multivariada, para o caso em que *J=2* (duas variáveis ou atributos).

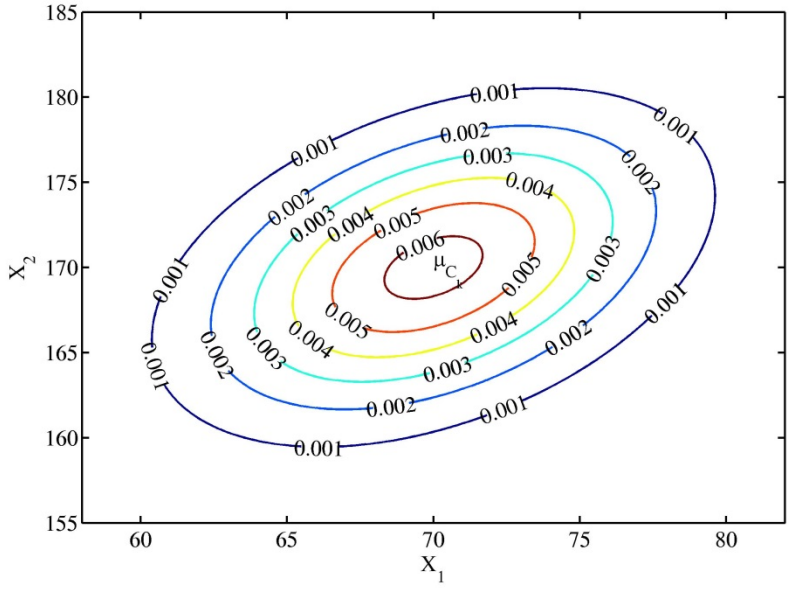


Figura 18. Exemplo de FDP para a Distribuição Normal Multivariada: caso Bivariado.

Para a FDP exibir o formato e posicionamento dispostos na Figura 18, foi necessário escolher os parâmetros para o vetor de médias e matriz de variâncias-covariâncias conforme abaixo:

$$\boldsymbol{\mu_{Mas}} = [70\ \ 170]^T \text{ e } \boldsymbol{\Sigma_{C_k}} = \begin{bmatrix} 25 & 11 \\ 11 & 30 \end{bmatrix}$$

Pode-se também notar que se plotou a densidade de probabilidade para uma escolha de valores para x_1 e x_2. Por exemplo, se fosse escolhido $x_1 = 65$ e $x_2 = 165$, teríamos uma densidade de aproximadamente 0,003. Quanto mais a escolha de x_1 e x_2 é próxima ao vetor de médias, maior a densidade calculada. Contudo, apesar dos valores da densidade ser informação relevante, um maior foco é dado aos parâmetros da distribuição. Por exemplo, a distribuição é um pouco mais dilatada no sentido da variável X_1, devido à relação entre variância e média ser maior nesse sentido do que na variável X_2. Ainda, ressalta-se que há uma correlação positiva (termo de covariância = 25) entre as duas variáveis, o que implica que o aumento de um atributo está associado ao crescimento conjunto do outro. Portanto, a ênfase da análise é voltada ao vetor de médias e matriz de variâncias-covariâncias, que definem o formato e locação da distribuição.

Com base nestes dois parâmetros – vetor de médias e matriz de variâncias-covariâncias – é explorada uma versão menos estrita da definição de estacionariedade exposta na expressão (9). Considere $\boldsymbol{X}^{(t)}$ um vetor aleatório (não necessariamente com Distribuição Normal Multivariada), com vetor de valores esperados e matriz de variâncias-covariâncias existentes e finitos.

Assim, um ambiente é tido como fracamente estacionário ou estacionário de 2ª ordem se:

$$E\left[\boldsymbol{X}^{(t)} \left| C_k^{(t)}\right.\right] = E\left[\boldsymbol{X}^{(t+\gamma)} \left| C_k^{(t+\gamma)}\right.\right] \quad (13)$$

$$V\left[\boldsymbol{X}^{(t)} \left| C_k^{(t)}\right.\right] = V\left[\boldsymbol{X}^{(t+\gamma)} \left| C_k^{(t+\gamma)}\right.\right] \quad (14)$$

Ou seja, se o vetor de valores esperados e matriz de variâncias-covariâncias da FDP conjunta condicional não variar, então o ambiente é tido como fracamente estacionário. Observa-se que esta definição é menos estrita, pois duas FDP conjuntas condicionais podem se igualar em termos de valores esperados e matriz de variâncias-covariâncias, mas não necessariamente serem as mesmas funções. Caso esta condição $\boldsymbol{X}^{(t)} | \, C_k^{(t)} \sim N_J(\, \boldsymbol{\mu_{C_k}}(t), \Sigma_{C_k}(t))$ seja assumida (ou atestada por meio de dados), é fácil demonstrar que a definição de estacionariedade estrita e a fraca são equivalentes. A única dificuldade nesta definição é incorporar os diferentes tipos de *drift*, em destaque o do tipo abrupto e gradual. Portanto, a seguir, é explorada uma versão semelhante dessa definição para *drifts* abruptos, que são o foco desta pesquisa. Para tal, será utilizado novamente o exemplo da investigação de peso e altura de indivíduos. A Figura 19 apresenta um exemplo com dois atributos: Altura (cm) e Peso (kg), cujo modelo escolhido foi de duas Distribuições Normais Bivariadas para descrever a configuração da população do sexo Masculino (curvas de nível em azul) e do sexo Feminino (curvas de nível em vermelho).

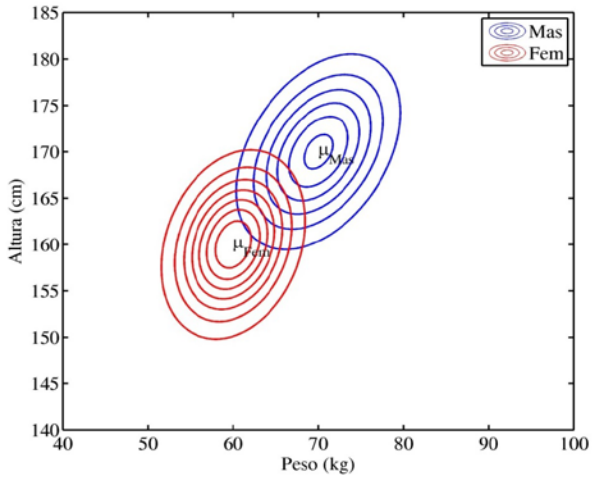


Figura 19. Exemplo de Distribuições Normais Bivariadas, representando o Grupo do Sexo Masculino e Feminino.

É possível observar que a média do Peso e da Altura dos indivíduos do sexo Masculino são maiores do que a do Feminino. Observa-se ainda que há uma variabilidade um pouco maior no grupo Masculino, bastando observar o maior dilatamento das curvas de nível da FDP conjunta condicional. Tais

características podem ser observadas a partir dos parâmetros das distribuições Normais Bivariadas, cujos valores são:

$$\boldsymbol{\mu_{Mas}} = [70kg \;\; 170cm]^T \text{ e } \boldsymbol{\mu_{Fem}} = [60kg \;\; 160cm]^T$$

$$\boldsymbol{\Sigma_{Mas}} = \begin{bmatrix} 25kg^2 & 11kg\,cm \\ 11cm\,kg & 30cm^2 \end{bmatrix} \text{ e } \boldsymbol{\Sigma_{Fem}} = \begin{bmatrix} 17kg^2 & 5kg\,cm \\ 5cm\,kg & 25cm^2 \end{bmatrix}$$

3.1.1.1.

Drift Abrupto no Vetor de Médias

Definição 1. Considera-se que ocorreu um ***drift*** **abrupto na média condicional** quando a seguinte igualdade não é satisfeita:

$$\boldsymbol{\mu_{C_k}}(t) = \boldsymbol{\mu_{C_k}}(t+1) \qquad (15)$$

logo, o vetor de médias condicionais à classe k difere do momento t para t+1.

Observação: a igualdade (15) mantém o mesmo princípio de um teste de hipótese: primeiramente, é necessário encontrar um estimador para $\boldsymbol{\mu_{C_k}}(t)$, $\boldsymbol{\mu_{C_k}}(t+1)$ e para os outros parâmetros envolvidos . Segundo, dados estes valores, submeta-os a um teste de hipótese que, fixado um nível de significância (α), compute a probabilidade de não rejeitar a igualdade entre $\boldsymbol{\mu_{C_k}}(t)$ e $\boldsymbol{\mu_{C_k}}(t+1)$. Se esta probabilidade é menor ou igual ao nível de significância, então ocorreu um *concept drift* abrupto no vetor de médias condicionais. A figura 20 ilustra um *drift* abrupto ocorrendo no vetor de médias condicionais à classe C_1. Nota-se que agora é mais difícil traçar uma região de discriminação entre as duas classes. O teste de hipótese usado para esta comparação é chamado T^2 Hotteling [Johnson & Wichern, 2014], apresentado na seção 3.1.2.1.

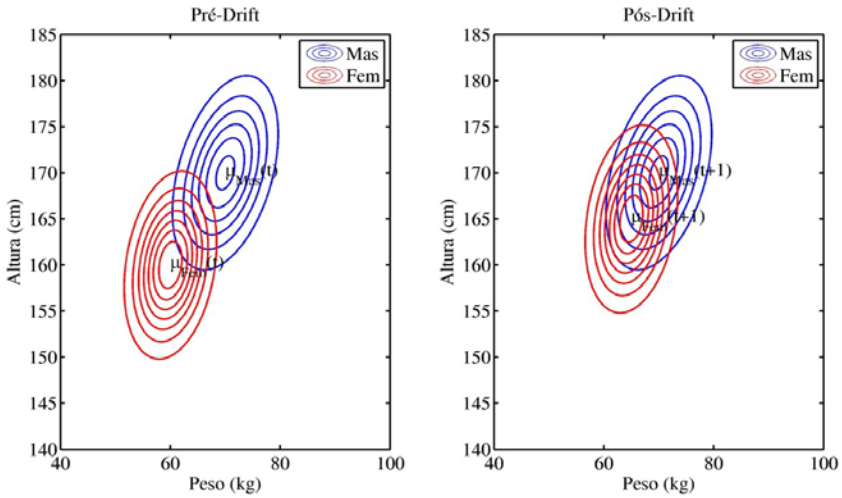


Figura 20. Exemplo de *drift* abrupto na média condicional dos atributos ao sexo Feminino.

Portanto, houve um aumento médio de 5kg e 5cm no grupo Feminino. Observa-se que a fronteira entre as classes se altera substancialmente, tornando-se mais difícil distinguir, dado um novo indivíduo, se ele pertence ao Sexo Masculino ou Feminino.

3.1.1.2.

Drift Abrupto na Matriz de Covariâncias

Definição 2. Considera-se que ocorreu um ***drift* abrupto na matriz de covariância condicional** quando a seguinte igualdade não é satisfeita:

$$\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t) = \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t+1) \qquad (17)$$

logo, a matriz de covariância condicional à classe k difere do momento t para t+1.

Observação: de forma similar, esta igualdade será verificada usando um teste de hipótese. A figura 21 ilustra dois exemplos de *drift* abrupto ocorrido na matriz de covariância. Nota-se que a variância condicional de X_1 à classe C_1 aumentou, dilatando as curvas de nível nesta direção, enquanto a covariância condicional de X_1 e X_2 à classe C_2 perde intensidade, rotacionando as curvas de nível. A seção 3.1.2.2 apresenta o teste de hipótese para realizar esta comparação, denominado teste de Box-M [Johnson & Wichern, 2014].

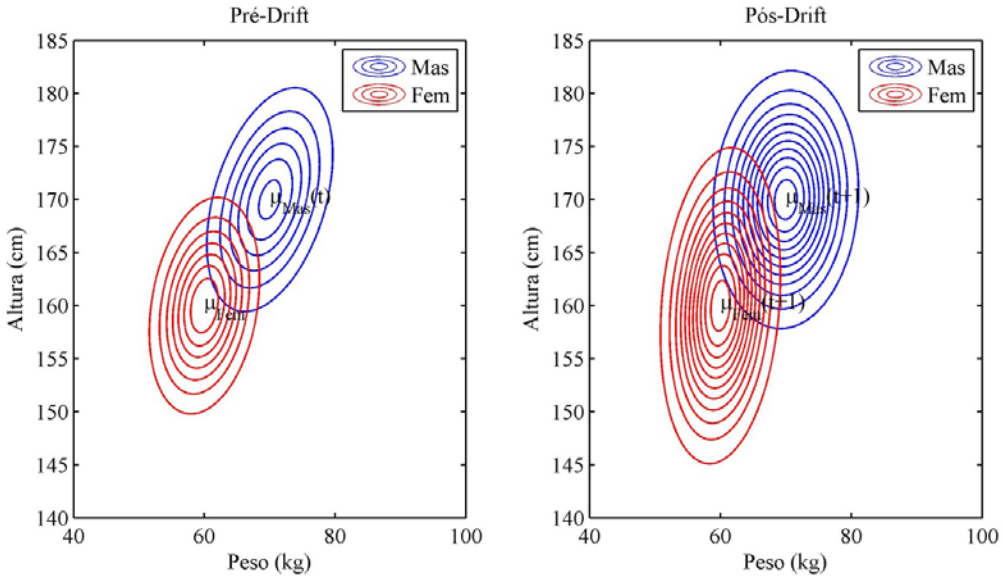


Figura 21. Exemplo de *drift* abrupto na matriz de covariância condicional dos atributos ao sexo Masculino e Feminino.

Portanto, observa-se uma dispersão maior no sentido da altura para o grupo dos elementos do sexo Feminino, e uma maior "independência" entre os atributos peso e altura para o sexo Masculino. Nota-se também que a fronteira entre as classes se altera substancialmente, tornando-se mais difícil distinguir, dado um novo indivíduo, se ele pertence ao Sexo Masculino ou Feminino.

3.1.2. Detecção de *Drifts*

Dado o conjunto de definições previamente elaboradas, um *drift* abrupto não ocorreu caso as igualdades:

$$\boldsymbol{\mu}_{\boldsymbol{C_k}}(t) = \boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1) \tag{19}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t) = \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t+1) \tag{20}$$

sejam honradas. Quando o vetor aleatório $\boldsymbol{X}^{(t)} = [X_1^{(t)}, \dots, X_J^{(t)}]$ segue uma distribuição condicional à classe *k* Normal Multivariada (i.e., $\boldsymbol{X}^{(t)}|C_k^{(t)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(t), \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t)\right)$), é fácil ver que o par de definições acima (estacionariedade de 2ª ordem) e a estacionariedade estrita (relacionada a distribuição conjunta) são equivalentes .

Contudo, raramente se conhece os parâmetros populacionais $\mu_{C_k}(t)$ e $\Sigma_{C_k}(t)$. O que se observa é uma amostra aleatória $\boldsymbol{X}_1^{(t)}, \dots, \boldsymbol{X}_n^{(t)}$ de tamanho *n* da população em estudo. A partir desta amostra, estima-se os parâmetros $\mu_{C_k}(t)$ e $\Sigma_{C_k}(t)$, usando os estimadores de máxima verossimilhança [Johnson e Wichern, 2014]:

$$\overline{\boldsymbol{X}}_{C_k}(t) = [\bar{X}_{1|C_k}(t), \dots, \bar{X}_{J|C_k}(t)] \tag{21}$$

$$\boldsymbol{S}_{C_k}(t) = \begin{bmatrix} s^2_{1|C_k}(t) & \cdots & s_{1,J|C_k}(t) \\ \vdots & \ddots & \vdots \\ s_{J,1|C_k}(t) & \cdots & s^2_{J|C_k}(t) \end{bmatrix} \tag{22}$$

onde $\overline{\boldsymbol{X}}_{C_k}(t)$ é um vetor, cujos elementos são as médias aritméticas $\overline{\boldsymbol{X}}_{j|C_k}(t)$ de cada j-ésimo atributo, enquanto $\boldsymbol{S}_{C_k}(t)$ é uma matriz, com entradas na diagonal principal idênticas às variâncias amostrais de cada j-ésimo atributo ($s^2_{j|C_k}(t)$), e os elementos fora da diagonal são as covariâncias amostrais entre o atributo *j* com o *l* ($s_{j,l|C_k}(t)$). Todos estes no instante *t* e condicionados à classe *k*.

Como $\boldsymbol{X}_1^{(t)}, \ldots, \boldsymbol{X}_n^{(t)}$ forma uma amostra aleatória de uma população com Distribuição Normal Multivariada, tem-se três resultados a partir desses estimadores:

- (1) $\bar{\boldsymbol{X}}_{\boldsymbol{C_k}}(\boldsymbol{t}) \sim \boldsymbol{N_j}\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(\boldsymbol{t}), \left(\frac{\boldsymbol{1}}{\boldsymbol{n}}\right)\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(\boldsymbol{t})\right)$;
- (2) $(\boldsymbol{n}-\boldsymbol{1})\boldsymbol{S}_{\boldsymbol{C_k}}(\boldsymbol{t}) \sim \boldsymbol{W}\left(\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(\boldsymbol{t}), \boldsymbol{n}-\boldsymbol{1}\right)$;
- (3) $\bar{\boldsymbol{X}}_{\boldsymbol{C_k}}(\boldsymbol{t})$ e $\boldsymbol{S}_{\boldsymbol{C_k}}(\boldsymbol{t})$ são independentes.

então, $\bar{\boldsymbol{X}}_{C_k}(t)$ segue uma Distribuição Normal Multivariada, com média idêntica à populacional $\mu_{C_k}(t)$ e uma matriz de covariância idêntica à populacional $\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t)$, mas cujo tamanho se reduz conforme se aumenta o número de amostras. A quantidade $(n-1)\boldsymbol{S}_{C_k}(t)$ segue uma Distribuição Wishart com parâmetro de escala $\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t)$ e $(n-1)$ graus de liberdade. Ainda, $\bar{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$ são independentes. A prova desses resultados é apresentada em [Johnson & Wichern, 2014].

Tendo em vista este conjunto de definições, estimadores e resultados, as próximas subseções apresentam as formas de caracterização de *drift* abrupto podem ocorrer na média ou na covariância.

3.1.2.1.

Drift Abrupto no Vetor de Médias

Suponha as duas hipóteses:

$$H_0: \boldsymbol{\mu}_{\boldsymbol{C_k}}(t) = \boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$$
$$H_1: \boldsymbol{\mu}_{\boldsymbol{C_k}}(t) \neq \boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$$

Se H_0 é rejeitada, então considera-se que ocorreu um drift abrupto no vetor de métias condicionais. Dadas duas amostras aleatórias $\boldsymbol{X}_1^{(t)}, \ldots, \boldsymbol{X}_{n_1}^{(t)}$ e $\boldsymbol{X}_1^{(t+1)}, \ldots, \boldsymbol{X}_{n_2}^{(t+1)}$ de tamanho n_1 e n_2 respectivamente, com $\boldsymbol{X}^{(t)}|C_k^{(t)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(\boldsymbol{t}), \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(\boldsymbol{t})\right)$ e $\boldsymbol{X}^{(t+1)}|C_k^{(t+1)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(\boldsymbol{t}+1), \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(\boldsymbol{t}+1)\right)$, o teste estatístico T^2 Hotteling é dado por:

$$T^2 = \left(\bar{\boldsymbol{X}}_{C_k}(t) - \bar{\boldsymbol{X}}_{C_k}(t+1)\right)^t \left(\frac{\boldsymbol{S}_{\boldsymbol{C_k}}(\boldsymbol{t})}{n_1} + \frac{\boldsymbol{S}_{\boldsymbol{C_k}}(\boldsymbol{t}+1)}{n_2}\right)^{-1} \left(\bar{\boldsymbol{X}}_{C_k}(t) - \bar{\boldsymbol{X}}_{C_k}(t+1)\right) \quad (23)$$

que compara os vetores de médias amostrais $\bar{\boldsymbol{X}}_{C_k}(t)$ e $\bar{\boldsymbol{X}}_{C_k}(t+1)$ (estimadores para $\boldsymbol{\mu_{C_k}(t)}$ e $\boldsymbol{\mu_{C_k}}(t+1)$) em dois diferentes instantes (blocos), de forma que se T^2 é muito grande, H_0 é rejeitada. $\boldsymbol{S_{C_k}(t)}$ e $\boldsymbol{S_{C_k}(t+1)}$ representam as matrizes de covariância amostrais. Dados os pressupostos por trás das amostras aleatórias e sobre H_0, o teste estatístico $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}\,T^2$ segue uma distribuição *F* com *J* e $(n_1+n_2)-J-1$ graus de Liberdade [Johnson & Wichern, 2014]. Com este conhecimento, é possível estabelecer uma zona de rejeição para H_0 a fim de identificar *drifts* abruptos no vetor de médias. A seguir, são descritos os passos para a execução do teste T^2 Hotteling:

(1) Calcular $\bar{\boldsymbol{X}}_{C_k}(t)$ e $\bar{\boldsymbol{X}}_{C_k}(t+1)$, assim como $\boldsymbol{S}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t+1)$.

(2) Computar a estatística de teste T^2 (expressão 23).

(3) Ocorreu *drift* abrupto (H_0 foi rejeitada) caso $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}\,T^2 > F_{J,(n_1+n_2)-J-1}(\alpha)$, onde $F_{J,(n_1+n_2)-J-1}(\alpha)$ é o percentil 100*(1-α) da distribuição $F_{J,(n_1+n_2)-J-1}$ acumulada.

3.1.2.2.

Drift Abrupto na Matriz de Covariâncias

Suponha duas hipóteses:

$$H_0: \boldsymbol{\Sigma_{C_k}}(t) = \boldsymbol{\Sigma_{C_k}}(t+1)$$

$$H_1: \boldsymbol{\Sigma_{C_k}}(t) \neq \boldsymbol{\Sigma_{C_k}}(t+1)$$

Se H_0 é rejeitada, então considera-se que ocorreu um *drift* abrupto na matriz de covariâncias condicionais. Dadas duas amostras aleatórias $\boldsymbol{X}_1^{(t)}, \dots, \boldsymbol{X}_{n_1}^{(t)}$ e $\boldsymbol{X}_1^{(t+1)}, \dots, \boldsymbol{X}_{n_2}^{(t+1)}$, com $\boldsymbol{X}^{(t)}|C_k^{(t)} \sim N_J\left(\mu_{C_k}(t), \Sigma_{C_k}(t)\right)$ e $\boldsymbol{X}^{(t+1)}|C_k^{(t+1)} \sim N_J\left(\mu_{C_k}(t+1), \Sigma_{C_k}(t+1)\right)$, considere o teste de razão de verossimilhança como:

$$\Lambda = \left(\frac{\det(\boldsymbol{S_{ck}}(t))}{\det(\boldsymbol{S_{pool}})}\right)^{\frac{n_1-1}{2}} * \left(\frac{\det(\boldsymbol{S_{ck}}(t+1))}{\det(\boldsymbol{S_{pool}})}\right)^{\frac{n_2-1}{2}} \tag{24}$$

Onde $\boldsymbol{S_{pool}} = \frac{(n_1-1)\boldsymbol{S_{ck}}(t)+(n_2-1)\boldsymbol{S_{ck}}(t+1)}{(n_1-1)+(n_2-1)}$ é a matriz de covariância combinada. O teste Box-M [Johnson & Wichern, 2014] usa a estatística:

$$M = -2\ln\Lambda \tag{25}$$

Se a hipótese nula é verdadeira, as matrizes de covariâncias não ser diferem substancialmente e, consequentemente, estas não devem diferir significativamente da matriz de covariância combinada, tornando Λ próximo a 1 e $M = 0$. Finalmente, define-se a quantidade *u* por:

$$u = \left[\frac{1}{(n_1 - 1)} + \frac{1}{(n_2 - 1)} - \frac{1}{(n_1 - 1) + (n_2 - 1)}\right]\left[\frac{2J^2 + 3J - 1}{6(J + 1)}\right] \quad (26)$$

Então, $C = (1 - u)M$ segue aproximadamente uma distribuição χ^2 com $\frac{1}{2}J(J + 1)$ graus de liberdade. Com este conhecimento, é possível estabelecer uma zona de rejeição para H_0 a fim de identificar *drift* abrupto na matriz de covariâncias. Como mostrado em [Johnson & Wichern, 2014], a aproximação χ^2 funciona bem quando $n_1, n_2 > 20$ e o número de atributos é menor que 5. Algumas bases de dados têm mais de 5 atributos, e nestes casos a aproximação pela distribuição *F* foi utilizada [Mardia, 1971]. Os passos para execução do teste Box-M são:

(1) Calcular $\boldsymbol{S}_{C_k}(t)$, $\boldsymbol{S}_{C_k}(t + 1)$ e $\boldsymbol{S}_{\boldsymbol{pool}}$.

(2) Computar a M (expressão 7) e *u* (expressão 25)

(3) Ocorreu *drift* se $C > \chi^2_{\frac{1}{2}J(J+1)}(\alpha)$, onde $\chi^2_{\frac{1}{2}J(J+1)}(\alpha)$ é o percentil 100*(1-α) da distribuição $\chi^2_{\frac{1}{2}J(J+1)}$.

A próxima subseção utiliza os testes estatísticos apresentados, de forma a detectar *drifts* quando se possui conhecimento das classes no instante $t + 1$ (detecção reativa) e quando não se possui tal conhecimento (detecção proativa).

3.1.3. Formas de Detecção de *Drifts*

Conforme já mencionado, todos os métodos de detecção citados na revisão bibliográfica do capítulo 2, assim como a maioria dos métodos encontrados na literatura, funcionam de maneira reativa, ou seja, após a ocorrência do *drift* e erro do modelo, uma vez que eles dependem dos rótulos de classes dos padrões de entrada. Logo, a efetividade de tais metodologias consiste na velocidade da identificação da ocorrência passada do *drift*, isto é, na redução do intervalo de tempo entre um *drift* e sua detecção. Tal característica se manifesta comumente em trabalhos de reconhecimento de padrões em fluxos de dados, no qual a acurácia de um modelo após um *drift* cai abruptamente, ou

se degenera aos poucos. Este trabalho busca ir além da detecção reativa de *drift*, tentando também detectar de forma proativa e se planejar e reagir adequadamente a esse cenário de mudança. A seguir, serão apresentados algoritmos para ambas as abordagens propostas neste trabalho: detecção reativa e detecção proativa de *drift* abrupto.

Para compreensão dos algoritmos de detecção, considere um conjunto de *n* padrões no instante *t* - $\boldsymbol{x_1}(t), \dots, \boldsymbol{x_n}(t)$ -, onde $\boldsymbol{x_i}(t) = [x_{i1}(t), \dots, x_{iJ}(t)]$ é o i-ésimo padrão composto por observações de *J* atributos (*i=1,...,n* e *j=1,...,J*). Deste conjunto de *n* padrões n_1 padrões pertencem a classe 1, n_2 padrões pertencem a classe 2, até n_K pertencem a classe K (*k=1,...,K*). Pode-se assumir que os *n* padrões formam uma amostra aleatória $\boldsymbol{X}_1^{(t)}, \dots, \boldsymbol{X}_n^{(t)}$ de uma população com Distribuição Normal Multivariada. Apesar de ser possível efetuar os testes de hipótese elencados nas subseções anteriores sem a necessidade de tal pressuposto, a sua importância se dá na definição correta da distribuição de probabilidade associada à estatística de teste considerada (a estatística T^2 ou *M*, por exemplo). Quando tal pressuposto não é verificado, pode-se estar aumentando/reduzindo a probabilidade de se rejeitar uma hipótese quando esta era verdadeira ou vice-versa. Somente com os testes em bases de dados reais será possível aferir a eficiência do método em cenários cuja distribuição do vetor aleatório não é idealizada.

3.1.3.1.

Detecção Reativa

A detecção reativa implica na existência das classes dos padrões no instante $t+1$. Seja $\boldsymbol{x_1}(t+1), \dots, \boldsymbol{x_n}(t+1)$ o conjunto de *n* padrões no instante $t+1$. Considere por α o nível de significância definido anteriormente pelo usuário.

Para a detecção reativa de um *drift* abrupto, deve-se realizar:

No vetor de médias condicionais:

(1) Calcule $\overline{\boldsymbol{X}}_{C_k}(t)$ e $\overline{\boldsymbol{X}}_{C_k}(t+1)$, assim como $\boldsymbol{S}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t+1)$.

(2) Compute a estatística de teste T^2 (expressão 23).

(3) Defina que ocorreu um *drift* caso $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}\ T^2 > F_{J,(n_1+n_2)-J-1}(\alpha)$

Na matriz de covariância condicional:

(1) Calcule $\boldsymbol{S}_{C_k}(t)$, $\boldsymbol{S}_{C_k}(t+1)$ e $\boldsymbol{S_{pool}}$.

(2) Compute a estatística de teste M (expressão 25) e a quantidade *u*, agrupando ambas no cálculo de $C = (1-u)M$.

(3) Defina que ocorreu um *drift* caso $C > \chi^2_{\frac{1}{2}J(J+1)}(\alpha)$

Após a detecção do *drift*, alguma medida precisa ser tomada, como por exemplo, o novo treinamento do classificador. Contudo, tal adequação pode demorar e os erros do classificador ocorridos podem reduzir a confiabilidade do procedimento de reconhecimento de padrões em fluxos de dados. A próxima subseção apresenta a abordagem proativa, cujo objetivo é tentar predizer aproximadamente a classe estimada dos padrões no instante $(t+1)$, a partir de um algoritmo não supervisionado, e então seguir o procedimento de teste de hipótese a partir deste rótulo de classes inferido.

3.1.3.2.

Detecção Proativa

A detecção proativa, isto é, antes que os efeitos do *drift* sejam percebidos pelo modelo, implica na ausência das classes dos padrões no instante $t+1$. Então, seja $\boldsymbol{x_1}(t+1), \ldots, \boldsymbol{x_n}(t+1)$ o conjunto de *n* padrões no instante $t+1$. Normalmente esses padrões são do bloco de dados que será usado para o teste do classificador, e, portanto, aguarda-se no próximo instante a chegada dos seus rótulos. Como não é possível calcular o vetor de médias $\overline{\boldsymbol{X}}_{C_k}(t+1)$ ou matriz de covariância $\boldsymbol{S}_{C_k}(t+1)$ condicionais à classe *k*, torna-se inviável em um primeiro momento a comparação dessas estimativas com as já calculadas em *t* ($\overline{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$).

Logo, faz-se necessário uma técnica que possa usar a informação contida em $\boldsymbol{x_1}(t+1), \ldots, \boldsymbol{x_n}(t+1)$ e propor um conjunto de rótulos para estes padrões. Uma abordagem seria usar o classificador treinado com os padrões de *t* e usá-lo para esta tarefa. Contudo, este classificador, por ter fixado uma fronteira de discriminação entre as classes que pode se alterar bruscamente após o *drift*, tenderá a ser uma escolha viesada para o processo de estimação dos rótulos de cada padrão. Outra abordagem, que não se baseia nos rótulos da classe para compreender sua distribuição é a partir de algoritmos de agrupamento [Everitt et al., 2001].

Os algoritmos para agrupamento de padrões utilizam um processo de aprendizado não supervisionado que, a partir da distribuição dos padrões no

espaço vetorial, forma os grupos a partir de uma noção de similaridade. Há basicamente dois tipos de métodos de agrupamento: os aglomerativos e os divisivos [Kaufman & Rousseeuw, 1990]. Na abordagem para a detecção proativa, os métodos de agrupamento que devem ser explorados são os aglomerativos por três motivos: (*i*) já se sabe de antemão o número de grupos que deverão ser formados (idêntico ao total de classes do problema); (*ii*) a condição inicial para o centroide de cada grupo será o vetor de médias condicionais de cada classe, o que irá facilitar no processo de formação dos grupos e na posterior identificação dos grupos como classes; e (*iii*) tendem a serem computacionalmente mais eficientes do que os métodos de agrupamento divisivos.

