Detecção de novidade em fluxos contínuos de dados multiclasse

Elaine Ribeiro de Faria Paiva

Detecção de novidade em fluxos contínuos de dados multiclasse

Elaine Ribeiro de Faria Paiva

Orientador: **Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho**
Co-Orientador: **Prof. Dr. João Manuel Portela da Gama**

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*.

USP – São Carlos
Junho de 2014

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

F224d

Faria, Elaine Ribeiro de
Detecção de novidade em fluxos contínuos de dados multiclasse / Elaine Ribeiro de Faria; orientador Andre Carlos Ponce de Leon Ferreira de Carvalho; co-orientador João Manuel Portela da Gama. -- São Carlos, 2014.
141 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014.

1. fluxos contínuos de dados. 2. detecção de novidade. I. Carvalho, Andre Carlos Ponce de Leon Ferreira de, orient. II. Gama, João Manuel Portela da, co-orient. III. Título.

Agradecimentos

Meus agradecimentos às pessoas que contribuíram para a realização deste trabalho.

Ao prof. André Carvalho agradeço por ter me aceitado como sua aluna de doutorado. Mesmo sabendo que eu seria uma aluna de dedicação parcial ele nunca colocou nenhum empecilho em me orientar. Nos momentos de maior dificuldade, nos quais eu não conseguia render quase nada, e me sentia totalmente impotente, ele estava sempre me incentivando. Qualquer pequeno avanço no trabalho era sempre comemorado por ele. Dentre as suas características as que mais se destacaram para mim é a humildade, o bom relacionamento com os alunos, educação e facilidade de trabalho em grupo. Agradeço pela paciência na leitura dos meus textos, por responder meus emails tão rapidamente e por me atender tão prontamente quando eu precisava.

Ao prof. João Gama agradeço por ter aceitado ser meu co-orientador e por ter me recebido tão bem nas duas vezes que estive em Portugal. O prof. João teve um papel muito importante neste trabalho, com sua experiência e conhecimentos na área. Sem dúvida este trabalho não seria o mesmo sem a sua orientação. Ele é aquele tipo de pessoa que queremos sempre conversar, transmite sabedoria, certeza e convicção nas sua ideias. Ele tem o dom de transformar assuntos complicados em definições simples.

Aos professores e técnicos que me ajudaram nesta caminhada. Sem dúvida o ICMC é um centro de excelência e me sinto muito feliz por ter conhecido e participado desse instituto.

Ao CNPq, programa Ciência Sem Fronteiras, e ao Banco Santander por terem financiado meus dois estágios na Universidade do Porto. Aos colegas do LIAAD/INESC pelo apoio durante o meu estágio na Universidade do Porto, em especial, a Isabel, pelo trabalho em equipe.

À Universidade Federal de Uberlândia (UFU) e à Faculdade de Computação (FACOM), que permitiram a flexibilização dos meus horários e o meu afastamento total no último ano de doutorado. Agradeço também aos meus colegas da FACOM, que muitas vezes me ajudaram nas minhas atividades na FACOM para que eu pudesse conciliá-las com o doutorado.

Aos meus colegas do BioCom, em especial ao Jonathan Silva, Ivani Negrão, Ricardo Cerri e Rodrigo Barros. Agradeço o apoio e amizade nestes anos de doutorado. Obrigada pelos trabalhos em conjunto, pelas discussões que ajudaram a evoluir meu trabalho, por me ajudarem quando eu chegava cansada das minhas viagens de Uberlândia. Foi muito importante ter o apoio de vocês.

Aos amigos Paula e Moacir, que foram meu ponto de apoio em São Carlos e me ajudaram muito nesta caminhada. Aos amigos João e Rachel da UFV por terem me ajudado tanto no início do doutorado. Vocês são grandes amigos e não imaginam o quanto foram importante para mim.

À minha família, por terem me apoiado e incentivado sempre. Agradeço aos meus pais, que mesmo com apenas o ensino primário, lutaram sempre para que os filhos estudassem.

Finalmente agradeço ao meu marido José Gustavo, que esteve sempre ao meu lado. Era ele que me ouvia quando eu estava cansada e sem ânimo para continuar. Era ele que estava do meu lado e vibrava a cada nova conquista. Obrigada por seu meu amigo, confidente e companheiro. Obrigada por ouvir pacientemente, os meus questionamentos e ideias para o algoritmo MINAS. Obrigada pelos conselhos, pela paciência e pelo carinho. Você é muito importante na minha vida.

Resumo

Mineração de fluxos contínuos de dados é uma área de pesquisa emergente que visa extrair conhecimento a partir de grandes quantidades de dados, gerados continuamente. Detecção de novidade é uma tarefa de classificação que consiste em reconhecer que um exemplo ou conjunto de exemplos em um fluxo de dados diferem significativamente dos exemplos vistos anteriormente. Essa é uma importante tarefa para fluxos contínuos de dados, principalmente porque novos conceitos podem aparecer, desaparecer ou evoluir ao longo do tempo. A maioria dos trabalhos da literatura apresentam a detecção de novidade como uma tarefa de classificação binária. Poucos trabalhos tratam essa tarefa como multiclasse, mas usam medidas de avaliação binária. Em vários problemas, o correto seria tratar a detecção de novidade em fluxos contínuos de dados como uma tarefa multiclasse, no qual o conceito conhecido do problema é formado por uma ou mais classes, e diferentes novas classes podem aparecer ao longo do tempo. Esta tese propõe um novo algoritmo – MINAS – para detecção de novidade em fluxos contínuos de dados. MINAS considera que a detecção de novidade é uma tarefa multiclasse. Na fase de treinamento, MINAS constrói um modelo de decisão com base em um conjunto de exemplos rotulados. Na fase de aplicação, novos exemplos são classificados usando o modelo de decisão atual, ou marcados como desconhecidos. Grupos de exemplos desconhecidos podem formar padrões-novidade válidos, que são então adicionados ao modelo de decisão. O modelo de decisão é atualizado ao longo do fluxo a fim de refletir mudanças nas classes conhecidas e permitir inserção de padrões-novidade. Esta tese também propõe uma nova metodologia para avaliação de algoritmos para detecção de novidade em fluxos contínuos de dados. Essa metodologia associa os padrões-novidade não rotulados às classes reais do problema, permitindo assim avaliar a matriz de confusão que é incremental e retangular. Além disso, a metodologia de avaliação propõe avaliar os exemplos *desconhecidos* separadamente e utilizar medidas de avaliação multiclasse. Por último, esta tese apresenta uma série de experimentos executados usando o MINAS e os principais algoritmos da literatura em bases de dados artificiais e reais. Além disso, o MINAS foi aplicado a um problema real, que consiste no reconhecimento de atividades humanas usando dados de acelerômetro. Os resultados experimentais mostram o potencial do algoritmo e da metodologia propostos.

Palavras-chave: Detecção de novidade, fluxos contínuos de dados, classificação multiclasse, metodologia de avaliação, evolução de conceitos.

Abstract

Data stream mining is an emergent research area that aims to extract knowledge from large amounts of continuously generated data. Novelty detection is a classification task that assesses if an example or a set of examples differ significantly from the previously seen examples. This is an important task for data streams, mainly because new concepts may appear, disappear or evolve over time. Most of the work found in the novelty detection literature presents novelty detection as a binary classification task. A few authors treat this task as multiclass, but even they use binary evaluation measures. In several real problems, novelty detection in data streams must be treated as a multiclass task, in which, the known concept about the problem is composed by one or more classes and different new classes may appear over time. This thesis proposes a new algorithm – MINAS – for novelty detection in data streams. MINAS deals with novelty detection as a multiclass task. In the training phase, MINAS builds a decision model based on a labeled data set. In the application phase, new examples are classified using the decision model, or marked with an unknown profile. Groups of unknown examples can be later used to create valid novelty patterns, which are added to the current decision model. The decision model is updated as new data arrives in the stream in order to reflect changes in the known classes and to allow the addition of novelty patterns. This thesis also proposes a new methodology to evaluate classifiers for novelty detection in data streams. This methodology associates the unlabeled novelty patterns to the true problem classes, allowing the evaluation of a confusion matrix that is incremental and rectangular. In addition, the proposed methodology allows the evaluation of unknown examples separately and the use multiclass evaluation measures. Additionally, this thesis presents a set of experiments carried out comparing the MINAS algorithm and the main novelty detection algorithms found in the literature, using artificial and real data sets. Finally, MINAS was applied to a human activity recognition problem using accelerometer data. The experimental results show the potential of the proposed algorithm and methodologies.

Keywords: Novelty detection, data streams, multiclass classification, evaluation methodology, concept evolution.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AM	*Aprendizado de Máquina*
CF	*Vetor de Características* (*Cluster Feature*)
CLAM	*CLAss based Micro classifier ensemble*
DETECTNOD	*DiscrETE Cosine Transform based NOvelty and Drift detection*
DN	*Detecção de Novidade* (*Novelty Detection*)
ECSMiner	*Enhanced Classifier for data Streams with novel class Miner*
FCD	*Fluxo Contínuo de Dados* (*Data Stream*)
KNN	*K Vizinhos Mais Próximos* (*K Nearest Neigbors*)
MCM	*Multi Class Miner in Data Streams*
MD	*Mineração de Dados*
MINAS	*Algoritmo de Aprendizado Multiclasses para Fluxos Contínuos de Dados* (*MultI-class learNing Algorithm for data Streams*)
OLINDDA	*OnLIne Novelty and Drift Detection*
RN	*Redes Neurais*
SVM	*Máquina de Vetor Suporte* (*Support Vector Machine*)

Sumário

CAPÍTULO

1

Introdução

Nas últimas três décadas, a pesquisa e a prática de aprendizado de máquina (AM) focou o aprendizado *batch*, em geral usando pequenas bases de dados (Gama, 2010). No cenário *batch*, um conjunto de dados representativos está disponível para a fase de aprendizado. Após o processamento desses dados, um modelo de decisão é gerado e pode ser usado para fazer predições futuras na tarefa abordada. Nesse tipo de cenário, os exemplos são gerados de acordo com uma distribuição de probabilidade estacionária (Gama, 2010).

No entanto, para problemas que possuem uma natureza dinâmica, o uso de algoritmos de aprendizado *batch* não é adequado. Em muitos problemas, os dados fluem continuamente ao longo do tempo, são os chamados Fluxos Contínuos de Dados (FCDs) – do inglês, *data streams*. FCDs são uma sequência ordenada de exemplos $(x_1, x_2, x_3,..., x_i,....)$ lidos em ordem crescente dos índices i (Guha et al., 2000). Um grande número de aplicações do mundo real lida com FCDs, como por exemplo, detecção de intrusos em uma rede de computadores, fluxo de transações financeiras, previsões do consumo de energia, monitoramento de um usuário via dados do seu celular e mineração de redes sociais.

Os FCDs possuem as seguintes características:

- Os exemplos chegam de forma contínua, e em muitos casos, em alta velocidade;
- O fluxo pode ser considerado infinito, e portanto os exemplos não podem ser armazenados na memória;
- O processo de geração de dados não é estacionário, isto é, a distribuição de probabilidade pode mudar ao longo do tempo.

> Devido a essas características, os algoritmos tradicionais de mineração de dados (MD) não podem ser facilmente aplicados ao domínio de FCD (Mahdiraji, 2009).

Uma propriedade desejável dos algoritmos que agem em ambientes com FCDs é a habilidade de adaptar seus modelos pela incorporação de novos dados. Muitos dos algoritmos desenvolvidos são naturalmente incrementais, outros precisam de adaptações para se tornarem incrementais. No entanto, segundo Gama (2010), em cenários não-estacionários, nos quais o conceito alvo pode mudar ao longo do tempo, algoritmos incrementais são necessários, mas não suficientes. Nesse caso, os algoritmos de mineração de dados devem ter mecanismos capazes de incorporar mudanças no conceito alvo sendo aprendido e adaptar o modelo de decisão para o estado recente do FCD.

Além disso, em cenários de aprendizado, em especial em FCDs, pode haver a introdução de novas classes do problema ao longo do tempo. Para esses cenários, são necessários algoritmos capazes de manter um modelo de decisão coerente com as mudanças e com as novas classes que podem surgir ao longo do tempo. Por outro lado, o modelo de decisão pode se tornar obsoleto e não mais contribuir para o estado atual do FCD, sendo necessário atualizá-lo constantemente.

Assim, novas soluções precisam ser desenvolvidas para atender problemas de AM que envolvam FCDs. Um *framework* geral para minerar FCDs requer tempo constante e pequeno para processar cada objeto, uso de uma quantidade fixa de memória, execução de no máximo uma varredura nos dados, atualização do modelo e inclusão de informações do passado, que não estão desatualizadas de acordo com cenário atual (Mahdiraji, 2009).

1.1 Contextualização

Detecção de novidade (DN), a habilidade de identificar situações novas ou desconhecidas, é uma tarefa útil para sistemas de AM, especialmente quando os dados são adquiridos incrementalmente (Perner, 2009). A DN torna possível reconhecer padrões-novidade, que podem indicar a presença de um novo conceito, uma mudança nos conceitos conhecidos ou a presença de ruídos (Gama, 2010). De acordo com Masud et al. (2011a), as tarefas mais desafiadoras em FCDs são mudança de conceito (*do inglês, concept drift*), em que o perfil das classes existentes muda, e a evolução de conceitos, o aparecimento de novas classes (do inglês, *concept evolution*).

A DN tem sido apresentada como uma tarefa de classificação com uma classe, cujo objetivo é distinguir os exemplos do conceito *normal* dos que não pertencem a esse conceito, pertencendo assim a uma classe chamada *não-normal* (Marsland et al., 2002), (Markou e Singh, 2003a), (Clifton et al., 2006), (Ahmed e Coates, 2007), (Spinosa et al., 2009) . A fase de treinamento (ou fase *offline*) induz um modelo de decisão a partir de exemplos rotulados como conceito ou classe normal, os quais representam o conceito conhecido do

problema. Na fase de aplicação (ou fase *online*), os exemplos não rotulados podem ser classificados no conceito normal, ou com o não-normal, também conhecido como anomalia ou novidade.

Neste trabalho, o problema de DN é apresentado de uma forma mais genérica do que uma tarefa de classificação com uma única classe. Em problemas envolvendo DN, o conceito conhecido pode ser composto por diferentes classes e diferentes novas classes podem aparecer ao longo do FCD, o que motiva o uso de técnicas de classificação multiclasse. Além disso, o modelo de decisão não pode ser estático, mas deve evoluir constantemente a fim de representar as classes emergentes e as mudanças nos conceitos já aprendidos.

Há muitos problemas reais envolvendo múltiplas classes, para os quais a aplicação de técnicas de DN pode ser útil. Dentre eles podemos destacar: detecção de intrusos em redes de computadores (Spinosa et al., 2008), (Khan et al., 2007), detecção de fraudes (Zhang et al., 2006), detecção de tipo de cobertura de floresta (Masud et al., 2011a), filtros para detecção de *spam* (Hayat e Hashemi, 2010) e classificação de texto (Li, 2006).

Alguns trabalhos recentes têm tratado DN em FCDs como uma tarefa de classificação multiclasse (Masud et al., 2011a), (Farid et al., 2013), (Al-Khateeb et al., 2012a). No entanto, eles supõem que apenas uma nova classe aparece por bloco de dados lido e processado. A distinção entre as diferentes novas classes que podem surgir ao longo do FCD é feita posteriormente, atualizando o modelo de decisão usando *feedback*. Para isso, é assumido que o rótulo verdadeiro de todos os exemplos do FCD sempre estará disponível. No entanto, obter o rótulo verdadeiro de todos os exemplos é uma tarefa que pode ser custosa e demandar muito tempo e, em vários cenários, pode ser impraticável.

Outro problema é que a maioria dos trabalhos para DN em FCDs usa medidas de classificação binárias para avaliar seus classificadores, mesmo aqueles que tratam a tarefa de classificação como multiclasse. Assim, essas medidas só conseguem avaliar se um exemplo é classificado como normal ou novidade, não diferenciando as classes que compõem cada um desses conceitos.

1.2 Hipóteses e Objetivos

As hipóteses dessa tese são:

- *Um algoritmo para DN em FCDs, que considere que essa tarefa possa ser multiclasse, que atualize o modelo de decisão sem usar feedback externo e que atenda às restrições para MD em FCDs com um desempenho superior aos algoritmos da literatura para alguns nichos de DN em FCDs.*

- *H2: Uma metodologia de avaliação para DN em FCDs que permita uma matriz de confusão incremental, na qual não há uma correspondência direta entre padrões-novidades não-rotulados e classes do problema e possam estar presentes exemplos*

marcados com o perfil desconhecido, permite avaliar melhor o erro ou acerto do classificador.

Os objetivos deste trabalho são:

- Desenvolver um novo algoritmo para a tarefa de DN em cenários multiclasse envolvendo FCDs. O algoritmo possui uma fase de aprendizado supervisionada *offline* e uma fase de aplicação não-supervisionada *online*. A fase de aprendizado cria um modelo de decisão que representa o cenário conhecido do problema, composto por uma ou mais classes. A fase de aplicação classifica novos exemplos que chegam constantemente ao longo do fluxo. Ela também detecta padrões-novidade e atualiza o modelo de decisão sem usar *feedback* externo, ou seja, sem exigir que o rótulo verdadeiro de todos os exemplos esteja disponível.

- Desenvolver uma nova metodologia de avaliação capaz de avaliar classificadores desenvolvidos para DN em FCDs.

- Aplicar o algoritmo e a metodologia de avaliação propostos em um problema real, envolvendo o reconhecimento de atividades humanas, bem como em bases de dados reais e artificiais públicas.