Há diversos métodos de agrupamento aglomerativos na literatura (k-means, Gaussian Mixture Model, etc.) [Bezdek et al., 1999; Bishop, 2006]. Como primeira abordagem, foi usado o método de agrupamento *k-means* [MacQueen, 1967]. Esse método é um dos mais clássicos, de fácil implementação e computacionalmente eficiente. Definido o método de agrupamento como o *k-means*, a seguir são dispostos os passos para sua implementação na tarefa de providenciar rótulos para os *n* padrões disponíveis:

1. Primeira base de dados (*t=1*), que possui *n*(t) padrões pertencentes a *K* classes. Esta primeira base de dados é comumente usada para treinar o classificador.
2. A partir dessa primeira base de dados, computa-se as médias condicionais para cada classe considerada no problema $\overline{\boldsymbol{X}}_{C_1}(t), \ldots, \overline{\boldsymbol{X}}_{C_K}(t)$.
3. Próximo instante (*t=t+1*): recepção dos novos *n*(*t*) padrões para serem classificados. Nesse instante faça:
    a. Agrupe os *n*(*t*) padrões usando o algoritmo *k-means*, usando como número de grupos o total de classes do problema (grupos = *K*) e como sugestão de centroides de cada grupo os vetores $\overline{\boldsymbol{X}}_{C_1}(t-1), \ldots, \overline{\boldsymbol{X}}_{C_K}(t-1)$. Opta-se pela distância de Mahalanobis como métrica de dissimilaridade, visando à formação de grupos tanto com formatos esféricos, assim como com formatos mais elípticos [Melnykov & Melnykov, 2014].
    b. Dado um agrupamento, defina os $n_1(t)$ padrões que ficaram próximos ao centroide iniciado em $\overline{\boldsymbol{X}}_{C_1}(t-1)$ como pertencentes à classe 1, os $n_2(t)$ padrões que ficaram próximos ao centroide iniciado em $\overline{\boldsymbol{X}}_{C_2}(t-1)$ como pertencentes a classe 2, e assim sucessivamente.
    c. Neste caso, quando se computar o vetor de médias e matriz de covariâncias condicionais à classe k, estes não são representados por $\overline{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$, mas sim por $\overline{\boldsymbol{X}}_{\hat{C}_k}(t)$ e $\boldsymbol{S}_{\hat{C}_k}(t)$, estimativas baseadas na classe estimada $\hat{C}_k$.

4. Considere por α o nível de significância fixado inicialmente pelo usuário. A detecção proativa de um *drift* abrupto deve-se realizar:
    a. No vetor de médias condicionais:
        i. Calcule $\overline{\boldsymbol{X}}_{\hat{C}_k}(t)$ e $\overline{\boldsymbol{X}}_{C_k}(t-1)$, assim como $\boldsymbol{S}_{\hat{C}_k}(t)$ e $\boldsymbol{S}_{C_k}(t-1)$.
        ii. Compute a estatística de teste T^2 (expressão 23).
        iii. Defina que ocorreu um *drift* caso $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}\,T^2 > F_{J,(n_1+n_2)-J-1}(\alpha)$
    b. Na matriz de covariância condicional:
        i. Calcule $\boldsymbol{S}_{\hat{C}_k}(t)$, $\boldsymbol{S}_{C_k}(t-1)$ e $\boldsymbol{S}_{\boldsymbol{pool}}$.
        ii. Compute a estatística de teste M (expressão 25) e a quantidade *u*, agrupando ambas no cálculo de $C = (1-u)M$.
        iii. Defina que ocorreu um *drift* caso $C > \chi^2_{\frac{1}{2}J(J+1)}(\alpha)$.
5. Retorne ao passo 3 até quando for necessário.

É importante notar que neste método não é necessário guardar os blocos de dados anteriores, mas simplesmente o número de padrões de cada classe, os vetores de média e de matriz de covariância condicional do instante passado. Após a detecção do *drift*, alguma medida precisa ser tomada, como por exemplo, o novo treinamento do classificador ou reajuste dos padrões contidos na fase de teste do classificador. Esse reajuste pode ser realizado tendo como base condicionar estes novos padrões para possuírem uma distribuição similar aos que foram usados no treinamento do classificador. Para exemplificar, considere a seguinte linha de raciocínio:

Suponha um problema com duas classes e dois atributos, além de um conjunto de *n* padrões. Foi usado o método de detecção proativa, e após identificados os padrões que possivelmente pertencem a classe 1 e 2, verificou-se a presença de um *drift* abrupto no vetor de médias condicionais à classe 1. Observou-se um incremento de 10 unidades na média do primeiro atributo em relação ao instante anterior. Para "corrigir o *drift*", isto é, fazer com que este não seja percebido pelo classificador, deve-se subtrair 10 unidades no primeiro atributo dos padrões que se credita a pertinência à classe 1 e, em seguida, utilizar o classificador antigo. Portanto, este processo de "correção do *drift*" é um dos diferenciais do método proposto de detecção proativa de *drift*.

Conforme já mencionado, o método de detecção proposto nesta tese detecta somente *drifts* abruptos. Por este motivo, ele foi denominado DetectA (do inglês, *Detect Abrupt*). No capítulo 5 são apresentados os experimentos realizados com o método DetectA.

4
NEVE: Modelo Neuroevolutivo para Aprendizagem em Ambientes Não Estacionários

Este capítulo apresenta o modelo neuroevolutivo para aprendizagem em ambientes não estacionários e detalha as suas quatro variações implementadas.

4.1.
Visão Geral

Conforme já explicado no Capítulo 1, com o objetivo de propor e desenvolver um modelo auto adaptável, flexível, com boa acurácia e adequado para aprendizado em ambientes não estacionários, foi criado um modelo neuroevolutivo com inspiração quântica, baseado em um comitê de redes neurais do tipo *Multi-Layer Perceptron* (MLP), para a aprendizagem em ambientes não estacionários. Neste modelo, cada rede neural membro do comitê é treinada e tem seus parâmetros (topologia, pesos, entre outros) otimizados através do algoritmo AEIQ-BR (já apresentado no Capítulo 2). Este modelo neuroevolutivo é denominado NEVE (do acrônimo em inglês *Neuro-EVolutionary Ensemble*) e é composto por três módulos principais, detalhados a seguir e ilustrados na Figura 22:

- Módulo de Detecção de *Drift*
- Módulo de Criação de Classificadores
- Módulo de Avaliação e Determinação dos Pesos dos membros do comitê.

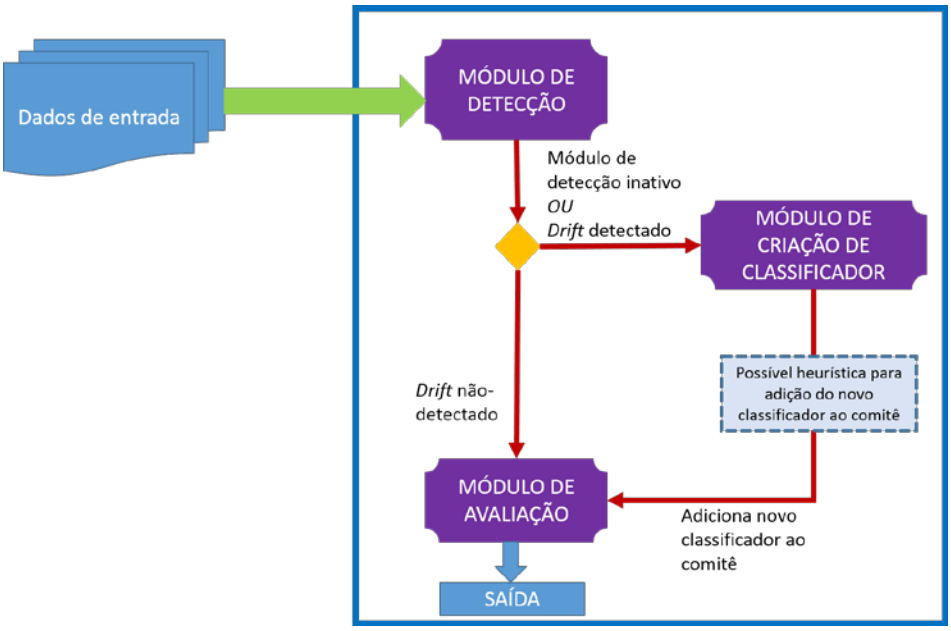


Figura 22. Estrutura Modular do NEVE.

O Módulo de Detecção de *Drift* tem sua implementação opcional, sendo possível ativá-lo ou não. Em caso afirmativo, a cada novo bloco de dados de entrada recebido, o módulo de detecção verifica se foi detectada alguma mudança. Optou-se por trabalhar com a entrada de dados em blocos de tamanho configurável; desta forma, caso seja necessário ou deseje-se trabalhar com a entrada de dados um padrão por vez, basta configurar o tamanho do bloco igual a 1. Entretanto, é importante mencionar que a estratégia de trabalhar com um padrão por vez não é a mais adequada para este modelo, pois pode comprometer o seu desempenho computacional. Duas formas de detecção foram propostas: proativa ou reativa, resultando em quatro abordagens implementadas para este módulo de detecção de *drift*, que serão melhor detalhadas nas próximas subseções:

- Sem detecção
- Detecção reativa
- Detecção proativa, usando a abordagem *Group Label*
- Detecção proativa, usando a abordagem *Pattern Mean Shift*

O Módulo de Criação de Classificadores é responsável por criar um novo classificador, o qual poderá ou não ser adicionado ao comitê, dependendo do tamanho máximo do comitê definido pelo usuário. Vale ressaltar que a decisão de criar uma nova rede está atrelada ao mecanismo de detecção de *drift*

utilizado, o que será melhor detalhado nas subseções a seguir. Caso criada, a nova rede só é adicionada ao comitê caso haja espaço disponível ou em substituição a uma rede antiga de pior acurácia. Isto dá ao comitê a habilidade de aprender o novo conjunto de dados sem precisar analisar os dados antigos, além de permitir o esquecimento dos dados que não são mais necessários. Em suma, o módulo de criação de classificadores determina como cada nova rede MLP membro do comitê será criada, utilizando o algoritmo AEIQ-BR (já apresentado no capítulo 2). O algoritmo irá selecionar as variáveis de entrada mais apropriadas, dentre as variáveis disponíveis no bloco de dados de entrada, irá determinar o número de neurônios da camada escondida (respeitando o limite máximo configurado pelo usuário) e irá determinar os pesos e as funções de ativação de cada neurônio. O número de neurônios de saída será igual ao número de classes do problema.

Finalmente, o Módulo de Avaliação é responsável por determinar a resposta final do comitê de classificadores, através da combinação dos resultados apresentados pelos classificadores membros do comitê. O algoritmo AEIQ-R (também já apresentado no capítulo 2) é utilizado para determinar, dinamicamente, o peso de votação mais adequado para cada classificador. A otimização dos pesos permite ao comitê de classificadores se adaptar facilmente a mudanças bruscas nos dados, através da atribuição de pesos maiores aos classificadores mais bem adaptados aos conceitos correntes que governam os dados. Foram implementadas 3 possibilidades de votação:

- **Combinação Linear:** Utiliza o algoritmo AEIQ-R para gerar um peso de votação para cada classificador, que é multiplicado pela saída de cada neurônio da rede do comitê (entre 0 e 1) em uma média ponderada. O resultado desta média ponderada é utilizado para determinar a resposta do comitê: no caso de problemas com apenas duas classes, a saída é atribuída à classe 0 se o resultado for menor que 0,5 e à classe 1 caso contrário; no caso de problemas com múltiplas classes, a classe será aquela que apresentar a saída com maior valor;
- **Votação Majoritária Ponderada:** Da mesma forma que no caso anterior, utiliza o algoritmo AEIQ-R para gerar um peso de votação para cada classificador. Entretanto, neste caso, as saídas dos neurônios de cada rede do comitê são, primeiramente, arredondadas (para os valores 0 ou 1) e, em seguida, multiplicadas

pelo peso da rede correspondente, formando assim uma média ponderada. Assim como na combinação linear, no caso de problemas com apenas duas classes, a saída é definida como classe 0 se o resultado da média ponderada for menor que 0,5 e como classe 1 caso contrário; no caso de problemas com múltiplas classes, define-se a classe associada à saída com maior valor;

- **Votação Majoritária Simples:** a saída de cada rede do comitê é arredondada para uma das possíveis classes de saída do problema e a saída final do comitê é a classe mais escolhida dentre todas as redes. Neste caso, não há necessidade de determinação de pesos de votação.

Em resumo, considerando o mecanismo de detecção utilizado, são quatro as possíveis variações do modelo NEVE propostas e detalhadas nas subseções a seguir:

- NEVE, sem detecção
- DE-NEVE, com detecção reativa
- DE-NEVE, com detecção proativa e a abordagem *Group Label*
- DE-NEVE, com detecção proativa e a abordagem *Pattern Mean Shift*

As seções a seguir detalham cada uma das quatro variações do NEVE propostas. Para cada variação, será apresentado um fluxograma para ilustrar seu funcionamento, seguido por um texto explicativo e pelo pseudocódigo do seu algoritmo.

4.2. NEVE (Sem Detecção)

A primeira variação do modelo NEVE será denominada, de agora em diante, de “NEVE Sem Detecção”. Como o nome indica, esta variação não utiliza mecanismo de detecção e consiste em um comitê de redes neurais MLP que, a cada novo bloco de dados recebido, treina uma nova MLP que será adicionada ao comitê se houver espaço disponível. O funcionamento do “NEVE Sem Detecção” pode ser ilustrado pela Figura 23 e resumido conforme os passos a seguir.

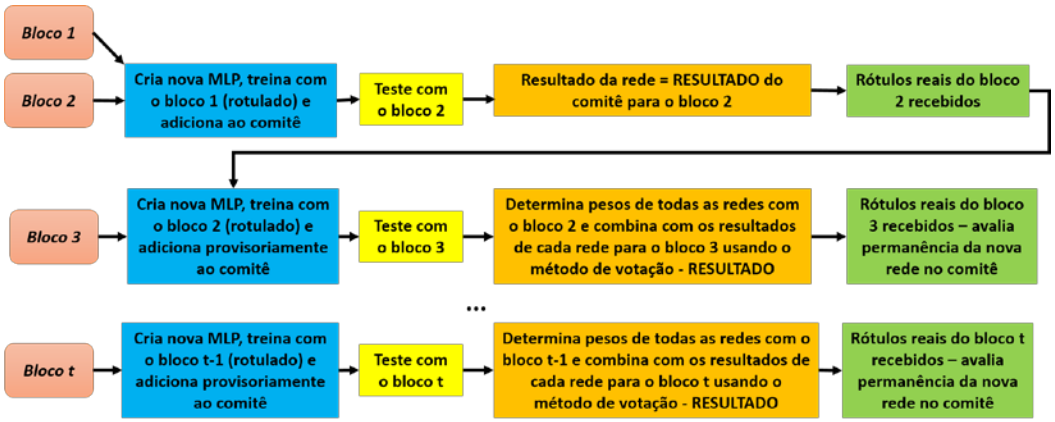


Figura 23. Estrutura do NEVE Sem Detecção.

O processo se inicia com a chegada dos dois primeiros blocos de dados, identificados respectivamente por bloco 1 e 2. Assume-se que o bloco 1 é rotulado e o bloco 2 é não-rotulado e então o algoritmo AEIQ-BR cria uma nova rede MLP treinada com o bloco 1 com os rótulos de classe reais e esta rede é adicionada ao comitê, que é testado com o bloco 2. Como há apenas uma rede no comitê, não há necessidade da determinação de pesos de votação, e o resultado final do comitê é o próprio resultado da rede, testada com o bloco 2. Finalmente, assume-se que os rótulos reais do bloco 2 tornam-se disponíveis.

Ao chegar o bloco 3, é criada uma nova rede MLP usando para treinamento o bloco 2 com os rótulos de classe reais. A nova rede é provisoriamente adicionada ao comitê e este é testado com o bloco 3. Os pesos de votação de todas as redes são determinados usando o bloco 2 e combinados com os resultados de teste de cada rede com o bloco 3 usando o método de votação escolhido para determinar o resultado final do comitê. Finalmente, assume-se que os rótulos reais do bloco 3 tornam-se disponíveis e, então, a permanência da nova rede no comitê é avaliada: ela permanece no comitê se houver espaço disponível ou se for melhor do que pelo menos uma das redes antigas, substituindo-a no comitê.

Assim, o funcionamento desta variação do modelo pode ser generalizado em: ao chegar o bloco t, é criada uma nova rede MLP usando o algoritmo AEIQ-BR e o bloco t-1 com os rótulos de classe reais. A nova rede é provisoriamente adicionada ao comitê e este é testado com o bloco t. Os pesos de votação de todas as redes são determinados usando o algoritmo AEIQ-R e o bloco t-1 e então combinados com os resultados de teste de cada rede com o bloco t usando o método de votação escolhido para determinar o resultado final do

comitê. Finalmente, assume-se que os rótulos reais do bloco t tornam-se disponíveis e então, a permanência da nova rede no comitê é avaliada.

O pseudocódigo do NEVE Sem Detecção é demonstrado na Figura 24:

```
Criar um comitê de classificadores vazio P
1. Definir o tamanho do comitê s
2. Criar um novo classificador MLP c' usando o bloco de dados
   D1 e o algoritmo quântico e adicioná-lo ao comitê
3. Testar c' com o bloco de dados D2
4. Decisão final do comitê = Decisão de c'
5. Receber rótulos reais do bloco de dados D2
6. Para cada bloco de dados Di; i = 3, 4..., m faça
6.1.  Criar um novo classificador MLP e treiná-lo usando o
      bloco de dados Di-1 e o algoritmo quântico
6.2.  Adicionar o novo classificador provisoriamente ao comitê
6.3.  Testar cada rede do comitê com o bloco de dados Di
6.4.  Evoluir os pesos de votação wj para cada classificador
      usando o último bloco de dados Di-1
6.5.  Determinar a decisão final do comitê usando o método de
      votação escolhido
6.6.  Receber rótulos reais do bloco de dados Di
6.7.  Calcular o erro de classificação E' para o novo
      classificador c' e Ej para cada classificador cj do
      comitê e o bloco de dados Di
6.8.  Calcular o erro de classificação do comitê
6.9.  Se o comitê estiver cheio (número de classificadores = s)
      então
   i)    Se (E' < max(Ej))
         a. Substituir o classificador com max(Ej) pelo novo
            classificador
```

Figura 24. Pseudocódigo do algoritmo NEVE Sem Detecção.

4.3. DE-NEVE com Detecção Reativa

A segunda variação do modelo NEVE foi denominada de “DE-NEVE Reativo”. Esta variação usa o mecanismo reativo de detecção, já detalhado no capítulo 3 (seção 3.1.3.1). A cada novo bloco de dados recebido, o comitê faz a classificação e, assim que seus rótulos de classe reais forem recebidos, o mecanismo de detecção verifica se houve ou não *drift* em relação ao bloco anterior. Em caso afirmativo, é criada uma nova MLP, que é adicionada ao comitê se houver espaço disponível. O funcionamento do “DE-NEVE Reativo” pode ser ilustrado pela Figura 25 e resumido conforme os passos a seguir.

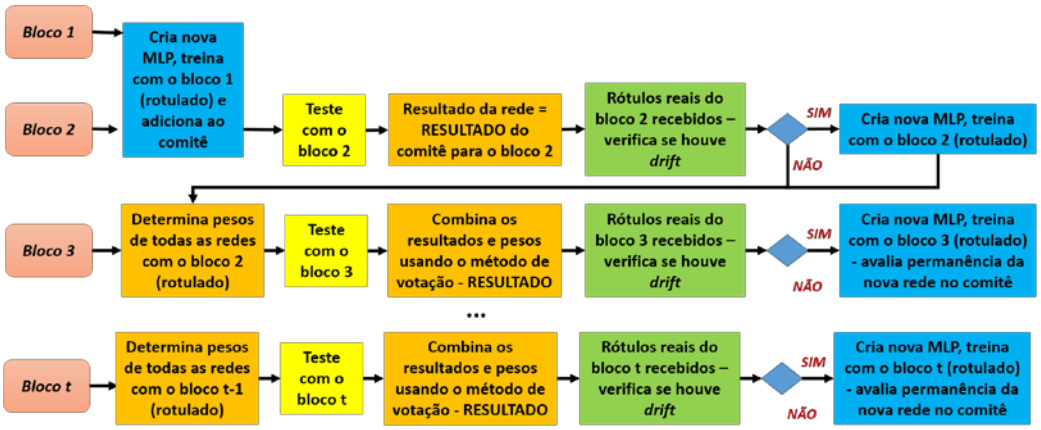


Figura 25. Estrutura do DE-NEVE Reativo.

O processo se inicia com a chegada dos dois primeiros blocos de dados, identificados respectivamente por bloco 1 e 2. Assume-se que o bloco 1 é rotulado e o bloco 2 é não-rotulado e então o algoritmo AEIQ-BR cria uma nova rede MLP treinada com o bloco 1 com os rótulos de classe reais e esta rede é adicionada ao comitê, que é testado com o bloco 2. Como há apenas uma rede no comitê, não há necessidade da determinação de pesos de votação, e o resultado final do comitê é o próprio resultado da rede, testada com o bloco 2. Assume-se que os rótulos reais do bloco 2 tornam-se disponíveis e, então, verifica-se se houve *drift* no bloco 2 em relação ao bloco 1. Em caso afirmativo, é criada uma nova rede MLP usando o algoritmo escolhido, treinada com o bloco 2 com os rótulos de classe reais. A permanência da nova rede é avaliada: ela é adicionada ao comitê se houver espaço disponível ou se for melhor do que pelo menos uma das redes antigas, substituindo-a no comitê.

Assim, o funcionamento desta variação do modelo pode ser generalizado da seguinte forma: ao chegar o bloco t, os pesos de votação de todas as redes do comitê são determinados usando o algoritmo AEIQ-R e o bloco t-1. O comitê é testado com o bloco t e os resultados são combinados com os pesos usando o método de votação escolhido para determinar o resultado final do comitê. Assume-se que os rótulos reais do bloco t já estão disponíveis e então, verifica-se se houve *drift* no bloco t em relação ao bloco t-1. Em caso afirmativo, é criada uma nova rede MLP usando o algoritmo AEIQ-BR e treinada a partir do bloco t com os rótulos de classe reais. A nova rede é adicionada ao comitê se houver espaço disponível ou se for melhor do que pelo menos uma das redes antigas, substituindo-a no comitê.

O pseudocódigo do DE-NEVE Reativo é demonstrado na Figura 26:

```
Criar um comitê de classificadores vazio P
1. Definir o tamanho do comitê s
2. Criar um novo classificador MLP c' usando o bloco de dados
   D1 e o algoritmo quântico e adicioná-lo ao comitê
3. Testar c' com o bloco de dados D2
4. Decisão final do comitê = Decisão de c'
5. Receber rótulos reais do bloco de dados D2
6. Se for detectado drift entre os blocos D1 e D2
6.1.  Criar um novo classificador MLP e treiná-lo usando o
      bloco de dados D2 e o algoritmo quântico
7. Para cada bloco de dados Di; i = 3, 4,..., m faça
7.1.  Evoluir os pesos de votação wj para cada classificador
      usando o último bloco de dados Di-1
7.2.  Testar cada rede do comitê com o bloco de dados Di
7.3.  Determinar a decisão final do comitê usando o método de
      votação escolhido
7.4.  Receber rótulos reais do bloco de dados Di
7.5.  Se for detectado drift entre os blocos Di-1 e Di
7.5.1.   Criar um novo classificador MLP e treiná-lo usando o
      bloco de dados Di e o algoritmo quântico
7.5.2.   Adicionar o novo classificador provisoriamente ao
      comitê
7.5.3.   Calcular o erro de classificação E' para o novo
      classificador c' e Ej para cada classificador cj do
      comitê e o bloco de dados Di
7.5.4.   Se o comitê estiver cheio (número de classificadores =
      s) então
   i)    Se (E' < max(Ej))
      a. Substituir o classificador com max(Ej) pelo novo
         classificador
7.6.  Calcular o erro de classificação do comitê
```

Figura 26. Pseudocódigo do algoritmo DE-NEVE Reativo.

4.4. DE-NEVE com Detecção Proativa, Estratégia *Group Label*

A terceira variação do modelo NEVE foi denominada de "DE-NEVE Proativo *Group Label*". Esta variação usa o mecanismo proativo de detecção, já detalhado no capítulo 3 (seção 3.1.3.2). Neste caso, a cada novo bloco de dados recebido, é realizado um agrupamento, usando como sugestão de centroides o bloco anterior rotulado, para determinar as classes previstas de cada padrão do novo bloco. Usando este agrupamento, o mecanismo de detecção verifica se houve *drift* em relação ao bloco anterior e, em caso afirmativo, cria uma nova MLP treinada com o novo bloco e os rótulos de classe sugeridos no agrupamento. O funcionamento do "DE-NEVE Proativo *Group Label*" pode ser ilustrado pela Figura 27 e resumido conforme os passos a seguir.

O processo se inicia com a chegada do primeiro bloco de dados, identificado por bloco 1. Assume-se que este bloco é rotulado e então é

realizado o agrupamento dos dados deste bloco utilizando os rótulos de classe reais. O algoritmo AEIQ-BR cria uma nova rede MLP treinada com este bloco e a rede é adicionada ao comitê.

Ao chegar o bloco 2, é realizado o agrupamento dos dados deste bloco utilizando como sugestão inicial de centroides as médias condicionais das classes do bloco 1, uma vez que os rótulos reais de classes do bloco 2 ainda não são conhecidos. Neste momento, verifica-se se houve *drift* no bloco 2 em relação ao bloco 1. Em caso afirmativo, é criada uma nova rede MLP treinada com o bloco 2 com os rótulos de classe previstos no agrupamento realizado. A nova rede é provisoriamente adicionada ao comitê. O comitê é testado com o bloco 2. Os pesos de votação de todas as redes são determinados usando o algoritmo AEIQ-R e o bloco 2 com os rótulos previstos no agrupamento. Os resultados são combinados com os pesos usando o método de votação escolhido para determinar o resultado final do comitê. Assume-se que os rótulos reais do bloco 2 já estão disponíveis e as sugestões de centroides para o próximo agrupamento são atualizadas, considerando agora os rótulos de classe reais do bloco 2. A permanência da nova rede no comitê é avaliada: ela é adicionada ao comitê se houver espaço disponível ou se for melhor do que pelo menos uma das redes antigas, substituindo-a no comitê.

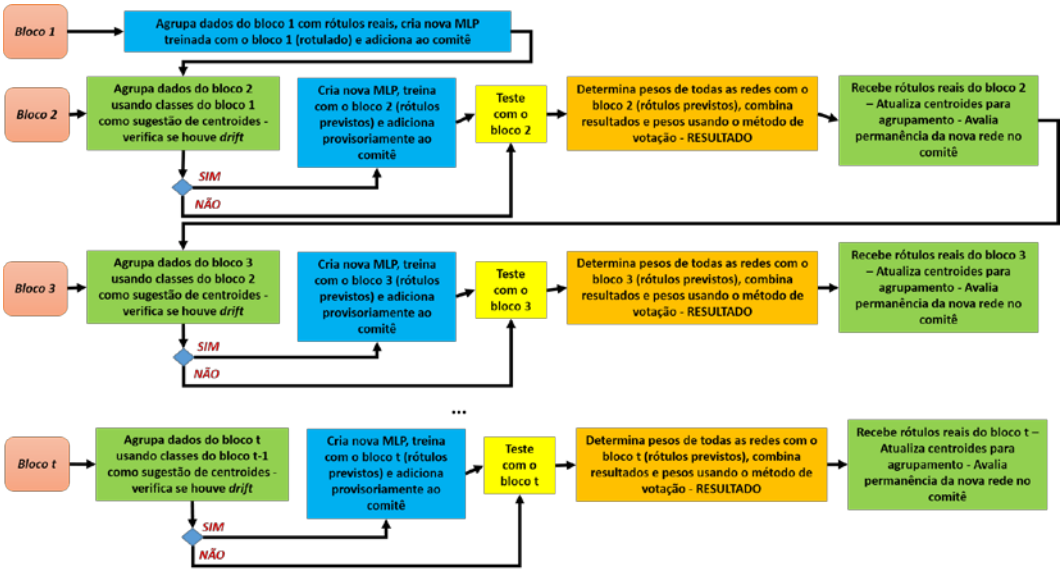


Figura 27. Estrutura do DE-NEVE Proativo *Group Label*.

Assim, o funcionamento desta variação do modelo pode ser generalizado como: ao chegar o bloco t, é realizado o agrupamento dos dados do bloco utilizando como sugestão inicial de centroides as classes do bloco t-1, já que os

rótulos reais de classes do bloco t ainda não são conhecidos. Neste momento, verifica-se se houve *drift* no bloco t em relação ao bloco t-1. Em caso afirmativo, é criada uma nova rede MLP com o algoritmo AEIQ-BR a partir do bloco t com os rótulos de classe previstos no agrupamento. A nova rede é provisoriamente adicionada ao comitê, que é testado com o bloco t. Os pesos de votação de todas as redes são determinados usando o algoritmo AEIQ-R e o bloco t com os rótulos previstos no agrupamento. Os resultados são combinados com os pesos usando o método de votação escolhido para determinar o resultado final do comitê. Assume-se que os rótulos reais do bloco t já estão disponíveis e as sugestões de centroides para o próximo agrupamento são atualizadas, considerando agora os rótulos de classe reais do bloco. A permanência da nova rede no comitê é avaliada: ela só se mantém no comitê se houver espaço disponível ou se for melhor do que pelo menos uma das redes antigas, substituindo-a no comitê. O pseudocódigo do DE-NEVE Proativo *Group Label* é demonstrado na Figura 28:


```
Criar um comitê de classificadores vazio P
1. Definir o tamanho do comitê s
2. Criar um novo classificador MLP usando o bloco de dados D1 e
   o algoritmo quântico e adicioná-lo ao comitê
3. Para cada bloco de dados Di; i = 2, 3,..., m faça
3.1.  Agrupar dados de Di usando as classes de Di-1 como
      sugestão de centroides
3.2.  Se for detectado drift entre os blocos Di-1 e Di
3.2.1.   Criar um novo classificador MLP e treiná-lo usando o
      bloco de dados Di com os rótulos previstos no agrupamento
      e o algoritmo quântico
3.2.2.   Adicionar o novo classificador provisoriamente ao
      comitê
3.3.  Testar cada rede do comitê com o bloco de dados Di
3.4.  Evoluir os pesos de votação wj para cada classificador
      usando o último bloco de dados Di com os rótulos
      previstos no agrupamento
3.5.  Determinar a decisão final do comitê usando o método de
      votação escolhido
3.6.  Receber rótulos reais do bloco de dados Di
3.7.  Calcular o erro de classificação E' para o novo
      classificador c' e Ej para cada classificador cj do
      comitê e o bloco de dados Di
3.8.  Atualizar as sugestões de centroides para o próximo
      agrupamento usando o bloco Di e rótulos reais
3.9.  Se foi criado novo classificador e se o comitê estiver
      cheio (número de classificadores = s) então
   i)    Se (E' < max(Ej))
      a. Substituir o classificador com max(Ej) pelo novo
         classificador
```

Figura 28. Pseudocódigo do algoritmo DE-NEVE Proativo *Group Label*.

4.5.
DE-NEVE com Detecção Proativa, Estratégia *Pattern Mean Shift*

A quarta variação do modelo NEVE foi denominada de "DE-NEVE Proativo *Pattern Mean Shift*". Esta variação usa o mecanismo proativo de detecção, já detalhado no capítulo 3 (seção 3.1.3.2). Assim, como na variação anterior, a cada novo bloco de dados recebido, é realizado um agrupamento, usando como sugestão de centroides o bloco anterior rotulado para determinar as classes previstas de cada padrão do novo bloco. O mecanismo de detecção verifica se houve *drift* em relação ao bloco anterior e, em caso afirmativo, cria uma nova MLP treinada com o bloco anterior rotulado, e o novo bloco é "ajustado" em direção ao bloco anterior. Ou seja, quando detectado um *drift*, em vez de criar uma nova MLP usando o novo bloco de dados (como feito na estratégia *Group Label*), é utilizado o bloco antigo para treinamento da rede e o *drift* ocorrido é "retirado" do novo bloco de dados. Enquanto na estratégia *Group Label* a nova rede é adequada aos novos dados, na estratégia *Pattern Mean Shift* os novos dados é que são adequados à rede treinada com os dados antigos. O funcionamento do "DE-NEVE Proativo *Pattern Mean Shift*" pode ser ilustrado pela Figura 29 e resumido conforme os passos a seguir.

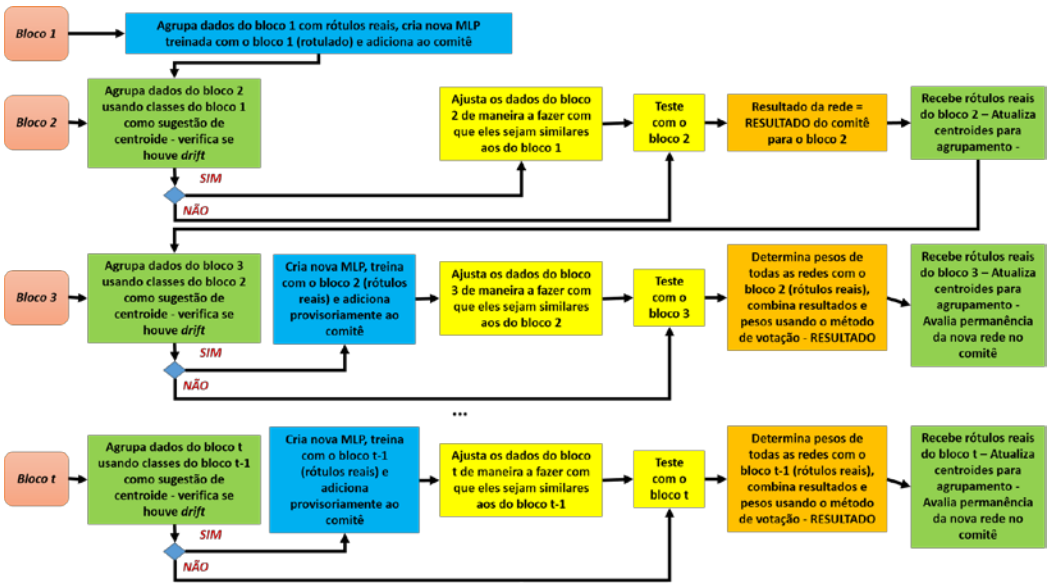


Figura 29. Estrutura do DE-NEVE Proativo *Pattern Mean Shift*.