O trabalho abordará alguns pontos importantes. São eles:

- Construção de um modelo de decisão representado por um conjunto de micro-grupos, estruturas sumárias dos dados que possuem a propriedade da incrementalidade e que não requerem que os dados lidos a partir do FCD sejam armazenados;

- Atualização do modelo de decisão a fim de representar as mudanças no conceitos já aprendidos ou a inserção de padrões-novidade. A atualização é feita usando a propriedade incremental dos micro-grupos ou pela adição/remoção de micro-grupos;

- Representação dos conceitos conhecidos do problema levando em conta que pode haver mais que uma classe;

- Processo de DN sem *feedback* externo e que distingue entre diferentes padrões-novidade que podem ser detectados ao longo do FCD, o qual é mais complexo do que simplesmente classificar novos exemplos como normal e não-normal;

- Processo para detecção de ruídos ou *outliers*, diferenciando-os de um novo padrão-novidade;

- Criação de um modelo de decisão único que consiga agregar todo o conhecimento do problema, aprendido de maneira *offline* ou *online*;

- Metologia de avaliação para algoritmos que tratam da tarefa de DN em cenários envolvendo FCDs de duas ou mais classes;

- Apresentação de uma formalização geral da tarefa de DN em FCD, bem como das propostas existentes na literatura a fim de padronizar os termos e abordagens usados na área.

É também intuito deste trabalho, aplicar as técnicas propostas nos seguintes cenários envolvendo FCDs:

- Bases de dados sintéticas, geradas com mudança de conceito e aparecimento/ desparecimento de classes;

- Bases de dados reais, disponíveis em repositórios públicos, e comumente utilizadas nos trabalhos já desenvolvidos na área para DN em FCDs;

- Base de dados de um problema real que consiste em reconhecer a ação que um usuário está executando usando um sensor de acelerômetro (*human activity recognition*).

1.3 Artigos Originados da Tese

Os artigos publicados são:

- J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. Carvalho, and J. Gama, "Data stream clustering: A survey", ACM Computing Surveys, vol. 46, no. 1, 13:1–13:31 ,2014.

- E. R. Faria, I. R. Gonçalves, J. Gama, and A. C. P. L. F. Carvalho, "Evaluation methodology for multiclass novelty detection algorithms,"in Proceedings of the 2nd Brazilian Conference on Intelligent Systems (BRACIS'13), pp. 19 - 25, 2013.

- E. R. Faria, J. Gama, and A. C. P. L. F. Carvalho, "Novelty detection algorithm for data streams multi-class problems,"in Proceedings of the 28th Symposium on Applied Computing (ACM SAC'13), 2013, pp. 795 - 800.

- Faria, E. R. , Barros, R. C. , Carvalho, A. C. P. L. F. , Gama, J., "Improving the Offline Clustering Stage of Data Stream Algorithms in Scenarios with Variable Number of Clusters", in Proceedings of the 27th ACM Symposium On Applied Computing, (SAC'12), 2012, pp. 829- 830 (pôster).

Os artigos em avaliação são:

- E. R. Faria, I. R. Gonçalves, A. C. P. L. F. Carvalho, J. Gama, "Novelty Detection in Data Streams", Artificial Intelligence Review.

- E. R. Faria, I. R. Gonçalves, A. C. P. L. F. Carvalho, J. Gama, "On Evaluating Multi-Class Novelty Detection Algorithms from Data Streams", IEEE Transactions on Knowledge and Data Engineering (TKDE).

1.4 Estrutura do Trabalho

O trabalho está organizado como segue.

- Capítulo 2: define o que são FCDs, suas características e aplicações. Também são mostrados os principais problemas envolvendo as tarefas de classificação e agrupamento em FCDs.

- Capítulo 3: mostra uma visão geral do que é DN, principais desafios a serem tratados, diferentes abordagens para tratar o problema em cenários envolvendo FCDs e exemplos de aplicações.

- Capítulo 4: descreve o algoritmo proposto neste trabalho – MINAS – desenvolvido para DN em cenários envolvendo duas ou mais classes e FCDs.

- Capítulo 5: apresenta a metodologia de avaliação proposta neste trabalho para avaliar algoritmos de DN em problemas com múltiplas classes em FCDs.

- Capítulo 6: apresenta os experimentos realizados com o algoritmo MINAS e com outros algoritmos disponíveis na literatura, usando bases reais e artificiais. Apresenta também uma aplicação real, que visa o reconhecimento das atividades que um usuário desempenha usando dados de acelerômetros.

- Capítulo 7: discute os principais resultados obtidos durante este trabalho de pesquisa, as principais deficiências dos métodos propostos, bem como possíveis trabalhos futuros.

CAPÍTULO

2

Fluxos Contínuos de Dados

2.1 Introdução

A área de Aprendizado de Máquina (AM) tem como objetivo estudar algoritmos que melhorem o seu desempenho para uma data tarefa de acordo com a sua experiência (Mitchell, 1997). Existem hoje várias técnicas que possibilitam o AM, sendo aplicadas a uma gama de problemas. As técnicas tradicionais de AM trabalham em cenários *batch*, no qual um conjunto de treinamento está disponível previamente e, uma vez aprendido um modelo de decisão, esse não muda com o passar do tempo.

Recentes avanços em hardware e software possibilitaram a aquisição de dados em larga escala (Aggarwal, 2006). Todavia, lidar com grandes quantidades de dados impõe desafios aos pesquisadores, devido às limitações físicas dos atuais recursos computacionais. Nas últimas décadas, tem-se visto um crescente interesse em gerenciar essas grandes quantidades de dados, que são sequências infinitas de dados, geradas continuamente (em geral, em alta velocidade), conhecidas como fluxos contínuos de dados (FCDs) (Aggarwal, 2006), (Gama, 2010), (Gama e Gaber, 2007). Uma característica importante dos FCD é que os conceitos podem mudar ao longo do tempo, ou até mesmo, novos conceitos podem surgir.

Extrair conhecimento útil a partir de FCDs é uma tarefa desafiadora. A maioria dos algoritmos de AM assumem que o conjunto de dados usado para o treinamento é finito, representativo e contempla todas as classes do problema, que os dados são gerados por uma distribuição de probabilidade estacionária e que estes podem ser armazenados fisicamente e analisados por um algoritmo que executa em múltiplos passos. Assim, torna-se

necessário o desenvolvimento de algoritmos para lidar com FCDs, que possam atender a todos os requisitos exigidos por esse tipo de dados, que não podem ser tratados usando a MD clássica.

Este capítulo apresenta os FCDs, suas principais características e alguns dos principais algoritmos de AM para trabalhar com FCDs. A Seção 2.2 mostra uma visão geral dos FCDs, listando as principais características desse cenário e exemplos de aplicações. A Seção 2.3 mostra uma visão geral da problemática envolvendo MD em FCDs. A Seção 2.4 descreve a tarefa de agrupamento em FCDs, mostrando os detalhes de um importante algoritmo da área que é o CluStream, usado neste trabalho. A Seção 2.5 descreve a tarefa de classificação em FCDs. Por último, a Seção 2.6 apresenta as considerações finais do capítulo. Muitas das seções apresentadas neste capítulo são baseadas no trabalho (Silva et al., 2013).

2.2 Principais Aspectos dos FCDs

2.2.1 Visão Geral dos FCDs

Segundo Gama (2010), durante muitos anos, a área de AM concentrou-se em resolver problemas de aprendizado não-incrementais (ou *batch*), no qual todo o conjunto de dados de treinamento está disponível para a fase de aprendizado. Após o processamento desses dados de treinamento, uma ou várias vezes, um modelo de decisão é gerado. O modelo gerado é estático e será usado para fazer predições assim que novos exemplos estiverem disponíveis. Esse cenário descrito é bastante utilizado em problemas em que a distribuição que gera os dados é estacionária. Nesse caso, o modelo preditivo permanecerá sempre o mesmo. Vários algoritmos como K-Means (Lloyd, 1982) (MacQueen, 1967), Algoritmos de Indução de Árvores de Decisão (Rokach e Maimon, 2008) e SVM (Steinwart e Christmann, 2008) foram desenvolvidos para trabalhar em ambientes *batch*.

No entanto, vários problemas do mundo real são dinâmicos. Nesse cenário, os dados fluem continuamente ao longo do tempo, e a distribuição que gera os exemplos pode mudar com o passar do tempo, ou seja, não é estacionária. Esse cenário é conhecido como FCDs. No contexto de classificação, segundo Gama e Rodrigues (2009), o grande problema no aprendizado a partir de FCDs é a habilidade em manter permanentemente um modelo de decisão com boa acurácia. Assim, o algoritmo de aprendizado deve modificar o modelo de decisão corrente sempre que novos dados são analisados, necessitando de algoritmos de aprendizado incremental.

Além de ser incremental, outros requisitos são necessários para o trabalho com FCDs. Em geral, vários dados chegam continuamente em um curto espaço de tempo, exigindo assim que o modelo de decisão seja rápido para tomar sua decisão. Também, a quantidade de dados gerados é muito grande, exigindo que os exemplos não sejam armazenados,

mas utilizados e em seguida descartados. Outro ponto a ser destacado é que, à medida que o tempo passa, o modelo de decisão evolui, sendo necessário esquecer os exemplos do passado e concentrar-se nos exemplos do presente. Por último, é válido lembrar que ruídos ou *outliers* podem estar presentes nos dados e o algoritmo deve ser capaz de tratá-los adequadamente. Essas são algumas das características que tornam necessária a construção de algoritmos de AM especialmente desenvolvidos para trabalhar com FCDs, e que tornam essa tarefa tão desafiadora.

Os problemas envolvendo MD e FCDs podem ser categorizados em três principais abordagens (Khalilian e Mustapha, 2010): algoritmos de aprendizado supervisionado, não supervisionado e extração de regras de associação. Em geral, os algoritmos clássicos para MD não atendem a todos os requisitos exigidos para MD em FCDs (a serem detalhados na Seção 2.2.2). Além disso, produzir um novo modelo de decisão, sempre que um novo exemplo é gerado, pode ser uma abordagem bastante custosa. Assim, na literatura encontram-se vários algoritmos especialmente desenvolvidos para atender à MD em FCDs (Aggarwal et al., 2003), (Domingos e Hulten, 2000) e (Chang e Lee, 2005).

Em geral, pequenas modificações em algoritmos existentes não são suficientes para atender a todos os requisitos exigidos pelo FCDs (a serem detalhados na Seção 2.2.2). Além disso, produzir um novo modelo de decisão, sempre que um novo exemplo é gerado, pode ser uma abordagem bastante custosa. Assim, na literatura encontram-se vários algoritmos especialmente desenvolvidos para atender à MD em FCDs (Aggarwal et al., 2003), (Domingos e Hulten, 2000) e (Chang e Lee, 2005).

2.2.2 Características dos FCDs

Segundo Babcock et al. (2002) FCDs diferem de dados convencionais de várias maneiras:

- Os dados chegam de forma contínua, e em geral, em alta velocidade;
- O sistema não tem controle sobre a ordem na qual os exemplos chegam para serem processados;
- FCDs têm tamanho ilimitado;
- Uma vez que um exemplo foi processado, ele é descartado ou armazenado, não podendo ser recuperado facilmente, a não ser que seja explicitamente armazenado na memória, que é relativamente pequena comparada com o tamanho do FCD.

Outra característica importante em FCDs é que, em geral, os dados evoluem ao longo do tempo, visto que são gerados por uma distribuição de probabilidade não estacionária, ocasionando mudanças nos conceitos aprendidos. Assim, novas classes/grupos podem surgir ou desaparecer e classes/grupos conhecidos podem evoluir ao longo do tempo. Portanto, é importante que os algoritmos para FCDs consigam decidir quais dados são relevantes

para o cenário atual, quais dados são redundantes e quais podem ser esquecidos, pois tornaram-se obsoletos para o problema atual.

É importante notar que tarefas clássicas do cenário *batch*, como encontrar a média ou a entropia de um conjunto de exemplos, tornam-se muito mais complicadas no contexto de FCDs. Elementos que dificultam o cálculo dessas medidas são o fato de que o fluxo não é finito e não há conhecimento a priori do número de classes.

2.2.3 Exemplos de Aplicações de FCDs

Atualmente, existem várias aplicações do mundo real que geram grandes quantidades de dados em um fluxo contínuo. São exemplos de aplicações:

- **Redes e telecomunicações**: inclui sistemas de monitoramento de rede, segurança, detecção de intrusos em redes de computadores, fluxos de cliques em páginas da web e registros de chamadas telefônicas;

- **Finanças**: um exemplo típico inclui a detecção de fraudes em cartões de crédito. Outro exemplo são os dados da bolsa de valores, que devem ser processados assim que recebidos gerando informações *online* sobre o mercado aos clientes da bolsa;

- **Descoberta da evolução de um doença**: o conhecimento sobre uma dada doença pode ter sido modelado, no entanto, o conhecimento pode evoluir com o passar do tempo, à medida que novos casos são estudados;

- **Meteorologia**: entender fenômenos meteorológicos que acontecem com a evolução do tempo;

- **Comércio**: descobrir a evolução da carga de trabalho de um servidor de comércio eletrônico;

- **Energia**: monitoramento, predição e planejamento do consumo de energia elétrica de uma cidade/estado/país feito por uma companhia de energia elétrica;

- **Dispositivos móveis**: monitoramento das atividades desempenhadas por um usuário em tempo real, usando dados do seu dispositivo móvel, com fins de segurança, monitoramento de saúde, monitoramento de idosos, etc.

A tendência é que com a evolução da Tecnologia da Informação, mais dados e informações serão gerados e coletados constantemente em FCDs. Assim, há cada vez mais necessidade de algoritmos que trabalham com esse tipo de cenário.

2.3 Mineração em FCDs

Muitos algoritmos de AM trabalham com a ideia de que todos os dados de treinamento encontram-se previamente disponíveis. Eles também consideram que, uma vez que o classificador seja construído, com base em exemplos fornecidos em uma fase de treinamento, ele não será mais modificado, ou seja, usam a ideia de algoritmos não-incrementais. Esses algoritmos apresentam as seguintes características:

- O conhecimento não pode ser atualizado constantemente;
- Dependem da massa de dados fornecida na etapa de treinamento;
- Em geral, preocupam-se mais com a precisão do que com a eficiência.

No entanto, muitas das aplicações do mundo real são dinâmicas. Nessas aplicações, os conceitos a cada dia são atualizados e a aprendizagem ocorre continuamente. Assim, segundo Fisher (1987), a ideia de aprendizagem incremental nasce a partir da observação do mundo real, na qual o conhecimento pode rapidamente ser atualizado a cada nova observação, sustentando uma base contínua para reagir a novos estímulos. O aprendizado incremental acredita que o ambiente pode mudar ao longo do processo e, portanto, necessita que o conhecimento adquirido seja constantemente atualizado. Assim, o modelo de aprendizado precisa ser preciso e eficiente para se adaptar a essas mudanças.

Uma das definições de algoritmo incremental é dada em Langley (1995). A definição diz que um algoritmo L é incremental se L recebe como entrada um exemplo de treinamento de cada vez, não reprocessa qualquer exemplo já processado e mantém somente uma estrutura de conhecimento na memória. Ainda segundo o autor, um algoritmo incremental:

- Pode ter sua fase de treinamento interrompida a qualquer momento para fazer uma predição;
- O tempo para processar cada exemplo deve ser constante;
- Não retém na memória os dados previamente analisados.

É possível enumerar algumas vantagens dos algoritmos incrementais (Pinto, 2005):

- Poupam memória do computador;
- O tempo de processamento para a geração do modelo de aprendizado não é desperdiçado;
- Possuem maior rapidez de resposta;
- São importante em alguns domínios, como por exemplo, MD;

- Aproximam-se da forma de aprendizado dos humanos.

Um dos problemas dos algoritmos incrementais é que eles são sensíveis à ordem na qual os exemplos são fornecidos. Dependendo da ordem com que são fornecidos os exemplos, diferentes modelos podem ser gerados.

Pode-se perceber que a definição de algoritmos incrementais mostra-se bastante interessante para aplicação em cenários com FCDs. No entanto, Gama e Rodrigues (2009) salientam que o aprendizado a partir de FCDs exige algoritmos de aprendizado incremental que também levem em conta a mudança de conceito envolvida no processo. Nesse caso, é necessário descartar exemplos antigos, que não são condizentes com o cenário atual e adaptar o modelo de decisão aos novos dados.

A mineração de dados clássica usa técnicas como agrupamento (*clustering*), classificação e mineração de regras de associação (mineração de padrões frequentes). Para cada uma dessas técnicas, novos algoritmos foram desenvolvidos para trabalhar com FCDs.

2.4 Agrupamento (*clustering*) em FCDs

A prática de classificar objetos de acordo com as similaridades percebidas é a base para grande parte da ciência. Organizar dados em grupos é um dos modos mais fundamentais de compreensão e aprendizagem (Jain e Dubes, 1988). Agrupamento é a tarefa de agrupar objetos em diferentes grupos de acordo com sua similaridade. Espera-se que os objetos de um grupo possuam alta similaridade e objetos de grupos diferentes possuam baixa similaridade. O agrupamento é uma ferramenta para explorar a estrutura dos dados e não exige os pressupostos comuns à maioria dos métodos estatísticos. A representação obtida pode ser investigada para ver se os grupos estão de acordo com ideias preconcebidas ou sugerir novos experimentos (Jain e Dubes, 1988).

Várias áreas do conhecimento têm usado o conceito de agrupamento de dados estimulando o desenvolvimento de uma grande variedade de algoritmos. Dentre os importantes algoritmos para agrupamento em cenários *batch* destacam-se: K-Means (Lloyd, 1982)(MacQueen, 1967), DBSCAN (Ester et al., 1996), OPTICS (Ankerst et al., 1999), PAM (Kaufman e Rousseeuw, 2005), CURE (Guha et al., 1998), ROCK (Guha et al., 1999), CHAMALEON (Karypis et al., 1999) e COBWEB (Fisher, 1987) . Uma das possíveis razões para a diversidade de algoritmos de agrupamento é que não há uma definição geral de grupos, e em geral, diferentes métodos de agrupamento se adaptam a diferentes tipos de aplicação (Kaufman e Rousseeuw, 2005).

Segundo Gama (2010), o problema de agrupamento em FCDs é definido como: manter continuamente um agrupamento de dados bom e consistente com a sequência de dados observadas, usando uma pequena quantidade de memória e tempo. As dificuldades são impostas pela chegada contínua de dados e a necessidade de analisá-los em tempo real.

Assim, essas características requerem algoritmos de agrupamentos incrementais, mantendo estruturas de grupos que evoluam ao longo do tempo. Além disso, algoritmos para FCDs exigem uma mineração *online*, na qual deseja-se minerar os dados de forma contínua, e uma mineração *offline*, baseada nas necessidades do usuário (Aggarwal, 2006). Segundo Barbará (2002) a natureza do FCD exige três requisitos em algoritmos de agrupamento:

- Compacidade da representação;
- Processamento rápido e incremental de novos dados;
- Identificação clara e rápida de *outliers*.

Para Aggarwal et al. (2003), o agrupamento de dados é um problema difícil a ser resolvido no domínio de FCD. A justificativa para esse fato é a grande quantidade de dados chegando constantemente, o que torna os algoritmos tradicionais muito ineficientes. Alguns algoritmos de agrupamento de dados que fazem apenas um varredura nos dados foram desenvolvidos para lidar com FCD. No entanto, ainda segundo Aggarwal et al. (2003), apesar desses algoritmos tratarem as questões de escalabilidade da tarefa de agrupamento, eles não consideram a evolução dos dados e não tratam as seguintes questões: a qualidade dos grupos é pobre quando os dados evoluem consideravelmente sobre o tempo e não oferecem funcionalidades para a descoberta e exploração dos grupos sobre diferentes partes do fluxo. Pode-se, ainda acrescentar como problemas dificultadores para o agrupamento em FCD, o fato de que um ruído pode ser confundido com uma mudança nos dados, e que as mudanças de conceito tanto podem ser abruptas como graduais (Khalilian e Mustapha, 2010).

De acordo com Chen e Tu (2007), algumas propostas para agrupamento em FCD usam um modelo que executa uma única varredura sobre os dados, mas que tratam o agrupamento em FCD com uma versão contínua dos agrupamento de dados estáticos, por exemplo (Guha et al., 2003). Esses algoritmos usam a estratégia dividir-para-conquistar, que particiona o FCD em segmentos e descobre grupos usando algoritmos baseados no K-Means em um espaço finito (Guha et al., 2000), (O'Callaghan et al., 2002). Uma limitação de tais esquemas é que eles dão o mesmo peso para dados antigos e recentes e não conseguem capturar as características que evoluem ao longo do fluxo.

Muitos algoritmos foram criados para agrupamento em FCD. Dentre os algoritmos de agrupamento para FCD destacam-se: CluStream (Aggarwal et al., 2003), DenStream (Cao et al., 2006) e StreamKM++ (Ackermann et al., 2012), D-Stream (Chen e Tu, 2007), ClusTree (Kranen et al., 2011).

2.4.1 Vetor de Características

Uma ideia muito utilizada em agrupamento em FCD é o conceito de vetor de características do grupo (*Cluster Feature Vector* - CF-Vetor), introduzido por Zhang et al. (1997). Um CF-Vetor, ou um micro-grupo, é uma representação compacta de um conjunto de exemplos. Um CF-vetor sumariza os exemplos de um grupo, mantendo uma tripla:

- N: número de exemplos no grupo;
- LS: soma linear dos N exemplos do grupo;
- SS: soma quadrada dos N exemplos do grupo.

As propriedades do CF-Vetor são (Gama, 2010):

- Aditividade: Seja $CF1$ e $CF2$ CF-vetores de dois grupos disjuntos. Então, o CF-Vetor do grupo que é formado pela união dos dois grupos disjuntos é:

$$CF1 + CF2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

- Incrementalidade: Se um exemplo x é adicionado ao grupo, então as estatísticas daquele grupo são atualizadas da seguinte forma:

$$\begin{gathered} LS \leftarrow LS + x \\ SS \leftarrow SS + x^2 \\ N \leftarrow N + 1 \end{gathered}$$

  Dado o CF-Vetor de um grupo X_i, é possível calcular medidas como o centróide e raio.

  – Centróide:

$$\begin{aligned} \vec{X}0 &= \frac{\sum_{i=1}^{N} \vec{X}_i}{N} \\ \vec{X}0 &= \frac{\vec{LS}}{N} \end{aligned} \tag{2.1}$$

  – Raio:

$$\begin{aligned} R &= \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X}0)^2}{N} \right)^{\frac{1}{2}} \\ R &= \left(\frac{\vec{SS}}{N} - \left(\frac{\vec{LS}}{N} \right)^2 \right)^{\frac{1}{2}} \end{aligned} \tag{2.2}$$

Para maiores detalhes de como calcular essas medidas usando apenas o CF-Vector, ver (Zhang et al., 1997).

2.4.2 CluStream

Existem vários algoritmos para agrupamento em FCDs. Neste trabalho será descrito apenas o algoritmo CluStream, pois ele foi utilizado na fase *offline* do algoritmo proposto neste trabalho. Uma descrição detalhada dos demais algoritmos para agrupamento pode ser encontrada em Silva et al. (2013).

CluStream (Aggarwal et al., 2003) é um *framework* desenvolvido para trabalhar com agrupamento em FCD que utiliza a ideia de micro-grupos. O agrupamento dos dados é dividido em uma fase *online* e uma fase *offline*. A fase *online* mantém um sumário estatístico dos dados, periodicamente, e a fase *offline* usa esse sumário a fim de fornecer ao analista uma rápida compreensão dos grupos no FCD. O sumário estatístico dos dados é mantido em micro-grupos, que são definidos como extensões temporais do CF-Vetor proposto no algoritmo BIRCH (Zhang et al., 1996). Em cada momento, q micro-grupos são armazenados na memória, sendo q um parâmetro definido pelo usuário.

Cada micro-grupo armazena além das 3 informações do CF-Vector (N, LS e SS), $CF1^t$ - soma dos marcadores de tempo (*timestamps*) e $CF2^t$ - soma quadrada dos marcadores de tempo. Lembrando que LS e SS correspondem a um vetor de d entradas, sendo d a dimensionalidade dos dados. Para obter os q micro-grupos iniciais, o algoritmo K-Means é executado sobre um conjunto inicial de dados. O tamanho desse conjunto inicial é indicado pelo parâmetro *InitNumber*. A seguir, para cada novo exemplo, o micro-grupo mais próximo é escolhido para absorvê-lo. O micro-grupo mais próximo é encontrado calculando a distância Euclidiana entre o exemplo e o centróide de cada micro-grupo e armazenando o micro-grupo com a menor distância. Se a distância entre o novo exemplo e o centróide do micro-grupo mais próximo está dentro de um limite máximo, então o novo objeto é absorvido. O limite máximo de um micro-grupo é calculado pelo desvio padrão da distância dos exemplos do grupo ao centróide do grupo (também chamado raio) multiplicado por um fator fat (em geral, o valor $fat = 2$ é usado). Se nenhum dos micro-grupos pode absorver o novo dado, então um novo micro-grupo é criado. Para criar um novo micro-grupo, o micro-grupo mais antigo deve ser removido ou dois micro-grupos devem ser unidos, de forma a manter sempre q micro-grupos na memória. O micro-grupo mais antigo é deletado se o seu marcador de tempo está abaixo de um dado limiar δ (parâmetro de entrada), sendo então considerado um *outlier* que deve ser removido. Assim, o algoritmo CluStream encontra o tempo de chegada (conhecido como *tempo de relevância*) de $m/(2N_i)$ porcento dos N_i exemplos em um micro-grupo i, cujos marcadores de tempo são assumidos serem normalmente distribuídos. Caso o micro-grupo mais antigo não seja removido, os dois mais próximos são unidos, usando a propriedade da aditividade do CF-vetor.

Na fase *online*, os q micro-grupos são armazenados em um dispositivo de armazenamento secundário de tempos em tempos, em intervalos de tempo que decrescem exponencialmente — α^l, onde α e l são parâmetros definidos pelo usuário — os chamados

snapshots. Esses *snapshots* permitem ao usuário pesquisar por grupos em diferentes horizontes de tempo, h, através de um conceito de janela de tempo piramidal (*pyramidal time window* (Aggarwal et al., 2003)).

O algoritmo Clustream também possibilita realizar um macro-agrupamentos dos micro-grupos mantidos na fase *online*. Para isso, uma fase *offline* é executa, na qual os micro-grupos são agrupados em k grupos, sendo k um parâmetro de entrada do algoritmo, representando por um valor menor que o número de micro-grupos. Para esse agrupamento, o algoritmo K-Means é executado, com algumas particularidades. São elas:

- Na fase inicial, as sementes não são escolhidas aleatoriamente, mas amostradas com probabilidade proporcional ao número de objetos em um dado micro-grupo. A semente correspondente é o centróide de cada micro-grupo;
- Na fase de particionamento, a distância de uma semente a um pseudo-objeto (ou micro-grupo) é igual à distância da semente ao centróide do correspondente micro-grupo;
- Na fase de ajuste da semente, uma nova semente para uma dada partição é definida como o centróide ponderado dos micro-grupos naquela partição.

2.5 Classificação em FCD

Problemas de classificação foram e continuam sendo muito estudados, sendo considerados uma das principais categorias das tarefas de análise de dados em AM, inferência estatística e MD (Aggarwal, 2006). Métodos de classificação fazem parte das técnicas de aprendizado supervisionado, na qual uma variável será predita com base em um conjunto de variáveis de entrada. Se a variável a ser predita é categórica, tem-se um problema de classificação. Se a variável a ser predita é numérica, tem-se um problema de regressão. Para a tarefa de classificação, primeiramente, um modelo de decisão é construído, baseado em um conjunto de dados de treinamento. O modelo obtido é avaliado utilizando um conjunto de testes.

Muitos métodos foram desenvolvidos para trabalhar com classificação em cenários *batch*, como por exemplo, árvores de decisão (Breiman et al., 1984)(Quinlan, 1993), métodos baseados em regras, redes neurais e máquinas de vetores suporte (SVM) (Vapnik, 1998). No entanto, a maioria das técnicas desenvolvidas para trabalhar com classificação consideram ambientes estáticos, onde uma vez criado o modelo de decisão, ele não muda (Aggarwal, 2006). Além disso, para a construção do modelo, os algoritmos assumem que todos os dados estão na memória e portanto, fazem várias varreduras sobre eles. Assim, os algoritmos de classificação projetados para ambientes estáticos precisam ser repensados para atender as características presentes em FCD.

O problema de classificação em FCD pode ser definido da seguinte forma (Chen et al., 2009). Assuma que o FCD consiste em uma sequência de exemplos, sendo que cada exemplo consiste em um conjunto de d atributos. Cada atributo pode ser um valor numérico ou categórico. A cada exemplo está associado um rótulo, representado por um valor discreto, que indica a classe a qual o exemplo pertence. O objetivo da classificação em FCD é predizer, com alta acurácia, a classe dos exemplos desconhecidos, que estão constantemente chegando na forma de FCD.

De acordo com Aggarwal (2006), o desafio mais importante encontrado na classificação em FCD está relacionado à mudança de conceito, ou seja, a evolução dos dados ao longo do tempo. Para Chen et al. (2009), o principal impulsionador da mineração de FCD no contexto de classificação é a exigência de uma única varredura nos dados. Assim, é impossível usar os algoritmos convencionais de classificação que assumem que os dados cabem na memória, fazem várias varredura sobre os dados e criam modelos de decisão estáticos.

Os algoritmos de classificação em FCD trabalham, em geral, em duas fases bem definidas, *offline* e *online*. Na fase *offline*, ocorre a indução de um modelo de decisão, baseado em um conjunto de exemplos previamente rotulados. Na fase *online*, acontece a predição da classe de um novo exemplo que acabou de chegar, baseado no modelo de decisão induzido na fase *offline*. Entretanto, devido a evolução dos dados ao longo do fluxo, o modelo de decisão precisa ser atualizado ao longo do tempo. Para realizar essa atualização, muitos algoritmos supõem que novos exemplos rotulados sempre estarão disponíveis a fim de serem usados para atualizar o modelo de decisão.

Segundo Aggarwal (2006), as principais questões que precisam ser tratadas em FCD são: alta velocidade do FCD e memória ilimitada, mudança de conceito, compromisso entre acurácia e eficiência, desafio para trabalhar com aplicações distribuídas, ambiente de mineração interativo para satisfazer às necessidades do usuário e visualização dos resultados da mineração do FCD. Além destas, pode-se acrescentar: tratamento de ruídos ou *outliers*, tratamento de dados categóricos e dados complexos, desenvolvimento de técnicas de pré-processamento de baixo custo computacional (Gaber et al., 2005), desenvolvimento de técnicas que lidem com a escassez de dados rotulados, alteração no número de classes ao longo do fluxo (também conhecido como evolução de conceitos)(Masud et al., 2011a).

Nos últimos anos, vários algoritmos de classificação para FCDs foram desenvolvidos usando propostas incrementais, que fazem uma única varredura nos dados e que atualizam o modelo de decisão de tempos em tempos (Domingos e Hulten, 2000) (Wang et al., 2003) (Aggarwal et al., 2004) (Gama et al., 2003) (Bifet et al., 2013). No entanto, a maioria desses trabalhos supõem que o rótulo verdadeiro de todas as instâncias estarão disponíveis para que novas fases de aprendizado supervisionada sejam executadas e o modelo de decisão seja atualizado. Além disso, esses trabalhos consideram que o número de classes é fixo e previamente conhecida.

DN é uma tarefa de classificação que pode ser aplicada em muitos problemas do mundo real envolvendo FCDs. A DN consiste em identificar exemplos (ou grupos de exemplos) que não são coerentes com o modelo de decisão previamente aprendido. A DN pode indicar que o conceito alvo aprendido mudou ou que novos conceitos estão surgindo. Em geral, os algoritmos para DN em FCD criam um classificador na fase de treinamento, usando um conjunto de exemplos rotulados, que representam os conceitos conhecidos do problema. Na fase de aplicação, exemplos não rotulados chegam constantemente ao longo do fluxo, e são rotulados de acordo com o modelo de decisão atual, ou então são marcados com o perfil *desconhecido*, indicando que o modelo de decisão não tem informação suficiente para classificá-los. Grupos de exemplos *desconhecidos* são usados para modelar novos conceitos ou extensões dos conceitos conhecidos.

2.6 Considerações Finais

Esse capítulo apresentou uma visão geral sobre FCD, suas principais características e aplicações. O capítulo também mostrou alguns algoritmos de MD desenvolvidos para lidar com FCD.

O próximo capítulo deste trabalho mostra uma revisão da literatura dos principais trabalhos sobre DN em FCD. As principais deficiências desses trabalhos são também apresentadas bem como as principais questões que precisam ser tratadas na área.

CAPÍTULO

3

Detecção de Novidade

3.1 Introdução

Detecção de Novidade (DN) é a habilidade para identificar que uma instância não rotulada (ou um conjunto delas) difere significativamente dos conceitos já aprendidos. Uma vez que esta é uma importante habilidade para sistemas de aprendizado, DN tem recebido considerável atenção dos pesquisadores de AM. Na literatura é possível encontrar várias definições para DN, tais como:

- É o reconhecimento que uma entrada difere de alguma forma das entradas anteriores (Perner, 2009).
- Está relacionada a identificar comportamentos anormais do sistema e mudanças abruptas de um regime para outro (Lee e Roberts, 2008).
- Torna possível reconhecer um conceito novidade, que pode indicar o surgimento de novo conceitos, uma mudança ocorrida nos conceitos conhecidos ou a presença de ruído (Gama, 2010).

Diversos trabalhos tratam o problema de DN em contextos *batch*, no qual é assumido que um conjunto de dados representativo está disponível (Markou e Singh, 2003a), (Markou e Singh, 2003b), (Marsland et al., 2002), (Schölkopf et al., 2000) e (Hoffmann, 2007). No entanto, atualmente, um importante cenário no qual a tarefa de DN representa um desafio a ser tratado são os FCDs.

FCDs são uma sequência de exemplos gerados continuamente. Eles são ilimitados, fluem em alta velocidade e a distribuição que gera os dados pode mudar ao longo do tempo (Silva et al., 2013). Em FCDs, novos conceitos podem aparecer, conceitos conhecidos podem desaparecer ou evoluir ao longo do tempo.

Uma vez que os conceitos não são constantes, a aplicação das técnicas de DN em FCDs apresentam muitos desafios (Gama, 2010) que incluem a presença de:

- Mudança de conceito, que dificulta distinguir novos conceitos dos conceitos conhecidos;
- Ruído ou *outlier*, que podem ser confundidos com o aparecimento de novos conceitos;
- Contextos recorrentes, os quais podem ser confundidos com o aparecimento de um novo conceito;
- Evolução de conceitos, que ocorre quando o número de classes do problema aumenta ao longo do tempo.