Assim, o funcionamento desta variação do modelo pode ser generalizado como a seguir: ao chegar o bloco t, é realizado o agrupamento dos dados deste bloco utilizando como sugestão inicial de centroides as classes do bloco t-1, uma

vez que os rótulos reais de classes do bloco t ainda não são conhecidos. Neste momento, verifica-se se houve *drift* no bloco t em relação ao bloco t-1. Em caso afirmativo, é criada uma nova rede MLP usando o algoritmo AEIQ-BR e o bloco t-1 com os rótulos reais e a nova rede é provisoriamente adicionada ao comitê. Neste caso, os dados do bloco t são ajustados de maneira a fazer com que eles sejam similares aos do bloco t-1. O comitê é testado com o bloco t. Os pesos de votação de todas as redes são determinados usando o algoritmo AEIQ-R e o bloco t-1 com os rótulos reais. Os resultados são combinados com os pesos usando o método de votação escolhido para determinar o resultado final do comitê. Assume-se que os rótulos reais do bloco t já estão disponíveis e as sugestões de centroides para o próximo agrupamento são atualizadas, considerando agora os rótulos de classe reais do bloco t. A permanência da nova rede no comitê é avaliada: ela só se mantém no comitê se houver espaço disponível ou se for melhor do que pelo menos uma das redes antigas, substituindo-a no comitê. O pseudocódigo do DE-NEVE Proativo *Pattern Mean Shift* é demonstrado na Figura 30:


```
Criar um comitê de classificadores vazio P
1. Definir o tamanho do comitê s
2. Criar um novo classificador MLP usando o bloco de dados D_1 e
   o algoritmo quântico e adicioná-lo ao comitê
3. Para cada bloco de dados D_i; i = 2, 3,..., m faça
3.1.  Agrupar dados de D_i usando as classes de D_i-1 como
      sugestão de centroides
3.2.  Se for detectado drift entre os blocos D_i-1 e D_i
3.2.1.   Criar um novo classificador MLP e treiná-lo usando o
      bloco de dados D_i-1 com os rótulos reais e o algoritmo
      quântico
3.2.2.   Adicionar o novo classificador provisoriamente ao
      comitê
3.2.3.   Ajustar os dados do bloco D_i de maneira a fazer com
      que eles sejam similares aos do bloco D_i-1
3.3.  Testar cada rede do comitê com o bloco de dados D_i
3.4.  Evoluir os pesos de votação w_j para cada classificador
      usando o bloco de dados D_i-1 com os rótulos reais
3.5.  Determinar a decisão final do comitê usando o método de
      votação escolhido
3.6.  Receber rótulos reais do bloco de dados D_i
3.7.  Calcular o erro de classificação E' para o novo
      classificador c' e E_j para cada classificador c_j do
      comitê e o bloco de dados D_i
3.8.  Atualizar as sugestões de centroides para o próximo
      agrupamento usando o bloco D_i e rótulos reais
3.9.  Se foi criado novo classificador e se o comitê estiver
      cheio (número de classificadores = s) então
   i)    Se (E' < max(E_j))
      a. Substituir o classificador com max(E_j) pelo novo
         classificador
```

Figura 30. Pseudocódigo do algoritmo DE-NEVE Proativo *Pattern Mean Shift*.

Resumidamente, a principal diferença do “DE-NEVE Proativo *Pattern Mean Shift*” para o “DE-NEVE Proativo *Group Label*” é que, quando um *drift* é detectado, é criada uma nova MLP usando o bloco anterior rotulado (e não o novo bloco com os rótulos previstos no agrupamento, como na abordagem *Group Label*). Em seguida, nesta abordagem, os dados do novo bloco são “ajustados” na direção do bloco anterior e submetidos à classificação do comitê (enquanto que na abordagem *Group Label*, o novo bloco é testado pelo comitê sem ajustes). Além disso, nesta abordagem, o bloco de dados usado para determinação dos pesos de cada MLP do comitê é o bloco anterior com os rótulos reais, enquanto na abordagem *Group Label* é usado o novo bloco com os rótulos previstos pelo agrupamento.

Este capítulo apresentou o modelo neuroevolutivo para aprendizagem em ambientes não estacionários proposto nesta tese e detalhou as suas quatro variações. O próximo capítulo detalha os experimentos realizados com o método de detecção proposto.

5
Experimentos com o Método de Detecção Proposto

Este capítulo detalha os experimentos realizados com o método de detecção DetectA, já apresentado no Capítulo 3, e apresenta seus resultados. Primeiramente, é apresentado o *Experimento 1*, que consiste em uma análise de sensibilidade do método de detecção proposto em relação às características da base de dados. Em seguida, é apresentado o *Experimento 2*, no qual o método de detecção é testado com diferentes bases de dados reais e artificiais.

5.1.
Experimento 1: Análise de Sensibilidade do Método de Detecção

Neste experimento, foi realizada uma análise de sensibilidade do mecanismo de detecção de *drift* proposto com base na alteração do número de atributos, número de padrões, taxa de desbalanceamento entre classes, entre outros parâmetros, a fim de compreender a influência de cada uma destas variáveis no desempenho do método e aprimorá-lo para que possa ser utilizado em modelos de aprendizagem em ambientes não estacionários.

Para tal, primeiramente foi realizada a geração de um conjunto de bases de dados artificiais, com a possibilidade de se manipular o momento da ocorrência e o tipo do *drift.* Estas bases de dados foram então utilizadas para avaliar como as variações nas suas características (número de atributos, padrões, etc.) podem influenciar na eficácia do método de detecção. O procedimento detalhado de geração das bases de dados pode ser encontrado no Anexo 2.

A eficácia de um método de detecção pode ser medida pela sua taxa de falsos positivos (alarmes falsos) e falsos negativos (alarmes defeituosos). Ambas as medidas são usadas para avaliar o método de detecção proposto. Pode-se também avaliar o tempo de atraso entre a ocorrência de um *drift* e a sua detecção, entretanto, tal medida não será considerada nesta análise, pois o número de blocos a serem usados na prova de conceito e avaliação do método de detecção é reduzido (10 blocos). A matriz de confusão do processo de detecção e ocorrência de *drifts* pode ser resumida na Tabela 2.

Tabela 2. Matriz de confusão para o processo de detecção de *drift*.

Detector/Drift	*Ausência de Drift (ND)*	*Presença de Drift (D)*
Não Alertou (NA)	#NAeND	#NAeD
Alertou (A)	#AeND	#AeD

A taxa de falsos positivos ou alarmes falsos é calculada por:

$$FP = \frac{\#AeND}{\#AeND+\#NAeND} \quad (27)$$

medindo, portanto, a relação entre o número de casos em que os alertas foram realizados de forma equivocada e o número total de casos sem *drift*. Por outro lado, a taxa de falsos negativos ou alarmes defeituosos é dada por:

$$FN = \frac{\#NAeD}{\#NAeD+\#AeD} \quad (28)$$

calculando a relação entre o número de erros cometidos pela não detecção de *drifts* ($\#NAeD$) e o número total de casos que possuíam *drift*. Um bom detector é o que minimiza ambas as quantidades. Para simplificar o experimento, foram geradas bases de dados binárias, ou seja, com duas classes possíveis (classe 1 e classe 2).

Deseja-se então mensurar a taxa de alarmes falsos e alarmes defeituosos ocorridos tanto para a classe 1 quanto para a classe 2. Desta forma, são quatro os indicadores de desempenho de interesse do método de detecção para este experimento, representados na Tabela 3.

Tabela 3. Indicadores de acurácia do método de detecção.

Indicador de Acurácia	Descrição
ALARME_FALSO_CLASSE1	Taxa de alarmes falsos (falsos positivos) na classe 1
ALARME_DEFEITUOSO_CLASSE1	Taxa de alarmes defeituosos (falsos negativos) na classe 1
ALARME_FALSO_CLASSE2	Taxa de alarmes falsos (falsos positivos) na classe 2
ALARME_DEFEITUOSO_CLASSE2	Taxa de alarmes defeituosos (falsos negativos) na classe 2

A análise de sensibilidade do método de detecção envolveu os seguintes parâmetros descritos na Tabela 4.

Tabela 4. Parâmetros analisados na análise de sensibilidade.

Parâmetro	Descrição	Valores usados
num_atrib	Número de atributos da base de dados	5, 15 e 25
num_pat	Número de padrões por bloco ou tamanho do bloco	150, 350 e 500
imb_rate	Proporção da ocorrência da classe 1 em relação a classe 2	0.2, 0.35 e 0.5
num_drift_classe1	Número de blocos onde há a ocorrência de *drift* na classe 1	1, 3, 5, 7
num_drift_classe2	Número de blocos onde há a ocorrência de *drift* na classe 2	0, 1, 3, 5, 7
num_atrib_drift	Proporção de atributos que sofrerão *drift* dentro do bloco	0.2, 0.35 e 0.5
alpha	Nível de significância, ou seja, nível mínimo que se aceita da hipótese alternativa H1 (ocorrência de *drift*) estar correta	0.01, 0.05 e 0.1

A seguir discute-se cada um dos parâmetros selecionados para a análise de sensibilidade, assim como algumas hipóteses sobre como os indicadores devem influenciar no desempenho do modelo proposto.

- **Número de atributos:** o número de atributos afeta o método de detecção em dois sentidos: no cálculo dos graus de liberdade para o teste estatístico e na quantidade de variações que se efetuam nas médias dos atributos. A hipótese é que quanto maior o número de atributos, mais reativo tenderá a ser o método de detecção, pois pequenas variações sentidas em muitos atributos são mais significativas do que as mesmas em poucos atributos.
- **Número de padrões:** o número de padrões é o conjunto de evidências que são usadas para se avaliar a presença de um possível distúrbio, ou seja, é o tamanho do bloco. Quanto maior o número de padrões, menor a probabilidade de se ter um alarme falso ou defeituoso.
- **Proporção de desbalanceamento entre classes:** como se está avaliando as médias condicionais dos atributos, a desigualdade na proporção de padrões entre as classes pode facilitar a detecção de um *drift* na classe majoritária em comparação à classe menos favorecida (neste caso, em geral a classe majoritária é a classe 2). Quanto mais equilibrado, mais similares os resultados de alarmes falsos e defeituosos entre as classes, e quanto menos equilibrado,

tais taxas devem se tornar mais acentuadas na classe menos favorecida.

- **Número de blocos com *drift*:** espera-se, obviamente, que quanto mais blocos com *drift* menor tenderá a ser a taxa de alarmes falsos (pois há menos blocos em que não há *drifts*).
- **Proporção de atributos que sofrerão *drift* dentro do bloco:** esse parâmetro define a quantidade de atributos que podem sofrer *drift* nas suas respectivas médias condicionais. A hipótese é de que quanto maior a proporção de atributos sofrendo *drift*, mais fácil é a tarefa de detecção por parte do algoritmo.
- ***Alpha*:** o nível de significância é o limiar estabelecido para que, em uma comparação entre blocos, a diferença (em termos de média dos atributos) seja tida como significativa. Portanto, quanto menor o *alpha*, maior devem ser as diferenças entre os blocos para que o método de detecção declare a presença de um *drift* abrupto na média. Em outros termos, quanto menor *alpha*, menor tenderá a ser a taxa de alarmes falsos, mas, por outro lado, maior tenderá a ser a taxa de alarmes defeituosos.

A partir dessas hipóteses, as próximas seções apresentam os resultados tendo em conta as análises individuais e interações entre os parâmetros. As análises individuais avaliam o quanto as variações de nível em um determinado parâmetro (por exemplo, passar de 5 para 15 atributos) afetam os indicadores de desempenho dispostos na Tabela 3. Por outro lado, as análises de interações vão além, observando variações mútuas de níveis entre dois parâmetros, e suas possíveis interdependências ou potencialização no desempenho do método de detecção.

5.1.1. Análises Individuais

As análises individuais a seguir têm o objetivo de avaliar a influência de cada uma das variáveis da Tabela 4 nos indicadores de desempenho apresentados na Tabela 3. Para tal, decidiu-se usar gráficos do tipo *boxplot*, pois além de exibir diversas estatísticas descritivas (como a mediana, quartis, mínimo e máximo) este tipo de gráfico possibilita a identificação da ocorrência de *outliers*. Estes gráficos, bem como a análise detalhada dos mesmos, podem ser

encontrados no Anexo 3. A Tabela 5 consolida estas informações, exibindo a média de cada um dos quatro indicadores de desempenho agrupada por cada possível valor de cada um dos parâmetros analisados.Em todos os casos, o desvio padrão observado foi inferior a 2%.

Tabela 5. Média dos indicadores agrupada por cada possível valor dos parâmetros.

	Número de atributos			Número de Padrões			Desbalanceam. entre Classes			Alpha		
	5	15	25	150	350	500	0.2	0.35	0.5	0.01	0.05	0.1
ALARME_FALSO_CLASSE1	0.48	0.09	0.08	0.19	0.22	0.24	0.25	0.21	0.19	0.15	0.2	0.31
ALARME_DEFEITUOSO_CLASSE1	0.21	0.44	0.39	0.38	0.34	0.32	0.34	0.35	0.35	0.75	0.21	0.08
ALARME_FALSO_CLASSE2	0.49	0.10	0.09	0.20	0.24	0.26	0.23	0.22	0.23	0.14	0.23	0.32
ALARME_DEFEITUOSO_CLASSE2	0.11	0.21	0.19	0.18	0.16	0.16	0.16	0.17	0.17	0.38	0.09	0.03
	Número de Blocos com *Drift* na Classe 1				Número de Blocos com *Drift* na Classe 2					Proporção de Atributos com Drift		
	1	3	5	7	0	1	3	5	7	0.2	0.35	0.5
ALARME_FALSO_CLASSE1	0.25	0.24	0.22	0.17	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22
ALARME_DEFEITUOSO_CLASSE1	0.34	0.34	0.35	0.35	0.35	0.35	0.34	0.34	0.34	0.35	0.35	0.34
ALARME_FALSO_CLASSE2	0.26	0.24	0.22	0.2	0.27	0.21	0.20	0.18	0.15	0.23	0.23	0.23
ALARME_DEFEITUOSO_CLASSE2	0.17	0.17	0.17	0.17	0.00	0.33	0.34	0.34	0.34	0.17	0.17	0.16

Analisando a Tabela 5, as seguintes observações podem ser verificadas sobre a influência da variação dos parâmetros nas taxas de alarmes falsos e defeituosos:

Número de atributos

- Quanto maior o número de atributos, a taxa de alarmes falsos tende a diminuir, indicando que ela vai se estabilizando quando o número de atributos é igual a 15, alcançando valores menores que 10% a partir daí. Ou seja, há uma indicação que mais atributos produzem melhores resultados quanto a alarmes falsos.
- Para os indicadores de alarmes defeituosos, a taxa de alarmes defeituosos não segue uma relação linear com o número de atributos. Ainda assim, observa-se que as taxas mais baixas ocorrem quando número de atributos é igual a 5.

Número de Padrões

- Observa-se que ao aumentar o número de padrões (tamanho do bloco), a média das taxas de alarmes falsos também aumenta,

ainda que de forma pouco expressiva. Entretanto, a análise dos *boxplots* do Anexo 3 mostram que a mediana, pelo contrário, vai diminuindo, também de forma pouco expressiva.

- Analisando tanto a média quanto a mediana, observa-se que a taxa de alarmes defeituosos tende a diminuir à medida que o número de padrões aumenta, confirmando a hipótese anterior.

Taxa de desbalanceamento entre classes

- No geral, a variação não segue um padrão fixo e é pouco expressiva; porém, a taxa de alarmes falsos para a classe 1 parece estar diminuindo à medida em que se aumenta a taxa de desbalanceamento.

Número de Blocos com *Drifts* na classe 1

- Ainda que de forma pouco expressiva, os valores das taxas de alarmes falsos tendem a diminuir à medida que se aumenta o número de blocos com *drift* na classe 1, enquanto que os valores das taxas de alarmes defeituosos permanecem praticamente constantes.

Número de Blocos com *Drift* na Classe 2

- A taxa de alarmes falsos para a classe 2 é a única que apresenta variação expressiva, diminuindo à medida em que diminui o número de blocos com *drift* na classe 2.

Proporção de Atributos com Drift

- A variação da média para as quatro taxas é mínima ou nula. Dentre as quatro taxas, a que apresentou pior média foi a taxa de alarmes defeituosos para a classe 1 (em torno de 35%) enquanto que as demais taxas ficaram com boa média, em torno de 20%.

Alpha

- Percebe-se que as taxas de alarmes falsos sobem à medida que *alpha* sobe, o que já era esperado, pois maiores valores de *alpha* tornam o modelo fica mais reativo, havendo maior probabilidade de alarmes falsos. Já as taxas de alarmes defeituosos descem à medida que *alpha* sobe, o que também reforça a hipótese anterior.

A análise individual dos parâmetros indica algumas influências que a sua variação pode provocar nas taxas de alarmes falsos e defeituosos. No geral, os resultados foram bons, principalmente para as taxas de alarmes falsos: a média foi 22% para a classe 1 e 23% para a classe 2 e a mediana foi 14% para ambas as classes, indicando que o modelo de detecção proposto tem baixa ocorrências de alarmes falsos. A taxa de alarmes defeituosos também foi consideravelmente baixa, com mediana de 19%. A Tabela 6 apresenta a média, mediana, variância e desvio padrão para os quatro indicadores.

Tabela 6. Estatísticas por indicadores.

	Média	Mediana	Variância	Desvio Padrão
ALARME_FALSO_CLASSE1	0.22	0.14	0.05	0.22
ALARME_DEFEITUOSO_CLASSE1	0.35	0.19	0.12	0.34
ALARME_FALSO_CLASSE2	0.23	0.14	0.05	0.22
ALARME_DEFEITUOSO_CLASSE2	0.17	0.00	0.09	0.30

Esta subseção apresentou as análises individuais, realizadas separadamente por cada parâmetro variado no modelo, e também algumas análises gerais dos indicadores. A próxima subseção irá apresentar análises de interações, combinando mais de um parâmetro por vez.

5.1.2. Análises de Interações

As análises das interações têm o objetivo de estudar o efeito conjunto de duas variáveis no resultado final. As interações exploradas neste trabalho foram escolhidas com base nas variáveis que parecem ter relação relevante, e serão apresentadas nas subseções a seguir:

- Número de padrões x número de atributos;
- Número de padrões x percentual de desbalanceamento entre classes;
- Número de atributos x proporção de atributos com *drift*.

5.1.2.1. Interação 1: Número de Padrões x Número de Atributos

A primeira interação combina as variáveis número de padrões e número de atributos. Os resultados obtidos são apresentados no gráfico da Figura 31,

mostrando a evolução das taxas de alarmes falsos e defeituosos para as classes 1 e 2. É importante mencionar que foi necessário normalizar os valores de número de padrões e número de atributos de forma a facilitar a visualização dos gráficos. Os valores de número de padrões (originalmente 150, 350 e 500) foram divididos por 1000 (resultando nos valores 0.15, 0.35 e 0.5) e os valores de número de atributos (originalmente 5, 15 e 25) foram divididos por 100 (resultando nos valores 0.05, 0.15 e 0.25).

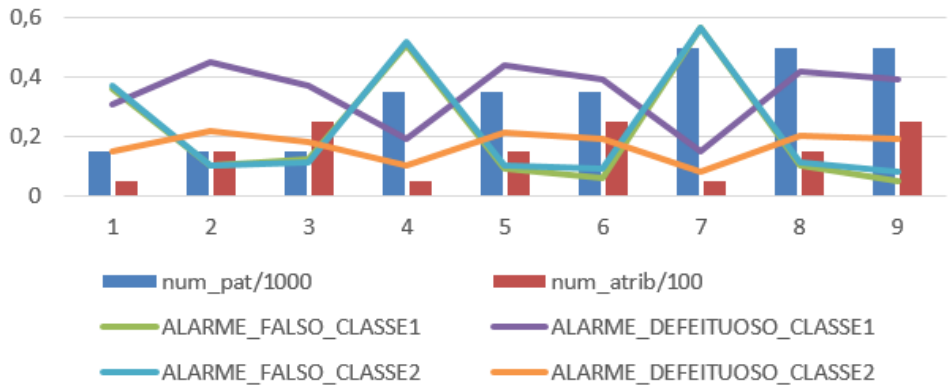


Figura 31. Evolução das taxas de alarmes falsos e defeituosos considerando número de padrões e número de atributos

Analisando o gráfico da Figura 31, observa-se que as taxas de alarmes falsos para a classe 1 praticamente coincide-se com a da classe 2, exceto por uma quantidade muito pequena de pontos. Ambas as taxas de alarmes falsos parecem estar mais relacionadas à variação do número de atributos do que à variação do número de padrões: percebe-se que elas diminuem à medida que se aumenta o número de atributos.

Também pode-se observar que, apesar das taxas de alarmes defeituosos da classe 1 serem sempre superiores às taxas de alarmes defeituosos da classe 2, elas seguem um mesmo padrão. Assim como para as taxas de alarmes falsos, o parâmetro que mais parece ter influenciado nas taxas de alarmes defeituosos é o número de atributos que, à medida que tem o seu valor aumentado, provoca a diminuição das taxas de alarmes defeituosos.

A análise desta interação reforça os resultados das análises individuais apresentados na subseção 5.1.1, onde se verificou que o número de atributos tem influência direta nos resultados das taxas de alarmes falsos e defeituosos, enquanto que o número de padrões (tamanho do bloco) tem menor influência na variação destas taxas.

5.1.2.2. Interação 2: Número de Padrões x Percentual de Desbalanceamento

A segunda interação combina as variáveis número de padrões e percentual de desbalanceamento entre classes. Os resultados obtidos são apresentados no gráfico da Figura 32, mostrando a evolução das taxas de alarmes falsos e defeituosos para as classes 1 e 2. É importante mencionar que, assim como na análise da iteração 1, foi necessário normalizar os valores de número de padrões de forma a facilitar a visualização dos gráficos. Os valores de número de padrões (originalmente 150, 350 e 500) foram divididos por 1000 (resultando nos valores 0.15, 0.35 e 0.5). Para o percentual de desbalanceamento, foram mantidos os valores originais, 0.2, 0.35 e 0.5.

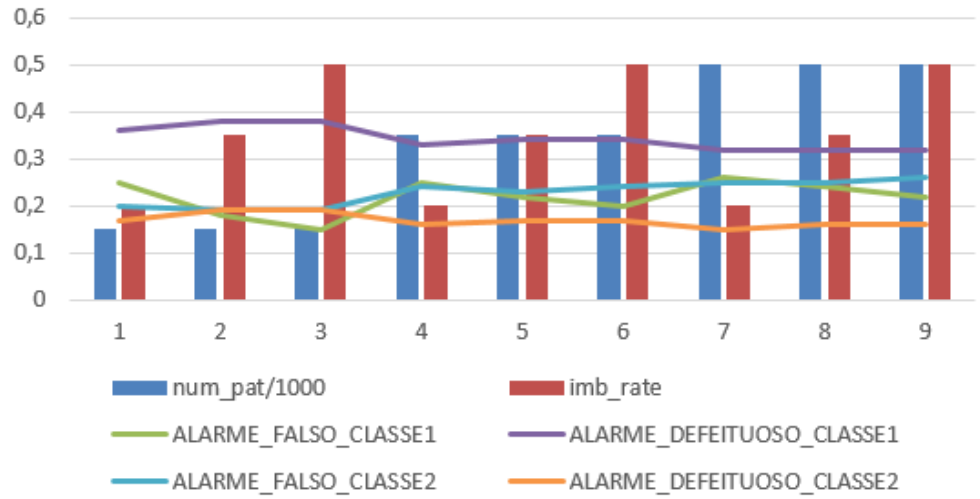


Figura 32. Evolução das taxas de alarmes falsos e defeituosos considerando número de padrões e taxa de balanceamento.

Observando o gráfico da Figura 32, nota-se que há uma pequena variação nas taxas de alarmes falsos em função da taxa de balanceamento: as taxas de alarmes falsos diminuem levemente à medida que as taxas de balanceamento crescem. Já para o número de padrões, a variação das taxas de alarmes falsos é menor ainda, indicando que as taxas de alarmes falsos aumentam levemente à medida que se aumenta o número de padrões. Para as taxas de alarmes defeituosos, as variações também são mínimas considerando a taxa de balanceamento, e um pouco mais expressivas para o número de padrões quando se observa a classe 1: à medida em que este valor sobe, as taxas de alarmes defeituosos descem.

A análise desta interação reforça os resultados das análises individuais apresentadas na subseção 5.1.1, onde se verificou que a taxa de balanceamento

teve pouca influência nos resultados das taxas de alarmes falsos e defeituosos, enquanto que o número de padrões (tamanho do bloco) influenciou um pouco mais a variação destas taxas.

5.1.2.3. Interação 3: Número de Atributos x Proporção de Atributos com *drift*

A terceira interação combina as variáveis número de atributos e proporção de atributos que sofrem *drift*. Os resultados obtidos são apresentados no gráfico da Figura 33, mostrando a evolução das taxas de alarmes falsos e defeituosos para as classes 1 e 2. É importante mencionar que, assim como nas análises das iterações 1 e 2, foi necessário normalizar os valores de número de atributos de forma a facilitar a visualização dos gráficos. Os valores de número de atributos (originalmente 5, 15 e 25) foram divididos por 100 (resultando nos valores 0.05, 0.15 e 0.25). Para a proporção de atributos que sofrem *drift*, foram mantidos os valores originais, 0.2, 0.35 e 0.5.

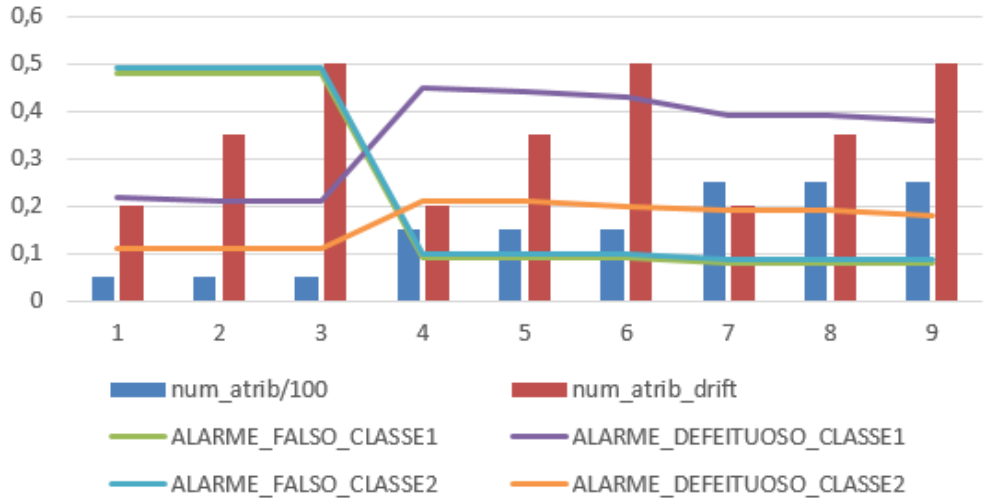


Figura 33. Evolução das taxas de alarmes falsos e defeituosos considerando número de atributos e proporção de atributos que sofrem *drift*.

Analisando o gráfico da Figura 33, observa-se que as taxas de alarmes falsos são mais altas quando o número de atributos é baixo (igual a 5), mas que as mesmas diminuem muito quando os valores deste parâmetro aumentam. E ainda, observa-se que as taxas de alarmes defeituosos são mais baixas quando o número de atributos é baixo, e as mesmas aumentam quando os valores do número de atributos aumentam. A proporção de atributos com *drift*, ao contrário, não parece ter impactos significativos nas taxas de alarmes falsos e defeituosos.

A análise desta interação reforça os resultados das análises individuais apresentados na subseção 5.1.1, onde se verificou que o número de atributos tem influência direta nos resultados das taxas de alarmes falsos e defeituosos, enquanto que o percentual de atributos com *drift* não indicou nenhuma influência na variação destas taxas.

Esta subseção apresentou as análises de interações, combinando mais de um parâmetro por vez, com o objetivo de complementar as análises individuais realizadas anteriormente. A próxima subseção irá apresentar os resultados das análises de variância realizadas.

5.1.3. Análise de Variância

A Análise de Variância (ANOVA) [Montgomery, 2013] é uma metodologia estatística que, aplicada ao contexto da análise de sensibilidade do método de detecção proposto, visa a aferir qual dos níveis de cada variável (número de atributos = {5, 15, 25}, por exemplo) contribuem para aumento ou redução média do desempenho do detector de forma significativa. Ainda, esta análise possibilita aferir quais variáveis (número de atributos, padrões, etc.) tendem a afetar com maior intensidade o desempenho do método de detecção. Portanto, a ANOVA é uma ferramenta bastante útil no propósito da análise de sensibilidade do detector proposto.

A Tabela 7 exibe os principais resultados da ANOVA para cada um dos parâmetros de forma isolada e também para as interações que foram consideradas, em função da taxa de alarmes falsos. Os parâmetros ou interações que mais afetam o modelo (maiores valores de soma dos quadrados) estão destacados em vermelho e os parâmetros ou interações significativas para o modelo (p-valor < 0,05) estão destacados em azul.

Analisando a Tabela 7, pode-se verificar que, considerando as taxas de alarmes falsos, o número de atributos é o parâmetro mais influente do modelo (com base no valor da soma dos quadrados). Em segundo lugar, está o parâmetro *alpha* que, por medir o nível de significância, está diretamente ligado ao fato do modelo ser mais ou menos reativo quanto a alarmes falsos, sendo então já esperado o impacto direto nos resultados do modelo. Em terceiro lugar, também já esperado, está o número de blocos com *drift*, indicando que quanto mais ocorrências de *drift* numa determinada classe, mais fácil será a detecção. Considerando o p-valor, os parâmetros ou interações que são influentes para as

taxas de alarmes falsos são: número de atributos, número de padrões, percentual de desbalanceamento entre classes (influencia apenas para a classe 1 – por ser em geral a classe minoritária), número de blocos com *drift*, *alpha*, interação 1 (Número de padrões x número de atributos) e interação 2 (Número de padrões x percentual de desbalanceamento entre classes).

Tabela 7. Análise de variância considerando alarmes falsos.

		Alarme Falso - Classe 1		Alarme Falso - Classe 2	
		Soma dos quadrados	p-valor	Soma dos quadrados	p-valor
Parâmetros	Número de atributos	209,37	< 2e-16	202,65	< 2e-16
	Número de padrões por bloco	2,91	< 2e-16	4,93	< 2e-16
	Percentual de desbalanceamento entre classes	5,53	< 2e-16	0,00	0,78
	Proporção de atributos com *drift*	0,01	0,35	0,00	0,86
	Número de blocos com *drift* na classe 1	5,94	< 2e-16	4,68	< 2e-16
	Número de blocos com *drift* na classe 2	0,00	0,76	14,69	< 2e-16
	alpha	36,84	< 2e-16	40,22	< 2e-16
Interações	1) Número de padrões x número de atributos	17,55	< 2e-16	11,44	< 2e-16
	2) Número de padrões x percentual de desbalanceamento entre classe	0,60	< 2e-16	0,11	0,01
	3) Número de atributos x proporção de atributos com *drift*	0,01	0,44	0,00	0,62

Estas premissas foram confirmadas com o teste de Tukey [Montgomery, 2013], um teste de comparação de médias entre níveis de uma determinada variável. O teste de Tukey foi realizado apenas para os resultados mais relevantes e com maiores diferenças na ANOVA: número de atributos, número de padrões, *alpha*, número de blocos com *drift*, interação 1.

As tabelas referentes aos testes de Tukey realizados podem ser encontradas no Anexo 3. Em resumo, pode-se notar que todos estes parâmetros são significativos (p-valor < 0.05) e também que os seguintes resultados das análises anteriores foram confirmados, para ambas as classes:

- Quanto menor o número de atributos, maiores são as taxas de alarmes falsos;
- Quanto maior o número de padrões, maiores as taxas de alarmes falsos;
- Quanto maior o *alpha*, maiores as taxas de alarmes falsos;

- Quanto maior número de blocos com *drift*, menores as taxas de alarmes falsos;
- Considerando a interação 1, o número de atributos é o parâmetro mais forte da interação quanto à influência de alarmes falsos.