Dentre as muitas aplicações nas quais o uso de técnicas de DN são importantes, é possível destacar: detecção de intrusos em uma rede de computadores (Coull et al., 2003), (Spinosa et al., 2008), detecção de falhas (Zhang et al., 2006), diagnóstico médico (Perner, 2009),(Spinosa e Carvalho, 2004), detecção de regiões de interesse em imagens (Singh e Markow, 2004), detecção de fraude (Wang et al., 2003), detecção de tipo de cobertura de florestas (Masud et al., 2011a), filtro de *spam* (Hayat et al., 2010), recuperação da informação (Gaughan e Smeaton, 2005) e classificação de texto (Li, 2006).

O objetivo desse capítulo é descrever e analisar os principais algoritmos para DN em FCDs. Esse capítulo é baseado em um artigo que foi submetido para a revista *Artificial Intelligence Review* e encontra-se em processo de avaliação. Há importantes revisões bibliográficas sobre DN (Markou e Singh, 2003a), (Markou e Singh, 2003b), (Marsland, 2003), (Pimentel et al., 2014), detecção de anomalia (Chandola et al., 2009) e detecção de *outlier* (Gogoi et al., 2011), (Hodge e Austin, 2004). Entretanto, esses trabalhos só tratam cenários *batch* e não tratam a tarefa de DN no contexto de FCDs. Além disso, esses trabalhos focam apenas no método usado para a DN, sem detalhar os demais aspectos da tarefa, como por exemplo, tratamento de ruídos e *outliers* e atualização do modelo.

Nos últimos anos, diversos pesquisadores propuseram diferentes trabalhos para DN em FCDs (Masud et al., 2011a), (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Farid e Rahman, 2012), (Al-Khateeb et al., 2012a), (Al-Khateeb et al., 2012b). Uma revisão dos principais aspectos desses trabalhos pode fornecer uma inspiração para o desenvolvimento de novas técnicas e identificar os principais desafios a serem tratados pelos pesquisadores. Para isso, esse capítulo cobre diferentes aspectos da DN em FCDs, tais como:

- Definição das principais propostas pra DN;
- Formalização da DN em FCDs, claramente distinguindo as fases *offline* e *online*;
- Taxonomia das principais propostas para DN em FCDs;
- Tarefa de aprendizado;
- Número de classes e classificadores considerados em cada uma das duas fases;
- *Feedback* externo para atualização do modelo;
- Mecanismos para esquecimento;
- Tratamento de ruídos e *outliers*;
- Tratamento de contextos recorrentes;
- Metodologias de avaliação.

3.2 Definições Importantes

Essa seção apresenta as diferenças entre detecção de novidade, anomalia e *outlier* segundo a interpretação seguida neste trabalho. Além disso, conceitos importantes relacionados a DN, tais como, mudança de conceito e evolução de conceito, são apresentados.

3.2.1 Detecção de Novidade, Detecção de Anomalia e Detecção de *Outlier*

Detecção de novidade, anomalia e *outlier* são termos co-relacionados. Enquanto em alguns contextos esses três termos parecem ter o mesmo significado e são usados indistintamente, neste trabalho, eles serão diferenciados. Em geral, os dois últimos termos são mais similares e frequentemente usados para expressar problemas muito próximos.

De fato, detecção de novidade, anomalia e *outlier* são termos que dizem respeito a encontrar padrões que são diferentes dos padrões normais. Enquanto os termos anomalia e *outlier* dão a ideia de um padrão indesejado, o termo novidade indica um conceito novo ou emergente que precisa ser incorporado ao padrão normal.

De acordo com Chandola et al. (2009), detecção de anomalia refere-se a tarefa de encontrar padrões nos dados que não estão de acordo com o comportamento esperado. Esses padrões são também referenciados como anomalias, *outliers*, observações discordantes, exceções, aberrações, surpresa, peculiaridades ou contaminantes. Ainda de acordo com Chandola et al. (2009), DN objetiva detectar padrões não observados (emergentes, novidades) nos dados, entretanto, esse termo se distingue de detecção de anomalia, uma

vez que no primeiro, os padrões-novidades são tipicamente incorporados ao modelo normal depois de detectados.

Em Aggarwal (2013), os autores definem *outlier* como um dado, que pode ser considerado como anormalidade ou ruído, sendo que anomalia refere-se a um tipo especial de *outlier*, o qual é de interesse do analista. De acordo com Gogoi et al. (2011), *outliers* podem ser considerados dados anômalos que podem afetar o sistema adversamente, tais como, produzindo resultados incorretos, especificação incorreta dos modelos e estimação dos parâmetros enviesada. Algumas das causas dos *outliers* (Chandola et al., 2009) são: atividade maliciosa, erro de instrumentação, mudanças no ambiente e erro humano. Para Gama (2010), um grupo conciso de exemplos deve ser requerido como uma evidência do aparecimento de um novo conceito, ou novidade. Por outro lado, exemplos esparsos e independentes, cujas características diferem muito daqueles que definem o modelo normal, devem ser considerados simplesmente como *outliers*, uma vez que não há garantia de que eles representam conceitos.

De acordo com Marsland et al. (2002), DN é útil em casos nos quais uma classe importante do problema é sub-representada no conjunto de treinamento. Para Markou e Singh (2003a), é uma importante tarefa, uma vez que, para muitos problemas, não é possível saber se os dados de treinamento disponíveis incluem todos as possíveis classes de exemplos. De acordo com Gama (2010), DN torna possível reconhecer novos perfis (conceitos) em dados não rotulados.

Neste trabalho, novidade será tratada como um grupo representativo de exemplos que representam um novo conceito a ser incorporado no modelo de decisão. Esse novo conceito é diferente dos conceitos conhecidos do problema e representa uma evolução, como por exemplo, o aparecimento de uma nova classe. Exemplos isolados, grupos de exemplos não representativos ou não coesos, que não são explicados pelo modelo normal, são considerados *outliers*, sendo necessário identificá-los, mas não adicioná-los ao modelo de decisão.

3.2.2 Evolução de Conceito e Mudança de Conceito

Em Aprendizado de Conceito, um conceito é uma função a ser aprendida pelo algoritmo de aprendizagem, definida sobre um conjunto de exemplos, a qual mapeia entradas em saídas (Mitchell, 1997). Em FCDs, que representam um ambiente não estacionário, os conceitos não são estáticos, mas eles evoluem ao longo o tempo. Assim, dois importantes fenômenos, nomeados mudança de conceito e evolução de conceitos, podem ocorrer.

De acordo com Elwell e Polikar (2011), mudança de conceito refere-se a uma mudança nas definições das classes (conceitos) ao longo do tempo, e portanto uma mudança na distribuição a partir da qual os dados são gerados. De acordo com Dries e Rückert (2009), mudança de conceito é um importante problema em AM e MD, e pode ser descrito como

uma mudança significativa na distribuição dos dados. Para Karnick et al. (2008), o ambiente é não-estacionário e portanto há mudança de conceito, se há uma alteração na distribuição de dados entre dois intervalos de tempo consecutivos.

Para Widmer e Kubat (1996), na tarefa de aprendizado, o conceito de interesse pode depender de algum contexto escondido, e mudanças no contexto escondido podem induzir mudanças no conceito alvo, produzindo mudança de conceito. Para Tsymbal (2004), um problema difícil ao tratar mudança de conceito é distinguir entre uma mudança de conceito real e um ruído. Por exemplo, se os conceitos são vistos como formas em um espaço de representação, eles podem mudar suas formas, tamanhos e localizações (Kolter e Maloof, 2007). Como exemplo de mudança de conceito, pode-se destacar: as mudanças no padrão de uma doença, as mudanças no tempo, mudanças no perfil de compras de um cliente ao longo dos anos, etc.

Em algoritmos de AM tradicionais, é assumido que o número de classes é previamente definido. Esses algoritmos assumem que os exemplos pertencem a no mínimo uma das classes dentre um conjunto de classes predefinidas (Park e Shim, 2010). Entretanto, em cenários envolvendo FCDs, essa suposição não é válida, uma vez que nem todas as classes são conhecidas na fase de treinamento e exemplos de novas classes podem aparecer ao longo do fluxo. O fenômeno da evolução de conceito, também conhecido como classes emergentes, pode ocorrer em problemas como detecção de intrusão em redes de computadores, filtro de *spam* e classificação de texto.

É importante destacar que em tarefas de DN é fundamental lidar com mudança de conceitos, evolução de conceitos e presença de *outliers*. Algoritmos para DN devem ser capazes de atualizar o modelo de decisão a fim de representar mudanças ocorridas nos dados, detectar novas classes emergentes e atualizar o modelo de decisão com essas novas classes e identificar ruídos e *outliers* e descartá-los.

3.3 Formalização do Problema

FCDs são um sequência infinita de exemplos que fluem de forma contínua (em geral em alta velocidade), e cuja distribuição pode variar ao longo do tempo.

Definição 3.1. Fluxo Contínuo de Dados (*Data Stream*) Um fluxo contínuo de dados $\mathcal{S}$ é uma sequência maciça de exemplos multidimensionais $\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_N,...$, que é potencialmente infinita, cujos dados chegam nos marcadores de tempo $\mathbf{T}_1,\mathbf{T}_2,...,\mathbf{T}_N,....$ Cada exemplo x_i é descrito por um vetor de atributos $n-$dimensional (Aggarwal et al., 2003).

Fluxos contínuos de dados, assim como séries temporais, são conjuntos de dados sequenciais. De acordo com Han (2005), um conjunto de dados sequenciais consiste em uma sequência de eventos ordenados, com ou sem uma noção concreta de tempo. FCD

são tipicamente sequências multivariadas, na qual a noção de tempo nem sempre está presente. Sequências de cliques em páginas web e sequências de transações de compras de um cliente são exemplos de conjuntos de dados sequenciais, mas que podem não ser dados de séries temporais (Han, 2005). Uma série temporal é uma sequência numérica de observações de uma variável aleatória lida sequencialmente no tempo, que apresenta uma característica intrínseca de dependência entre dois exemplos adjacentes (Box e Jenkins, 1994). Em FCD essa dependência nem sempre ocorre e, em diversas situações, exemplos adjacentes são completamente independentes. Entretanto, séries temporais nas quais os dados fluem continuamente em alta velocidade, podem ser consideradas FCD.

Em geral, algoritmos para DN em FCDs trabalham em duas fase, conhecidas como *offline* e *online*. Na fase *offline*, um conjunto de exemplos rotulados são usados para induzir um classificador. Esses exemplos representam os conceitos conhecidos do problema. De agora em diante, os conceitos conhecidos do problema serão referenciados somente como conceitos conhecidos. Usualmente os conceitos conhecidos são composto por exemplos de uma única classe, chamada de classe normal. Na fase *online*, sempre que um novo exemplo chega, ele é classificado na classe normal ou então é rejeitado (ou classificado como anormal, anomalia ou novidade). Esse é o clássico cenário de classificação com uma classe (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Tax e Duin, 2008), (Denis et al., 2005), (Liu et al., 2003), (Yeung e Ding, 2003), (Yeung e Chow, 2002), (Aregui e Denœux, 2007), (Tan et al., 2011).

Recentemente, diversos autores estenderam esse *framework* para um contexto multi-classe (Farid e Rahman, 2012), (Masud et al., 2010a), (Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Al-Khateeb et al., 2012b). Nessa nova generalização, na fase *offline*, cada exemplo do conjunto de treinamento tem um rótulo y_i, onde $y_i \in Y^{tr}$, com $Y^{tr} = \{C_{knw_1}, C_{knw_2}, .., C_{knw_L}\}$, onde C_{knw_i} representa a $i-$ésima classe conhecida do problema e L é o número de classes conhecidas. Na fase *online*, assim que um novo exemplo chega, novas classes novidade podem ser detectadas, expandindo o conjunto de rótulos de classes para $Y^{all} = \{C_{knw_1}, C_{knw_2}, .., C_{knw_L}, C_{nov_1}..., C_{nov_K}\}$, onde C_{nov_i} representa a $i-$ésima *classe novidade* e K é o número de classes novidade, o qual é previamente desconhecido.

Definição 3.2. Classe Novidade: Uma classe que não está disponível na fase de treinamento (*offline*), aparecendo somente na fase *online*.

Inicialmente, um classificador consegue efetivamente classificar somente exemplos das classes de treinamento. Quando exemplos pertencentes às classes novidade aparecem ao longo do fluxo, eles são temporariamente classificados como *desconhecidos*.

Definição 3.3. *Desconhecido*: Um exemplo não explicado pelo modelo de decisão atual. Em classificação com uma classe, ele é nomeado como anormal, rejeitado ou anomalia, o

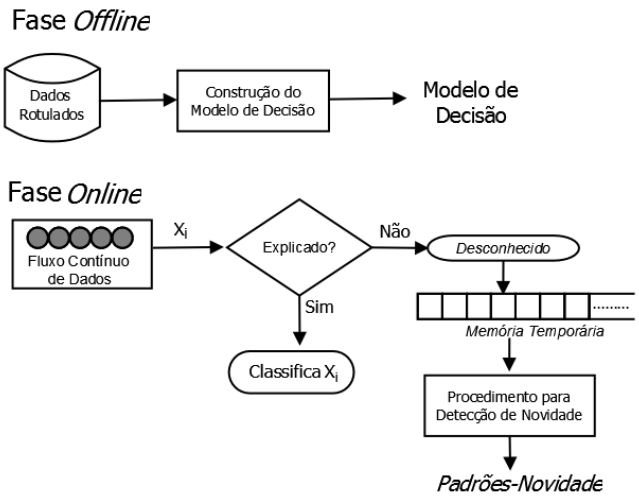


Figura 3.1: Visão geral da tarefa de DN.

exemplo não pertence ao conceito normal. Em alguns contextos, isso é suficiente. Entretanto, em problemas multiclasse, um grupo de exemplos *desconhecidos* pode ser usado para modelar novos conceitos.

Os exemplos *desconhecidos* são submetidos a um processo de DN a fim de produzir diferentes padrões-novidade. A Figura 3.1 mostra uma visão geral da tarefa de DN.

Definição 3.4. Padrão-novidade: Um padrão identificado em exemplos não rotulados, previamente considerados como *desconhecidos* pelo sistema de classificação.

É importante notar, que em FCDs, além da emergência de classes novidade, fenômeno conhecido como evolução de conceito, outro fenômeno deve ser considerado, a mudança de conceito. Neste caso, os conceitos conhecidos pode mudar ao longo do tempo.

Definição 3.5. Mudança de Conceito: Toda instância $\mathbf{x}_t$ é gerada a partir de um fonte que corresponde a uma certa distribuição D_t. Se para cada quaisquer dois exemplos x_1 e x_2, com marcadores de tempo t_1 e t_2, $D_1 \neq D_2$, então uma mudança de conceito ocorre (Farid et al., 2013).

Assim, é necessário evoluir o modelo de decisão atual a fim de representar estas mudanças. Do contrário, toda mudança no conceito conhecido do problema será detectada como um padrão-novidade. Além disso, também é necessário esquecer estruturas de classificação antigas, que não estão contribuindo para a atividade atual do fluxo.

Além disso, ruídos e *outliers* não devem ser considerados como padrões-novidade, uma vez que eles representam exemplos isolados ou grupos de exemplos não-coesos.

A próxima seção fornece uma visão geral dos principais trabalhos encontrados na literatura que investigam a tarefa de DN em FCDs.

3.4 Visão Geral da DN em FCDs

Essa seção apresenta a taxonomia usada para classificar os algoritmos de DN para FCDs. Essa taxonomia cobre a maioria dos estudos encontrados na literatura, e que são apresentados neste trabalho. A taxonomia caracteriza esses trabalhos de acordo com:

- Fase *offline*:
  - Número de classes (uma ou mais que uma)
  - Número de classificadores (um ou comitê de classificadores)
  - Tarefa de aprendizado (supervisionada ou não supervisionada)
- Fase *online*:
  - Classificação:
    - Classificação com a opção de perfil *desconhecido* (sim ou não)
    - Número de classificadores (um ou comitê de classificadores)
  - Detecção de novos padrões:
    - Número de classes novidade (uma ou mais que uma)
  - Atualização do modelo de decisão:
    - Com *feedback* externo (sim ou não)
    - Número de classificadores (um ou comitê de classificadores)
    - Mecanismo de esquecimento (sim ou não)
- Outros aspectos
  - Tratamento de *outliers*
  - Tratamento de contextos recorrentes
  - Medidas de avaliação

Os trabalhos a serem analisados são apresentados na Tabela 3.1.

Em geral, as técnicas para DN trabalham em duas fase *offline* e *online*. A fase *offline* detalhada na Seção 3.5, recebe um conjunto de treinamento e induz um modelo de decisão, assim como nas tarefas tradicionais de AM. Três importantes aspectos dessa fase são o tipo de aprendizado usado (supervisionado ou não supervisionado), o número de classificadores construídos (um ou comitê) e o número de classes que compõem o conceito conhecido do problema (uma ou mais que uma).

A fase *online*, detalhada na Seção 3.6 recebe um FCD não rotulado e executa três tarefas: classificação de novos exemplos, detecção de padrões-novidade e atualização do

Tabela 3.1: Algoritmos para DN em FCDs.

Algoritmo	Referência
1- ECSMiner - *Enhanced Classifier for Data Streams with novel class Miner*	(Masud et al., 2011a)
2- MCM - *Multi Class Miner in Data Streams*	(Masud et al., 2010a)
3- CLAM - *CLAss-based Micro classifier ensemble*	(Al-Khateeb et al., 2012a)
4- OLINDDA - *OnLine Novelty and Drfit Detection Algorithm*	(Spinosa et al., 2009)
5- DETECTNOD - *DiscrETE Cosine Transform based NOvelty and Drift detection*	(Hayat e Hashemi, 2010)
6- Tree for ND	(Farid e Rahman, 2012)
7- Ensemble Tree for ND	(Farid et al., 2013)
8- HS-Trees - *Streaming Half-Space-Trees*	(Tan et al., 2011)
9- SONDE - *Self-Organizing Novelty Detection*	(Albertini e de Mello, 2007)
10- Neural Network (NN) for ND	(Rusiecki, 2012)
11- Adaptive WOCSVM - *Weighted One-Class Support Vector Machine*	(Bartosz Krawczyk, 2013)

modelo de decisão. A tarefa de classificação usa o modelo de decisão atual para classificar novos exemplos que chegam constantemente ao longo fluxo. Algumas propostas classificam todo novo exemplo, enquanto outras optam por marcar com o perfil *desconhecido*, os exemplos classificados com baixa confiança. Além disso, a tarefa de classificação pode ser baseada em um classificador ou em um comitê de classificadores. Algumas propostas consideram que o conceito novidade é composto por somente uma classe, enquanto que outras distribuem o conceito novidade em mais que uma classe.

Outra diferença entre as propostas existentes é se elas usam *feedback* para atualizar o modelo de decisão. Além disso, algumas técnicas usam mecanismos de esquecimento, que descartam dados antigos que não representam o comportamento atual do FCDs.

Outros aspectos relevantes para DN são discutidos na Seção 3.7. O primeiro aspecto é o tratamento de *outliers*, que não é tratado pela maioria dos algoritmos. O segundo é o tratamento de contextos recorrentes. Por último, as medidas de avaliação e metodologia experimental usadas são também um aspecto importante a ser tratado.

A Tabela 3.2 classifica cada um dos algoritmos apresentados neste trabalho de acordo com a taxonomia previamente apresentada.

3.5 Fase *Offline*

A fase *offline* representa a fase de aprendizado estático do algoritmo. Nessa fase, os conceitos conhecidos são aprendidos usando um conjunto de dados rotulados. Essa fase é descrita neste trabalho com foco em três aspectos (veja Figura 3.2), número de classes que compõem os conceitos conhecidos, número de classificadores a serem construídos e tarefa de aprendizado.

3.5.1 Número de classes e número de classificadores

Diferentes propostas da literatura consideram que os conceitos conhecidos são composto por somente uma classe, também chamada de classe normal ou conceito normal. Assim, essas propostas empregam técnicas de classificação com uma classe para induzir

Tabela 3.2: Taxonomia dos algoritmos para DN em FCDs

Algoritmo	Fase *Offline*			Outros Aspectos		
	Nro de classes	Nro de classificadores	Tarefa de aprendizado	Trata contextos recorrentes	Trata *outliers*	Medidas de avaliação
ECSMiner	mais-que-uma	comitê	supervisionado	não	sim	Fnew, Mnew, Err
MCM	mais-que-uma	comitê	supervisionado	não	sim	Fnew, Mnew, Err, AUC
CLAM	mais-que-uma	comitê	supervisionado	sim	sim	Fnew, Mnew, Err
OLINDDA	uma	um	não-supervisionado	não	sim	Fnew, Mnew, Err
DETECTNOD	uma	um	não-supervisionado	não	não	Fnew, Mnew, Err
Árvores para DN	mais-que-uma	um	supervisionado	não	não	Fnew, Mnew, Err
Comitê de Árvores	mais-que-uma	comitê	supervisionado	não	não	Fnew, Mnew, Err
HS-Trees	uma	comitê	não-supervisionado	não	não	AUC
SONDE	uma	um	não-supervisionado	não	não	precisão, revocação, f-measure
RNs para DN	uma	um	não-supervisionado	não	não	2D-plot (sinal x resultados da DN)
WOCSVM Adaptativo	uma	um	não-supervisionado	não	não	acurácia

Algoritmo	Fase *Online*					
	Classificação		Detecção de Novidade	Atualização do Modelo de Decisão		
	Classificação com opção de rótulo desconhecido	Nro de classificadores	Nro de classes novidade	*Feedback* externo	Nro de classificadores	Mecanismo de esquecimento
ECSMiner	sim	comitê	uma	sim	comitê	sim
MCM	sim	comitê	mais-que-uma	sim	comitê	sim
CLAM	sim	comitê	uma	sim	comitê	sim
OLINDDA	sim	um	uma	não	um	não
DETECTNOD	sim	um	uma	não	um	não
Árvore para DN	não	um	uma	sim	um	não
Comitê de Árvores	sim	comitê	uma	sim	comitê	sim
HS-Trees	não	comitê	uma	não	comitê	sim
SONDE	não	um	uma	não	um	sim
RNs para DN	não	um	uma	sim	um	não
WOCSVM Adaptativo	não	um	uma	sim	um	sim

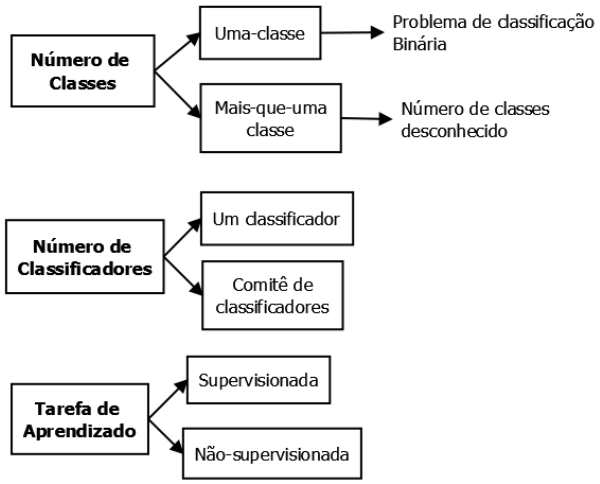


Figura 3.2: Taxonomia usada na fase *offline*.

um modelo de decisão, nomeado de modelo normal. Essa estratégia é usada por algoritmos como (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Tan et al., 2011), (Albertini e de Mello, 2007), (Rusiecki, 2012), (Bartosz Krawczyk, 2013).

Outras propostas consideram que os conceitos conhecidos são composto por mais de uma classe e usam técnicas de classificação multiclasse. Assim, o modelo de decisão induzido é capaz de distinguir entre as diferentes classes que compõem o conhecimento sobre o problema. Essa estratégia é usada por algoritmos como (Al-Khateeb et al., 2012a), (Masud et al., 2011a), (Masud et al., 2010a), (Farid e Rahman, 2012), (Farid et al., 2013).

O segundo aspecto a ser analisado na fase *offline* é o número de classificadores. As técnicas propostas nos trabalhos (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Albertini e de Mello, 2007), (Rusiecki, 2012), (Farid e Rahman, 2012), (Bartosz Krawczyk, 2013) criam um único classificador nessa fase, o qual será atualizado ao longo do FCD de forma

incremental. Técnicas encontradas nos trabalhos (Masud et al., 2010a), (Masud et al., 2011a), (Tan et al., 2011), (Al-Khateeb et al., 2012a), (Al-Khateeb et al., 2012b), (Farid et al., 2013) são baseadas em um comitê de classificadores. Já nos trabalhos (Al-Khateeb et al., 2012a), (Al-Khateeb et al., 2012b) um comitê é usado por classe. Em todos esse trabalhos, um bloco de dados de exemplos é usado para induzir cada classificador.

3.5.2 Tarefa de Aprendizado

Apesar dos rótulos dos exemplos estarem disponíveis na fase *offline*, muitos algoritmos não usam essa informação para construir o modelo de decisão. Assim, eles supõem que todos os exemplos do conjunto de treinamento pertencem ao conceito normal e portanto, não estão aptos a distinguir entre diferentes classes que podem vir a compor este conceito. Para esses algoritmos, a tarefa de aprendizado é não-supervisionada. Em oposição, diversos algoritmos usam propostas supervisionadas na construção de modelos de decisão, isto é, eles levam em consideração a informação sobre os rótulos dos exemplos. Assim, o modelo de decisão pode classificar exemplos de diferentes classes conhecidas. Essas propostas são detalhadas a seguir.

É importante destacar que se o modelo de decisão utilizado for estático, este deve ser substituído por um novo modelo a fim de representar as mudanças nos dados. Se o modelo for dinâmico, ele deve ser atualizado incrementalmente com a chegada de novos exemplos.

Também é importante destacar que na fase de aprendizado, muitos algoritmos da literatura usam métodos de agrupamento para construir o classificador, ou seja, um classificador é composto por um conjunto de grupos. A ideia é que usar grupos para representar as classes do problema possibilita a evolução do modelo de decisão, incorporando novos grupos, esquecendo grupos antigos ou ainda atualizando o centro e o raio dos grupos existentes.

Propostas Supervisionadas

As propostas supervisionadas usam a informação sobre os rótulos dos exemplos do conjunto de treinamento a fim de construir um modelo de decisão. Assim, os modelos de decisão construídos são capazes de representar uma ou mais classes conhecidas do problema. As técnicas existentes seguem propostas distintas para construir o modelo de decisão.

CLAM (Al-Khateeb et al., 2012a) cria um comitê de M classificadores para cada classe do problema. Ao todo, são criados c comitês, sendo que c representa o número de classes conhecidas. Nessa proposta, não são usados classificadores comumente empregados da literatura, mas sim um classificador que é composto por um conjunto de micro-grupo. Cada micro-grupo é sumarizado pelo seu raio, centro e número de elementos. Classificar um elemento, significa identificar o micro-grupo cujo centro está mais próximo ao elemento.

Cada classificador do comitê é construído a partir de diferentes blocos de dados. Um bloco de dados é uma porção de dados da base, sendo o seu tamanho definido pelo usuário. Cada bloco de dados é dividido em subconjuntos, cada um representando uma das classes presentes no bloco. Para cada classe, um conjunto de grupos é construído, usando o algoritmo K-Means.

MCM (Masud et al., 2010a) e ECSMiner (Masud et al., 2011a) usam somente um comitê para representar as classes conhecidas. No MCM (Masud et al., 2010a) o comitê é composto por M classificadores KNN. Já no ECSMiner (Masud et al., 2011a), o comitê é composto por M árvores de decisão. Cada classificador é construído usando os dados de um bloco e consideram todas as classes presentes nesse bloco. O tamanho de cada bloco de dados assim como número de classificadores no comitê M são parâmetros de entrada. No MCM (Masud et al., 2010a), cada classificador é formado por K grupos, obtidos usando o algoritmo K-Means, e um sumário de cada grupo (centróide, raio e frequência de ocorrência de cada classe) é armazenado. No ECSMiner, após construir a árvore de decisão, um agrupamento é realizado em cada nó folha da árvore, o qual é usado para identificar exemplos não explicados pelo modelo atual (*desconhecidos*). Em Masud et al. (2011a), os autores também propõem o uso de um comitê de classificadores KNN.

No trabalho de Farid et al. (2013), que é uma extensão do trabalho de (Farid e Rahman, 2012) que usa somente uma árvore de decisão, um comitê de árvores de decisão é proposto. O primeiro passo é inicializar um peso para cada exemplo do conjunto de treinamento D. O peso usado é o valor da probabilidade resultante da classificação desse exemplo usando um classificador Naive Bayes, construído a partir dos dados de treinamento. A primeira árvore é construída usando um conjunto de dados, nomeado D_{new}, obtido a partir de uma técnica de seleção com reposição aplicada em D. Para construir as duas outras árvores, os exemplos com os maiores pesos são selecionados, criando um novo conjunto de treinamento. O peso dos exemplos é atualizado, de acordo como sua classificação, realizada usando a última árvore de decisão. Cada nó folha da árvore é também submetido a um agrupamento e além disso, é computado a porcentagem de exemplos nesta folha em relação ao conjunto de treinamento. Um peso T é associado a cada árvore, baseado na sua taxa de acurácia obtida ao classificar os exemplos do conjunto de treinamento original.

Propostas não-supervisionadas

Diferentes técnicas para DN em FCDs não consideram o rótulo verdadeiro dos exemplos para construir o modelo de decisão na fase *offline*. Elas supõem que todos os exemplos do conjunto de treino pertencem a uma mesma classe, a chamada classe normal. Agrupamento e redes neurais (RN) são algumas das técnicas usadas por esses algoritmos.

OLINDDA (Spinosa et al., 2009) cria um conjunto de grupos para representar o conceito normal, sem distinguir entre diferentes classes normais. O algoritmo de agrupamento (por exemplo, o K-Means) produz um conjunto de k grupos, os quais são representados por um

centro e um raio. Além disso, uma macro-hiperesfera é criada ao redor dos grupos que compõem o modelo normal, cujo centro é o centróide dos grupos e o raio é a distância ao elementos mais distantes do centro. Essa macro-hiperesfera será usada na fase *online* para separar elementos do conceito extensão e do conceito novidade. Assim, o conceito normal é composto por somente uma classe, representada por um conjunto de grupos. DETECTNOD (Hayat e Hashemi, 2010) também usa um algoritmo de agrupamento para criar um conjunto de k grupos. Entretanto, um algoritmo de agrupamento é executado em cada grupo a fim de produzir sub-grupos. Usando interpolação, o mesmo número de exemplos é atribuído a cada sub-grupo. A fim de decrementar o uso da memória uma proposta baseada em DCT (*Discrete Cosine Transform*) é executada em cada um dos sub-grupos.

No trabalho de Albertini e de Mello (2007), os autores propõem uma RN não supervisionada, nomeada SONDE (*Self-Organizing Novelty Detection*), para representar a classe normal. Apesar dos autores não proporem seu uso para FCDs, a SONDE pode ser facilmente usada em FCDs. A SONDE consiste de três camadas: camada de entrada, que normaliza os dados, camada competitiva, na qual os neurônios competem para fornecer uma saída aos dados normalizados e BMU (*Best Matching Unit*), na qual o neurônio vencedor é usado pra representar um novo exemplo recebido. Neurônios são usados tanto para classificar novos dados de entrada quanto para detectar novidades. Cada neurônio é representado por um centróide, um raio médio e um nível de similaridade para reconhecer novos exemplos. Padrões de entrada similares são associados a um mesmo neurônio. Os exemplos usados para o treinamento da RN representam o conceito normal.

No trabalho de Rusiecki (2012), duas RNs auto-regressivas *feedforward* induzem o modelo de decisão. A primeira RN é treinada com um algoritmo de aprendizado robusto, que tenta minimizar o erro, decrementando a influência de *outliers* no conjunto de treinamento. A segunda usa o algoritmo tradicional de *backpropagation* para minimizar o erro quadrático mínimo. Uma janela de tempo de tamanho predefinido que se move em períodos discretos é usada para representar os dados mais recentes. Os exemplos usados no treinamento da RN representam o conceito normal.

No trabalho de Bartosz Krawczyk (2013), os autores usam uma técnica baseada em SVMs (Vapnik, 1998), nomeada WOCSVM (*Weighted One-Class Support Vector Machine*) (Bicego e Figueiredo, 2009), para representar a classe normal. Baseada no OCSVM (*One-Class Support Vector Machine*) (Schölkopf et al., 2001), uma adaptação das SVMs para a classificação com uma classe, WOCSVM adiciona um peso a cada exemplo do conjunto de treinamento. Os pesos são usados para minimizar o volume da hiperesfera e ainda envolver todos os dados de treinamento. Na fase de treinamento, o primeiro bloco de dados é usado para induzir o classificador WOCSVM. A função de minimização é apresentada nas Equações 3.1 e 3.2 e o peso associado a cada exemplo na Equação 3.3. Nessas equações, C representa o centro da hiperesfera, R, é o seu raio, w_i é o peso associado ao

i-ésimo exemplo, N é o número de exemplos, ξ é a variável *slack*, O é um parâmetro que controla o processo de otimização (maiores valores implicam em menos *outliers*) e δ é um parâmetro, que quando maior que 0 evita a divisão por 0 na Equação 3.3.

$$\Theta(C,R) = R^2 + O\sum_{i=1}^{N} w_i\xi_i \tag{3.1}$$

$$\forall_{1\leq i\leq N} : \|x_i - C\|^2 \leq R^2 + \xi_i \tag{3.2}$$

$$w_i = \frac{|x_i - x_{mean}|}{R + \delta} \tag{3.3}$$

Um comitê de *Half-Space-Trees* (HS-Tree)(Ting et al., 2009) é usado para representar a classe normal no trabalho proposto por Tan et al. (2011). HS-Trees são árvores binárias, construídas sem o uso dos exemplos de treinamento, mas usando as dimensões do espaço de dados. HS-Trees são compostas por um conjunto de nós, sendo que cada nó contém o número de exemplos (massa) que alcançaram esse nó no bloco de dados de referência (massa r) e no bloco de dados mais recente (massa l). Após construir as M primeiras árvores HS-Tree, o sistema usa o primeiro bloco de dados de exemplos para atualizar a massa r de cada árvore. Para isso, cada exemplo percorre cada uma das M árvores HS-Tree da raiz até um nó folha, atualizando a massa r no seu nó correspondente.

3.6 Fase *Online*

Na fase *online*, novos dados não rotulados chegam constantemente (em geral em alta velocidade). Nessa fase, três tarefas são executadas: classificação de novos exemplos, detecção de padrões-novidade, e atualização do modelo de decisão. A Figura 3.3 mostra uma visão geral das tarefas executadas nessa fase.

3.6.1 Classificação

A tarefa de classificação verifica se um novo exemplo pode ser explicado pelo modelo de decisão atual. A Figura 3.4 sumariza a taxonomia usada nessa fase.