Obviamente, os resultados do teste de Tukey confirmam os resultados das análises individuais. Porém, com relação somente ao número de atributos e número de padrões, as hipóteses levantadas inicialmente foram contrariadas. Esperava-se que o aumento do número de atributos aumentasse as taxas de alarmes falsos, mas o contrário ocorreu. Isto se deve, provavelmente, ao fato de um número maior de atributos fornecer mais informações sobre o problema em estudo e, consequentemente, facilitar a tomada de decisões corretas sobre ele (produzindo menos alarmes falsos). Outra possibilidade que precisa ser analisada em trabalhos futuros é que mais atributos facilitam o processo de agrupamento e, consequentemente, a detecção de *drift*.

Em relação ao número de padrões, esperava-se que com mais padrões, as taxas de alarmes falsos reduziriam devido ao aumento do poder dos testes de hipótese, mas o contrário ocorreu. Isto pode ser talvez explicado pela maior dificuldade de se realizar o agrupamento em muitos dados, o que pode ser investigado em trabalhos futuros.

A Tabela 8 apresenta os principais resultados da ANOVA para cada uma das variáveis do modelo e também para as interações que foram consideradas, considerando agora as taxas de alarmes defeituosos. Os parâmetros ou interações que mais afetam o modelo (maiores valores de soma dos quadrados) estão destacados em vermelho e os parâmetros ou interações significativas para o modelo (p-valor < 0,05) estão destacados em azul.

Tabela 8. Análise de variância considerando alarmes defeituosos.

		Alarme Defeituoso - Classe 1		Alarme Defeituoso - Classe 2	
		Soma dos quadrados	p-valor	Soma dos quadrados	p-valor
Parâmetros	Número de atributos	37,70	< 2e-16	7,50	< 2e-16
	Número de padrões por bloco	4,00	< 2e-16	0,90	< 2e-16
	Percentual de desbalanceamento entre classes	0,20	0,04	0,30	0,03
	Proporção de atributos com *drift*	0,10	0,05	0,10	0,25
	Número de blocos com *drift* na classe 1	0,10	0,22	0,00	0,79
	Número de blocos com *drift* na classe 2	0,00	0,31	135,10	< 2e-16
	alpha	557,00	< 2e-16	156,00	< 2e-16
Interações	1) Número de padrões x número de atributos	7,40	< 2e-16	1,10	< 2e-16
	2) Número de padrões x percentual de desbalanceamento entre classe	0,10	0,25	0,00	0,64
	3) Número de atributos x proporção de atributos com *drift*	0,00	0,95	0,00	0,94

Analisando a Tabela 8, referente à taxa de alarmes defeituosos, o parâmetro *alpha* é o mais influente no modelo (com base no valor da soma dos quadrados). Em segundo lugar, está número de blocos com *drift* na classe 2 (para a classe 2) e em terceiro lugar, o número de atributos. Os parâmetros ou interações que são significativas para as taxas de alarmes defeituosos são: número de atributos, número de padrões, percentual de desbalanceamento, número de blocos com *drift* na classe 2 (apenas para a classe 2), *alpha* e interação 1 (número de padrões x número de atributos). Estas premissas foram confirmadas com o teste de Tukey, que foi realizado para os resultados mais relevantes e com maiores diferenças na ANOVA: número de atributos, *alpha*, número de padrões e interação 1. As tabelas dos testes realizados também podem ser encontradas no Anexo 3. Em resumo, pode-se notar que todos estes parâmetros são significativos (p-valor < 0.05) e também que os seguintes resultados das análises anteriores foram confirmados, para ambas as classes:

- A taxa de alarmes defeituosos não parece seguir uma relação linear com o número de atributos;
- Quanto maior o valor de *alpha*, menores são as taxas de alarmes defeituosos;

- Quanto maior o valor de número de padrões, menores são as taxas de alarmes defeituosos;
- Considerando a interação 1, o número de atributos é o parâmetro mais forte da interação quanto à influência de alarmes defeituosos.

Os resultados do teste de Tukey para a taxa de alarmes defeituosos confirmou as hipóteses levantadas inicialmente.

Como conclusão, os resultados deste primeiro experimento mostraram que o detector se mostrou robusto e eficiente para:

- Bases de dados de alta dimensionalidade, pois as taxas de alarmes falsos foram menores para mais atributos e as de alarmes defeituosos não estão diretamente associadas ao número de atributos;
- Blocos de tamanho intermediário, pois as taxas de alarmes falsos aumentam com o aumento do bloco (número de padrões) enquanto as taxas de alarmes defeituosos diminuem;
- Bases de dados com qualquer proporção de *drift*, uma vez que este parâmetro não influenciou de forma significativa as taxas de alarmes falsos ou defeituosos;
- Bases de dados com qualquer balanceamento de classes, devido aos resultados não terem sofrido variações significativas com a mudança deste parâmetro.

Esta seção apresentou os resultados do primeiro experimento realizado com o método de detecção DetectA. A próxima seção apresentará os resultados do segundo experimento.

5.2. Experimento 2: Detecção de *Drift* em Bases de Dados

Com o objetivo de checar a influência da utilização do modelo de detecção proposto na classificação, considerando a acurácia e o desempenho computacional, foram usadas seis diferentes bases de dados com as quais foram efetuadas diversas simulações em cenários distintos. As bases escolhidas são conhecidas na literatura e já foram utilizadas em diversos modelos.

5.2.1.
Descrição das Bases de Dados

As bases de dados utilizadas neste experimento são brevemente apresentadas a seguir. No total, seis bases de dados foram selecionadas, duas bases artificiais (*SEA Concepts* e *Rotating Checkboard*), apresentadas na subseção 5.2.1.1, e quatro bases reais (*Nebraska*, *Electricity*, *Cover Type* e *Poker Hand*), descritas na subseção 5.2.1.2.

5.2.1.1.
Bases Artificiais

5.2.1.1.1.
SEA Concepts

A base de dados *SEA Concepts* foi criada artificialmente em [Street & Kim, 2001] e tem sido usada por diferentes algoritmos como um *benchmark* para estudar modelos que identificam *concept drift*. A base de dados, disponível em [Polikar & Elwell, 2013], é caracterizada por extensos períodos sem grandes alterações no ambiente, mas com ocasionais e bruscas alterações nas regiões de discriminação das classes (*concept drift*). A base de dados consiste em 50000 padrões criados aleatoriamente em um plano tri-dimensional (3 atributos). Os padrões para cada atributo estão no intervalo [0, 10], mas somente dois dos três atributos são relevantes para a determinação da classe de saída. Esses padrões são divididos em quatro blocos, com diferentes conceitos. As classes são atribuídas a cada padrão baseado na soma dos atributos relevantes (no caso, dois atributos); um determinado padrão pertence à classe 1 se a soma dos atributos relevantes for maior que um limiar; e pertence à classe 2, caso contrário. Esse limiar é alterado ao longo do tempo, em intervalos regulares, cujos valores são (8→9→7.5→9.5), criando, portanto, *drifts* abruptos nas regiões de discriminação das classes.

Esta base de dados foi usada em [Street & Kim, 2001] com o algoritmo SEA, em [Elwell & Polikar, 2011] com o algoritmo Learn++.NSE, em [Kolter & Maloof, 2007] com o algoritmo DWM e em [Gonçalves Júnior, 2013] com o algoritmo RCD.

5.2.1.1.2. Rotating Checkboard

A base de dados *Rotating Checkboard*, também disponível em [Polikar & Elwell, 2013], é um *dataset* não-gaussiano derivado do problema canônico XOR, e é similar a um tabuleiro de xadrez que gira. A rotação torna este problema particularmente desafiador, uma vez que o ângulo e a localização das fronteiras de decisão mudam drasticamente a cada poucos passos de tempo. A fim de evitar o treinamento de blocos idênticos de dados e para aumentar a complexidade, foi introduzido 10% de ruído de forma aleatória.

Esta base de dados tem quatro variações, representando quatro diferentes cenários de taxas de *drift*: 1) taxa de *drift* constante de 2pi/400 = 0.016rad/*time step*; 2) taxa de *drift* que aumenta exponencialmente (o tabuleiro gira cada vez mais rápido); 3) taxa de *drift* variando de forma sinusoidal; e 4) Taxa de *drift* em forma de pulso gaussiano (devagar-rápido-muito rápido-rápido-lento) *drift*. Cada uma das quatro variações da base de dados consiste em 400 blocos com 25 padrões cada de treino e 400 blocos com 1024 padrões de teste. Os filmes que ilustram as quatro variações da base de dados podem ser encontrados em [Polikar & Elwell, 2013].

Esta base de dados foi usada em [Elwell & Polikar, 2011] com o algoritmo Learn++.NSE, entre outros modelos.

5.2.1.2. Bases Reais

5.2.1.2.1. Nebraska

A base de dados Nebraska, também disponível em [Polikar & Elwell, 2013], apresenta uma compilação de medidas de clima a partir de 9000 estações climáticas espalhadas mundialmente pela *U.S. National Oceanic and Atmospheric Administration* desde 1930, provendo uma grande quantidade de períodos climáticos. As medições diárias incluem uma variedade de atributos (como temperatura, pressão, velocidade do vento, etc.), indicadores de precipitação e outras informações e eventos relacionados ao clima. Como esta base de dados possui um tamanho significativo, foram escolhidos para este experimento os dados provenientes da subestação *Offutt Air Force Base em Bellevue*, Nebraska, devido à grande quantidade de padrões (50 anos de dados diários entre 1949–1999) e diversos atributos climáticos, sendo esta base um problema real de classificação/previsão com *concept drift*. Os rótulos das classes

são baseados em uma indicação binária, onde para cada dia (padrão) há a possibilidade de chover (31% dos exemplos), ou não chover (61% dos exemplos). Cada bloco de treinamento consiste em 30 amostras (dias), com os subsequentes 30 dias como base de teste. O classificador deve então prever se nos 30 dias seguintes choverá ou não, e estes dados se tornarão dados de treinamento para o bloco seguinte. Esta base de dados inclui 583 blocos consecutivos de treinamento e teste, cada um com 30 dias, cobrindo os 50 anos.

Esta base de dados foi usada em [Elwell & Polikar, 2011] com o algoritmo Learn++.NSE e em [Gonçalves Júnior, 2013] com o algoritmo RCD.

5.2.1.2.2. Electricity

Esta base de dados, disponível em [MOA Datasets, 2015], é composta de 45.312 padrões e oito atributos extraídos do *Australian New South Wales Electricity Market*, coletados a cada 30 minutos entre 7 de maio de 1996 e 5 de dezembro de 1998. Dentre os oito atributos de entrada, dois deles representam o dia da semana (um inteiro entre 1 e 7) e o período do dia (um inteiro entre 1 e 48, representando os 48 períodos de 30 minutos existentes em um dia).

Neste mercado de eletricidade, os preços não são fixos, variando com base na oferta e na demanda do mercado. Os preços são definidos a cada cinco minutos e o rótulo da classe define a mudança do preço relacionada a uma média móvel das últimas 24 horas. O objetivo do problema é prever se o preço irá subir ou descer. Esta base de dados inclui 944 blocos consecutivos de treinamento e teste, cada um com 48 padrões.

Esta base foi usada em [Kolter & Maloof, 2007] com o algoritmo DWM e em [Gonçalves Júnior, 2013] com o algoritmo RCD.

5.2.1.2.3. Cover Type

Esta base de dados, disponível também em [MOA Datasets, 2015], é composta de 581.012 padrões e 54 atributos. Ela contém células de informação que correspondem a uma cobertura florestal de 30x30 metros, determinada pelo Sistema de Informações de recursos da Região 2 do US Forest Service (USFS). O objetivo do problema é predizer o tipo de cobertura florestal a partir das variáveis cartográficas. A saída é uma classe categórica com sete possíveis valores.

Esta base foi usada em [Gonçalves Júnior, 2013], com o algoritmo RCD.

5.2.1.2.4. Poker Hand

Esta base de dados, disponível também em [MOA Datasets, 2015], é composta de 829.201 padrões e dez atributos (cinco categóricos e cinco numéricos). A saída é uma classe categórica com dez possíveis valores representando a mão de poker, que contém 5 cartas. Esta base de dados então representa o problema de identificar a classe de uma mão de Poker dentre as dez possibilidades (por exemplo, *Full House*, *Flush*, etc). Esta base foi também usada em [Gonçalves Júnior, 2013], com o algoritmo RCD.

5.2.2. Descrição do Experimento

A fim de investigar a influência do tipo de detector utilizado na acurácia e no desempenho computacional de um modelo de classificação, optou-se por utilizar uma rede neural do tipo *multi-layer perceptron (MLP)* simples como classificador base. Esta escolha deveu-se ao fato do modelo de classificação proposto nesta tese (NEVE), já apresentado no capítulo 4, ser baseado em um comitê de redes neurais do tipo *MLP*. Para o modelo de detecção de *drift*, foram usadas as estratégias proativa (com as abordagens *Group Label* e *Pattern Mean Shift*) e reativa, já descritos no capítulo 3. Para fins de comparação de resultados, também foram realizados experimentos com uma *MLP* simples sem nenhum detector.

Decidiu-se pela utilização da abordagem em blocos para o treinamento e teste dos modelos para manter a consistência com o modelo NEVE, que trabalha também em blocos. Os valores de tamanho de bloco e número de blocos utilizados para cada base de dados estão representados na Tabela 9. Foram escolhidos valores já utilizados em outros trabalhos da literatura, como por exemplo [Elwell & Polikar, 2011] e [Brzezinski & Stefanowski, 2014].

Tabela 9. Tamanho do bloco e número de blocos utilizados no experimento.

Base de Dados	Tamanho do Bloco	Número de Blocos
SEA Concepts	250	400
Rotating Checkboard	25 (treino), 1024 (teste)	400
Nebraska	30	583
Electricity	48	944
Cover Type	500	1162
Poker Hand	500	1658

Para os modelos que utilizaram o mecanismo de detecção de *drift*, foram utilizadas duas diferentes abordagens de treinamento:

- "*Esquece passado após detecção*": consiste em retreinar o classificador a cada novo bloco de dados com todos os blocos passados, porém, no caso da detecção de um *drift*, o treinamento é realizado apenas do ponto de detecção em diante, e todo o passado é esquecido para o treinamento.
- "*Somente treina com detecção*": consiste em somente retreinar o classificador quando há detecção de *drift*, utilizando para treinamento o bloco onde o *drift* foi detectado.

No caso da utilização do classificador simples (sem detecção), a abordagem de treinamento tradicional é utilizada: a cada novo bloco de dados, o classificador é retreinado com todos os blocos passados.

Desta forma, serão executadas simulações utilizando sete variações do modelo de detecção, resumidas na tabela 10. Para cada uma das sete variações, foram executadas 30 simulações criando uma rede MLP com 5 neurônios na camada escondida e 30 simulações criando uma rede MLP com 10 neurônios na camada escondida, totalizando 420 execuções para cada uma das bases de dados.

Tabela 10. Modelos e abordagens utilizadas no experimento.

#	Modelo utilizado	Abordagem de treino
1	Sem detecção	Tradicional
2	Detecção reativa	Esquece passado após detecção
3	Detecção reativa	Somente treina com detecção
4	Detecção proativa – *Group Label*	Esquece passado após detecção
5	Detecção proativa – *Group Label*	Somente treina com detecção
6	Detecção proativa – *Pattern Mean Shift*	Esquece passado após detecção
7	Detecção proativa – *Pattern Mean Shift*	Somente treina com detecção

Para melhor compreensão do funcionamento de cada uma das sete variações, as próximas subseções apresentam um exemplo do fluxo de execução de cada abordagem. Este exemplo mostrará o que acontece em cada passo de tempo, a partir do primeiro bloco de dados recebido.

5.2.2.1. Abordagem 1 – Rede MLP simples (sem detecção)

Esta abordagem consiste em apenas uma MLP simples, sem detecção de *drift*, e seu funcionamento pode ser resumido conforme descrito a seguir. Inicialmente, é criada uma rede MLP, com 5 ou 10 neurônios na camada escondida, e esta é treinada com o bloco 1 utilizando os rótulos de classes reais. A seguir, a rede é testada com o bloco 2. Assumindo que em algum momento após este passo chegarão os rótulos verdadeiros do bloco 2, a rede é retreinada com os blocos 1 e 2 e os respectivos rótulos verdadeiros. A rede é então testada com o bloco 3 e, da mesma forma, assume-se que em algum momento após este passo chegarão os rótulos verdadeiros do bloco 3, e assim por diante. A Figura 34 ilustra este processo.

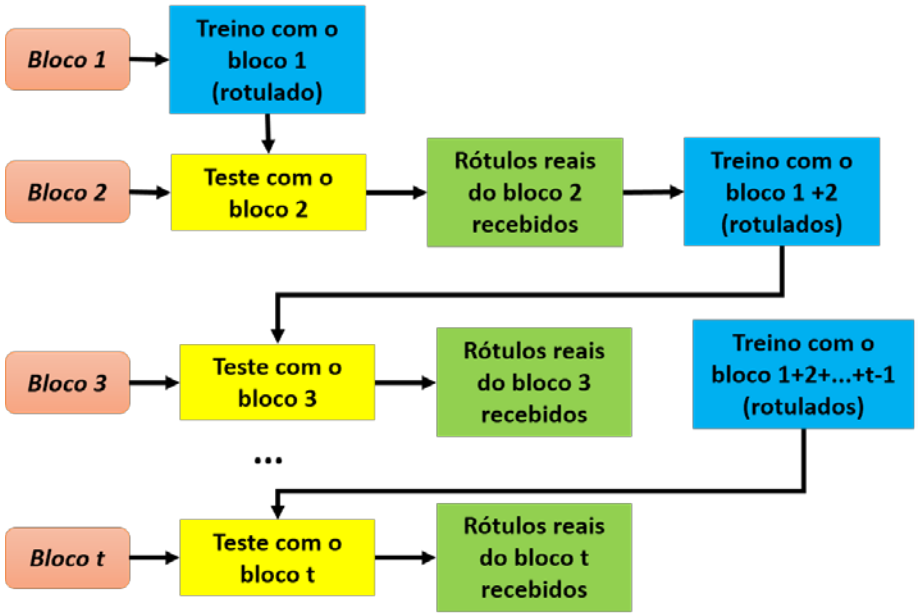


Figura 34. Exemplo de iteração da abordagem MLP simples (sem detecção).

5.2.2.2. Abordagem 2 – Rede MLP com detecção reativa de *drift* – Esquece passado após detecção

Nesta abordagem, inicialmente, é criada uma rede MLP com 5 ou 10 neurônios na camada escondida e esta é treinada com o bloco 1 utilizando os rótulos de classes reais. A seguir, a rede é testada com o bloco 2. Assumindo que em algum momento após este passo chegarão os rótulos verdadeiros do bloco 2, verifica-se se o bloco 2 apresentou *drift* em relação ao bloco 1. Em caso afirmativo, a rede é retreinada apenas com o bloco 2 e os respectivos rótulos

verdadeiros. Caso contrário, a rede é retreinada com os blocos 1 e 2 e os respectivos rótulos verdadeiros. A rede é então testada com o bloco 3 e, da mesma forma, assume-se que em algum momento após este passo chegarão os rótulos verdadeiros do bloco 3 e verifica-se se o bloco 3 apresentou *drift* em relação ao bloco 2. Em caso afirmativo, a rede é retreinada apenas com o bloco 3 e os respectivos rótulos verdadeiros. Caso contrário, a rede é retreinada com os blocos 1, 2 e 3 e os respectivos rótulos verdadeiros (caso não tenha ocorrido *drift* do bloco 1 para o bloco 2) ou com os blocos 2 e 3 e os respectivos rótulos verdadeiros (caso tenha ocorrido *drift* do bloco 1 para o bloco 2). A rede é testada com o bloco 4, e assim por diante. A Figura 35 ilustra este processo.

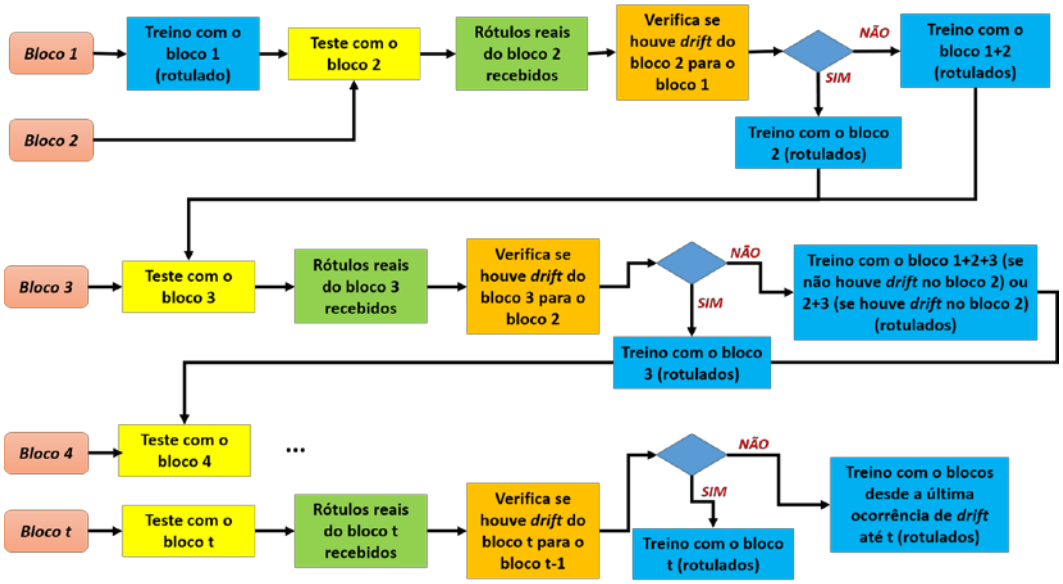


Figura 35. Exemplo de iteração da abordagem MLP com detecção reativa (Esquece passado após detecção).

5.2.2.3. Abordagem 3 – Rede MLP com detecção reativa de *drift* – Somente treina com detecção

Esta abordagem é similar à anterior (abordagem 2), diferenciando-se apenas na consequência da detecção ou não de *drift*. Quando o mecanismo de detecção é executado, caso seja detectado *drift* no bloco t em relação ao bloco t-1, a rede é retreinada apenas com o bloco t e os respectivos rótulos verdadeiros. Caso contrário, a rede antiga continua sendo utilizada, sem novo treino. A Figura 36 ilustra este processo.

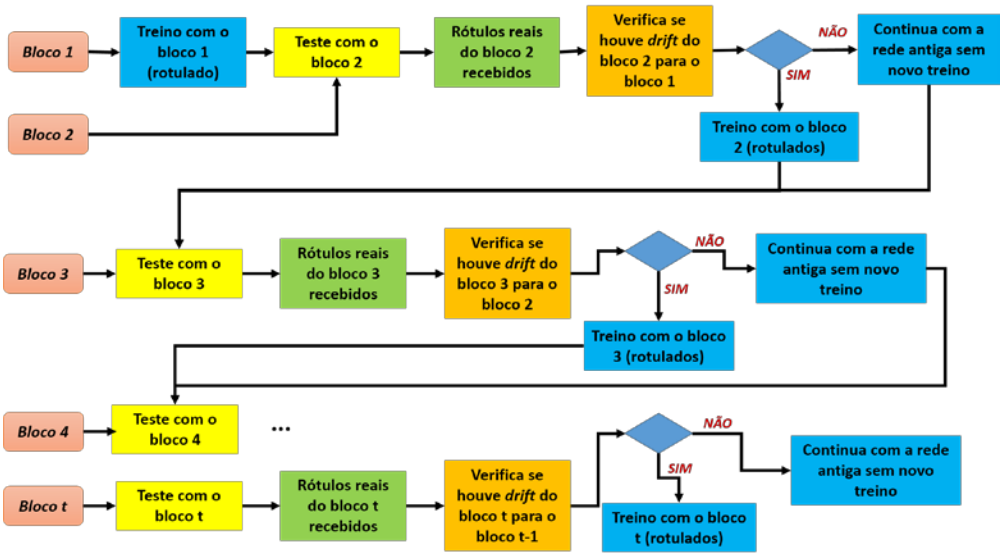


Figura 36. Exemplo de iteração da abordagem MLP com detecção reativa (Somente treina com detecção).

5.2.2.4. Abordagem 4 – Rede MLP com detecção proativa de *drift* - *Group Label* – Esquece passado após detecção

Nesta abordagem, inicialmente é criada uma rede MLP com 5 ou 10 neurônios na camada escondida, treinada com o bloco 1 utilizando os rótulos de classes reais. Ao chegar o bloco 2, é feito o agrupamento para se obter as classes previstas do bloco 2 e verifica-se se o bloco 2 apresentou *drift* em relação ao bloco 1. Em caso afirmativo, a rede é retreinada apenas com o bloco 2 e os respectivos rótulos previstos no agrupamento. A rede é testada com o bloco 2 e assume-se que, em seguida, chegarão os rótulos verdadeiros deste bloco, quando então é realizado o ajuste dos centroides para o próximo agrupamento considerando o bloco 2 e os rótulos reais. Se não foi detectado *drift* no bloco 2 em relação ao bloco 1, a rede é retreinada com os blocos 1 e 2 e os respectivos rótulos verdadeiros. A seguir, ao chegar o bloco 3, é feito o agrupamento para se obter as classes previstas do bloco 3 e verifica-se se o bloco 3 apresentou *drift* em relação ao bloco 2, e assim por diante. A Figura 37 ilustra este processo.

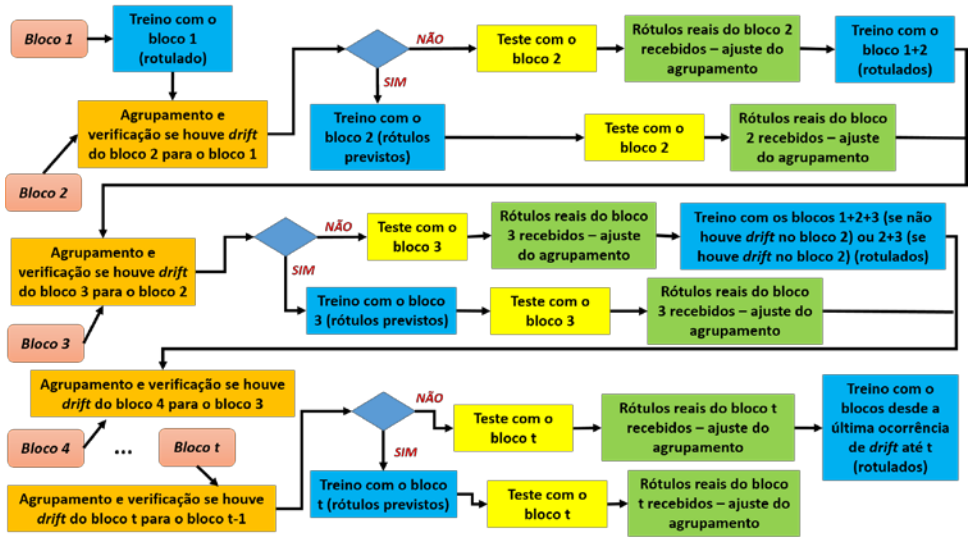


Figura 37. Exemplo de iteração da abordagem MLP com detecção proativa *Group Label* (Esquece passado após detecção).

5.2.2.5. Abordagem 5 – Rede MLP com detecção proativa de *drift* - *Group Label* – Somente treina com detecção

Esta abordagem é similar à anterior (abordagem 4), diferenciando-se apenas na consequência da detecção ou não de *drift*. Quando o mecanismo de detecção é executado, caso seja detectado *drift* no bloco t em relação ao bloco t-1, a rede é retreinada apenas com o bloco t e os rótulos de classe previstos no agrupamento. Caso contrário, a rede antiga continua sendo utilizada, sem novo treino. A Figura 38 ilustra este processo.

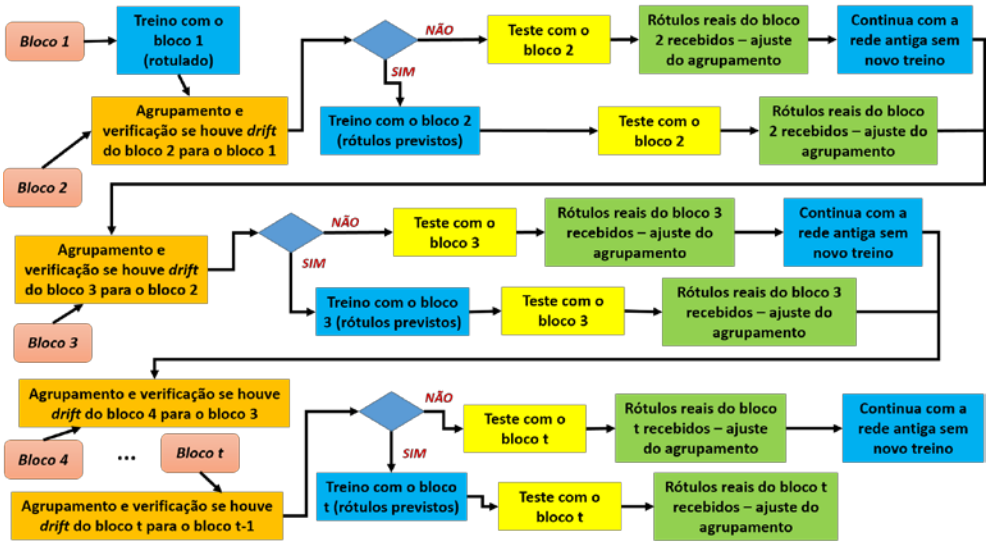


Figura 38. Exemplo de iteração da abordagem MLP com detecção proativa *Group Label* (Somente treina com detecção).

5.2.2.6. Abordagem 6 – Rede MLP com detecção proativa de *drift* - *Pattern Mean Shift* – Esquece passado após detecção

Nesta abordagem, inicialmente, cria-se uma rede MLP, com 5 ou 10 neurônios na camada escondida, e treina-a com o bloco 1 utilizando os rótulos de classes reais. Ao chegar o bloco 2, é feito o agrupamento para se obter as classes previstas do bloco 2 e verifica-se se o bloco 2 apresentou *drift* em relação ao bloco 1. Em caso afirmativo, a rede é testada com o bloco 2 "ajustado" em direção ao *drift* detectado; assume-se que em seguida, chegarão os rótulos verdadeiros deste bloco, quando então o agrupamento do bloco 2 é ajustado considerando os rótulos reais e, em seguida, a rede é retreinada apenas com o bloco 2 e os rótulos reais. Se não foi detectado *drift* no bloco 2 em relação ao bloco 1, a rede é testada com o bloco 2 com seus valores originais e assume-se que, em seguida, chegarão os rótulos verdadeiros deste bloco, quando então é realizado o ajuste das sugestões de centroides para o próximo agrupamento considerando o bloco 2 e os rótulos reais. Neste caso, neste momento a rede é retreinada com os blocos 1 e 2 e os respectivos rótulos verdadeiros. A seguir, ao chegar o bloco 3, é feito o agrupamento para se obter as classes previstas do bloco 3 e verifica-se se o bloco 3 apresentou *drift* e relação ao bloco 2, e assim por diante. A Figura 39 ilustra este processo.

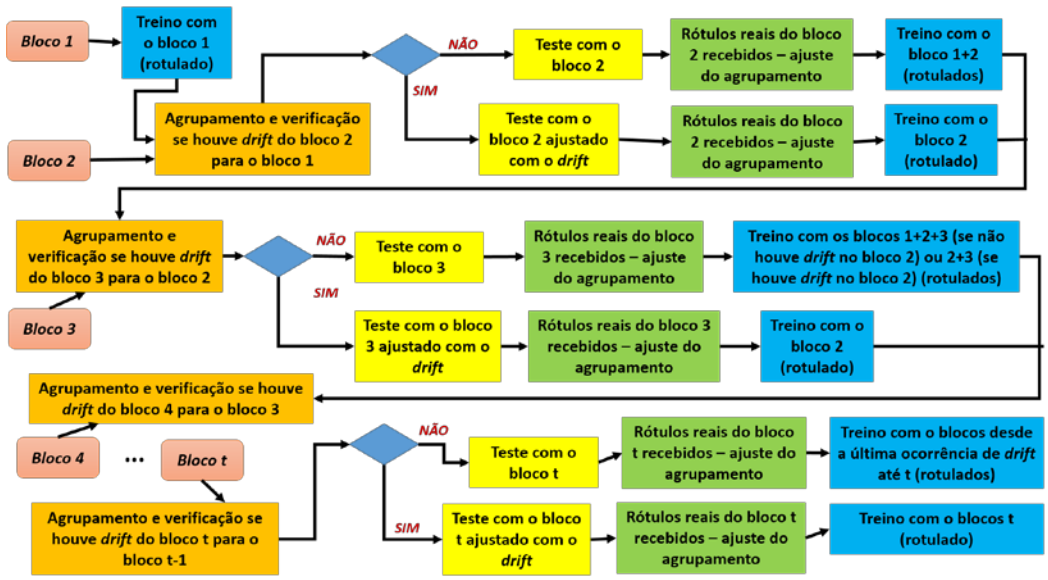


Figura 39. Exemplo de iteração da abordagem MLP com detecção proativa *Pattern Mean Shift* (Esquece passado após detecção).