Algumas propostas como (Albertini e de Mello, 2007), (Bartosz Krawczyk, 2013), (Tan et al., 2011), (Farid e Rahman, 2012), (Rusiecki, 2012) consideram todo novo exemplo como normal, se este for explicado pelo modelo de decisão atual, ou como novidade (anômalo ou anormal), se não explicado. Nessas propostas, uma novidade é detectada pela presença de um único novo exemplo não explicado pelo modelo de decisão atual. Outras propostas como (Faria et al., 2013b), (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Farid et al., 2013), (Masud et al., 2010a), (Masud et al., 2011a), (Al-Khateeb et al., 2012a) não classificam imediatamente todo novo exemplo, mas associam o perfil *desconhecido* àqueles

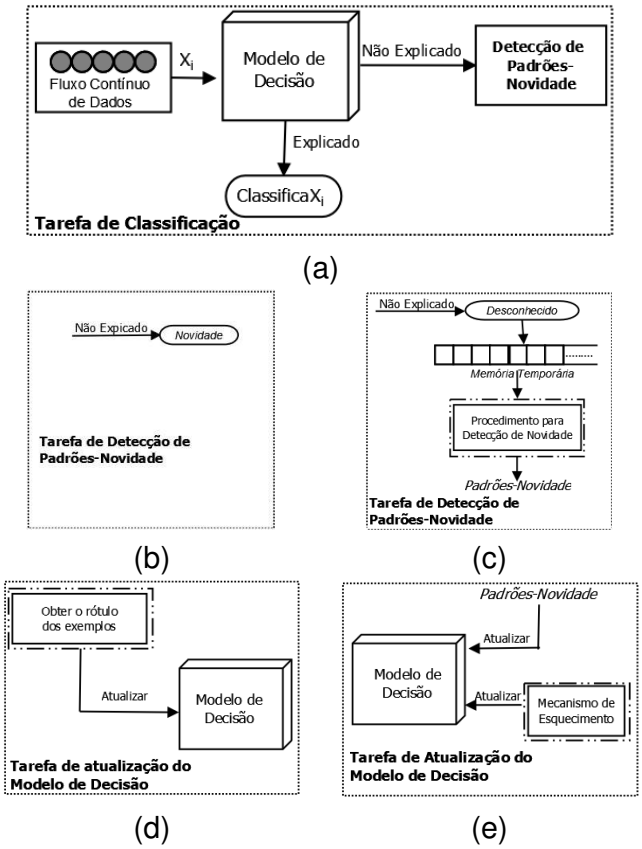


Figura 3.3: Tarefas executadas na fase *online* (a) Classificação (b) Detecção de padrões-novidade sem a opção de rótulo *desconhecido* (c) Detecção de padrões-novidade com a opção de rótulo *desconhecido* (d) Atualização do modelo de decisão com *feedback* (e) Atualização do modelo de decisão sem *feedback*.

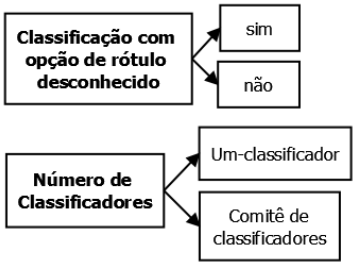


Figura 3.4: Taxonomia usada na tarefa de classificação.

exemplos não explicados pelo modelo de decisão atual. Esses exemplos são colocados em uma memória temporária para análise posterior e podem ser usados futuramente para modelar padrões-novidade. Nessas propostas, um padrão-novidade é composto por um conjunto coeso e representativo de exemplos não explicados pelo modelo atual. Em (Spinosa et al., 2009) e (Hayat e Hashemi, 2010), os exemplos marcados como *desconhecido* são também usados para modelar extensões dos conceitos conhecidos.

Adicionalmente, duas diferentes propostas têm sido usadas para classificar novos exemplos, nomeadamente as baseadas em um único classificador e as baseada em um comitê de classificadores.

Um único classificador

As técnicas propostas em (Spinosa et al., 2009) e (Hayat e Hashemi, 2010) usam um modelo de decisão representado por um conjunto de grupos e permitem a classificação com a opção de perfil *desconhecido*. O modelo de decisão é composto por 3 submodelos: normal, representa o conceito conhecido do problema, extensão, representa extensões do modelo normal, ou seja, conceitos próximo ao normal, e novidade, representa novos conceitos nos dados. Em (Spinosa et al., 2009), para a classificação de um novo exemplo, a distância entre o novo exemplo e um dos grupos do modelo de decisão é obtida. Se a distância é menor que o raio do grupo, então o exemplo é explicado pelo modelo de decisão. Caso contrário, ele é marcado com o perfil *desconhecido*. Nesse caso, como o modelo de decisão é composto pelos submodelos normal, extensão e novidade, cada um desses submodelos é verificado. O exemplo é rotulado com a classe associada ao submodelo do grupo no qual o exemplo está inserido, permitindo assim três possíveis classificações: normal, extensão ou novidade. Em (Hayat e Hashemi, 2010), o grupo mais próximo é escolhido para classificar um novo exemplo e um escore é computado para cada novo exemplo indicando o quanto ele é *desconhecido* . Esse escore representa a diferença entre a representação DCT do sub-grupo antes e depois da inclusão do novo exemplo. Se esse escore está acima de um limiar, o exemplo é rotulado como *desconhecido*.

Dois estudos, (Albertini e de Mello, 2007) e (Rusiecki, 2012), propõem o uso de um único classificador, baseado em RNs, sem a opção de rótulo desconhecido. Em (Albertini e de Mello, 2007), classificar um novo exemplo significa encontrar o neurônio que melhor representa esse exemplo. Para isso, a distância entre o novo exemplo e cada neurônio é calculada e a seguir usada na definição de valor de ativação do neurônio. O neurônio com o maior valor de ativação é escolhido para classificar o exemplo. Entretanto, esse neurônio deve apresentar um nível de similaridade mínimo. Caso contrário, ele não pode ser usado para classificar o novo exemplo, indicando a presença de uma novidade. Em (Rusiecki, 2012), um novo exemplo é submetido a duas RNs. Se a diferença entre os resultados produzidos por essas duas RNs é menor que um limiar, então o exemplo é classificado como pertencente ao conceito normal. Caso contrário, ele é rotulado como novidade.

No trabalho proposto por Bartosz Krawczyk (2013), um único classificador é também usado, sem a opção de rótulo *desconhecido*. Nesse trabalho, a classificação de um novo exemplo durante a fase *online* verifica se o mesmo está dentro da hiperesfera criada na fase *offline* usando um único classificador SVM. Se sim, o novo objeto é rotulado como pertencente ao conceito normal.

Em outro trabalho (Farid e Rahman, 2012), uma árvore de decisão é usada para classificar um novo exemplo, sem a opção de rótulo *desconhecido*. Quando um novo exemplo alcança um nó folha, a porcentagem de exemplos classificados por essa folha é atualizada. Se esse valor aumenta, isto pode ser um indicativo da chegada de um nova classe.

Comitê de classificadores

Nas propostas que usam um comitê de classificadores, cada classificador corresponde a um voto e a decisão final é feita baseada em uma regra, que combina a saída dos múltiplos classificadores (Menahem et al., 2013). Algumas dessas regras referenciadas na literatura são votação da maioria, voto médio, voto médio ponderado, regra max, etc. (Menahem et al., 2013), (Tax e Duin, 2001), (Juszczak e Duin, 2004).

Em Masud et al. (2011a), Masud et al. (2010a), Al-Khateeb et al. (2012a), Farid et al. (2013) um comitê é usado para classificar um novo exemplo usando a opção de rótulo *desconhecido*. Em Masud et al. (2011a), a primeira tarefa é verificar se o exemplo é *desconhecido*, usando para isso um comitê de árvores de decisão. Se o exemplo está fora de todos os grupos presentes no nó folha alcançado pelo exemplo, para todos os classificadores do comitê, então este exemplo é marcado como *desconhecido*. Caso contrário ele é classificado usando o voto da maioria dos classificadores do comitê. Os autores também propõem o uso de um comitê de classificadores KNN. Nessa variação, para classificar um novo exemplo, é necessário inicialmente verificar se ele deve ser marcado como *desconhecido* . Isso ocorre quando o exemplo está fora do raio do seu grupo mais próximo para todos os classificadores do comitê. Caso contrário, o rótulo do exemplo será a classe com mais alta frequência do grupo usado para classificá-lo. Entretanto, a decisão final é dada pela voto da maioria dos classificadores no comitê.

Uma proposta similar é usada em Masud et al. (2010a). Entretanto, um espaço *slack* é criado para cada grupo, sendo que um grupo é representado por uma hiperesfera (centro e raio). Se um exemplo está fora do raio da hiperesfera, mas dentro do espaço *slack*, ele é considerado pertencente a essa hiperesfera. O limiar usado para definir o espaço *slack* é determinado por um parâmetro do usuário.

O trabalho de Farid et al. (2013) também usa uma estratégia similar ao de Masud et al. (2011a), na qual um novo exemplo é classificado usando o voto da maioria ou como *desconhecido*, quando ele não pertence a nenhum dos grupos presentes no nó folha.

Outra técnica que usa um comitê de classificadores, nomeada CLAM (Al-Khateeb et al., 2012a), emprega um comitê de classificadores por classe, sendo que cada classificador é

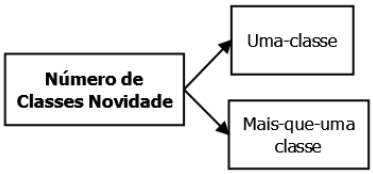


Figura 3.5: Taxonomia usada na detecção de padrões-novidade.

representado por um conjunto de grupos. Quando o CLAM recebe um novo exemplo, cada comitê verifica se o exemplo é *desconhecido*. Um exemplo é considerado *desconhecido* por um comitê se ele está fora do limite de decisão da maioria dos seus classificadores. Um novo exemplo é rotulado como *desconhecido* se todos os comitês consideram-no como *desconhecido*. Caso contrário, o exemplo é classificado em uma das classes conhecidas do problema. Para isso, para cada comitê, a distância entre o exemplo e os grupos que compõem cada classificador no comitê é computada. O exemplo é classificado na classe do comitê com a menor distância.

Um comitê de HS-Trees sem a opção de rótulo desconhecido é usada em Tan et al. (2011). Quando um novo exemplo precisa ser classificado, ele é associado a um nó folha em cada um das t HS-Trees, sendo que um escore é calculado para cada árvore, usando a massa r (uma informação calculada na fase de treinamento para cada nó). A soma dos escores de um exemplo, obtidas em cada árvore, define um escore de anomalia, que é usada para decidir se um novo exemplo é anômalo ou não.

3.6.2 Detecção de Padrões-Novidade

Em geral, esta tarefa usa exemplos não rotulados, não explicados pelo modelo de decisão atual, a fim de identificar padrões-novidade. Em sistemas de detecção de anomalia, a presença de somente um exemplo não explicado pelo modelo de decisão é suficiente para identificar um comportamento anômalo. Entretanto, várias propostas consideram que uma novidade é composta por um conjunto coeso e representativo de exemplo não explicados pelo modelo atual. A taxonomia usada nesta tarefa é apresentada na Figura 3.5.

A maioria das propostas da literatura (Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Farid e Rahman, 2012), (Farid et al., 2013), (Albertini e de Mello, 2007), (Rusiecki, 2012), (Tan et al., 2011), (Bartosz Krawczyk, 2013) consideram que o conceito novidade é composto por somente uma classe. Nesse caso, um exemplo ou um conjunto de exemplos não explicados pelo modelo de decisão são classificados como novidade. Diferentemente, Masud et al. (2010a) considera que diferentes classes novidade, compostas por um ou mais padrões-novidade, podem aparecer ao longo do tempo. Assim, os diferentes padrões-novidade devem ser diferenciados quando eles aparecem. Como resultado, não é suficiente classificar um novo exemplo somente como novidade, mas ele deve ser rotulado com o padrão-novidade ao qual ele pertence.

Diferentes propostas têm sido usadas para identificar padrões-novidade, a partir de exemplos não rotulados. Alguns deles usam algoritmos de agrupamento para encontrar grupos de exemplos similares a fim de identificar novidades. No OLINDDA (Spinosa et al., 2009), cada vez que um novo exemplo é marcado com o perfil *desconhecido*, o algoritmo K-Means é executado, produzindo k grupos. Cada grupo é analisado a fim de verificar se ele é valido. Caso contrário, o grupo é descartado. Um grupo é considerado válido se é coeso e representativo. A coesão avalia o nível de similaridade entre os exemplos do grupo. Representatividade estabelece um número mínimo de exemplos em um grupo válido. Limiares são usados para eliminar grupos não representativos (limiar definido pelo usuário) e não coesos (limiar baseado no modelo normal). Quando um grupo válido é identificado, é verificado se ele representa uma novidade ou uma extensão do modelo normal. Se o centróide do novo grupo válido está dentro da macro-hiperesfera, criada usando os grupos da classe normal, então o grupo é considerado uma extensão do conceito normal. Caso contrário ele é considerado uma novidade. Assim, a novidade é composta por um conjunto de grupos, mas não há distinção entre diferentes padrões-novidade.

DETECTNOD (Hayat e Hashemi, 2010) executa um algoritmo de agrupamento nos exemplos *desconhecidos*. A seguir, ele interpola os grupos e produz uma representação baseada em DCT (*Discrete Cosine Transform*). Para decidir entre uma extensão ou uma novidade, ele usa a distância entre o novo grupo e o seu sub-grupo mais próximo do modelo normal. Se a distância é menor que um limiar, o novo grupo é considerado uma extensão, caso contrário ele é considerado uma novidade. O conceito novidade também é composto por um conjunto de grupos, mas não há distinção entre diferentes padrões-novidade.

Depois de agrupar os exemplos na memória temporária, ECSMiner (Masud et al., 2011a), calcula uma medida interna de validação do agrupamento, nomeada q-NSC, para cada grupo em cada classificador do comitê (ver Equação 3.4). Essa medida é uma combinação entre coesão e separação e produz valores no intervalo [-1,+1]. Nessa Equação $\overline{D}_{c_{out},q(x)}$ é a distância média de um exemplo *desconhecido* x a $\lambda_{c_{out},q(x)}$, $\lambda_{c_{out},q(x)}$ é o conjuntos do q-vizinhos mais próximos de x considerando os exemplos *desconhecidos*, e $\overline{D}_{c_{min},q(x)}$ é o mínimo entre todos os $\overline{D}_{c,q(x)}$, sendo que c é uma classe existente. Um valor positivo indica que os exemplos desconhecidos são mais coesos e mais distantes das classes existentes. Se o número de grupos em um classificador com valor positivo q-NSC é maior que um limiar, então o classificador vota para a detecção de uma nova classe. Se todos os classificadores votam para a detecção de uma nova classe, uma novidade é identificada. Para cada bloco de dados, se mais que duas novas classes aparecem simultaneamente, elas são consideradas somente como uma novidade. CLAM (Al-Khateeb et al., 2012a) usa um procedimento similar para DN, baseado na medida q-NSC. CLAM, assim como o ECSMiner, não distingue entre diferente padrões-novidade no mesmo bloco de dados.

$$q - NSC(x) = \frac{\overline{D}_{c_{min},q(x)} - \overline{D}_{c_{out},q(x)}}{max(\overline{D}_{c_{min},q(x)}, \overline{D}_{c_{out},q(x)})} \tag{3.4}$$

Outra técnica que também usa a medida q-NSC é o MCM (Masud et al., 2010a). Para cada exemplo marcado como desconhecido, o MCM calcula seu Coeficiente Gini, cujo valor é usado para separar os exemplos que representam uma evolução de conceito de um ruído ou mudança de conceito. Se o Coeficiente Gini é maior que um limiar, os exemplos representam uma evolução do conceito. A seguir, o q-NSC é calculado para cada exemplo (ver Equação 3.4). Se o q-NSC para um exemplo desconhecido é negativo, o exemplo é removido. A seguir, a medida Nscore (ver Equação 3.5) é calculada para cada exemplo que tem um valor positivo para q-NSC. Nessa Equação, R é o raio do grupo mais próximo ao exemplo, d é a distância entre um exemplo e o centro do grupo mais próximo e $minweight$ é o peso mínimo entre todos os exemplos *desconhecidos* com valor positivo para q-NSC. Um valor alto para Nscore indica que há uma alta probabilidade do exemplo ser considerado uma novidade. Ao fim, o Nscore(x) é discretizado em n intervalos, uma função de distribuição cumulativa (CDF) é calculada, e o Coeficiente Gini discreto é calculado. Se o valor do Coeficiente Gini é maior que $\frac{n-1}{3n}$, uma nova classe é detectada. MCM distingue entre diferentes classes novidade computando os componentes conectados de um grafo, construído a partir dos grupos dos exemplos marcados como novidade.

$$weight(x) = e^{R-d} \tag{3.5}$$

$$Nscore(x) = \frac{1 - weight(x)}{1 - minweight} q - NSC(x) \tag{3.6}$$

Sistemas de detecção de anomalia são propostos em (Albertini e de Mello, 2007), (Rusiecki, 2012), (Farid e Rahman, 2012), (Farid et al., 2013) (Tan et al., 2011), (Bartosz Krawczyk, 2013). Nesses sistemas, a presença de um único exemplo *desconhecido* indica um comportamento anômalo. Apesar da maioria desses trabalhos usarem o termo detecção de novidade para definir esse comportamento anômalo, uma definição mais adequada para essa tarefa é detecção de anomalia. A motivação para isso é que, de acordo com a definições dadas neste trabalho, uma novidade corresponde a um grupo coeso e representativo de exemplos, não explicados pelo modelo de decisão.

Em (Albertini e de Mello, 2007), quando nenhum neurônio é capaz de classificar um novo exemplo, um neurônio é criado, indicando que uma novidade foi detectada. O centroide do novo neurônio corresponde ao vetor de atributos do novo exemplo, o nível de similaridade mínima recebe um valor predefinido α, e o raio é definido com o valor $-\ln(\alpha)$.

Em (Rusiecki, 2012), quando a diferença entre os resultados produzidos pelas duas RNs para um novo exemplo é maior que um dado limiar T, uma novidade é detectada. A fim de encontrar o melhor valor para esse limiar, uma proposta baseada no desvio padrão

das diferenças entre as saídas das RNs é empregado, veja Equação 3.7. Nessa Equação, $y_{mse}(x_i)$ é a saída da RN tradicional e $y_{mls}(x_i)$ é a saída da RN robusta para o i-ésimo exemplo, sendo k uma constante definida pelo usuário.

$$T = k * Std(|y_{mse}(x_i) - y_{mls}(x_i)|) \tag{3.7}$$

Os trabalhos propostos por Farid e Rahman (2012), Farid et al. (2013) e Tan et al. (2011) usam uma árvore de decisão para detectar novidades. No trabalho de Farid e Rahman (2012), o algoritmo proposto para DN verifica se adicionar um novo exemplo a um nó folha aumenta a porcentagem p (proporção dos exemplos no nó folha considerando o número de exemplos no conjunto de treinamento) já calculada. Um aumento indica a presença de uma nova classe. Se o novo exemplo está fora de todos os grupos dessa folha, ele pertence a uma classe novidade que acabou de chegar. No trabalho de Farid et al. (2013), se um novo exemplo não pertence a nenhum dos grupos na sua folha correspondente, ele é colocado na memória temporária e o valor 1 é atribuído ao *flag* novidade. Se o número de exemplos classificados por um nó folha aumenta ou decrementa (em comparação com o valor previamente calculado) e o *flag* novidade tem valor 1, o exemplo pertence a uma classe novidade.

Em Tan et al. (2011), o escore acumulado é calculado para cada novo exemplo (descrito na Seção 3.6.1). Se esse valor é maior que um limiar, o exemplo é considerado anômalo. Em Bartosz Krawczyk (2013), se um novo exemplo é rotulado como não pertencente ao modelo normal, ele é considerado um exemplo anômalo.

3.6.3 Atualização do Modelo de Decisão

Algoritmos para DN em FCD devem atualizar seu modelo de decisão continuamente, a fim de tratar mudança de conceito e evolução do conceito. A Figura 3.6 mostra três aspectos a ser considerados nessa fase:

- Tipo de atualização, o qual pode ser executado com ou sem *feedback* externo;
- Número de classificadores;
- Uso de mecanismos de esquecimento a fim de remover dados antigos.

O uso de um único classificador (Spinosa et al., 2009), (Albertini e de Mello, 2007), (Hayat e Hashemi, 2010), (Farid e Rahman, 2012), (Rusiecki, 2012), (Karnick et al., 2008) em oposição ao uso de um comitê de classificadores (Farid e Rahman, 2012), (Masud et al., 2010a), (Masud et al., 2011a), (Al-Khateeb et al., 2012a) afeta a atualização do modelo de decisão. Enquanto algoritmos com um único classificador usualmente são induzidos por algoritmos incrementais, em comitês, sempre que um classificador é treinado, um classificador antigo é usualmente removido.

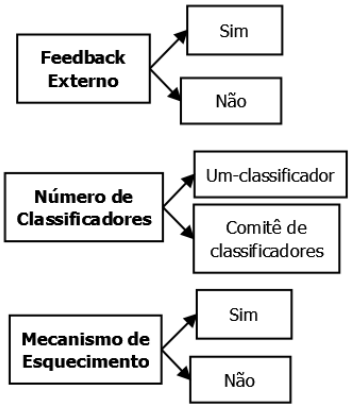


Figura 3.6: Taxonomia usada na tarefa de atualização do modelo de decisão.

O uso de *feedback* externo na atualização do modelo de decisão atual é também um importante aspecto a ser considerado. Técnicas que usam *feeback* assumem que o rótulo real de todos os exemplos estará disponível após um período. Além disso, há técnicas que perguntam ao usuário o rótulo real de um subconjunto de exemplos, os exemplos importantes do FCD (aprendizado ativo). Por outro lado, nas técnicas que não usam *feedback*, o modelo de decisão é atualizado sem informação do rótulo real dos exemplos. É importante observar que obter os rótulos reais é uma tarefa cara e que consome tempo. Em muitos problemas reais pode ser impraticável.

1. **Número de classificadores e *feedback* externo:**

   Um classificador sem *feedback* externo:
   Em alguns algoritmos, como (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Albertini e de Mello, 2007) o modelo de decisão é composto por um único classificador, o qual é atualizado incrementalmente. Esses algoritmos não usam *feedack* externo para atualizar o modelo de decisão.

   Em (Spinosa et al., 2009), sempre que um novo grupo válido é obtido usando os exemplos *desconhecidos*, ele é incluído no modelo de decisão. Há um modelo de decisão para representar o conceito normal, um para o conceito extensão e um para o conceito novidade. Em (Hayat e Hashemi, 2010), o modelo normal é atualizado com os grupos que representam uma mudança de conceito. A estratégia usada é substituir um grupo antigo por um novo. Em (Albertini e de Mello, 2007), quando um novo exemplo é classificado por um neurônio, este neurônio representa o exemplo, e portanto tem o seu raio e centroide atualizados. Quando nenhum dos neurônios existentes pode explicar um novo exemplo, um novo neurônio é criado.

   Um classificador com *feedback* externo:
   Os trabalhos propostos em (Farid e Rahman, 2012), (Bartosz Krawczyk, 2013), (Rusiecki, 2012) também usam um único classificador, entretanto, eles são atualizados usando *feedback* externo. Em (Farid e Rahman, 2012), um novo exemplo é conside-

rado novidade quando nenhum dos grupos presentes no nó folha, alcançado por um exemplo, consegue explicá-lo. Nesse caso, o exemplo é adicionado ao conjunto de exemplos de treinamento e o modelo de decisão é atualizado. Em (Bartosz Krawczyk, 2013), os autores propõem um único classificador baseado em SVMs. Uma técnica de aprendizado incremental passiva é usada para atualizar o modelo de decisão. Essa técnica adiciona cada novo exemplo ao conjunto de treinamento com um peso associado. Duas abordagens são propostas para calcular o peso de cada exemplo. A primeira usa a Equação 3.3, a segunda associa os maiores valores de peso aos exemplos mais recentes. Como resultado, todos os exemplos no bloco de dados mais recente tem peso igual a 1. Em (Rusiecki, 2012), os autores argumentam que atualizar a RN toda vez que um novo exemplo chega é computacionalmente caro. Assim, eles argumentam que uma melhor proposta é treinar a RN somente uma vez para um dado período de tempo. Exemplos de treinamento são acumulados até que um dado tamanho é alcançado, determinado por um parâmetro nomeado tamanho da janela. A seguir, os pesos da RN são atualizados usando esses exemplos. Outro parâmetro define a distância entre duas janelas de tempo, uma vez que nem todos os exemplos são usados para o treinamento.

Comitê de classificadores com *feedback* externo:
Outros algoritmos que usam um comitê de classificadores são (Masud et al., 2010a), (Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Farid et al., 2013), (Tan et al., 2011). A maioria deles assume que, com um certo atraso, o rótulo verdadeiro de todos os exemplos estará disponível. Assim, de tempos em tempos, eles atualizam de forma *offline* o modelo de decisão. Esses algoritmos são capazes de tratar mudança de conceito e incorporar novas classes ao modelo de decisão usando a informação de *feedback*. Entretanto, se o rótulo verdadeiro dos exemplos não está disponível ou se está disponível após um longo atraso, esses algoritmos tem seu desempenho decrementado.

Em Masud et al. (2010a),Masud et al. (2011a), a fim de tratar o tema mudança de conceito, o comitê é continuamente atualizado, substituindo no comitê o modelo com mais alto erro de classificação por um novo modelo. Em (Farid et al., 2013), cada árvore de decisão do comitê tem um peso associado. Valores pequenos para o peso são associados com classificadores com baixa acurácia, os quais podem ser substituídos por um novo classificador. O comitê de classificadores é atualizado, se uma nova árvore de decisão tem um peso maior que todas as outras árvores de decisão do comitê. Em Al-Khateeb et al. (2012a), sempre que um novo classificador é treinado, ele substitui um dos classificadores do seu correspondente comitê. Os classificadores com mais alto erro preditivo, considerando o subconjunto correspondente no último bloco de dados de treinamento, são removidos.

Por último, algumas propostas baseadas em comitê de classificadores assumem que os exemplos rotulados são limitados, uma vez que rotular todos os exemplos é uma tarefa que consome muito tempo. Em Masud et al. (2010b), depois de classificar cada exemplo de um bloco de dados em uma das classes conhecidas ou como novidade, um conjunto de exemplos fracamente classificados é selecionado para ser rotulado por um especialista. Esses exemplos são colocados em uma memória de exemplos de treinamento. Quando há um número suficiente de exemplos nessa memória, um nova fase de aprendizado supervisionada atualiza o modelo de decisão atual.

Comitê de classificadores sem *feedback* externo:
No comitê de classificadores usado em Tan et al. (2011), um classificador pode ser atualizado, mas não há substituição dos classificadores no comitê. Além disso, a estrutura da árvore não é modificada durante a atualização, sendo que somente os atributos massa l e r são atualizados. Assim, o *feedback* externo não é usado para atualizar o comitê de classificadores.

2. **Mecanismo de Esquecimento:**
Além de atualizar o modelo de decisão, alguns trabalhos usam um mecanismo para esquecimento dos dados desatualizados. As propostas baseadas em um comitê identificam e removem esses dados toda vez que um novo classificador é treinado, substituindo um modelo antigo. Nas propostas que usam um único classificador, um mecanismo específico para tal é incluído.

Em Albertini e de Mello (2007), como os conceitos conhecidos podem mudar, os neurônios adaptam-se, esquecendo informação do passado. Quando um neurônio é atualizado porque ele foi usado para classificar um novo exemplo, seu centróide e raio são atualizados seguindo a proposta EWMA (*Exponential Weighted Moving Averages*), a qual usa parâmetros que definem a influência de exemplos antigos no comportamento corrente do FCD.

Em Bartosz Krawczyk (2013), um esquecimento incremental reduz os pesos dos exemplos do bloco de dados anterior. Após ler alguns blocos de dados, os exemplos antigos tem peso igual a 0 e são removidos do conjunto de treinamento, assumindo que eles não são importantes para o FCD atual. Duas abordagens foram propostas para calcular o peso dos exemplos do bloco de dados anterior. A primeira delas decresce o peso dos exemplos de acordo com sua importância inicial, isto é, objetos como pesos iniciais diferentes serão removidos em tempos diferentes. A segunda usa um decaimento alinhado que não considera a importância dos pesos iniciais, isto é, todos os exemplos de um mesmo bloco de dados serão removidos no mesmo marcador de tempo.

3.7 Outros Aspectos Relevantes dos Algoritmos de DN

Esta seção apresenta alguns aspectos que são importantes para a tarefa de DN, mas que foram tratados por poucos trabalhos da literatura. O primeiro deles é a detecção de contextos recorrentes, que pode levar a altas taxas de falsos positivos, quando uma classe recorrente é confundida com o aparecimento de uma nova classe. A segunda é o tratamento de ruídos e *outliers*, que podem ser confundidos com mudança de conceito ou evolução de conceito. Por último, as medidas de avaliação são um importante aspecto a ser tratado, uma vez que as medidas clássicas de classificação podem não ser interessantes para DN em cenários envolvendo FCDs.

3.7.1 Detecção de Contextos Recorrentes

Contextos recorrentes são um importante fenômeno observado em muitas aplicações do mundo real, tais como clima, demanda de eletricidade, mudanças no tempo, hábitos dos compradores e novos tópicos em textos. Em tais aplicações, as definição das classes podem mudar, sendo que um cenário anterior pode recorrer de forma periódica ou aleatória, depois de um certo período de tempo (Elwell e Polikar, 2011). De acordo com Katakis et al. (2010), contextos recorrentes são um tipo especial de mudança de conceito no qual o que apareceu no passado pode ocorrer novamente no futuro.

Em cenários nos quais os conceitos reaparecem, é uma perda de esforço reaprender um conceito antigo para cada recorrência (Widmer e Kubat, 1996). Mesmo em cenários de detecção de anomalia, o tratamento de contextos recorrentes é um importante desafio a ser tratado (Srivastava, 2006). Em contextos recorrentes, ao invés de se esquecer os conceitos antigos, eles devem ser salvos e reexaminados depois de algum tempo a fim de melhorar o desempenho preditivo e a relação custo-benefício do classificador.

Para Masud et al. (2011b) e Al-Khateeb et al. (2012a), uma classe recorrente é um caso especial de evolução do conceito no qual uma classe aparece no fluxo, desaparece por um período de tempo, e aparece novamente. Um erro comum cometido pelos sistemas é não abordar classes recorrentes e tratá-las como se fossem novidades. De acordo com Katakis et al. (2010), essa estratégia cria efeitos indesejáveis como o aumento na taxa de falsos alarmes e em esforços humanos para analisar esses falso alarmes, esforços computacionais extras para executar a tarefa de DN e em aprender uma nova classe que já foi previamente aprendida.

Apesar do fenômeno de reaparecimento de conceitos ser muito comum em problemas do mundo real, somente poucas metodologias de classificação baseadas em fluxo levam em consideração esse fenômeno (Katakis et al., 2010). No contexto de DN em FCDs, esse fenômeno foi tratado nos seguintes trabalhos (Masud et al., 2011b), (Al-Khateeb et al., 2012a), (Faria et al., 2013b), (Katakis et al., 2010).

Em Masud et al. (2011b), um algoritmo baseado em blocos de dados é proposto para classificar novos exemplos em ambientes com classes recorrentes. Um comitê de M classificadores é usado para classificar novos exemplos, mas também um comitê auxiliar de tamanho M^A é mantido a fim de detectar classes recorrentes. Quando um novo exemplo chega, se ele não é considerado como *desconhecido* pelo comitê primário, ele é classificado usando esse comitê, em uma das classe conhecidas. Se o exemplo é considerado *desconhecido*, isto é, não explicado pelo comitê primário, o comitê auxiliar é consultado. Se o comitê auxiliar consegue explicar o exemplo, ele é classificado de acordo com esse comitê em uma classe recorrente. Caso contrário, ele é adicionado a uma memória temporária e mais tarde submetido à tarefa de DN.

De acordo com Al-Khateeb et al. (2012a), técnicas baseadas em blocos de dados para DN não conseguem detectar classes recorrentes. Isto acontece porque se uma classe desaparece por um período de tempo, o comitê descarta todos os modelos treinados com essa nova classe. Assim, quando exemplos dessa classe reaparecem, nenhum dos modelos no comitê consegue classificá-los, e provavelmente eles serão identificados como padrões-novidade. A fim de tratar esse problema, em Al-Khateeb et al. (2012a), os autores propõem uma nova técnica, nomeada comitê baseado em classes, a fim de classificar exemplos em cenários que envolvem classes recorrentes e superar as desvantagens das propostas baseadas em blocos de dados. Assim, os autores propõem criar c comitês, um por classe, e cada comitê é composto por M micro-classificadores (modelos). Assim, se c é o número de classes vistas até o momento no FCDs, então há c comitês. Sempre que um novo modelo é treinado, ele substitui um modelo antigo. Entretanto, mesmo se uma classe desaparece por um período de tempo no fluxo, seu comitê correspondente não é deletado. Nesse caso, a classe é sempre considerada ativa, isto é, ela nunca desaparece. Assim, a proposta de Al-Khateeb et al. (2012a) identifica classe recorrentes como classes já existentes.

Uma aplicação de DN para filtrar emails usando comitê de classificadores, capaz de lidar com contextos recorrentes, é encontrada em Katakis et al. (2010). O FCD é dividido em blocos de exemplos. Cada bloco é usado por uma função de transformação para produzir um novo modelo de representação conceitual, nomeado vetor conceitual. Vetores conceituais descrevem os conceitos presentes em um bloco particular de exemplos. Se dois blocos são conceitualmente similares, então seus vetores conceituais estão próximos. Um algoritmo de agrupamento é usado para agrupar os blocos de exemplos, representados por vetores conceituais, em conceitos, o que permite a identificação de contextos recorrentes. Para cada conceito detectado pelo algoritmo ao longo do FCD, um classificador incremental é criado e adicionado ao comitê. Para cada grupo, há um classificador associado. Assim, novos exemplos são sempre classificados usando o classificador que corresponde ao grupo do vetor conceitual anterior.

3.7.2 Tratamento de Outliers

Outliers são dados isolados, esparsos e não existem em um número representativo (Spinosa et al., 2009). Propostas para detecção de *outliers* procuram por exemplos que não seguem o o comportamento normal e que podem representar um padrão indesejado. Por outro lado, algoritmos de DN buscam um conjunto de exemplos coesos e representativos que descrevem um novo conceito emergente. Técnicas de DN devem abordar uma importante questão, o tratamento de ruídos e *outliers*, os quais podem ser confundidos com o aparecimento de um novo conceito ou uma mudança nos conceitos conhecidos. De acordo com Rusiecki (2012), detecção de *outliers* é mais complicado em FCDs , uma vez que eles podem ser um ruído, que precisa ser removido dos dados, ou produzidos por uma mudança de conceito, representando importantes alterações no comportamento do sistema. Alguns algoritmos tratam essa questão desenvolvendo técnicas que conseguem identificar ruídos e *outliers*.

Uma proposta usada por técnicas como OLINDDA (Spinosa et al., 2009), ECSMiner (Masud et al., 2011a) e MCM (Masud et al., 2010a) é armazenar os exemplos que não são explicados pelo modelo de decisão atual em uma memória temporária e rotulá-los como *desconhecidos*. Os exemplos *desconhecidos* são futuramente analisados a fim de decidir se eles representam um ruído ou *outliers* ou um padrão-novidade.