5.2.2.7.
Abordagem 7 – Rede MLP com detecção proativa de *drift -Pattern Mean Shift* – Somente treina com detecção

Esta abordagem é similar à anterior (abordagem 6), diferenciando-se apenas na consequência da detecção ou não de *drift*. Quando o mecanismo de detecção é executado, caso seja detectado *drift* no bloco t em relação ao bloco t-1, a rede é retreinada apenas com o bloco t "ajustado" em direção ao *drift* detectado. Caso contrário, a rede antiga continua sendo utilizada, sem novo treinamento. A Figura 40 ilustra este processo.

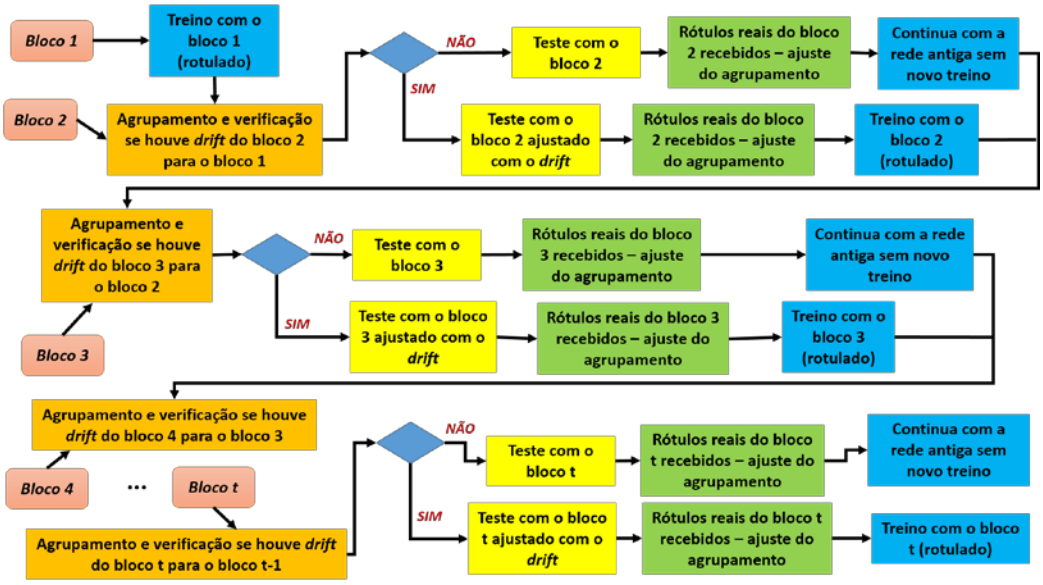


Figura 40. Exemplo de iteração da abordagem MLP com detecção proativa Pattern Mean Shift (Somente treina com detecção).

5.2.3.
Resultados do Experimento

A seguir são apresentados os resultados de acurácia média, e desempenho computacional médio em segundos (tempo médio de execução) para cada uma das sete abordagens utilizadas. As Tabelas 11 e 12 mostram os resultados das bases de dados artificiais, e as Tabelas 13 e 14, das bases de dados reais. A escala de cores utilizada vai do vermelho (piores valores), passando pelo laranja e amarelo (valores médios) e chegando até o verde (melhores valores). Obviamente, os melhores valores considerados são os maiores valores de acurácia e menores valores de desempenho computacional.

As análises dos resultados serão apresentadas a seguir. Em todos os casos, o desvio padrão observado foi inferior a 2%.

Tabela 11. Resultados das bases de dados artificiais (*SEA Concepts*).

Base de dados	SEA			
Escondida	5	10	5	10
	Acurácia		Tempo	
MLP simples (sem detecção)	0,6410	0,6382	42	41
MLP + DE reativa - Esquece pass.	0,8761	0,8637	78	65
MLP + DE reativa - Só treina c/ det.	0,8105	0,8463	10	7
MLP + DE pro. GL - Esquece pass.	0,7134	0,7164	205	192
MLP + DE pro. GL - Só treina c/ det.	0,7169	0,7140	194	209
MLP + DE pro. PM - Esquece pass.	0,8424	0,8354	116	104
MLP + DE pro. PM - Só treina c/ det.	0,8387	0,8325	177	188

Tabela 12. Resultados das bases de dados artificiais (*Rotating Checkboard*).

Base de dados	CB-Const				CB-Exp			
Escondida	5	10	5	10	5	10	5	10
	Acurácia		Tempo		Acurácia		Tempo	
MLP simples (sem detecção)	0,5324	0,5258	80	78	0,5314	0,5298	79	78
MLP + DE reativa - Esquece pass.	0,6610	0,6858	102	110	0,6721	0,6898	102	109
MLP + DE reativa - Só treina c/ det.	0,5541	0,5352	14	15	0,5486	0,5732	16	17
MLP + DE pro. GL - Esquece pass.	0,5220	0,5225	678	752	0,5244	0,5241	753	838
MLP + DE pro. GL - Só treina c/ det.	0,5415	0,5485	647	706	0,5392	0,5528	642	706
MLP + DE pro. PM - Esquece pass.	0,6794	0,7095	164	170	0,6541	0,6982	180	186
MLP + DE pro. PM - Só treina c/ det.	0,6769	0,6953	466	527	0,6531	0,6882	613	639
Base de dados	**CB-Pulse**				**CB-Sin**			
Escondida	5	10	5	10	5	10	5	10
	Acurácia		Tempo		Acurácia		Tempo	
MLP simples (sem detecção)	0,5163	0,5130	368	79	0,5229	0,5269	82	322
MLP + DE reativa - Esquece pass.	0,6941	0,7142	458	307	0,6608	0,6924	101	319
MLP + DE reativa - Só treina c/ det.	0,6389	0,6474	17	19	0,5996	0,5977	16	17
MLP + DE pro. GL - Esquece pass.	0,5149	0,5172	930	903	0,5112	0,5123	739	942
MLP + DE pro. GL - Só treina c/ det.	0,5299	0,5359	801	842	0,5283	0,5350	736	805
MLP + DE pro. PM - Esquece pass.	0,7009	0,7166	243	246	0,6824	0,6929	204	211
MLP + DE pro. PM - Só treina c/ det.	0,6801	0,7050	694	701	0,6684	0,6970	652	721

Tabela 13. Resultados das bases de dados reais (*Nebraska e Electricity*).

Base de dados	Nebraska				Electricity			
Escondida	5	10	5	10	5	10	5	10
	Acurácia		Tempo		Acurácia		Tempo	
MLP simples (sem detecção)	0,6655	0,6633	115	116	0,4254	0,4254	371	164
MLP + DE reativa - Esquece pass.	0,6795	0,6858	210	166	0,7640	0,7754	574	537
MLP + DE reativa - Só treina c/ det.	0,6855	0,6834	173	159	0,7614	0,7626	531	312
MLP + DE pro. GL - Esquece pass.	0,5482	0,5448	503	388	0,6887	0,6884	651	534
MLP + DE pro. GL - Só treina c/ det.	0,5478	0,5456	294	513	0,7032	0,7063	875	825
MLP + DE pro. PM - Esquece pass.	0,6589	0,6545	233	216	0,7397	0,7400	464	462
MLP + DE pro. PM - Só treina c/ det.	0,6485	0,6541	405	366	0,7379	0,7399	691	744

Tabela 14. Resultados das bases de dados reais (*Poker Hand e Cover Type*).

Base de dados	Poker Hand				Cover Type			
Escondida	5	10	5	10	5	10	5	10
	Acurácia		Tempo		Acurácia		Tempo	
MLP simples (sem detecção)	0,6194	0,6737	1423	1677	0,7686	0,8244	832	869
MLP + DE reativa - Esquece pass.	0,6714	0,6885	2007	2419	0,8036	0,8405	1242	1150
MLP + DE reativa - Só treina c/ det.	0,6668	0,6941	2061	2256	0,8098	0,8393	1184	1134
MLP + DE pro. GL - Esquece pass.	0,1317	0,1415	2848	2631	0,5890	0,6025	2154	1743
MLP + DE pro. GL - Só treina c/ det.	0,1372	0,1368	2923	2979	0,5875	0,6027	2217	1467
MLP + DE pro. PM - Esquece pass.	0,6699	0,6929	2193	2746	0,8045	0,8386	1501	1638
MLP + DE pro. PM - Só treina c/ det.	0,6650	0,6909	2177	2676	0,8117	0,8376	1440	1576

Analisando as Tabelas 11 a 14, as seguintes considerações podem ser extraídas para cada base de dados:

SEA Concepts:

- As diferenças nos valores de acurácia e desempenho computacional, comparando as redes com 5 ou 10 neurônios na camada escondida, são muito pequenas;
- A melhor acurácia média é para o modelo com detecção reativa (abordagem esquece passado após detecção), mas os modelos com detecção proativa *Pattern Mean Shift* (ambas as abordagens) e com detecção reativa (abordagem só treina com detecção) também apresentam boas acurácias;
- O melhor desempenho computacional médio é o modelo com detecção reativa (abordagem só treina com detecção), e sua acurácia média está entre as melhores para esta base de dados.

Os modelos sem detecção e com detecção reativa (abordagem esquece passado após detecção) também apresentam um bom desempenho computacional;

- A pior acurácia média ocorre para o modelo sem detecção e o pior desempenho médio ocorre para os modelos com detecção proativa *Group Label* (ambas as abordagens);
- MELHOR ESCOLHA: detecção reativa (abordagem esquece passado após detecção), devido à boa acurácia e desempenho computacional razoável;
- PIOR ESCOLHA: sem detecção, devido à péssima acurácia.

Rotating Checkboard:

- As diferenças nos valores de acurácia e desempenho computacional comparando as redes com 5 ou 10 neurônios na camada escondida são pequenas na maioria das vezes, mas os valores para 5 neurônios são ligeiramente melhores considerando o desempenho;
- As melhores acurácias médias ocorrem no modelo com detecção reativa (abordagem esquece passado após detecção). O modelo com detecção proativa *Pattern Mean Shift* (ambas as abordagens) também tem boas acurácias, em alguns casos até melhores, porém, demorando muito mais tempo;
- O melhor desempenho computacional médio ocorre no modelo com detecção reativa (abordagem só treina com detecção), e os piores tempos ocorrem no modelo com detecção proativa *Group Label* (abordagem só treina com detecção);
- As piores acurácias médias ocorrem com o modelo sem detecção e com o modelo com detecção proativa *Group Label* (ambas as abordagens);
- MELHOR ESCOLHA: detecção reativa (abordagem esquece passado após detecção), devido à boa acurácia e desempenho computacional razoável;

- PIOR ESCOLHA: detecção proativa *Group Label* (ambas as abordagens), devido à péssima acurácia e desempenho computacional.

Nebraska:

- As diferenças nos valores de acurácia comparando as redes com 5 ou 10 neurônios na camada escondida são muito pequenas na maioria das vezes, mas os valores para 5 neurônios são ligeiramente melhores. Quanto ao desempenho computacional, há diferença significativa para determinadas configurações e em alguns casos as redes com 5 neurônios têm um desempenho computacional superior, e em outros casos, inferior.
- As melhores acurácias médias ocorrem nos modelos com detecção reativa (ambas as abordagens) e o melhor desempenho computacional médio ocorre no modelo sem detecção, sendo que os tempos dos modelos com detecção reativa (ambas as abordagens) também são consideravelmente bons;
- As piores acurácias médias ocorrem nos modelos com detecção proativa *Group Label* (ambas as abordagens), e os piores desempenhos computacionais médios ocorrem nos modelos com detecção proativa *Group Label* (ambas as abordagens) e no modelo com detecção proativa *Pattern Mean Shift* (abordagem só treina com detecção);
- MELHOR ESCOLHA: detecção reativa (ambas as abordagens), devido à boa acurácia e desempenho computacional razoável;

  - PIOR ESCOLHA: detecção proativa *Group Label* (ambas as abordagens), devido à fraca acurácia e desempenho computacional.

Electricity:

- As diferenças nos valores de acurácia comparando as redes com 5 ou 10 neurônios na camada escondida são muito pequenas na maioria das vezes, mas os valores para 5 neurônios são ligeiramente melhores. Quanto ao desempenho computacional, há diferença significativa para determinadas configurações, sendo que

as redes com 10 neurônios apresentam desempenho computacional superior na maioria das vezes;

- A melhor acurácia média ocorre nos modelos com detecção reativa (ambas as abordagens) e a pior, no modelo sem detecção;
- O melhor desempenho computacional médio ocorre no modelo sem detecção e o pior, no modelo com detecção proativa *Group Label* (abordagem só treina com detecção) e no modelo com detecção proativa *Pattern Mean Shift* (abordagem só treina com detecção);
- MELHOR ESCOLHA: detecção reativa (ambas as abordagens), devido à boa acurácia e desempenho computacional razoável;
- PIOR ESCOLHA: Sem detecção, devido à péssima acurácia.

Poker Hand e Cover Type:

- As diferenças nos valores de acurácia comparando as redes com 5 ou 10 neurônios na camada escondida são muito pequenas na maioria das vezes, mas os valores para 10 neurônios são ligeiramente melhores. Quanto ao desempenho computacional, observam-se diferenças significativas, sendo que algumas configurações têm o desempenho computacional melhor com 5 neurônios e outras, com 10 neurônios;
- As melhores acurácias ocorrem nos modelos com detecção reativa (ambas as abordagens) e com detecção proativa *Pattern Mean Shift* (ambas as abordagens). Já o melhor desempenho computacional ocorre no modelo sem detecção;
- Tanto a pior acurácia média quanto o pior desempenho computacional médio ocorrem no modelo com detecção proativa *Group Label* (ambas as abordagens);
- MELHOR ESCOLHA: detecção reativa (ambas as abordagens), devido à boa acurácia e desempenho computacional razoável;
- PIOR ESCOLHA: detecção proativa *Group Label* (ambas as abordagens), devido à fraca acurácia e desempenho computacional.

A tabela 15 mostra o resultado consolidado para o experimento com as bases de dados. Na parte superior está o resultado detalhado, onde foi atribuído 1 ponto positivo cada vez que um modelo foi mencionado como melhor (ou como um dos melhores), em termos de acurácia ou desempenho computacional, para uma base de dados. Ao contrário, cada vez que um modelo foi mencionado como pior (ou como um dos piores) em termos de acurácia ou desempenho computacional, foi atribuído 1 ponto negativo. Na parte inferior da tabela (Geral), cada vez que um modelo foi mencionado como a melhor escolha para a base de dados, foram atribuídos 2 pontos positivos e cada vez que foi mencionado como a pior escolha, 2 pontos negativos. As células das colunas referentes a cada base de dados foram coloridas de verde para somas de pontos positivas, amarelo para somas neutras e vermelho para somas negativas, sendo que as células em branco são aquelas que não receberam nenhuma pontuação, nem negativa nem positiva. A coluna “PONTUAÇÃO” foi colorida com uma escala de cores que varia de vermelho (valores mais baixos) a verde (valores mais altos).

Tabela 15. Resultados consolidados do experimento com as bases de dados.

Detalhado	**SEA**	**CB**	**Neb**	**Elec**	**Pok**	**Cov**		**PONTUAÇÃO**
MLP simples (sem detecção)	0	-1	1	0	1	1		**2**
MLP + DE reativa - Esquece pass.	2	1	2	1	1	1		**8**
MLP + DE reativa. - Só treina c/ det.	1	1	2	1	1	1		**7**
MLP + DE proat. GL - Esquece pass.	-1	-1	-2		-2	-2		**-8**
MLP + DE proat. GL - Só treina c/ det.	-1	-2	-2	-1	-2	-2		**-10**
MLP + DE proat. PMS - Esquece pass.	1	1			1	1		**4**
MLP + DE proat. PMS - Só treina c/ det.	1	1	-1	-1	1	1		**2**
Geral	**SEA**	**CB**	**Neb**	**Elec**	**Pok**	**Cov**		**PONTUAÇÃO**
MLP simples (sem detecção)	-2			-2				**-4**
MLP + DE reativa - Esquece pass.	2	2	2	2	2	2		**12**
MLP + DE reativa. - Só treina c/ det.			2	2	2	2		**8**
MLP + DE proat. GL - Esquece pass.		-2	-2		-2	-2		**-8**
MLP + DE proat. GL - Só treina c/ det.		-2	-2		-2	-2		**-8**
MLP + DE proat. PMS - Esquece pass.								**0**
MLP + DE proat. PMS - Só treina c/ det.								**0**

Observa-se que o padrão é similar tanto para os resultados detalhados quanto para os resultados gerais: os melhores modelos são, nesta ordem: com detecção reativa (abordagem esquece passado após detecção), com detecção reativa (abordagem só treina com detecção), com detecção proativa *Pattern Mean Shift* (abordagem esquece passado após detecção) e com detecção proativa *Pattern Mean Shift* (abordagem só treina com detecção). Os piores

modelos são, nesta ordem: com detecção proativa *Group Label* (abordagem só treina com detecção), com detecção proativa *Group Label* (abordagem esquece passado após detecção) e sem detecção.

Após este experimento, pode-se concluir que:

- Em geral, os melhores resultados obtidos foram usando algum tipo de detecção. Como as bases de dados utilizadas possuem *drift*, o procedimento de detecção ajuda na acurácia e no desempenho computacional do classificador, bastando comparar os resultados com a abordagem sem detecção;
- Comparando as abordagens proativas, a abordagem *Pattern Mean Shift* teve a acurácia e o desempenho computacional muito superiores do que a *Group Label*. Isto se deve ao fato da abordagem *Group Label* ser muito mais agressiva com relação ao treinamento dos classificadores, não sendo a melhor opção para estas bases de dados escolhidas;
- Comparando as abordagens "Esquece passado após detecção" com "Só treina com detecção", observa-se que a acurácia da primeira é sensivelmente superior, enquanto o desempenho computacional da segunda é superior na maioria dos casos. Obviamente, a primeira abordagem produz uma melhor acurácia devido ao treino mais exaustivo, contudo, tomando mais tempo;
- Analisando as diferenças entre o número de neurônios na camada escondida, verificou-se que, para as bases de dados *Poker Hand* e *Cover Type*, a acurácia com 10 neurônios é superiori do que a com 5 neurônios sem que isso acarrete em um custo computacional muito superior. Nas demais bases de dados, não foram observadas diferenças muito significativas. Isto pode ser explicado pelo fato dessas bases serem maiores e mais complexas sendo, para estes casos, mais adequado utilizar uma rede neural mais complexa.

Este capítulo apresentou os resultados dos experimentos realizados com o método de detecção proposto. O próximo capítulo apresenta os resultados dos experimentos realizados com o modelo neuroevolutivo NEVE.

6
Experimentos com o Modelo Neuroevolutivo Proposto

Este capítulo apresenta os resultados dos experimentos realizados com o modelo neuroevolutivo proposto, NEVE. Foram realizados experimentos usando quatro abordagens diferentes do NEVE, detalhados nas seções a seguir.

6.1.
Visão Geral

Com o objetivo de checar a habilidade do modelo NEVE em aprender em ambientes não estacionários e também de verificar as melhores variações e configurações dos modelos em termos de acurácia e desempenho computacional, foram usadas seis diferentes bases de dados com as quais foram efetuadas diferentes simulações e cenários. As bases de dados utilizadas foram as mesmas já descritas no experimento com o método de detecção proposto na seção 5.2.1, assim como o número de blocos e tamanho do bloco utilizados. Para os experimentos, foram utilizadas as 4 variações do modelo NEVE já descritas no capítulo 4: NEVE Sem Detecção, DE-NEVE Reativo, DE-NEVE Proativo *Group Label* e DE-NEVE Proativo *Pattern Mean Shift*.

6.2.
Detalhes de Execução

Todas as execuções começam em t=0 e terminam quando T consecutivos blocos de padrões são apresentados para treinamento e teste, sendo que cada bloco pode sofrer diferentes cenários de *concept drift*, com taxas e naturezas desconhecidas.

Conforme já apresentado no Capítulo 4, o algoritmo quântico AEIQ-BR evolui a topologia da nova rede neural, que é criada seguindo os critérios de cada variação do modelo NEVE. Para determinar o número de variáveis de entrada, o AEIQ-BR faz uma seleção entre todas as variáveis de cada base de dados, sendo 3 entradas para a base *SEA Concepts*, 8 entradas para a *Nebraska*, 8 entradas para a *Electricity*, 2 entradas para a *Rotating Checkboard*,

10 entradas para a *Poker* e 54 entradas para a *CoverType*, representando os atributos de entrada de cada base de dados. Para todas as bases de dados, foi usada uma única camada escondida, cujo número de neurônios é evoluído, tendo como valor máximo um parâmetro informado pelo usuário e, neste experimento, foram utilizados os valores 5 e 10. O número de neurônios da camada de saída é igual ao número de saídas possíveis de cada base de dados. Os pesos sinápticos e as funções de ativação da camada escondida e da camada de saída também são evoluídos pelo AEIQ-BR.

Os parâmetros dos algoritmos evolutivos AEIQ-R (usado para evoluir os pesos de votação para cada classificador do comitê) e AEIQ-BR (usado para evoluir a topologia da nova rede neural criada) foram os mesmos utilizados por [Abs da Cruz, 2007] e [Pinho, 2010].

O método de votação utilizado é definido pelo usuário dentre os 3 métodos implementados e já detalhados no Capítulo 4: combinação linear, votação majoritária ponderada e votação majoritária simples. O número máximo de redes do comitê também é um parâmetro definido pelo usuário e, neste experimento, foram utilizados 3 valores: 5, 10 e ilimitado.

Assim, para cada base de dados utilizada no experimento, foram utilizadas 72 diferentes configurações do modelo (4 x 3 x 3 x 2), representando cada uma das possíveis combinações dos parâmetros que se deseja avaliar, conforme ilustra a Tabela 16. Para cada configuração, foram executadas 30 simulações e foi calculada a média da acurácia e do tempo computacional destas execuções.

Tabela 16. Configurações usadas nos experimentos.

Parâmetro	Possíveis Valores
Variação do modelo	NEVE Sem Detecção
	DE-NEVE Reativo
	DE-NEVE Proativo *Group Label*
	DE-NEVE Proativo *Pattern Mean*
Método de votação	Combinação Linear
	Votação Majoritária Ponderada
	Votação Majoritária Simples
Tamanho do comitê	5
	10
	Ilimitado
Número de neurônios na camada escondida	5
	10

6.3.
Resultados dos Experimentos

Neste estudo, buscou-se investigar a diferença entre a acurácia e o desempenho computacional obtidos usando cada uma das 4 variações do modelo NEVE, além do impacto da variação dos parâmetros método de votação, tamanho do comitê e número de neurônios na camada escondida. Desta forma, o objetivo é analisar como estas modificações impactam nos resultados dos modelos para cada uma das bases de dados.

As Tabelas 17 a 24 exibem os resultados dos experimentos realizados considerando a acurácia e o desempenho computacional medido em segundos e, a seguir, é realizada uma análise dos resultados. Cabe ressaltar que só foi possível analisar o tempo de execução para as bases de dados *SEA Concepts*, *Nebraska*, *Electricity* e *Rotating Checkboard*, pois as demais bases utilizadas (*Poker Hand* e *Cover Type*), por terem um tamanho considerável, necessitaram da paralelização das execuções em diversos computadores, tornando a comparação do tempo de execução entre as simulações inviável. Em todos os casos, o desvio padrão observado foi inferior a 2%.

Tabela 17. Resultados da base de dados *SEA Concepts*.

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
	Sem Detecção						**Sem Detecção**					
comblin	0,9052	0,9053	0,9052	0,9042	0,8703	0,8706	270	311	356	364	527	549
majpond	0,9031	0,9036	0,8948	0,8906	0,8347	0,8510	438	401	421	424	590	603
majsimples	0,9019	0,9019	0,9003	0,8994	0,8631	0,8634	378	367	387	378	486	521
	Reativo						**Reativo**					
comblin	0,9042	0,9038	0,9047	0,9053	0,8697	0,8685	289	328	374	379	536	568
majpond	0,9030	0,9019	0,8961	0,8821	0,8645	0,8590	444	408	429	429	611	599
majsimples	0,9015	0,9018	0,8993	0,8984	0,8637	0,8627	279	274	282	280	292	289
	Proativo GL						**Proativo GL**					
comblin	0,7249	0,7256	0,7344	0,7357	0,7521	0,7523	364	377	382	374	469	459
majpond	0,6922	0,6938	0,6845	0,6826	0,5897	0,5868	403	414	447	443	607	605
majsimples	0,7592	0,7613	0,7692	0,7670	0,8108	0,7978	288	288	290	290	307	310
	Proativo PMS						**Proativo PMS**					
comblin	0,8715	0,8707	0,8699	0,8702	0,8459	0,8463	276	334	366	394	568	567
majpond	0,8722	0,8756	0,8592	0,8550	0,6111	0,6375	422	411	438	444	612	598
majsimples	0,8684	0,8684	0,8687	0,8685	0,8461	0,8459	288	296	297	287	302	305

Tabela 18. Resultados da base de dados *Nebraska*.

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
	Sem Detecção						**Sem Detecção**					
comblin	0,6919	0,6875	0,6998	0,7003	0,7013	0,6999	1052	1047	1030	1041	1277	1283
majpond	0,6749	0,6756	0,6783	0,6725	0,7001	0,6958	1169	1161	1170	1178	1668	1630
majsimples	0,6928	0,6954	0,6969	0,6935	0,6973	0,6980	1058	1056	951	863	1196	1231
	Reativo						**Reativo**					
comblin	0,6885	0,6865	0,6988	0,6995	0,7020	0,7010	1022	1025	1017	1008	1252	1253
majpond	0,6761	0,6714	0,6812	0,6756	0,6922	0,7005	1124	1127	1122	1125	1620	1620
majsimples	0,6927	0,6958	0,6980	0,6970	0,6965	0,6973	800	792	794	737	753	749
	Proativo GL						**Proativo GL**					
comblin	0,5330	0,5339	0,5322	0,5268	0,5486	0,5499	1020	1024	1022	1037	1266	1261
majpond	0,4794	0,4778	0,4320	0,4396	0,3345	0,3416	1137	1145	1154	1140	1594	1437
majsimples	0,5514	0,5529	0,4892	0,4984	0,5611	0,5674	660	668	688	689	785	786
	Proativo PMS						**Proativo PMS**					
comblin	0,6526	0,6483	0,6662	0,6649	0,6788	0,6772	1047	1055	1034	1034	1262	1261
majpond	0,6304	0,6326	0,6282	0,6289	0,3561	0,3691	1149	1147	1157	1165	1592	1591
majsimples	0,6686	0,6679	0,6698	0,6695	0,6860	0,6875	824	816	817	758	768	761

Tabela 19. Resultados da base de dados *Electricity*.

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
comblin	0,5439	0,5352	0,5396	0,5472	0,5575	0,5009	1428	1451	1523	1645	2907	2490
majpond	0,5536	0,5534	0,5998	0,5911	0,6203	0,6199	1673	1660	1694	1691	2947	2754
majsimples	0,5458	0,5265	0,5523	0,5583	0,4365	0,4869	1481	1444	1423	1486	2495	2592
comblin	0,5348	0,5324	0,5416	0,5518	0,4999	0,5175	1366	1376	1389	1393	2431	2599
majpond	0,5649	0,5570	0,5897	0,5878	0,6304	0,6215	1671	1629	1580	1592	2793	2829
majsimples	0,5223	0,5245	0,5544	0,5262	0,4479	0,4486	1124	1116	1101	1089	1384	1416
comblin	0,7077	0,7072	0,7121	0,7141	0,7155	0,7161	1391	1390	1443	1456	2506	2513
majpond	0,7145	0,7158	0,7098	0,7168	0,6554	0,6600	1614	1543	1557	1579	2712	2604
majsimples	0,6863	0,6811	0,7004	0,6961	0,7112	0,6892	1049	1036	1031	1050	1335	1348
comblin	0,5301	0,5328	0,5338	0,5361	0,5387	0,4603	1382	1363	1347	1352	2308	2626
majpond	0,5509	0,5529	0,5734	0,5804	0,5761	0,5766	1703	1615	1562	1561	2566	2601
majsimples	0,5150	0,5176	0,5422	0,5374	0,4435	0,4609	1121	1083	1077	1067	1297	1282

Tabela 20. Resultados da base de dados *Poker Hand*.

	Acurácia					
Escondida	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim
	Sem Detecção					
comblin	0,5938	0,5976	0,6081	0,5957	0,6294	0,6255
majpond	0,5851	0,5880	0,6009	0,5981	0,6434	0,6328
majsimples	0,5646	0,5604	0,5407	0,5375	0,5308	0,5430
	Reativo					
comblin	0,5938	0,5976	0,6081	0,5957	0,6294	0,6255
majpond	0,5912	0,5829	0,6096	0,6044	0,6377	0,6364
majsimples	0,5655	0,5590	0,5398	0,5369	0,5418	0,5448
	Proativo GL					
comblin	0,2453	0,2457	0,3665	0,3819	0,4064	0,3485
majpond	0,2283	0,2187	0,4145	0,4146	0,3835	0,3950
majsimples	0,4694	0,4709	0,4752	0,4617	0,3952	0,4140
	Proativo PMS					
comblin	0,5789	0,5681	0,5617	0,5678	0,5777	0,5710
majpond	0,5655	0,5676	0,5648	0,5686	0,5902	0,5882
majsimples	0,5456	0,5482	0,5252	0,5236	0,5058	0,5282

Tabela 21. Resultados da base de dados *Cover Type*.

	Acurácia					
Escondida	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim
	Sem Detecção					
comblin	0,7430	0,7290	0,7483	0,7341	0,5799	0,5282
majpond	0,7530	0,7404	0,7292	0,7097	0,5949	0,5837
majsimples	0,6925	0,7034	0,7007	0,7093	0,4971	0,4926
	Reativo					
comblin	0,7430	0,7290	0,7483	0,7341	0,5904	0,5424
majpond	0,7464	0,7461	0,7426	0,7346	0,6055	0,5644
majsimples	0,6935	0,7106	0,7026	0,6934	0,5097	0,5170
	Proativo GL					
comblin	0,5114	0,5101	0,5372	0,5424	0,4718	0,4855
majpond	0,4976	0,4876	0,5381	0,5488	0,5008	0,4958
majsimples	0,5592	0,5704	0,5476	0,5639	0,4620	0,4234
	Proativo PMS					
comblin	0,7225	0,6986	0,6982	0,7159	0,5899	0,5273
majpond	0,7169	0,7248	0,7044	0,7098	0,5629	0,5462
majsimples	0,7145	0,6978	0,6975	0,6873	0,4852	0,4947

Tabela 22. Resultados da base de dados *Rot. Checkboard* (Constante).

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
	Sem Detecção						**Sem Detecção**					
comblin	0,6098	0,6078	0,6178	0,6187	0,5663	0,5641	580	606	610	600	711	684
majpond	0,6136	0,6126	0,6006	0,6016	0,5107	0,5077	630	639	660	661	921	995
majsimples	0,6104	0,6088	0,6044	0,6065	0,5112	0,5099	696	650	654	662	848	833
	Reativo						**Reativo**					
comblin	0,6087	0,6088	0,6162	0,6203	0,5687	0,5658	557	563	558	551	684	689
majpond	0,6123	0,6173	0,6080	0,6003	0,5095	0,5084	623	615	613	646	867	849
majsimples	0,6155	0,6141	0,6078	0,6111	0,5164	0,5125	426	428	433	438	521	522
	Proativo GL						**Proativo GL**					
comblin	0,5286	0,5293	0,5304	0,5323	0,5207	0,5187	769	773	784	781	1081	1063
majpond	0,5303	0,5305	0,5322	0,5331	0,5058	0,5058	944	932	984	1007	2847	2837
majsimples	0,5435	0,5420	0,5438	0,5489	0,5224	0,5227	611	596	599	596	674	679
	Proativo PMS						**Proativo PMS**					
comblin	0,6017	0,6094	0,6143	0,6240	0,5581	0,5563	540	538	549	556	669	677
majpond	0,6128	0,6142	0,5935	0,5952	0,5071	0,5100	641	646	643	632	838	877
majsimples	0,6067	0,5981	0,6124	0,6129	0,5188	0,5162	460	466	453	448	517	518

Tabela 23. Resultados da base de dados *Rot. Checkboard* (Exponencial).

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
	Sem Detecção						**Sem Detecção**					
comblin	0,6109	0,6089	0,6167	0,6124	0,5722	0,5646	570	553	552	545	653	646
majpond	0,6147	0,6143	0,5998	0,5897	0,5055	0,5078	594	625	649	654	850	874
majsimples	0,6114	0,6150	0,6023	0,6081	0,5237	0,5221	632	630	637	611	804	825
	Reativo						**Reativo**					
comblin	0,6054	0,6075	0,6145	0,6110	0,5689	0,5659	558	567	608	594	725	720
majpond	0,6090	0,6097	0,5954	0,5958	0,5053	0,5096	659	648	647	642	788	783
majsimples	0,6081	0,6126	0,5975	0,6048	0,5289	0,5263	407	405	402	390	464	479
	Proativo GL						**Proativo GL**					
comblin	0,5238	0,5224	0,5217	0,5194	0,5127	0,5126	735	747	758	758	1071	1082
majpond	0,5221	0,5193	0,5193	0,5210	0,4968	0,4963	947	943	977	967	2754	2788
majsimples	0,5297	0,5329	0,5355	0,5354	0,5092	0,5118	607	610	611	613	704	696
	Proativo PMS						**Proativo PMS**					
comblin	0,6027	0,6040	0,6128	0,6185	0,5656	0,5624	588	584	589	597	763	756
majpond	0,6095	0,6110	0,5899	0,5966	0,5008	0,5021	677	671	681	667	877	818
majsimples	0,6016	0,6010	0,6049	0,6089	0,5290	0,5286	419	419	420	419	475	476

Tabela 24. Resultados da base de dados *Rot. Checkboard* (Pulso).

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
	Sem Detecção						**Sem Detecção**					
comblin	0,6483	0,6466	0,6705	0,6517	0,6161	0,6034	599	591	566	589	736	765
majpond	0,6551	0,6664	0,6605	0,6474	0,5032	0,5123	662	676	683	685	919	846
majsimples	0,6189	0,6508	0,6341	0,6550	0,5988	0,6072	592	593	597	596	764	781
	Reativo						**Reativo**					
comblin	0,6446	0,6429	0,6558	0,6660	0,6169	0,6011	532	534	533	531	667	685
majpond	0,6642	0,6580	0,6579	0,6591	0,5161	0,5130	622	612	599	606	815	813
majsimples	0,6353	0,6281	0,6463	0,6487	0,6042	0,5898	408	413	440	441	537	531
	Proativo GL						**Proativo GL**					
comblin	0,5122	0,5135	0,5143	0,5124	0,5095	0,5052	749	740	742	683	819	778
majpond	0,5157	0,5146	0,5181	0,5110	0,4997	0,4977	696	663	661	654	1955	1772
majsimples	0,5209	0,5258	0,5243	0,5218	0,4740	0,4591	371	368	340	341	390	392
	Proativo PMS						**Proativo PMS**					
comblin	0,6254	0,6328	0,6297	0,6461	0,6227	0,5905	538	557	575	570	679	674
majpond	0,6302	0,6275	0,6246	0,6259	0,5042	0,5064	650	650	656	636	841	860
majsimples	0,6111	0,6052	0,6187	0,6056	0,5962	0,5993	457	444	443	439	535	545

Tabela 25. Resultados da base de dados *Rot. Checkboard* (Sinusoidal).

	Acurácia						Tempo de Execução					
Escondida	5	10	5	10	5	10	5	10	5	10	5	10
Comitê	5	5	10	10	Ilim	Ilim	5	5	10	10	Ilim	Ilim
	Sem Detecção						**Sem Detecção**					
comblin	0,6286	0,6255	0,6355	0,6337	0,5797	0,5654	563	562	546	535	559	488
majpond	0,6479	0,6347	0,6263	0,6174	0,5050	0,5023	428	413	404	427	582	590
majsimples	0,6088	0,6205	0,6192	0,6178	0,5114	0,5099	391	406	442	433	551	527
	Reativo						**Reativo**					
comblin	0,6310	0,6313	0,6305	0,6331	0,5757	0,5868	584	574	579	593	708	671
majpond	0,6368	0,6292	0,6240	0,6221	0,5058	0,5035	590	589	595	594	790	788
majsimples	0,6080	0,6199	0,6241	0,6233	0,5090	0,5010	404	404	399	395	421	387
	Proativo GL						**Proativo GL**					
comblin	0,5160	0,5118	0,5151	0,5152	0,5101	0,5089	423	422	428	428	564	565
majpond	0,5089	0,5154	0,5172	0,5145	0,4985	0,4991	677	826	899	932	2634	2926
majsimples	0,5178	0,5155	0,5151	0,5169	0,4737	0,4661	615	610	601	602	689	679
	Proativo PMS						**Proativo PMS**					
comblin	0,6134	0,6133	0,6367	0,6275	0,5632	0,5778	617	606	606	611	717	719
majpond	0,6229	0,6318	0,6087	0,6026	0,5047	0,5031	649	608	611	612	804	800
majsimples	0,6042	0,5954	0,6172	0,6153	0,5083	0,5173	432	439	438	430	488	486

A análise das Tabelas 17 a 25 mostra que:

- Em geral, as abordagens “Sem detecção”, “Reativa” e “Proativa *Pattern Mean Shift*” tiveram as melhores acurácias e a abordagem “Proativa *Group Label*” teve as piores acurácias. É interessante destacar que na base *Electricity*, a abordagem “Proativa *Group Label*” teve disparadamente as melhores acurácias. Na base *Cover Type*, além dos bons resultados das abordagens “Sem detecção” e “Reativa”, observam-se boas acurácias também na “Proativa *Group Label*”;

- Considerando o desempenho computacional, as abordagens “Sem detecção”, “Reativa” e “Proativa *Pattern Mean Shift*” apresentaram os melhores tempos computacionais, e a abordagem “Proativa *Group Label*”, o pior. Observou-se, no entanto, que a base de dados também tem grande influência neste critério: a mais lenta foi a *Electricity*, que é a base que possui o maior número de atributos e também maior quantidade de blocos dentre as bases avaliadas;

- Percebe-se que os melhores métodos de votação em termos de acurácia são, nesta ordem: combinação linear, seguido por majoritária ponderada e majoritária simples (empatados). Isto mostra que o algoritmo quântico está contribuindo positivamente na acurácia do modelo através da determinação dos pesos de votação das redes. Possivelmente, o arredondamento precoce feito na majoritária ponderada contribuiu para a sua acurácia média ficar abaixo da combinação linear;

- Quanto ao desempenho computacional, o método de votação que teve execuções mais rápidas foi a majoritária simples, o que pode ser explicado pelo fato deste método não realizar a determinação de pesos via algoritmo quântico;

- Observa-se que, em geral, a estratégia de comitê ilimitado possui acurácia média menor do que os comitês limitados. Não foi observada diferença significativa de acurácia entre os comitês de 5 e 10 redes, o que é um ponto positivo, pois as estratégias de comitê limitado apresentaram também o pior desempenho computacional, como já era esperado. Provavelmente, o motivo do comitê ilimitado apresentar pior acurácia se deve ao fato do

algoritmo quântico ter dificuldade na determinação de pesos quando há muitas redes: basta observar que, nas bases de dados utilizadas, há no mínimo 400 blocos, o que possibilita comitês de 400 redes para o caso ilimitado;

- Não foram observadas diferenças substanciais nem na acurácia média nem no desempenho computacional médio considerando as estratégias de 5 e 10 neurônios na camada escondida, o que provavelmente se deve ao fato de não se necessitar de redes individuais tão complexas quando utilizada a abordagem de comitê.

Considerando as bases binárias, também foi feita uma análise adicional considerando apenas as bases SEA, *Nebraska* e *Electricity*, ilustrada pela figura 41. Neste caso, as bases *Rotating Checkboard* foram desconsideradas pois possuem tamanho de bloco de treino e de teste muito diferentes, o que compromete o tempo computacional para a abordagem "Proativa *Group Label*". Observou-se que, considerando este subconjunto de bases, o tempo computacional da abordagem "Sem detecção" é superior aos demais, enquanto que as abordagens com algum tipo de detecção apresentam um tempo computacional médio similar para uma mesma base de dados. Isto mostra que o mecanismo de detecção proposto contribui para reduzir o tempo de execução médio dos modelos.

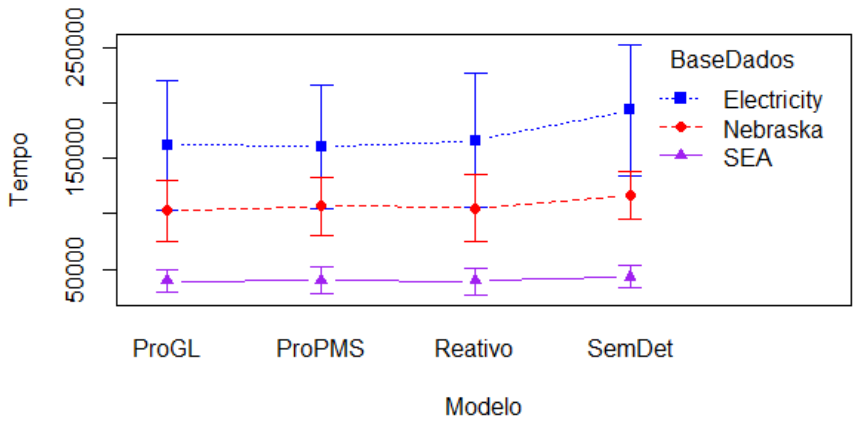


Figura 41. Análise Comparativa do Tempo de Execução

Para fins de comparação de acurácia, também foram realizadas simulações com os modelos DWM, Learn++.NSE e RCD, já apresentados no Capítulo 2. Foram utilizados 3 diferentes detectores de *drift* para o algoritmo RCD: DDM, EDDM e ECDD, também já apresentados no Capítulo 2. Estas

simulações foram realizadas com auxílio do MOA [MOA, 2015], um *framework open source* para mineração de dados que inclui diversos algoritmos de aprendizagem implementados para classificação, regressão, clusterização, detecção de *concept drift*, entre outros; e ferramentas para avaliação. Relacionado ao projeto WEKA, o MOA também é escrito na linguagem Java. Para esta comparação, foram utilizadas todas as bases de dados apresentadas (com exceção da base de dados *Rotating Checkboard*) utilizando o mesmo tamanho de bloco escolhido para as simulações do NEVE. A base de dados *Rotating Checkboard* não foi utilizada para comparação porque, conforme explicado anteriormente, nas simulações do NEVE foram utilizados diferentes tamanhos de bloco para treino e teste, e o MOA não permite esta personalização. Também não foi possível realizar simulações com os algoritmos SEA e DDD, também apresentados no Capítulo 2, pois suas implementações não estão disponíveis ou não estão funcionando no MOA. Para realizar uma comparação mais coerente com o NEVE e para descartar a influência do classificador base na acurácia do modelo, em todos os demais modelos também foram utilizadas redes neurais MLP como classificadores base. Todos os parâmetros dos modelos foram utilizados conforme indicado pelos respectivos autores.

As Tabelas 26, 27 e 28 apresentam os resultados da melhor configuração, da configuração média e da pior configuração (em termos de acurácia) de cada variação do NEVE, respectivamente, comparadas com o resultado da execução dos demais modelos. Para cada base de dados, foi destacado em verde o modelo com maior acurácia, em azul o segundo lugar e em vermelho o modelo com pior acurácia.

Tabela 26. Comparação dos resultados: melhor caso do NEVE x outros modelos.

Melhor	NEVE (Sem)	NEVE (Reat)	NEVE (ProGL)	NEVE (ProPMS)	RCD (DDM)	RCD (EDDM)	RCD (ECDD)	DWM	Learn++ .NSE
SEA	90,53	90,53	81,08	87,56	82,50	80,22	80,37	60,67	35,31
Nebraska	70,13	70,20	56,74	68,76	40,28	46,16	64,00	50,08	30,05
Electricity	62,03	63,04	71,68	58,04	52,41	48,22	49,41	67,16	42,46
Poker	64,34	63,77	47,52	59,02	60,37	59,30	40,44	73,78	47,76
Covtype	75,30	74,83	57,04	72,48	46,34	33,78	57,19	81,91	28,05

Tabela 27. Comparação dos resultados: média do NEVE x outros modelos.

Média	NEVE (Sem)	NEVE (Reat)	NEVE (ProGL)	NEVE (ProPMS)	RCD (DDM)	RCD (EDDM)	RCD (ECDD)	DWM	Learn++ .NSE
SEA	88,71	88,83	72,33	83,62	82,50	80,22	80,37	60,67	35,31
Nebraska	69,18	69,17	49,72	62,68	40,28	46,16	64,00	50,08	30,05
Electricity	54,83	54,19	70,05	53,78	52,10	48,22	49,41	67,16	42,46
Poker	58,75	58,89	37,42	55,81	60,37	59,30	40,44	73,78	47,76
Covtype	66,49	66,96	51,41	64,97	46,34	33,78	57,19	81,91	28,05

Tabela 28. Comparação dos resultados: pior caso do NEVE x outros modelos.

Pior	NEVE (Sem)	NEVE (Reat)	NEVE (ProGL)	NEVE (ProPMS)	RCD (DDM)	RCD (EDDM)	RCD (ECDD)	DWM	Learn++ .NSE
SEA	83,47	85,90	58,68	61,11	82,50	80,22	80,37	60,67	35,31
Nebraska	67,25	67,14	32,45	35,61	40,28	46,16	64,00	50,08	30,05
Electricity	43,65	44,79	65,54	44,35	52,41	48,22	49,41	67,16	42,46
Poker	53,08	53,69	21,87	50,58	60,37	59,30	40,44	73,78	47,76
Covtype	49,26	50,97	42,34	48,52	46,34	33,78	57,19	81,91	28,05

Verifica-se que, analisando a Tabela 26, o NEVE obteve o melhor resultado em 3 bases de dados e o segundo melhor em 4. Em nenhum caso obteve os piores resultados. Aparentemente, as abordagens “Sem Detecção” e “Reativa” possibilitam resultados uniformemente superiores em termos de acurácia. Observa-se que a abordagem “Proativa *Group Label*” tem melhores resultados para a base de dados *Electricity*. A Tabela 27 mostra que os resultados médios do NEVE são os melhores em 3 bases de dados e os segundos melhores também em 3. Isso demonstra que os resultados médios do NEVE não estão tão distantes dos melhores resultados. Por fim, mesmo considerando os piores resultados (Tabela 28), ainda assim o NEVE é o melhor modelo em 2 bases de dados e o segundo melhor em 3, sendo o pior modelo em apenas 1 base de dados. O que se nota em forma geral é que o modelo DWM é o principal competidor do NEVE em termos de acurácia.

Após a análise destes experimentos comparativos, destaca-se que o NEVE obteve bons resultados sem necessidade de um método de detecção, mas que a sua adição, em muitos casos, possibilitou ganhos substanciais de acurácia e de desempenho computacional. Este fato reforça que a abordagem por comitê neuroevolutivo foi uma escolha robusta para situações em que as bases de dados estão sujeitas a mudanças repentinas de comportamento.

Este capítulo apresentou os resultados dos experimentos realizados com o modelo neuroevolutivo NEVE com suas quatro variações, usando diversas bases de dados. O próximo capítulo apresenta as conclusões do trabalho, bem como os trabalhos futuros que podem ser realizados.

7
Conclusão

Este trabalho apresentou um novo modelo neuroevolutivo com inspiração quântica, baseado em um comitê de redes neurais do tipo *Multi-Layer Perceptron* (MLP), para a aprendizagem em ambientes não estacionários, denominado NEVE (*Neuro-EVolutionary Ensemble*). Também foi apresentado um novo mecanismo de detecção de *concept drift*, denominado DetectA (*Detect Abrupt*) com a capacidade de detectar mudanças tanto de forma proativa quanto de forma reativa. Este método pode ser utilizado em conjunto com o NEVE ou com outros modelos de aprendizagem, possibilitando a reação e o ajuste do modelo sempre que forem detectadas mudanças nos dados. Ainda, foi proposta uma metodologia para geração de bases de dados, com ou sem *drift*, no vetor de médias e na matriz de covariâncias condicionais.

O uso do algoritmo evolutivo com inspiração quântica híbrido AEIQ-BR em conjunto com o modelo NEVE possibilita a geração automática de novos classificadores para o comitê, determinando a topologia mais adequada para a nova rede, a seleção das variáveis de entrada mais apropriadas e a determinação de todos os pesos da rede neural MLP. O algoritmo evolutivo com inspiração quântica e representação real AEIQ-R é responsável pela determinação dos pesos de votação de cada rede neural membro do comitê.

Foram implementadas quatro diferentes abordagens do NEVE: Sem detecção, com detecção reativa, com detecção proativa - *Group Label* e com detecção proativa - *Pattern Mean Shift*. Estas abordagens se diferem uma da outra pela forma de detectar e tratar os *drifts* ocorridos.

Foram realizados dois experimentos com o método DetectA: uma análise de sensibilidade com base na alteração de diversos parâmetros a fim de compreender a influência de cada uma destas variáveis no desempenho do método e um experimento que utilizou o método de detecção e redes neurais MLP para aprendizagem em bases de dados reais e artificiais. A análise de sensibilidade mostrou que o detector se mostrou robusto e eficiente para bases de dados de alta dimensionalidade, blocos de tamanho intermediário, bases de dados com qualquer proporção de *drift* e com qualquer balanceamento de classes. O segundo experimento indicou que, em geral, os melhores resultados

obtidos foram usando algum tipo de detecção, sendo que dentre as abordagens proativas, a abordagem *Pattern Mean Shift* teve a acurácia e o desempenho computacional superiores. Também mostrou que a abordagem de treinamento "Esquece passado após detecção", que retreina o classificador a cada novo bloco de dados com todos os blocos passados e no caso da detecção de um *drift*, realiza o treinamento apenas do ponto de detecção em diante tem melhor acurácia, mas pior desempenho computacional comparada à abordagem "Só treina com detecção", que somente retreina o classificador quando há detecção de *drift*, utilizando para treinamento o bloco onde o *drift* foi detectado. Além disso, mostrou que para bases de dados maiores e mais complexas, é mais adequado utilizar uma MLP com maior número de neurônios na camada escondida.

As quatro diferentes abordagens implementadas do modelo NEVE também foram utilizadas em experimentos com bases de dados reais e artificiais a fim de avaliar quais configurações do modelo alcançaram melhores resultados. Foram variados os parâmetros: método de votação utilizado, número máximo de neurônios na camada escondida e tamanho máximo do comitê. Verificou-se que as abordagens "Sem detecção", "Reativa" e "Proativa *Pattern Mean Shift*" tiveram os melhores resultados em termos de acurácia, ao passo que a abordagem "Proativa *Group Label*" em geral teve os piores resultados. Quanto ao desempenho computacional, observou-se que as bases de dados com maior número de atributos e padrões tendem a ser mais lentas, como já esperado, e que as abordagens com melhor desempenho computacional foram a "Sem detecção", "Reativa" e "Proativa *Pattern Mean Shift*." Também foi percebido que a combinação linear é o melhor método de votação em termos de acurácia e a votação majoritária simples, a melhor em termos de desempenho computacional. A estratégia de comitê ilimitado possui acurácia e desempenho computacional piores do que os comitês limitados, não sendo observada diferença significativa entre os comitês de 5 e 10 redes nem entre as estratégias de 5 e 10 neurônios na camada escondida.

Comparando a acurácia das quatro abordagens do NEVE com outros modelos consolidados da literatura, verificou-se que o NEVE teve acurácia superior na maioria dos casos, seja considerando a configuração de melhor acurácia, a acurácia média ou a configuração de pior acurácia. Aparentemente, percebeu-se que as abordagens "Sem Detecção"e "Reativa" possibilitam resultados uniformemente superiores em termos de acurácia, porém, a adição do método de detecção, em alguns casos, possibilitou ganhos substanciais. Este

fato reforça que a abordagem por comitê neuroevolutivo foi uma escolha robusta para situações em que as bases de dados estão sujeitas a mudanças repentinas de comportamento.

Como possibilidades de trabalhos futuros, pode-se destacar:

- Implementar novos métodos de agrupamento no mecanismo de detecção, buscando aprimorar o processo de agrupamento do mecanismo de detecção proativo;
- Diminuir o espaço de busca do algoritmo quântico AEIQ-BR com o objetivo de melhorar os resultados da evolução, por exemplo, tirando do cromossomo a função de ativação das camadas da rede neural, ou a evolução do número de neurônios na camada escondida;
- Ainda no algoritmo quântico AEIQ-BR, realizar um número maior de avaliações, buscando encontrar indivíduos mais aptos no fim do processo evolutivo;
- Integrar, em uma única evolução, a criação da rede neural e a determinação de pesos de votação das redes do comitê, a fim de tornar a evolução um único processo integrado;
- Na abordagem reativa *Pattern Mean Shift* do NEVE, buscar estratégias de implementação da correção do bloco de dados na matriz de covariância condicional, além da média, quando for detectado um *drift*;
- Realizar novos experimentos em bases de dados artificialmente geradas, onde se conheça com detalhes os momentos e os tipos de *drift* ocorridos, a fim de avaliar em que tipo de *drift* o modelo NEVE e o mecanismo DetectA apresentam melhores resultados. Esta análise não é possível de ser realizada em bases de dados reais, uma vez que não se sabe ao certo quando ocorre um *drift*, e nem a sua natureza. Ainda, pretende-se utilizar o NEVE para aplicações reais, de maneira a validar o seu uso prático.

Referências

[Abs da Cruz et al., 2006] A. V. Abs da Cruz, M. M. B. R. Vellasco e M. A. C. Pacheco, “Quantum-Inspired Evolutionary Algorithm for Numerical Optimization”. 2006 IEEE International Conference on Evolutionary Computation, Vancouver. pp. 2630-2637, 2006.

[Abs da Cruz et al., 2010] A. V. Abs da Cruz, M. M. B. R. Vellasco e M. A. C. Pacheco, “Quantum-Inspired Evolutionary Algorithms applied to numerical optimization problems”. 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona. pp. 1-6, 2010.

[Abs da Cruz, 2007] A. V. Abs da Cruz, “Algoritmos evolutivos com inspiração quântica para otimização de problemas com representação numérica”. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

[Bach & Maloof, 2008] S. H. Bach e M. A. Maloof, “Paired Learners for Concept Drift”. Proc. of the 8th IEEE Int. Conf. on Data Mining (ICDM). IEEE, 23–32. Charts for Detecting Concept Drift. Pattern Recogn. Lett. 33, 2, pp. 191–198, 2012.

[Baena-García et al., 2006] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo e A. Bifet, “Early drift detection method”. Proc. of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS’06), Berlin, Germany, pp. 77–86, 2006.

[Baringhaus & Henze, 1988] Baringhaus, L.; Henze, N., "A consistent test for multivariate normality based on the empirical characteristic function". Metrika 35 (1): pp. 339–348, 1998.

[Bezdek et al., 1999] J. C. Bezdek, M. R. Pal, J. Keller e R. Krisnapuram. 1999. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[Bishop, 2006] C. M. Bishop. “Pattern Recognition and Machine Learning”. New York, Springer, 2006.

[Blum, 1997] A. Blum, “Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a CalendarScheduling Domain”. Mach. Learn. 26, 1, pp. 5–23, 1997.

[Botelho, 2014] A. C. O. P. Botelho, “Algoritmo Evolutivo com Inspiração Quântica, Representação Mista e Decaimento de Pesos Aplicado a Neuroevolução”. Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, Internal Research Reports. Número 38, Rio de Janeiro, Outubro de 2014.

[Botev et al., 2010] Z. I. Botev, J. F. Grotowski e D. P. Kroese, “Kernel density estimation via diffusion”. The Annals of Statistics, v. 38, n. 5, pp. 2916-2957, 2010.

[Bouchachia, 2011] A. Bouchachia, “Incremental Learning with Multi-Level Adaptation”. Neurocomp. 74, 11, pp. 1785–1799, 2011.

[Breiman, 1999] L. Breiman, “Pasting Small Votes for Classification in Large Databases and On-Line”. Mach. Learn. , pp. 85–103, 1999.

[Brzeziński, 2010] D. Brzezinski, “Mining Data Streams with Concept Drifts”, Dissertação de Mestrado, Poznan University of Technology, 2010.

[Brzezinski & Stefanowski, 2014] D. Brzezinski e J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm." Neural Networks and Learning Systems, IEEE Transactions on 25.1, pp. 81-94, 2014.

[Carvalho & Cohen 2006] V. Carvalho e W. Cohen. “Single-Pass Online Learning: Performance, Voting Schemes and Online Feature Selection”. Proc. of the 12th ACM SIGKDD Int. Conf. on Knowl. Disc. and DataMining (KDD) ACM, pp. 548–553, 2006.

[Chen & He, 2011] S. Chen e H. He, “Toward incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach”, Evolv. Syst., vol. 2, no. 1, pp. 35–50, 2011.

[Chen, 2000] S. X. Chen, “Probability density function estimation using gamma kernels” .Annals of the Institute of Statistical Mathematics, v. 52, n. 3, pp. 471-480, 2000.

[Cherubini et al., 2004] U. Cherubini, E. Luciano, W. Vecchiato, “Copula Methods in Finance”. John Wiley & Sons, 2004.

[Connoly et al., 2013] J. F. Connolly, E. Granger, R. Sabourin, “Dynamic multi-objective evolution of classifier ensembles for video face recognition”, Applied Soft Computing, Volume 13, Issue 6, pp. 3149-3166, 2013.

[Elwell & Polikar, 2011] R. Elwell, R. Polikar, “Incremental Learning of Concept drift in Nonstationary Environments”. IEEE Transactions on Neural Networks 22(10): pp. 1517-1531, 2011.

[Escovedo et al., 2013] T. Escovedo, A. Abs da Cruz, M. Vellasco, A. Koshiyama, “Learning Under Concept Drift Using a Neuro-Evolutionary Ensemble”. International Journal of Computational Intelligence and Applications, 12(04), 2013.

[Everitt et al., 2001] B. S. Everitt, S. Landau e M. Leese, “Cluster Analysis,” 4th Edition, Oxford University Press, Inc., New York; Arnold, London, 2001.

[Fan, 2004] W. Fan, “StreamMiner: a classifier ensemble-based engine to mine conceptdrifting data streams,” in Proceedings of the 30th International Conference on Very Large Data Bases, pp. 1257–1260, 2004.

[Fan, 2004b] W. Fan, “Systematic data selection to mine concept-drifting data streams,” in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 128–137, 2004.

[Fern & Givan, 2003] A. Fern e R. Givan, “Online Ensemble Learning: An Empirical Study”. Mach. Learn. 53, 1–2, 71–109, 2003.

[Gama et al., 2004] J. Gama, P. Medas, G. Castillo e P. Rodrigues, “Learning with drift detection”, Proc. of the 7th Brazilian Symposium on Artificial Intelligence (SBIA’04) - Lecture Notes in Computer Science, vol. 3171. São Luiz do Maranhão, Brazil: Springer, pp. 286–295, 2004.

[Gama et al., 2014] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy e A. Bouchachia. "A survey on concept drift adaptation." ACM Computing Surveys (CSUR) 46, no. 4: 44, 2014.

[Gonçalves Júnior, 2013] P. M. Gonçalves Júnior, “Multivariate Non-Parametric Statistical Tests to Reuse Classifiers in Recurring Concept Drifting Environments”. Tese de Doutorado, Universidade Federal de Pernambuco, Recife, 2013.

[Han & Kim, 2000] K. Han, J. Kim, “Genetic quantum algorithm and its application to combinatorial optimization problem”, Proceedings Of The 2000 Congress On Evolutionary Computation, vol. 2, pp. 1354–1360, 2000.

[Han & Kim, 2002] K. Han, J. Kim, “Quantum-inspired evolutionary algorithm for a class of combinatorial optimization”, IEEE Trans. Evolutionary Computation, vol. 6, no. 6, pp. 580–593, 2002.

[Han & Kim, 2003] K. Han, J. Kim, “On setting the parameters of qea for practical applications: Some guidelines based on empirical evidence”, GECCO, pp. 427–428, 2003.

[Han & Kim, 2004] K. Han, J. Kim, “Quantum-inspired evolutionary algorithms with a new termination criterion, He gate, and two-phase scheme,” IEEE Trans. Evolutionary Computation, vol. 8, no. 2, pp. 156–169, 2004.

[Hastie et al., 2005] T. Hastie, R. Tibshirani, J. Friedman and J. Franklin, “The elements of statistical learning: data mining, inference and prediction”, The Mathematical Intelligencer, 27(2), 83-85, 2005.

[Hulten et al., 2001] G. Hulten, L. Spencer e P. Domingos, “Mining time-changing data streams,” In Proc. Of The 2001 Acm Sigkdd Intl. Conf. On Knowledge Discovery And Data Mining, pp. 97–106, 2001.

[Johnson & Wichern, 2014] R. A. Johnson, D. W. Wichern, “Applied Multivariate Statistical Analysis”. Pearson Education Limited, 2014.

[Karnick et al., 2008] M. T. Karnick, M. Ahiskali, M. Muhlbaier e R. Polikar, “Learning concept drift in nonstationary environments using an ensemble of classifiers based approach”, in IJCNN, pp. 3455–3462, 2008.

[Kaufman & Rousseeuw, 1990] L. Kaufman e P. J. Rousseeuw, “Finding groups in data: An introduction to cluster analysis”, John Wiley & Sons, New York, 1990.

[Kolter & Maloof, 2003] J. Kolter e M. Maloof, “Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift”. Proc. of the 3rd IEEE Int. Conf. on Data Mining (ICDM). IEEE, pp. 123–130, 2003.

[Kolter & Maloof, 2005] J. Kolter e M. Maloof, “Dynamic weighted majority: a new ensemble method for tracking concept drift”, Proceedings of the 3rd International IEEE Conference on Data Mining, pp. 123–130, 2003.

[Kolter & Maloof, 2005b] Kolter e M. Maloof, “Using additive expert ensembles to cope with concept drift,” in Proceedings of the 22nd International Conference on Machine Learninig, pp. 449–456, 2005.

[Kolter & Maloof, 2007] Kolter e M. Maloof, “Dynamic weighted majority: An ensemble method for drifting concepts”, Journal of Machine Learning Research, vol. 8, pp. 2755–2790, 2007.

[Kuncheva, 2004] L. I. Kuncheva, “Classifier ensemble for changing environments,” in Multiple Classifier Systems, vol. 3077. New York: Springer-Verlag, 2004.

[Kuncheva, 2008] L. I. Kuncheva, “Classifier ensemble for detecting concept change in streaming data: Overview and perspectives,” in Proc. Eur. Conf. Artif. Intell, pp. 5–10, 2008.

[Kuncheva & Faithfull, 2014] L. I. Kuncheva e W. J. Faithfull, "PCA Feature Extraction for Change Detection in Multidimensional Unlabeled Data," Neural Networks and Learning Systems, IEEE Transactions on , vol.25, no.1, pp.69,80, 2014.

[Linden, 2012] R. Linden. “Algoritmos Genéticos”. 3a edição, Editora Ciência Moderna, 2012.

[Littlestone, 1988] N. Littlestone, “Learning quickly when irrelevant attributes abound: A new linear threshold algorithm”. Machine Learning, 2: pp. 285-318, 1988.

[MacQueen, 1967] J. MacQueen. "Some methods for classification and analysis of multivariate observations." Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14. 1967.

[Maloof, 2010] M. A. Maloof, “The AQ Methods for Concept Drift”, Advances in Machine Learning: Dedicated to the Memory of Professor Ryszard S. Michalski. Springer, Berlin, pp. 23–47, 2010.

[Mardia, 1970] K. V. Mardia, "Measures of multivariate skewness and kurtosis with applications". Biometrika 57 (3): pp. 519–530, 1970.

[Mardia, 1971] K. V. Mardia, “The effect of nonnormality on some multivariate tests and robustness to nonnormality in the linear model”. Biometrika, v. 58, n. 1, pp. 105-121, 1971.

[Melnykov & Melnykov, 2014] I. Melnykov e V. Melnykov, “On k-means algorithm with the use of Mahalanobis distances”, Statistics & Probability Letters, Volume 84, pp. 88-95, 2014.

[Minku & Yao, 2012] L. L. Minku e X. Yao, "DDD: A New Ensemble Approach for Dealing With Concept Drift". IEEE Transactions on Knowledge and Data Engineering, IEEE, v. 24, n. 4, pp. 619-633, 2012.

[Mitchell, 1997] T. Mitchell, “Machine Learning”. McGraw Hill, 1997.

[MOA Datasets, 2015] MOA Datasets, “MOA – Massive Online Analysis”. Disponível em: http://moa.cms.waikato.ac.nz/datasets/. Último acesso em Janeiro de 2015.

[Montgomery, 2013] D. C. Montgomery, “Applied Statistics and Probability for Engineers”, 6th edition. Wiley, 2013.

[Moreno-Torres et al., 2012] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla e F. Herrera. "A unifying view on dataset shift in classification." Pattern Recognition 45, no. 1, pp. 521-530, 2012.

[Nishida & Yamauchi, 2007] K. Nishida e K. Yamauchi, “Adaptive classifiers-ensemble system for tracking concept drift,” in Proceedings of the Sixth International Conference on Machine Learning and Cybernetics (ICMLC’07), Honk Kong, pp. 3607–3612, 2007.

[Nishida & Yamauchi, 2007b] K. Nishida e K. Yamauchi, "Detecting concept drift using statistical testing", Discovery Science. Springer Berlin Heidelberg, 2007.

[Nishida, 2008] K. Nishida, “Learning and detecting concept drift”. Tese de Doutorado, Hokkaido University, Japan, 2008.

[Oza, 2001] N. C. Oza, “Online Ensemble Learning,” Dissertation, University of California, Berkeley, 2001.

[Pinho, 2010] A. G. Pinho, “Algoritmo evolucionário com inspiração quântica e representação mista aplicado a Neuroevolução”. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2010.

[Polikar & Elwell, 2013] R. Polikar e R. Elwell. “Benchmark Datasets for Evaluating Concept drift/NSE Algorithms”. Disponível em: http://users.rowan.edu/~polikar/research/NSE. Último acesso em Dezembro de 2013.

[Ross et al., 2012] G. J. Ross, N. M. Adams, D. K. Tasoulis e D. J. Hand. 2012. “Ross, Gordon J., et al. "Exponentially weighted moving average charts for detecting concept drift”, Pattern Recognition Letters 33.2, pp. 191-198, 2012.

[Schlimmer & Granger, 1986] J. Schlimmer e R. Granger, “Incremental learning from noisy data”. Machine Learning, 1(3): pp. 317-354, 1986.

[Scholz & Klinkenberg, 2005] M. Scholz e R. Klinkenberg, “An ensemble classifier for drifting concepts,” in Proceedings of the 2nd International Workshop on Knowledge Discovery in Data Stream, pp. 53–64, 2005.

[Scholz & Klinkenberg, 2007] M. Scholz e R. Klinkenberg, “Boosting classifiers for drifting concepts,” Intelligent Data Analysis, vol. 11, no. 1, pp. 3–28, 2007.

[Stanley, 2003] K. O. Stanley, “Learning concept drift with a committee of decision trees,” Department of Computer Sciences, University of Texas at Austin, Tech. Rep. AI-03-302, 2003.

[Street & Kim, 2001] W. N. Street e Y. S. Kim, “A streaming ensemble algorithm (SEA) for largescale classification,” in Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–382, 2001.

[Trivedi & Zimmer, 2007] P. K. Trivedi, D. M. Zimmer, “Copula modeling: an introduction for practitioners”. Now Publishers Inc, 2007.

[Tsymbal, 2004] A. Tsymbal, “The problem of concept drift: Definitions and related work”, Tech. Rep., 2004.

[Widmer & Kubat, 1996] G. Widmer e M. Kubat, “Learning in the presence of concept drift and hidden contexts”, Mach. Learn., vol. 23, no. 1, pp. 69-101, 1996.

[Xingquan et al. 2004] Z. Xingquan, W. Xindong e Y. Ying, “Dynamic classifier selection for effective mining from noisy data streams”, Proc. 4th IEEE Int. Conf. Data Min., pp. 305–312, 2004.

[Yang et al., 2006] Y. Yang, X. Wu e X. Zhu. “Mining in anticipation for concept change: Proactive-reactive prediction in data streams”. Data Min. Knowl. Discov., 13(3): pp. 261-289, 2006.

[Zliobaite, 2009] I. Zliobaite, “Learning under Concept Drift: An Overview”. Tech. rep. Vilnius University, 2009.

Anexo 1 – Definições

Definição 1. Variável Aleatória

Seja $\mathrm{X}: \Omega \rightarrow \mathbb{R}$, uma função que associa cada resultado ω do espaço amostral Ω, a um número real, i.e., $\mathrm{X}(\omega) = \mathrm{x}$. Então, X é dita Variável Aleatória.

Comentário: uma variável aleatória X é uma forma de associar resultados no espaço amostral (ex.: ω ={cara, coroa}, num experimento de jogo de moedas) a números reais (ex.: $X(cara) = 1$). Uma variável aleatória é dita discreta, quando podem ser enumerados ou listados o seu conjunto de valores admissíveis; caso contrário, esta é dita contínua. Exemplo: a variável aleatória que define o resultado do experimento de lançar uma moeda, ou sacar uma carta em um baralho, contar o número de insetos em uma árvore, etc. é dito discreta. Já a variável aleatória que define o resultado do experimento de medir a temperatura de uma criança, auferir a concentração de oxigênio em uma solução, etc., são ditas contínuas.

Definição 2. Função Distribuição Acumulada

Seja X uma variável aleatória. Considere $F_X: X \rightarrow [0,1]$, uma função dotada dessas 2 propriedades:

$lim_{x \rightarrow -\infty} F_X(x) = 0$ e $lim_{x \rightarrow \infty} F_X(x) = 1$ – *limitada*

$F_X(x)$ é contínua superiormente (ou à direita), i.e., $lim_{x \downarrow x_0} F_X(x) = F_X(x_0)$

Então F_X é uma função distribuição acumulada para a variável aleatória X.

Definição 3. Função Densidade de Probabilidade

Seja X uma variável aleatória. Considere $f_X: X \rightarrow [a, b]$, uma função dotada dessas 2 propriedades:

$f_X(x) \geq 0$ – semi-positiva

$\int_X f_X(x) = 1$ (contínua) ou $\sum_x f_X(x) = 1$ (discreta) – normalizada

Então f_X é uma função densidade de probabilidade para a variável aleatória X.

Comentário: A definição 3 de fato é um teorema, que demonstra a equivalência (se, e somente se) entre a definição original:

Caso discreto: $P(X = x) = f_X(x)$

Caso contínuo: $P(X \leq x) = F_X(x) = \int_{-\infty}^{x} f_X(t)\, dt$

e as propriedades que a f_X deve possuir para ser definida como Função Densidade de Probabilidade.

Definição 4. Amostra Aleatória

Seja $X_1, X_2, \ldots, X_n$ uma sequência de variáveis aleatórias. Uma sequência de variáveis aleatórias é dita ser uma Amostra Aleatória se:

$X_1, X_2, \ldots, X_n$ são identicamente distribuídas

$X_1, X_2, \ldots, X_n$ são conjuntamente independentes

Anexo 2 – Procedimento de Geração de Bases de Dados

Geração de Bases de Dados sem *Drift*

Uma típica base de dados de um problema de classificação é apresentada na Tabela 29. Esta é constituída por:

- *n* padrões: $x_1, ..., x_n$ onde $x_i = [x_{11}, ..., x_{nJ}]$, com $i=1,...,n$.
- *J* atributos: $X_j = \{x_{1j}, ..., x_{nj}\}$, com $j=1,...,J$.
- Cada padrão está associado a uma das *K* classes.

Tabela 29. Típica base de dados para um problema de classificação.

Padrões	**Atributos**			**Classe**
	Atributo 1	***...***	***Atributo J***	
$\mathbf{x_1}$	x_{11}	...	x_{1J}	1
$\mathbf{x_2}$	x_{21}	...	x_{2J}	1
...	...	...	...	...
$\mathbf{x_n}$	x_{n1}	...	x_{nJ}	*K*

Ainda, outra informação é o nível de desbalanceamento entre as classes. Por exemplo, para um caso de duas classes ($K=2$) e 100 padrões ($n=100$) se o nível de desbalanceamento é de 60%, isso significa que há alguma classe que tem 60 padrões, enquanto que a outra tem 40 padrões. Defina por $\lambda_1, ..., \lambda_K$ como a proporção de cada uma das *K* classes na base de dados.

Seja $DB(t)$ a base de dados (ou bloco de dados) no instante de tempo *t*. Dado os métodos de detecção de *drift*, testes estatísticos e definições apresentadas anteriormente, sabe-se caracterizar de forma precisa quando $DB(t+\gamma)$ sofreu um determinado *drift*, quando os pressupostos acerca dos testes são seguidos. Portanto, a sequência de bases de dados: $DB(1)$, ..., $DB(T)$, $t=1,...,T$, deve ser composta por elementos (padrões) que advenham de uma mesma distribuição de probabilidade. Logo, cada padrão de $DB(t)$ será uma observação de um vetor aleatório $\boldsymbol{X}^{(t)}|C_k^{(t)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(t), \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t)\right)$ com distribuição Normal Multivariada condicional á classe *k*.

Definida a distribuição de probabilidade na qual os padrões serão gerados, cabe ainda escolher os parâmetros da distribuição Normal Multivariada. Esta escolha deve ser feita de maneira a tornar o processo de investigação do desempenho dos métodos de detecção o mais genérico e isento de intervenções e escolhas propositais. Neste sentido, o método apresentado a seguir é baseado na geração aleatória de valores para definir $\boldsymbol{\mu_{C_k}}(t)$ e $\boldsymbol{\Sigma_{C_k}}(t)$, seguindo algumas restrições impostas pela natureza destes dois objetos. Portanto, considere o esquema abaixo:

(1) Definindo alguns parâmetros: o usuário deve definir para a t=1-primeira base – as quantidades: número de padrões (*n*), número de atributos (*J*), número de classes (*K*) e a proporção das *K* classes na base de dados (λ1, ..., λK).

(2) Definindo os valores para $\boldsymbol{\Sigma_{C_k}}(t)$: Como $\boldsymbol{\Sigma_{C_k}}(t)$ é uma matriz de covariância, por definição, esta deve ser escolhida de forma simétrica[1] e positiva[2]. A seguir é apresentada uma típica matriz de covariância:

$$\boldsymbol{\Sigma_{C_k}}(t) = \begin{bmatrix} \sigma^2_{X_1|C_k} & \sigma_{X_1,X_2|C_k} & \cdots & \sigma_{X_1,X_J|C_k} \\ \sigma_{X_2,X_1|C_k} & \ddots & & \vdots \\ \vdots & & & \\ \sigma_{X_J,X_1|C_k} & \cdots & & \sigma^2_{X_J|C_k} \end{bmatrix}$$

A diagonal de $\boldsymbol{\Sigma_{C_k}}(t)$ é composta pelas variâncias de cada j-ésimo atributo (valores positivos -- $\sigma^2_{X_j|C_k}$) e os elementos fora da diagonal são formados pelas covariâncias. A covariância entre duas variáveis deve obedecer à desigualdade de Cauchy-Schwarz: $-\sigma^2_{X_1|C_k}\sigma^2_{X_2|C_k} \leq \sigma_{X_1,X_2|C_k} \leq \sigma^2_{X_1|C_k}\sigma^2_{X_2|C_k}$ – a covariância está limitada superiormente (inferiormente) pelo produto das variâncias de cada atributo (multiplicado por -1). Tendo essas duas observações, o processo de geração é sumarizado:

(a) Gere cada $\sigma^2_{X_j|C_k}$ *(j=1,...,J)* a partir de uma realização de uma variável aleatória com distribuição χ^2_1 com um grau de liberdade[3].

(b) Gere cada elemento $\sigma_{X_j,X_l|C_k}$ da parte triangular superior de $\boldsymbol{\Sigma_{C_k}}(t)$, fazendo $\sigma_{X_j,X_l|C_k} = \sigma^2_{X_j|C_k}\sigma^2_{X_l|C_k}u$, onde $u \sim U(-0{,}75, 0{,}75)$ representa o grau de

[1] Seja A uma matriz *nxn* e real. Diz-se que A é simétrica quando $A^T = A$, onde T é o operador de transposição.

[2] Seja A uma matriz *nxn*, real e simétrica. Diz-se que A é positiva quando $v^TAv > 0$, ou det(A)>0, ou todos os autovalores de A são estritamente positivos.

[33] Uma forma mais simples é gerar uma realização *z* de uma variável aleatória com distribuição *N*(0,1), e fazer com que $\sigma^2_{X_j|C_k} = z^2$.

correlação a ser manifestado na covariância entre as variáveis. Optou-se pela correlação máxima ser igual a $0{,}75$, para que se restrinja o caso de quase perfeita (valores muito próximos de 1) correlação entre as variáveis.

(c) Gere a parte triangular inferior de $\boldsymbol{\Sigma_{C_k}}(t)$, de forma a tornar a matriz simétrica. Observação final: a simetria de $\boldsymbol{\Sigma_{C_k}}(t)$ está garantida, mas não necessariamente esse processo de elaboração a torna positiva. Caso esta não seja positiva, basta somar a $\boldsymbol{\Sigma_{C_k}}(t)$ uma matriz diagonal D com valores positivos, até que a definição seja verificada. Repita esse processo para *k = 1,...,K*.

(d) Um comentário: a escolha da $\boldsymbol{\Sigma_{C_k}}(t)$ dimensiona a dispersão e a correlação dos diferentes atributos considerados. Dado que cada $\sigma^2_{X_j|C_k}$ está baseada na observação de uma variável aleatória com distribuição normal padrão ao quadrado (qui-quadrado com um grau de liberdade), implica que a variância não deverá tomar valores muito altos, ficando muito provavelmente (95% do tempo) em no máximo 4 unidades.

(3) Definindo os valores para $\boldsymbol{\mu_{C_k}}(t)$: O vetor de médias condicionais foi gerado, a partir de uma realização de uma variável aleatória $\boldsymbol{X}^{(t)} \sim N_J\left(\mathbf{0}, \boldsymbol{\Sigma_{C_k}}(t)\right)$. Portanto, um conjunto de médias possíveis, para um problema de três atributos seria: $\boldsymbol{\mu_{C_k}}(t) = [2, -1, 2.4]$, com uma escolha da matriz de covariância (conforme estágio prévio) em $\boldsymbol{\Sigma_{C_k}}(t) = \begin{bmatrix} 2{,}2 & 1{,}32 & -0.87 \\ 1{,}32 & 1{,}2 & 0{,}2 \\ -0.87 & 0{,}2 & 1{,}8 \end{bmatrix}$, por exemplo. Repita esse processo para *k=1,...,K*.

(4) Gere $\boldsymbol{DB}(\boldsymbol{t})$: Após estabelecido o conjunto de parâmetros necessários para a formação da primeira base de dados -- $\boldsymbol{\mu_{C_k}}(t)$ e $\boldsymbol{\Sigma_{C_k}}(t)$ – colete $n_1 = \lfloor n * \lambda_1 \rfloor$ realizações do vetor aleatório $\boldsymbol{X}^{(t)} \sim N_J\left(\boldsymbol{\mu_{C_1}}(t), \boldsymbol{\Sigma_{C_1}}(t)\right)$ e forme o conjunto de $\mathbf{x_1}, \ldots, \mathbf{x_{n1}}$ padrões da classe 1; colete $n_2 = \lfloor n * \lambda_2 \rfloor$ realizações do vetor aleatório $\boldsymbol{X}^{(t)} \sim N_J\left(\boldsymbol{\mu_{C_2}}(t), \boldsymbol{\Sigma_{C_2}}(t)\right)$ e forme o conjunto de $\mathbf{x_1}, \ldots, \mathbf{x_{n2}}$ padrões da classe 2; e assim sucessivamente para todas as *K* classes. Concatene todos os elementos e forme a $\boldsymbol{DB}(\boldsymbol{t})$**.**

Formada a primeira base de dados ($\boldsymbol{DB}(\boldsymbol{t})$, para *t*=1) é imediatamente possível calcular $\overline{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$. Faça *t*=*t*+1e siga para o próximo passo:

(5) Gere uma nova $\boldsymbol{DB(t)}$: como o objetivo dessa etapa é gerar uma base de dados sem *drift*, colete $n_1 = \lfloor n * \lambda_1 \rfloor$ realizações do vetor aleatório $\boldsymbol{X}^{(t)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_1}}(t), \boldsymbol{\Sigma}_{\boldsymbol{C_1}}(t)\right)$ e forme o conjunto de $\mathbf{x_1}, \ldots, \mathbf{x_{n1}}$ padrões da classe 1; colete $n_2 = \lfloor n * \lambda_2 \rfloor$ realizações do vetor aleatório $\boldsymbol{X}^{(t)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_2}}(t), \boldsymbol{\Sigma}_{\boldsymbol{C_2}}(t)\right)$ e forme o conjunto de $\mathbf{x_1}, \ldots, \mathbf{x_{n2}}$ padrões da classe 2; e assim sucessivamente para todas as *K* classes. Concatene todos os elementos e forme a $\boldsymbol{DB(t)}$. Obviamente que os novos padrões gerados não terão valores iguais aos prévios, mas parâmetros como $\overline{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$ bastante similares, quanto melhor for o tamanho e o método de amostragem.

(6) Verifique se o processo de amostragem não ocasionou *drift* em $\boldsymbol{DB(t)}$: devido ao processo de amostragem não ser efetivo em pequenas e médias amostras em distribuições multivariadas, pode ocorrer que $\overline{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$ sejam significativamente diferentes de $\overline{\boldsymbol{X}}_{C_k}(t-1)$ e $\boldsymbol{S}_{C_k}(t-1)$, ou dos demais instantes anteriores caso houver. Portanto, deve-se checar se as definições de ausência de *drift* abrupto, usando os testes estatísticos de T^2-Hotteling e o Box-M para o vetor de médias e matriz de covariância condicional, respectivamente. Dado um nível de significância α=0,05, faça:

(a) Aplique os testes para comparação. Caso não tenha ocorrido *drift* siga para o passo 7. Caso contrário vá para b.

(b) Refaça o passo 5, gerando uma nova base de dados.

(c) Aplique novamente os testes para comparação. Caso não tenha ocorrido *drift* siga para o passo 7. Caso contrário volte para b.

(7) Repita do passo 5 ao 7 até que *t*=*T* (número de bases de dados) seja atingida.

Seguindo esta sequência de passos é possível elaborar um conjunto de bases de dados que não possuem *drift* nas distribuições condicionais de cada classe. Contudo, deseja-se gerar bases de dados que possuam determinada característica de *drift*, de maneira a avaliar os métodos de detecção propostos. O passo 6 apresenta o caminho para a geração de bases de dados com imposição de *drifts* em determinados instantes de tempo. A próxima subseção trata dessa prática para o caso de geração de bases de dados com *drift* abrupto no vetor de médias condicionais

Geração de Bases de Dados com *Drift* Abrupto no Vetor de Médias Condicionais

De forma a ocasionar *drifts* abruptos no vetor de médias condicionais é necessário definir os seguintes parâmetros:

- Em quais instantes de tempo ocorrerão os *drifts* abruptos.
    - Dado um instante para a ocorrência de *drift*, quais classes do conjunto disponível são alvos do *drift*.
        - Dada uma classe, quais atributos que sofreram a alteração em suas respectivas médias.

A tarefa do usuário é determinar esses parâmetros para que o processo de geração de *drifts* se mantenha o mais controlado possível. Por simplificação, suponha um problema de classificação com duas classes (*K=2*), composto por 10 atributos (*J=10*) e um total de blocos de dados igual a 10 (*T=10*). Considere ainda que o usuário tenha definido dois momentos de *drift* abrupto: no bloco 3 e no bloco 7 para a classe 1. De forma sucinta, o que deve se garantir na geração dos 10 blocos de dados é que:

1. $\boldsymbol{\Sigma_{C_k}}(1) = \boldsymbol{\Sigma_{C_k}}(2) = \cdots = \boldsymbol{\Sigma_{C_k}}(9) = \boldsymbol{\Sigma_{C_k}}(10)$ para k=1 e 2.
2. $\boldsymbol{\mu_{C_2}}(1) = \boldsymbol{\mu_{C_2}}(2) = \cdots = \boldsymbol{\mu_{C_2}}(9) = \boldsymbol{\mu_{C_2}}(10)$.
3. $\left[\boldsymbol{\mu_{C_1}}(1) = \boldsymbol{\mu_{C_1}}(2)\right] \neq \left[\boldsymbol{\mu_{C_1}}(3) = \cdots = \boldsymbol{\mu_{C_1}}(6)\right] \neq [\boldsymbol{\mu_{C_1}}(7) = \cdots = \boldsymbol{\mu_{C_1}}(10)]$.

onde a igualdade nas expressões acima tem a mesma conotação que em um teste de hipótese: a partir dos estimadores $\overline{\boldsymbol{X}}_{C_k}(t)$ e $\boldsymbol{S}_{C_k}(t)$ para $\boldsymbol{\mu_{C_k}}(t)$ e $\boldsymbol{\Sigma_{C_k}}(t)$, verificar se a probabilidade da hipótese nula (ausência de diferença significativa entre os dois instantes) é superior/inferior ao nível de significância α. O que se deseja é que esta probabilidade seja superior a α para os casos número 1 e 2 acima, além das situações entre colchetes no número 3. Deseja-se o contrário nos momentos: $\boldsymbol{\mu_{C_1}}(2) \neq \boldsymbol{\mu_{C_1}}(3)$ e $\boldsymbol{\mu_{C_1}}(6) \neq \boldsymbol{\mu_{C_1}}(7)$.

Portanto, para as situações de ausência de *drift*, basta seguir a formulação apresentada na seção anterior que apresentou a forma de gerar bases de dados sem *drift*. Para os casos em que se deseja a presença de *drift*, e em destaque abrupto no vetor de médias condicionais, considere o procedimento a seguir:

(1) Defina por τ o momento do *drift* abrupto, *l* a classe em que ocorrerá o *drift* na média e o número de atributos que sofrerão variações na média como ω.

(2) Faça inicialmente $\boldsymbol{\mu_{C_l}}(\tau) = \boldsymbol{\mu_{C_l}}(\tau - 1)$. Selecione aleatoriamente ω componentes do vetor $\boldsymbol{\mu_{C_l}}(\tau) = [\mu_{X_1|C_l}(\tau), \mu_{X_2|C_k}(\tau), \ldots, \mu_{X_J|C_k}(\tau)]$ que sofrerão o *drift*. Defina um incremento ϵ pequeno, por volta de 0,01.

(3) Para todos os ω componentes selecionados, faça:

$v = 1$, se $u \sim U(0,1) > 0{,}5$ e $v = -1$.

$$\mu_{X_j|C_l}(\tau) = \mu_{X_j|C_l}(\tau) + v\left(\mu_{X_j|C_l}(\tau) * \epsilon\right)$$

(4) Colete $n_l = \lfloor n * \lambda_l \rfloor$ realizações do vetor aleatório $\boldsymbol{X}^{(t)} \sim N_J\left(\boldsymbol{\mu_{C_l}}(\tau), \boldsymbol{\Sigma_{C_l}}(\tau)\right)$ e forme o conjunto de $\mathbf{x_1}, \ldots, \mathbf{x_{nl}}$ padrões da classe *l*.

(5) Aplique os teste T^2-Hotteling para comparar $\bar{\boldsymbol{X}}_{C_l}(\tau)$ com $\bar{\boldsymbol{X}}_{C_l}(\tau - 1)$. Caso não tenha ocorrido *drift*, refaça o passo 3 (contudo, use os mesmos v's do instante anterior). Caso contrário prossiga para $\tau = \tau + 1$.

Se em $\tau + 1$ não houver a necessidade de novo *drift* abrupto, basta seguir o processo de geração de bases de dados sem *drift*. Caso contrário, reaplique o processo anterior para gerar uma intervenção na média condicional das classes. Cabe ressaltar que o processo apresentado exime de total responsabilidade o usuário de definir quais atributos vão ter a média alterada, a sua quantidade e sentido de crescimento. Um dos parâmetros que será estudado nos resultados é o número de atributos que sofrerão *drift*, de forma a verificar o seu impacto nos resultados do método de detecção. Seguindo essas linhas apresentadas, o usuário consegue gerar um conjunto de bases de dados em que se sabe de antemão o momento de ocorrência de um *drift* abrupto na média, ao mesmo tempo em que todas as demais condições de estacionariedade estão sendo satisfeitas.

Anexo 3 – Análise de Sensibilidade do DETECTA – Análises adicionais

Número de Atributos

A Figura 42 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se o número de atributos.

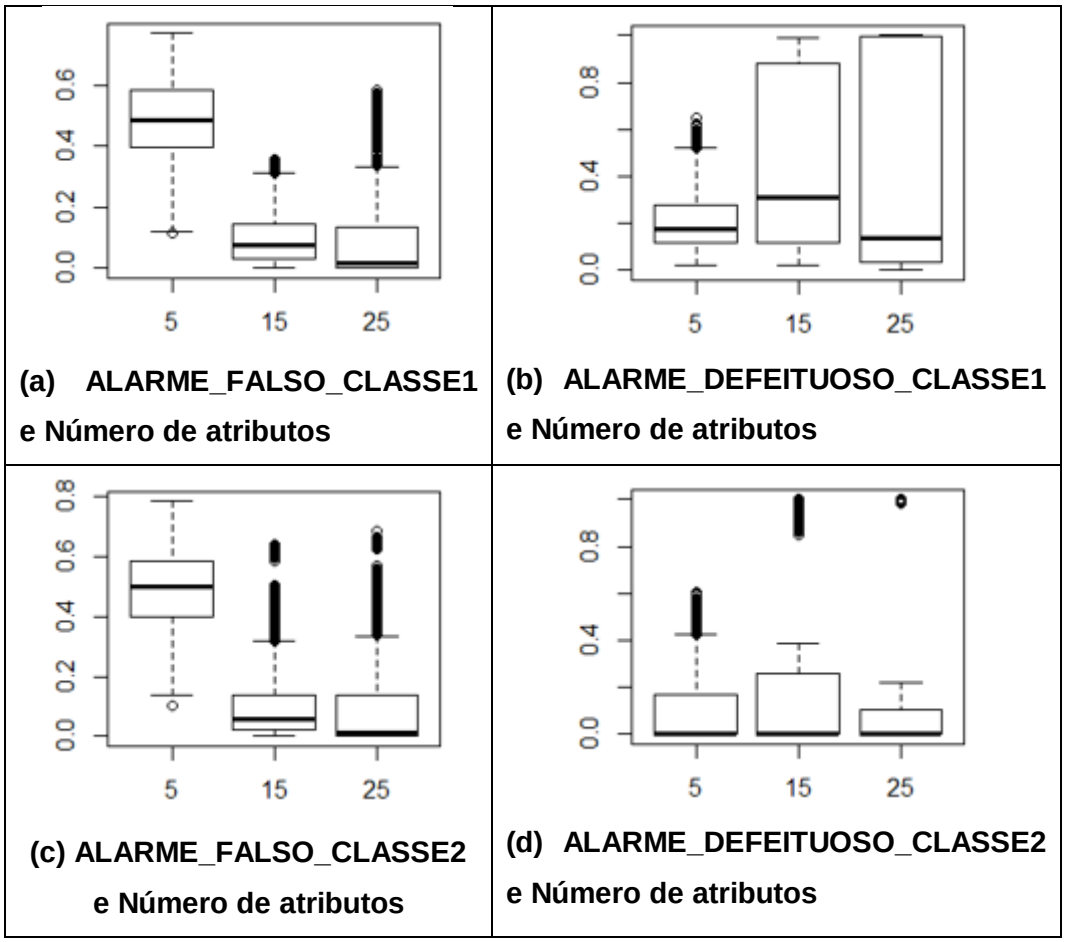


Figura 42. Distribuição dos resultados por número de atributos.

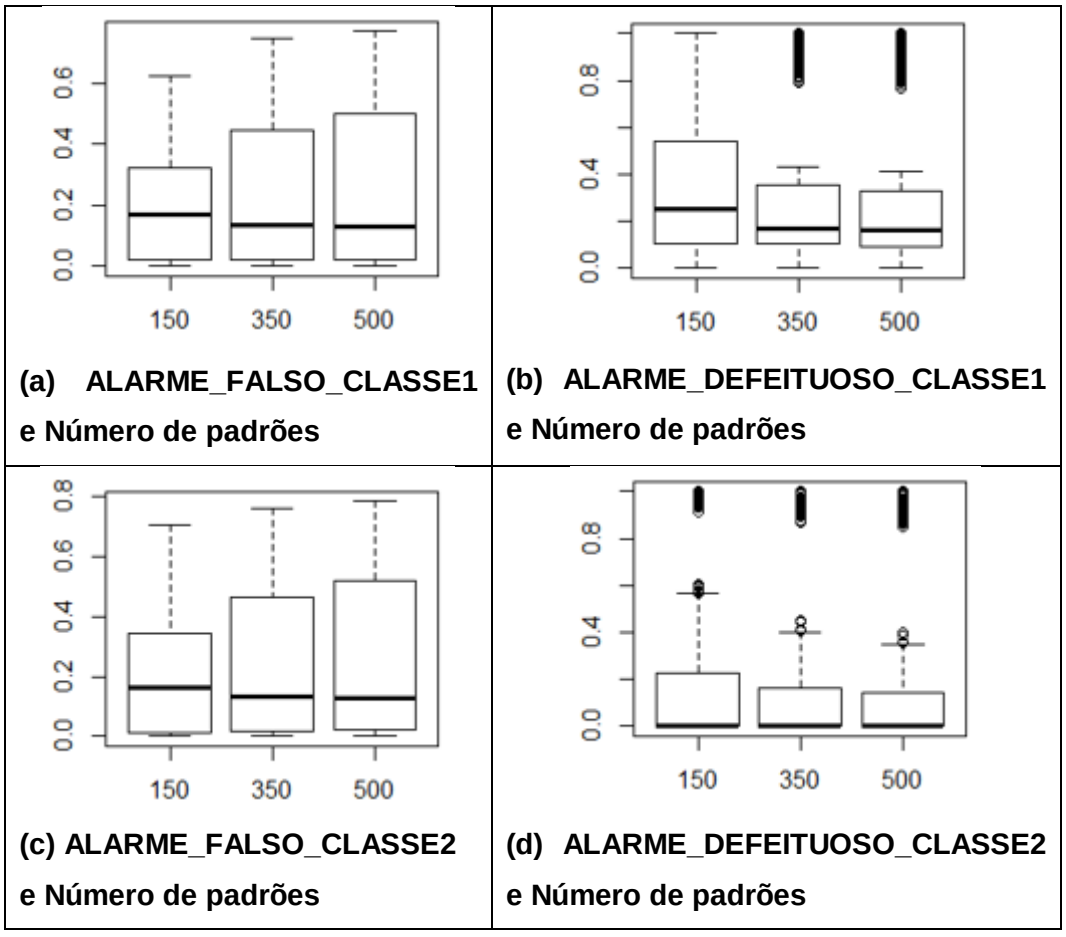
Observa-se que na Figura 42, as configurações com 5 atributos tiveram as maiores taxas de alarmes falsos tanto para a classe 1 quanto para a classe 2, concentrando-se entre 40% e 60% (região mais estreita entre os quartis, incluindo a mediana). Já com 15 e 25 atributos, a taxa de alarmes falsos mantém-se bem baixa, geralmente abaixo de 20%. Contudo, observa-se a ocorrência de um número razoável de *outliers* tanto para a classe 2 quanto para a classe 1, principalmente quando o parâmetro assume valor igual a 25. Já para

as taxas de alarmes defeituosos, observa-se que para a classe 1, as melhores taxas são alcançadas quando o número de atributos é igual a 5, tendendo a piorar quando o número de atributos aumenta (15) mas depois tendendo a diminuir quando o número de atributos aumenta ainda mais (25). Para a classe 2, não foi possível numa primeira análise identificar um padrão, porém, para os 3 valores de número de atributos utilizados, a taxa de alarmes defeituosos manteve-se razoavelmente baixa.

No geral, analisando-se os valores das medianas, pode-se verificar que para a taxa de alarmes falsos nas classes 1 e 2, há uma indicação que seu valor vai descendo à medida em que o número de atributos aumenta. Observa-se também que a taxa de alarmes defeituosos para a classe 2 em geral manteve-se bem baixa, com grande concentração dos dados próxima de zero.

Número de Padrões

A Figura 43 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se o número de padrões.



(a) ALARME_FALSO_CLASSE1 e Número de padrões

(b) ALARME_DEFEITUOSO_CLASSE1 e Número de padrões

(c) ALARME_FALSO_CLASSE2 e Número de padrões

(d) ALARME_DEFEITUOSO_CLASSE2 e Número de padrões

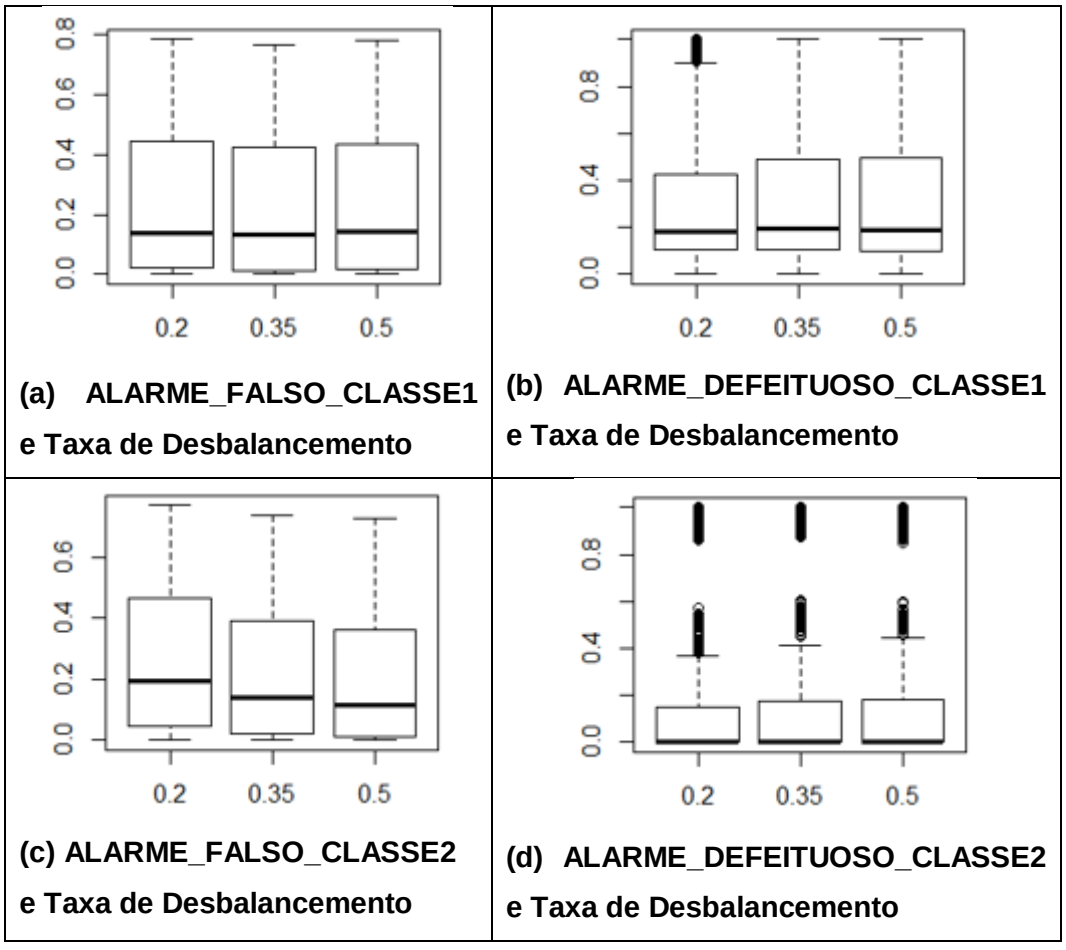
Figura 43. Distribuição dos resultados por número de padrões.

Ao avaliar as taxas de alarmes falsos da Figura 43, verifica-se que tanto para a classe 1 quanto para a classe 2, a sua amplitude (intervalo interquartílico) vai aumentando à medida em que o tamanho do bloco aumenta, enquanto a mediana desta taxa vai baixando, ainda que esta diminuição seja pouco expressiva.

Já para as taxas de alarmes defeituosos, a amplitude vai diminuindo à medida em que se aumenta o tamanho do bloco. Para a classe 1, a mediana vai diminuindo, indicando melhores taxas quando se aumenta o valor deste parâmetro. Para a classe 2 a mediana mantém-se muito baixa e praticamente constante; para todas as configurações são alcançadas taxas de alarmes defeituosos baixas.

Percentual de Desbalanceamento entre Classes

A Figura 44 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se o percentual de desbalanceamento entre as classes.



(a) ALARME_FALSO_CLASSE1 e Taxa de Desbalancemento

(b) ALARME_DEFEITUOSO_CLASSE1 e Taxa de Desbalancemento

(c) ALARME_FALSO_CLASSE2 e Taxa de Desbalancemento

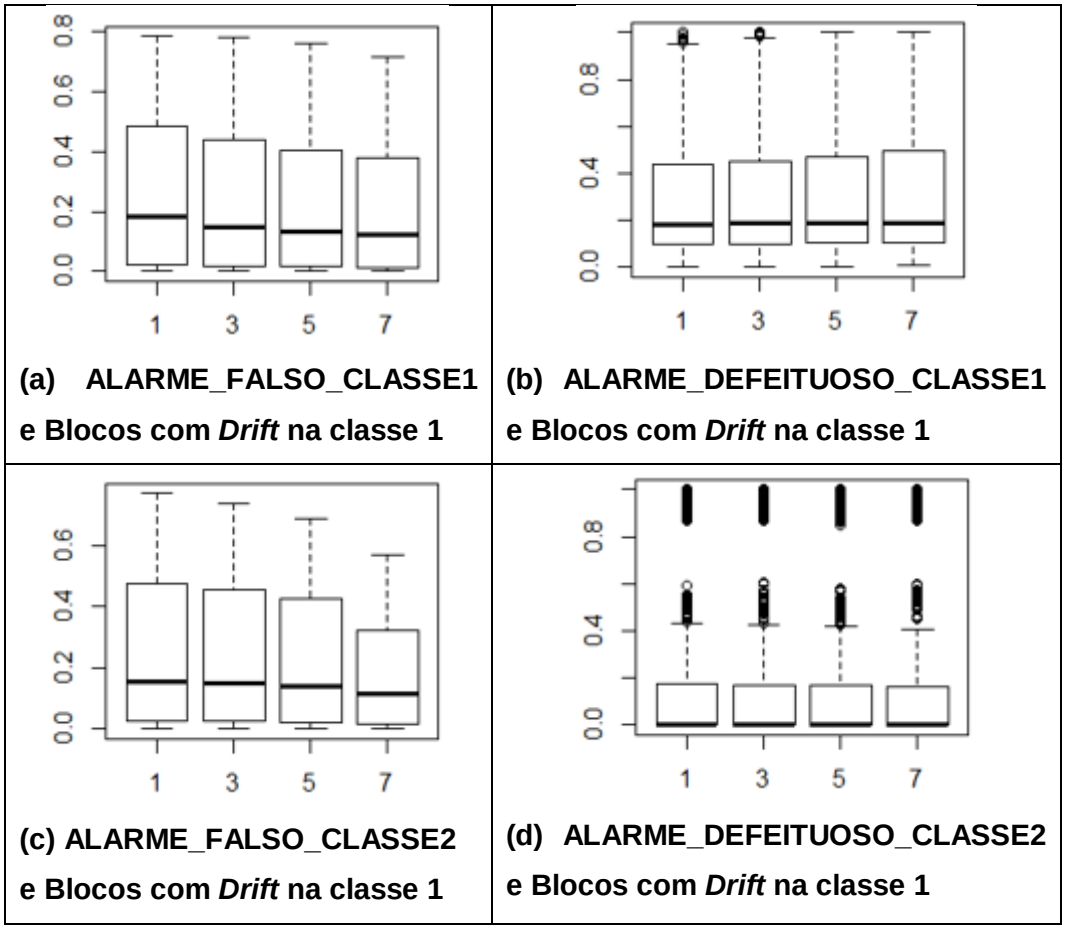
(d) ALARME_DEFEITUOSO_CLASSE2 e Taxa de Desbalancemento

Figura 44. Distribuição dos resultados por taxa de desbalanceamento.

A análise da Figura 44 mostra que a variação tanto das taxas de alarmes falsos quanto das taxas de alarmes defeituosos para ambas as classes é muito pequena. As taxas de alarmes falsos da classe 2 apresentam uma diminuição pouco expressiva à medida em que se aumenta a taxa de desbalanceamento, assim como a mediana, que é levemente diminuída. As taxas de alarmes defeituosos praticamente não sofreram variação, mas observa-se que para um valor baixo de desbalanceamento (0.2) na classe 1, a amplitude é menor e ocorreram mais *outliers*.

Número de blocos com *Drift* na Classe 1

A Figura 45 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se o número blocos com *drift* na classe 1.



(a) ALARME_FALSO_CLASSE1 e Blocos com *Drift* na classe 1

(b) ALARME_DEFEITUOSO_CLASSE1 e Blocos com *Drift* na classe 1

(c) ALARME_FALSO_CLASSE2 e Blocos com *Drift* na classe 1

(d) ALARME_DEFEITUOSO_CLASSE2 e Blocos com *Drift* na classe 1

Figura 45. Distribuição dos resultados por número de blocos com *drift* na classe 1.

Após a análise da Figura 45, observa-se que a taxa de alarmes falsos tende a diminuir à medida em que aumentamos o número blocos com *drift* na

classe 1. A mediana também vai diminuindo, ainda que de forma pouco expressiva. Isto indica que quanto mais *drifts* na classe 1, melhores os resultados considerando as taxas de alarmes falsos.

A taxa de alarmes defeituosos para a classe 1, ao contrário, parece estar aumentando à medida em que aumentam o número blocos com *drift* na classe 1. Para a classe 2, os resultados são praticamente os mesmos independentemente do valor do parâmetro, mantendo-se em valores baixos. Destaca-se a ocorrência de considerável número de *outliers* para a classe 2.

Número de Blocos com *Drift* na Classe 2

A Figura 46 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se o número blocos com *drift* na classe 2.

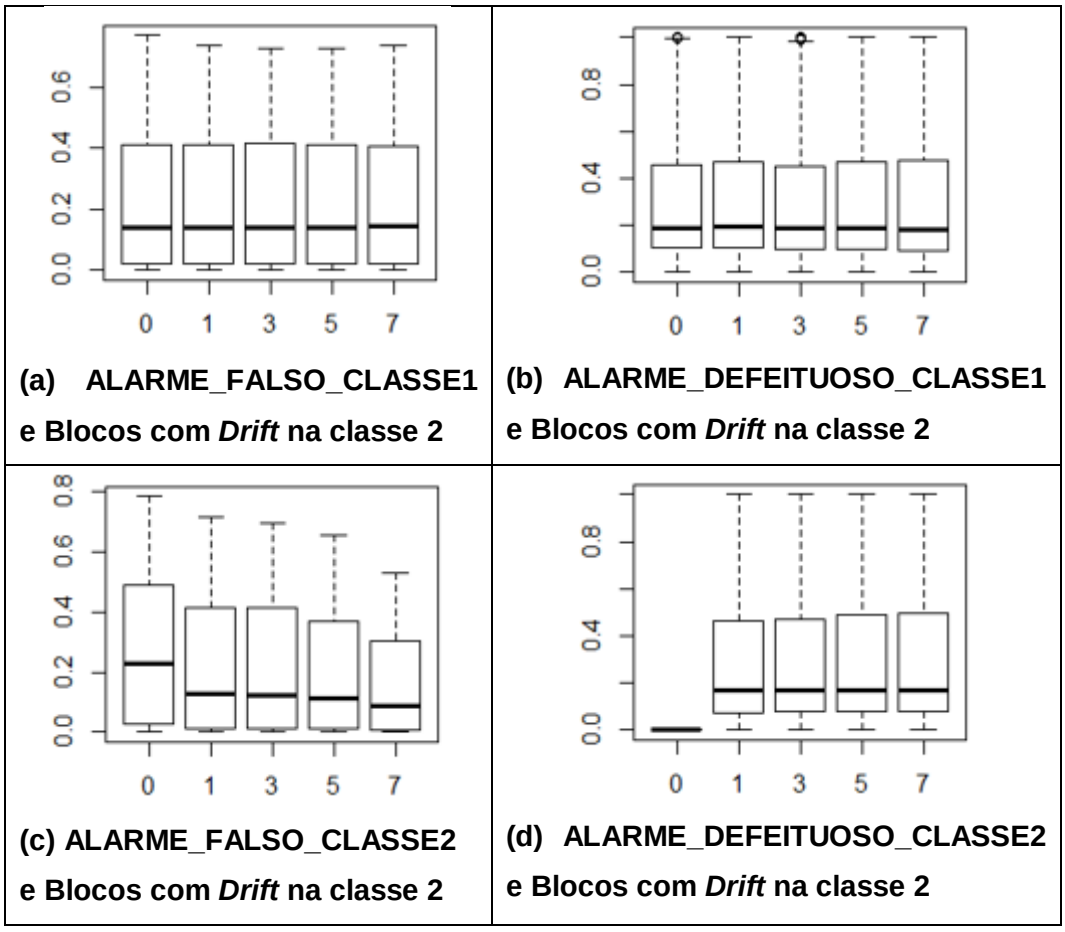


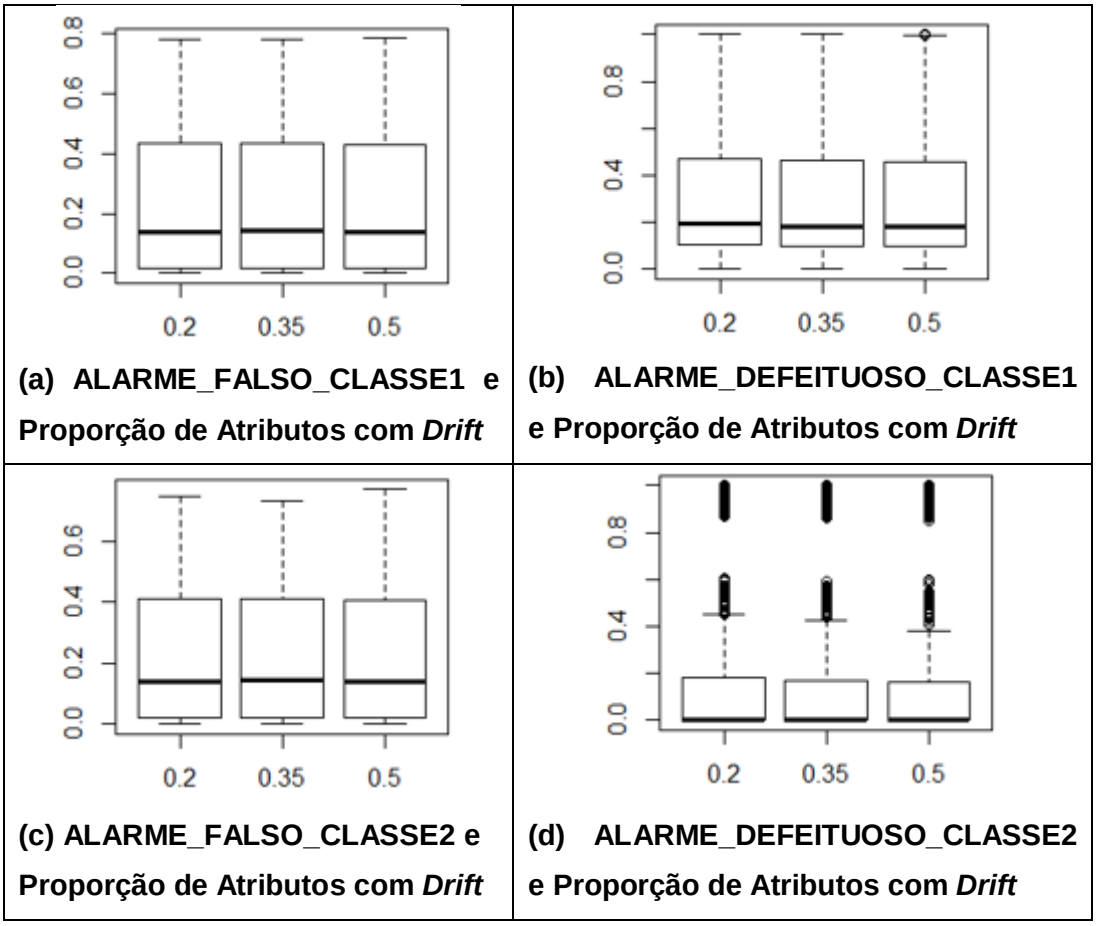
Figura 46. Distribuição dos resultados por número de blocos com *drift* na classe 2.

Nota-se uma diminuição gradativa na taxa de alarmes falsos para a classe 2 à medida em que se aumentam o número de blocos com *drift* na classe 2. As demais taxas (taxa de alarmes falsos para a classe 1, taxa de alarmes

defeituosos para as classes 1 e 2) permanecem praticamente constantes, com considerável amplitude (aproximadamente entre 0% e 40%) porém com mediana baixa (em torno ou abaixo de 20%).

Proporção de Atributos com *Drift*

A Figura 47 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se a proporção de atributos com *drift*.



(a) ALARME_FALSO_CLASSE1 e Proporção de Atributos com *Drift*

(b) ALARME_DEFEITUOSO_CLASSE1 e Proporção de Atributos com *Drift*

(c) ALARME_FALSO_CLASSE2 e Proporção de Atributos com *Drift*

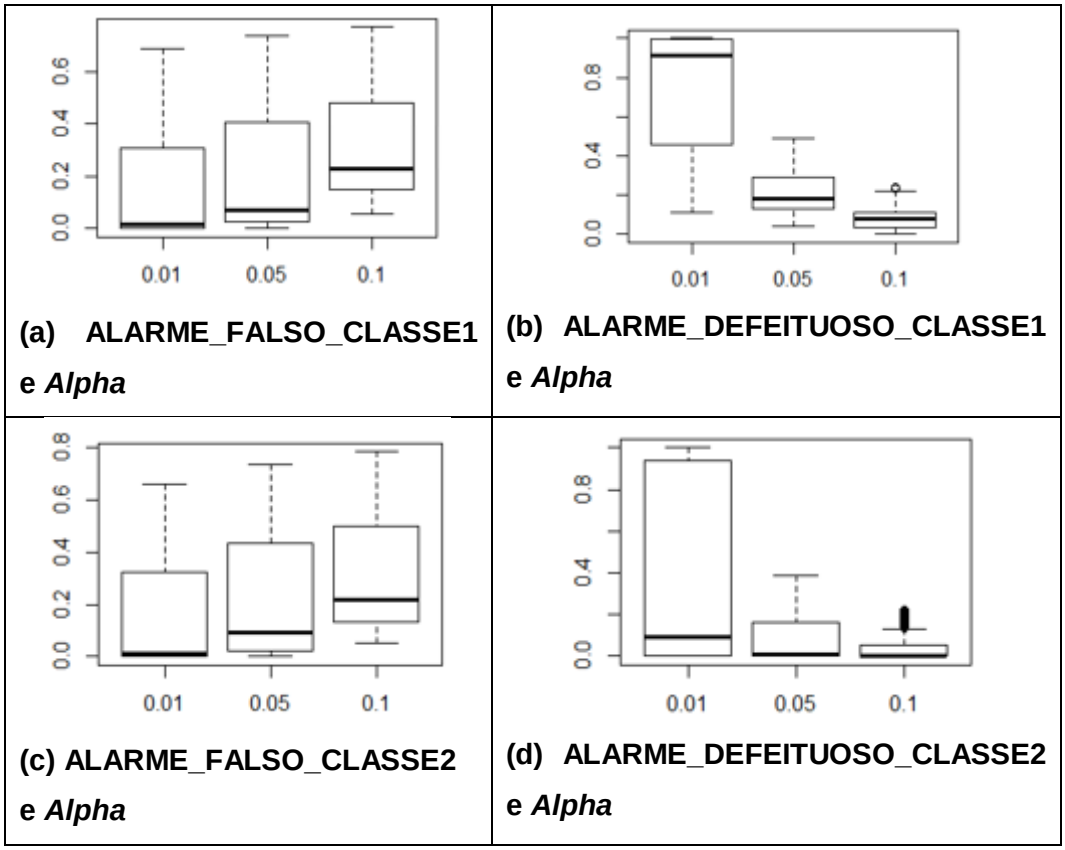
(d) ALARME_DEFEITUOSO_CLASSE2 e Proporção de Atributos com *Drift*

Figura 47. Distribuição dos resultados por proporção de atributos com *drift*.

Analisando a Figura 47, não foi encontrada variação expressiva para nenhuma das taxas a medida em que se varia o valor da proporção de atributos com *drift*. Todas as quatro taxas se encontram em torno ou abaixo de 40% e a mediana apresenta valores razoáveis: sempre abaixo de 20% pata as taxas de alarmes falsos, em torno de 20% para a taxa de alarmes defeituosos na classe 1 e tendendo a 0% na taxa de alarmes defeituosos da classe 2.

Alpha

A Figura 48 apresenta quatro gráficos *boxplot*, de forma que cada um representa um dos indicadores analisados e suas variações conforme aumenta-se ou diminui-se o valor de *alpha*.



(a) ALARME_FALSO_CLASSE1 e *Alpha*

(b) ALARME_DEFEITUOSO_CLASSE1 e *Alpha*

(c) ALARME_FALSO_CLASSE2 e *Alpha*

(d) ALARME_DEFEITUOSO_CLASSE2 e *Alpha*

Figura 48. Distribuição dos resultados por *alpha*.

Analisando a Figura 48, percebe-se uma clara influência na variação dos valores de *alpha* nos resultados das taxas de alarmes falsos e defeituosos. As taxas de alarmes falsos tanto da classe 1 quanto da classe 2 seguem um mesmo padrão: à medida em que o valor de *alpha* sobe, as taxas de alarmes falsos pioram, bem como a mediana, que vai aumentando progressivamente. Isto já era esperado, uma vez que como *alpha* indica o nível de significância, maiores valores indicam que o modelo fica mais reativo, havendo maior probabilidade de alarmes falsos.

Já para as taxas de alarmes defeituosos, observa-se que tanto para a classe 1 quanto para a classe 2 a mediana vai diminuindo à medida em que o valor de *alpha* aumenta. Também pode-se notar que a amplitude vai diminuindo com o aumento de alpha, indicando que quanto maior o valor de *alpha*, menor o número de alarmes defeituosos.

Teste de Tukey – Alarmes Falsos

As tabelas a seguir mostram o Teste de Tukey para os parâmetros: número de atributos, número de padrões, alpha, número de blocos com *drift* na classe 1 e 2 e para a interação 1, considerando as taxas de alarmes falsos.

Tabela 30. Teste de Tukey para número de atributos e alarmes falsos.

ALARME_FALSO_CLASSE1		
Número de Atributos	**Diferença**	**p-valor**
15-5	-0,387	< 0,05
25-5	-0,402	< 0,05
25-15	-0,015	< 0,05

ALARME_FALSO_CLASSE2		
Número de Atributos	**Diferença**	**p-valor**
15-5	-0,385	< 0,05
25-5	-0,395	< 0,05
25-15	-0,011	< 0,05

Tabela 31. Teste de Tukey para número de padrões e alarmes falsos.

ALARME_FALSO_CLASSE1		
Número de Padrões	**Diferença**	**p-valor**
350-150	0,030	< 0,05
500-150	0,047	< 0,05
500-350	0,017	< 0,05

ALARME_FALSO_CLASSE2		
Número de Padrões	**Diferença**	**p-valor**
350-150	0,041	< 0,05
500-150	0,061	< 0,05
500-350	0,020	< 0,05

Tabela 32. Teste de Tukey para *alpha* e alarmes falsos.

ALARME_FALSO_CLASSE1		
Alpha	**Diferença**	**p-valor**
0,05-0,01	0,052	< 0,05
0,1-0,01	0,167	< 0,05
0,1-0,05	0,115	< 0,05

ALARME_FALSO_CLASSE2		
Alpha	**Diferença**	**p-valor**
0,05-0,01	0,083	< 0,05
0,1-0,01	0,176	< 0,05
0,1-0,05	0,094	< 0,05

Tabela 33. Teste de Tukey para número de blocos com *drift* na classe 1 e 2 e alarmes falsos.

ALARME_FALSO_CLASSE1		
Número de blocos com *drift* na classe 1	**Diferença**	**p-valor**
3-1	-0,010	< 0,05
5-1	-0,029	< 0,05
7-1	-0,076	< 0,05
5-3	-0,019	< 0,05
7-3	-0,066	< 0,05
7-5	-0,047	< 0,05

ALARME_FALSO_CLASSE2		
Número de blocos com *drift* na classe 2	**Diferença**	**p-valor**
1-0	-0,067	< 0,05
3-0	-0,072	< 0,05
5-0	-0,089	< 0,05
7-0	-0,123	< 0,05
3-1	-0,006	< 0,05
5-1	-0,022	< 0,05
7-1	-0,056	< 0,05
5-3	-0,016	< 0,05
7-3	-0,051	< 0,05
7-5	-0,035	< 0,05

Tabela 34. Teste de Tukey para a interação 1 (número de atributos x número de padrões) e alarmes falsos.

ALARME_FALSO_CLASSE1		
Número de atributos x número de padrões	Diferença	p-valor
25:500-5:500	-0,518	< 0,05
15:500-5:500	-0,472	< 0,05
25:500-5:350	-0,462	< 0,05
25:350-5:350	-0,449	< 0,05
15:350-5:350	-0,424	< 0,05
15:500-5:350	-0,416	< 0,05
25:500-5:150	-0,308	< 0,05
25:350-5:150	-0,296	< 0,05
15:350-5:150	-0,271	< 0,05
15:150-5:150	-0,264	< 0,05
15:500-5:150	-0,263	< 0,05
25:150-5:150	-0,239	< 0,05
25:500-25:150	-0,070	< 0,05
25:350-25:150	-0,057	< 0,05
25:500-15:500	-0,045	< 0,05
25:500-15:150	-0,044	< 0,05
25:500-15:350	-0,037	< 0,05
15:350-25:150	-0,032	< 0,05
25:350-15:150	-0,032	< 0,05
25:350-15:350	-0,025	< 0,05
15:500-25:150	-0,024	< 0,05
25:500-25:350	-0,012	< 0,05
15:350-15:150	-0,007	< 0,05
15:500-15:150	0,001	< 0,05
15:500-15:350	0,008	< 0,05
25:150-15:150	0,025	< 0,05
15:500-25:350	0,033	< 0,05
5:500-5:350	0,056	< 0,05
5:350-5:150	0,154	< 0,05
5:500-5:150	0,210	< 0,05
5:350-25:150	0,392	< 0,05
5:350-15:150	0,418	< 0,05
5:500-25:150	0,448	< 0,05
5:500-15:150	0,473	< 0,05
5:500-15:350	0,480	< 0,05
5:500-25:350	0,505	< 0,05

ALARME_FALSO_CLASSE2		
Número de atributos x número de padrões	Diferença	p-valor
25:500-5:500	-0,491	< 0,05
15:500-5:500	-0,464	< 0,05
25:500-5:350	-0,435	< 0,05
25:350-5:350	-0,431	< 0,05
15:350-5:350	-0,416	< 0,05
15:500-5:350	-0,408	< 0,05
25:500-5:150	-0,292	< 0,05
25:350-5:150	-0,287	< 0,05
15:150-5:150	-0,275	< 0,05
15:350-5:150	-0,272	< 0,05
25:150-5:150	-0,265	< 0,05
15:500-5:150	-0,265	< 0,05
25:500-15:500	-0,027	< 0,05
25:500-25:150	-0,027	< 0,05
25:350-25:150	-0,023	< 0,05
25:500-15:350	-0,019	< 0,05
25:500-15:150	-0,017	< 0,05
25:350-15:350	-0,015	< 0,05
25:350-15:150	-0,013	< 0,05
15:350-25:150	-0,007	< 0,05
25:500-25:350	-0,004	< 0,05
15:500-25:150	0,000	< 0,05
15:350-15:150	0,003	< 0,05
15:500-15:350	0,007	< 0,05
25:150-15:150	0,010	< 0,05
15:500-15:150	0,010	< 0,05
15:500-25:350	0,023	< 0,05
5:500-5:350	0,055	< 0,05
5:350-5:150	0,144	< 0,05
5:500-5:150	0,199	< 0,05
5:350-25:150	0,408	< 0,05
5:350-15:150	0,419	< 0,05
5:500-25:150	0,464	< 0,05
5:500-15:350	0,471	< 0,05
5:500-15:150	0,474	< 0,05
5:500-25:350	0,486	< 0,05

Teste de Tukey – Alarmes Defeituosos

As tabelas a seguir mostram o Teste de Tukey para os parâmetros número de atributos, alpha, número de padrões, e para a interação 1, considerando as taxas de alarmes defeituosos.

Tabela 35. Teste de Tukey para número de atributos e alarmes defeituosos.

ALARME_DEFEITUOSO_CLASSE1		
Número de atributos	**Diferença**	**p-valor**
15-5	0,223195	< 0,05
25-5	0,17056	< 0,05
25-15	-0,05264	< 0,05

ALARME_DEFEITUOSO_CLASSE2		
Número de atributos	**Diferença**	**p-valor**
15-5	0,09902535	< 0,05
25-5	0,07630743	< 0,05
25-15	-0,02271792	< 0,05

Tabela 36. Teste de Tukey para *alpha* e alarmes defeituosos.

ALARME_DEFEITUOSO_CLASSE1		
Alpha	**Diferença**	**p-valor**
0,05-0,01	-0,54327	< 0,05
0,1-0,01	-0,67227	< 0,05
0,1-0,05	-0,129	< 0,05

ALARME_DEFEITUOSO_CLASSE2		
Alpha	**Diferença**	**p-valor**
0,05-0,01	-0,2948452	< 0,05
0,1-0,01	-0,35625786	< 0,05
0,1-0,05	-0,06141266	< 0,05

Tabela 37. Teste de Tukey para número de padrões e alarmes defeituosos.

ALARME_DEFEITUOSO_CLASSE1		
Número de padrões	**Diferença**	**p-valor**
350-150	-0,03712	< 0,05
500-150	-0,05492	< 0,05
500-350	-0,0178	< 0,05

ALARME_DEFEITUOSO_CLASSE2		
Número de padrões	**Diferença**	**p-valor**
350-150	-0,018800632	< 0,05
500-150	-0,026536229	< 0,05
500-350	-0,007735597	< 0,05

Tabela 38. Teste de Tukey para a interação 1 (número de atributos x número de padrões) e alarmes defeituosos.

ALARME_DEFEITUOSO_CLASSE1		
Número de atributos x número de padrões	Diferença	p-valor
5:500-15:150	-0,30187	< 0,05
5:500-15:350	-0,29343	< 0,05
5:350-15:150	-0,26015	< 0,05
5:500-25:350	-0,24156	< 0,05
5:500-25:150	-0,22565	< 0,05
5:350-25:150	-0,18393	< 0,05
5:500-5:150	-0,16056	< 0,05
5:350-5:150	-0,11884	< 0,05
25:150-15:150	-0,07622	< 0,05
25:350-15:150	-0,06031	< 0,05
25:500-15:150	-0,05512	< 0,05
25:350-15:350	-0,05187	< 0,05
25:500-15:350	-0,04668	< 0,05
5:500-5:350	-0,04172	< 0,05
25:500-15:500	-0,02982	< 0,05
15:500-15:150	-0,0253	< 0,05
15:500-15:350	-0,01686	< 0,05
15:350-15:150	-0,00844	< 0,05
25:500-25:350	0,005192	< 0,05
25:350-25:150	0,01591	< 0,05
25:500-25:150	0,021103	< 0,05
15:500-25:350	0,035009	< 0,05
15:500-25:150	0,05092	< 0,05
25:150-5:150	0,065087	< 0,05
15:350-25:150	0,06778	< 0,05
25:350-5:150	0,080998	< 0,05
25:500-5:150	0,08619	< 0,05
15:500-5:150	0,116007	< 0,05
15:350-5:150	0,132867	< 0,05
15:150-5:150	0,141307	< 0,05
25:350-5:350	0,19984	< 0,05
25:500-5:350	0,205033	< 0,05
15:500-5:350	0,23485	< 0,05
25:500-5:500	0,246753	< 0,05
15:350-5:350	0,25171	< 0,05
15:500-5:500	0,27657	< 0,05

ALARME_DEFEITUOSO_CLASSE2		
Número de padrões x número de atributos	Diferença	p-valor
5:500-15:150	-0,134824405	< 0,05
5:500-15:350	-0,126413139	< 0,05
5:350-15:150	-0,117690697	< 0,05
5:500-25:350	-0,104551477	< 0,05
5:500-25:150	-0,102978946	< 0,05
5:350-25:150	-0,085845238	< 0,05
5:500-5:150	-0,06669687	< 0,05
5:350-5:150	-0,049563161	< 0,05
25:150-15:150	-0,031845459	< 0,05
25:350-15:150	-0,030272928	< 0,05
25:500-15:150	-0,029601962	< 0,05
25:350-15:350	-0,021861662	< 0,05
25:500-15:350	-0,021190697	< 0,05
5:500-5:350	-0,017133708	< 0,05
15:500-15:150	-0,015155313	< 0,05
25:500-15:500	-0,014446649	< 0,05
15:350-15:150	-0,008411265	< 0,05
15:500-15:350	-0,006744048	< 0,05
25:500-25:350	0,000670966	< 0,05
25:350-25:150	0,001572531	< 0,05
25:500-25:150	0,002243497	< 0,05
15:500-25:350	0,015117615	< 0,05
15:500-25:150	0,016690146	< 0,05
15:350-25:150	0,023434193	< 0,05
25:150-5:150	0,036282077	< 0,05
25:350-5:150	0,037854608	< 0,05
25:500-5:150	0,038525573	< 0,05
15:500-5:150	0,052972222	< 0,05
15:350-5:150	0,05971627	< 0,05
15:150-5:150	0,068127535	< 0,05
25:350-5:350	0,087417769	< 0,05
25:500-5:350	0,088088735	< 0,05
15:500-5:350	0,102535384	< 0,05
25:500-5:500	0,105222443	< 0,05
15:350-5:350	0,109279431	< 0,05
15:500-5:500	0,119669092	< 0,05