No OLINDDA (Spinosa et al., 2009), um algoritmo de agrupamento é aplicado nos exemplos *desconhecidos*. Os grupos obtidos são avaliados para decidir se eles representam um padrão-novidade. Cada grupo tem sua representatividade e coesão avaliados por um critério de validação. Grupos com baixa avaliação, nomeados grupos inválidos, são removidos. Entretanto seus exemplos permanecem na memória temporária, que tem uma capacidade limitada. Se um novo exemplo deve ser armazenado mas não há espaço disponível, o exemplo mais antigo é removido. Como os exemplos removidos não foram usados recentemente, há uma grande chance de que esses exemplos sejam ruídos ou *outliers*.

ECSminer (Masud et al., 2011a), MCM (Masud et al., 2010a) e CLAM (Al-Khateeb et al., 2012a) também aplicam um algoritmo de agrupamento nos exemplos *desconhecidos* e usam os grupos obtidos para identificar padrões-novidade. Para cada grupo, a medida q-NSC, definida como a combinação entre coesão e separação, é calculada. Se o número de grupos como um valor positivo para q-NSC é maior que um limiar, seu classificador associado vota para a detecção de uma nova classe. Se todos os classificadores votam pela detecção de uma nova classe, uma novidade é identificada. Caso contrário, aqueles exemplos provavelmente representam ruídos, *outliers* ou mudanças nos conceitos conhecidos.

3.7.3 Avaliação em DN

Apesar de muitos algoritmos terem sido propostos para lidar com DN em cenários envolvendo FCDs, pouca atenção foi dedicada à avaliação da performance preditiva dos mesmos.

Metodologia Experimental

Em cenários *batch*, o procedimento de avaliação usa metodologias bem conhecidas e discutidas por décadas. Alguns exemplos dessas metologias incluem propostas como validação cruzada, *leave-one-out*, *bootstrap*, entre outros. Em FCDs, as metodologias de avaliação, bem como as medidas de avaliação são importantes aspectos a serem tratados. Estratégias de amostragem, como por exemplo, validação cruzada, não são aplicáveis a DN devido às restrições requeridas nesse cenário, tais como, dados potencialmente infinitos e distribuição de dados não-estacionária (Gama et al., 2013).

Para a classificação em FCDs, em geral, duas propostas têm sido usadas, nomeadamente *holdout* e *prequential* (sequencial preditivo, do inglês *predictive sequential*)(Dawid, 1984). No *holdout*, os dados são divididos em treinamento e teste. O modelo de decisão atual é aplicado ao conjunto de treinamento, em intervalos regulares de tempo. No *prequential*, para cada exemplo do FCD, o modelo faz uma predição. O erro é computado pela soma acumulada da função perda entre a predição fornecida pelo classificador e a classe verdadeira do exemplo.

Em geral, os trabalhos da literatura para DN em FCDs não discutem sobre a metodologia experimental usada. Alguns deles usam metodologias de cenários *batch* enquanto que outros usam metodologias novas, mas não justificam seu uso.

No OLINDDA (Spinosa et al., 2009), os autores propõem usar validação cruzada de 10 partições. O conjunto de treinamento é composto por exemplos de somente uma classe, a classe normal. O conjunto de teste é composto por exemplos das classes novidade mais o restante dos exemplos da classe normal. DETECTNOD (Hayat e Hashemi, 2010) usa uma estratégia similar ao OLINDDA.

No ECSMiner (Masud et al., 2011a), MCM (Masud et al., 2010a) e CLAM (Al-Khateeb et al., 2012a), na fase *offline*, um conjunto de treinamento, composto pelas primeiras $init$ janelas de dados, é usado, sendo que o tamanho de cada janela ($windowsize$) é um parâmetro definido pelo usuário. Os elementos restantes são usados na fase *online*. Nesses trabalhos, todos os exemplos de teste são usados no cálculo das medidas de avaliação.

Medidas de avaliação

Alguns trabalhos para DN em FCDs usam medidas de classificação clássicas para avaliar o desempenho preditivo dos classificadores investigados. Em Tan et al. (2011), por exemplo, a medida AUC (*Area Under the Curve*) é usada. As instâncias do conjunto de teste são

ordenadas de acordo com o seu escore de anomalia. Usando esse escore e o rótulo real, a medida AUC é calculada. Em Bartosz Krawczyk (2013), a acurácia é usada. Entretanto, os autores não fornecem mais informações de como essa medida é calculada em contextos para DN.

As medidas de avaliação supervisionada precisão, revocação e f-measure são usadas em Albertini e de Mello (2007). No contexto de DN, essas medidas são calculadas de acordo com as Equações 3.8, 3.9, and 3.10. Nessa Equações, $TNov$ é o número de novidades detectadas pelo algoritmo, que realmente são novidades, $DNov$ é o número de exemplos detectados como novidade pelo classificador, Nc é o número total de exemplos pertencentes a classes novidade no FCD.

$$Precisao = \frac{TNov}{DNov} \tag{3.8}$$

$$Revocacao = \frac{TNov}{Nc} \tag{3.9}$$

$$FMeasure = 2 * \frac{precisao \times revocacao}{precisao + revocacao} \tag{3.10}$$

Os experimentos realizados em Spinosa et al. (2009), Hayat e Hashemi (2010), Masud et al. (2010a), Masud et al. (2011a), Farid et al. (2013), Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b), Faria et al. (2013b) usam medidas de avaliação desenvolvidas para a tarefa de DN. Entretanto, essas medidas são mais adequadas para cenários estáticos e contextos nos quais a DN é considerada uma tarefa de classificação com uma classe. Além disso, essas medidas não conseguem avaliar classificadores com a opção de rótulo *desconhecido*.

Outros trabalhos, Spinosa et al. (2009) e Hayat e Hashemi (2010), tratam DN com uma tarefa de classificação com uma classe e portanto usam medidas de avaliação binárias. Medidas como a porcentagem de exemplos das novas classes incorretamente classificados como classe normal (*Mnew*) e a porcentagem de exemplos das classes normais incorretamente classificados como novidade ou extensão (*Fnew*), veja Equações 3.11 e 3.12, são usados. Nessas Equações, FP é o número total de elementos do conceito normal incorretamente classificados como novidade, extensão ou desconhecido, FN é o número total de exemplos das classes novidade classificados como classe normal, Nc é o número total de exemplos das classes novidade e N é o número total de exemplos no FCD. Outra medida comumente usada é a porcentagem de classificações incorretas, veja Equação 3.13. Essa medida inclui além dos erros FP e FN, o FE que representa o total de exemplos das classes existentes incorretamente classificados (além de FP).

$$Mnew = \frac{FN * 100}{Nc} \tag{3.11}$$

$$Fnew = \frac{FP * 100}{N - Nc} \tag{3.12}$$

$$Err = \frac{(FP + FN + FE) * 100}{N} \tag{3.13}$$

Em Masud et al. (2010a), Masud et al. (2011a), Farid et al. (2013), Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b), apesar da DN ser tratada como uma tarefa de classificação multiclasse, medidas de avaliação binária são usadas. Medidas de classificação binária não são capazes de avaliar adequadamente cenários nos quais o conceito novidade é multiclasse. Nesse cenário, não é suficiente classificar um exemplo de uma classe novidade apenas como novidade. Um erro deve ser computado quando exemplos de diferentes classes novidade são classificados no mesmo padrão-novidade.

Os experimentos realizados por Rusiecki (2012) usam somente conjuntos de dados artificiais 2D, no qual o classificador é avaliado sem o uso de medidas de avaliação. Ao contrário, os autores propõem o uso de um gráfico 2D com duas curvas, uma representando o sinal, isto é, o conjunto de dados 2D, e outra representando os resultados da tarefa de DN, isto é, os marcadores de tempo nos quais uma novidade foi detectada.

Como outra limitação, a maioria dos algoritmos para DN encontrados na literatura inicialmente classificam os exemplos não explicados pelo modelo de decisão atual como *desconhecido*. A seguir, esses exemplos podem ser rotulados como novidade. Assim, uma importante questão a ser tratada é como incorporar esses exemplos marcados com o perfil *desconhecido* nas medidas de avaliação. Uma situação similar é vista nos classificadores com opção de rejeição (Pillai et al., 2011), (Nadeem et al., 2010), (Marrocco et al., 2007) e (Tax e Duin, 2008). Nesses classificadores, um exemplo é rejeitado se sua classe não pode ser predita com confiança (Nadeem et al., 2010). É considerado melhor rejeitar o exemplo do que classificá-lo incorretamente. Para essas situações, as taxas de acurácia e erro podem ser calculadas considerando todos os exemplos ou somente os exemplos aceitos pelo classificador.

Uma proposta comumente usada para detectar padrões-novidade é agrupar exemplos *desconhecidos* similares usando um algoritmo de agrupamento. Um grupo ou um conjunto de grupos podem compor um padrão-novidade, o qual é usado futuramente para classificar novos exemplos. Entretanto, um importante problema a ser tratado é como associar os padrões-novidade detectados pelo algoritmo de aprendizado às classes do problema, a fim de que medidas de avaliação sejam computadas.

Mais um ponto a ser destacado é que à medida que novos padrões-novidade são detectados, o número de colunas da matriz de confusão é incrementado. Além disso, uma nova classe pode ser representada por um ou mais padrões novidade, o que pode levar a uma matriz de confusão não quadrada. O Capítulo 5 irá apresentar os problemas relacionados com as metodologias de avaliação existentes, bem como uma nova metodologia de

avaliação proposta neste trabalho.

3.8 Aplicações

Recentes avanços em hardware e software tem permitido a aquisição de dados em uma grande variedade de aplicações (Gaber et al., 2005). Em geral, esses dados são gerados continuamente criando grandes bases de dados que precisam ser mineradas a fim de extrair informação útil. Modelos induzidos por técnicas de mineração de dados têm sido aplicados a esses dados para extrair informação nova e útil. Além disso, o desenvolvimento de diferentes e poderosos sensores permitiu o monitoramento de muitos eventos em tempo real (Aggarwal, 2006). Uma vez que esses conjuntos de dados apresentam um comportamento dinâmico, é necessário construir modelos que possam não só representar esses dados, mas que também possam evoluir ao longo do tempo. Em geral, esses modelos aprendem a partir do cenário conhecido sobre um problema particular, e eles precisam reagir a mudanças, esquecer conceitos antigos e aprender novos. Dentre a grande variedade de aplicações a respeito de DN em FCDs, é possível mencionar detecção de intrusão/anomalias em redes, detecção de fraude em cartão de crédito, detecção de falhas em máquinas, segurança usando vídeos, etc. A tabela 3.3 mostra alguns conjuntos de dados usado na tarefa de DN em FCDs.

As técnicas propostas na literatura para DN em FCDs são frequentemente comparadas usando dados de aplicações reais. Uma das aplicações mais comum usadas nessas comparações é a base de dados de detecção de intrusos em uma rede de computadores.

Diversos trabalhos usam essa base de dados (Spinosa et al., 2008),(Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Spinosa et al., 2009),(Tan et al., 2011), (Farid e Rahman, 2012), (Farid et al., 2013), (Wang et al., 2008), (Rios et al., 2011) e (Perdisci et al., 2006). A detecção de intrusão têm sido empregada para induzir classificadores capazes de identificar atividades não autorizadas em uma rede de computadores, separando o acesso normal de ataque. Nessa aplicação, na fase de treinamento, um classificador é induzido usando exemplos do acesso normal (e em alguns casos um conjunto de ataques conhecidos). Na aplicação, fase online, o classificador deve ser capaz de reconhecer novos tipos de ataque, que podem aparecer a qualquer momento.

A maioria dos experimentos reportados na literatura para essa aplicação usa somente atributos numéricos. Entretanto, o trabalho de Shyu et al. (2005) apresenta uma proposta para lidar também com os atributos nominais da base. A base de dados pública mais popular para detecção de intrusão é o KDD Cup 99, disponível no UCI (Frank e Asuncion, 2010). Ela consiste na simulação de diferentes tipo de ataque em uma ambiente de rede militar. Ela é usada em (Spinosa et al., 2008), (Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Spinosa et al., 2009), (Tan et al., 2011), (Shyu et al., 2005). Um subconjunto dessa base de dados, nomeado NLS-KDD é usado em (Farid e Rahman, 2012), (Farid et

Tabela 3.3: Bases de dados usadas para DN em FCDs.

Bases de dados	Referências
KDD Cup 99	(Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Spinosa et al., 2009), (Shyu et al., 2005), (Wang et al., 2008), (Tan et al., 2011), (Farid e Rahman, 2012), (Farid et al., 2013)
Cobertura de Floresta	(Masud et al., 2011a) , (Al-Khateeb et al., 2012a), (Masud et al., 2010a), (Tan et al., 2011)
Análise de Vídeo	(Singh e Markou, 2005), (Ramezani et al., 2008), (Tavakkoli et al., 2006)
Twitter	(Masud et al., 2010a)
Mineração de Texto	(Yang et al., 2002)

al., 2013). O NLS-KDD foi proposto por Tavallaee et al. (2009) para superar os problemas encontrados no KDD Cup 99.

Outra base de dados clássica usada para avaliar os algoritmos de DN, disponível no repositório UCI, é o tipo de cobertura de floresta. Essa base de dados foi obtida de um problema real para detectar novos tipos de floresta usando um conjunto de atributos geo-espaciais. Em geral, um classificador é induzido a partir de um conjunto de diferentes tipos de cobertura de floresta. O modelo induzido pode ser usada para identificar novos tipos de cobertura de floresta. São exemplos de trabalhos que investigaram o desempenho de algoritmos para DN nessa base de dados (Masud et al., 2010a), (Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Tan et al., 2011).

Algoritmos para DN também têm sido aplicados para identificar novidades em imagens de vídeos (Singh e Markou, 2005), (Ramezani et al., 2008), (Tavakkoli et al., 2006) e análise de imagem (Farid et al., 2013). Alguns desses trabalhos, como por exemplo (Ramezani et al., 2008) focam em segurança baseada em vídeos, que é uma importante aplicação para tarefas como segurança humana, controle de terrorismo e controle de tráfego. Outro trabalho, desenvolvido por Gaughan e Smeaton (2005) objetiva usar *closed captions* para organizar resultados de busca de noticiários baseado no nível de novidade do tópico.

Mineração de texto em FCDs é também uma importante aplicação para DN. Em (Yang et al., 2002), os autores propõem um novo sistema capaz de classificar documentos *online* usando aprendizado supervisionado em um conjunto predefinido de tópicos. Em (Masud et al., 2010a), os autores propõem o sistema MCM para DN em FCDs. Uma aplicação usada para avaliar o desempenho do classificador é um fluxo de texto do Twitter, no qual o classificador é treinado com um número predefinido de classes (tópicos) e ao longo do FCD, novos tópicos aparecem e o sistema precisa ser capaz de detectar esses novos tópicos e incorporá-los ao modelo de decisão.

3.9 Desafios e Trabalhos Futuros

Ainda que diferentes algoritmos para DN tenham sido propostos e investigados para diferentes aplicações, muitas questões ainda precisam ser tratadas. Primeiramente, apesar da metodologia experimental estar bem estabelecida em cenários tradicionais *batch* de AM e MD, em FCDs, diferentes metologias tem sido adotadas por diversos autores. Esta falta

de padrão torna a comparação desses diferentes estudos uma tarefa difícil. Além disso, muitas questões importantes a respeito da metologia experimental em FCDs precisam ser tratadas ou melhor exploradas.

Um importante desafio a ser tratado é a adaptação e o desenvolvimento de medidas de avaliação capazes de fornecer uma avaliação significativa do desempenho das técnicas de DN. Em cenários envolvendo evolução de conceito e naqueles no qual a DN é considerada um problema multiclasse, as medidas de avaliação representam um importante desafio, especialmente porque o processo de DN usa exemplos não rotulados. Além disso, essas medidas de avaliação precisam lidar com a presença de exemplos rotulados como *desconhecidos*.

A maioria das técnicas disponíveis na literatura consideram que o rótulo verdadeiro das instâncias sempre estará disponível. Entretanto, esta não é uma suposição realística, uma vez que obter o rótulo verdadeiro é uma tarefa que consume tempo e esforço, e muitas vezes precisa da intervenção de um especialista de domínio. Uma forma de contorna esse problema é perguntar pelo rótulo verdadeiro de um subconjunto dos exemplos, o que pode ser feito usando técnicas de aprendizado ativo. Apesar de sua importância, poucos trabalhos tratam essa questão.

Diferentes trabalhos tratam DN como um problema de classificação com uma classe, no qual os exemplos de treino pertencem somente a um conceito normal e cujo objetivo é discriminar exemplos do conceito normal dos não-normais. Em problemas de classificação multiclasse, o conceito não normal pode estar associado com uma ou mais classes. Assim, propostas com uma classe precisam ser adaptadas para serem usadas nessas situações. Alguns exemplos de problemas de classificação multiclasse que se enquadram nessa situação são detecção de intrusos em redes de computadores, detecção de fraudes, detecção de falhas, detecção de tipo de cobertura de floresta, filtros de spam e classificação de texto.

Além disso, é também importante desenvolver propostas capazes de tratar ruídos e *outliers* em ambientes não estacionários, nos quais os conceitos podem evoluir gradual ou abruptamente. Distinguir entre um ruído ou *outlier* de um conceito novidade é um aspecto importante a ser considerado em um sistema de DN, uma vez que um conceito novidade é composto por um conjunto de exemplos coesos e representativos, e ruídos e *outliers* precisam ser removidos. Além disso, os conceitos conhecidos podem evoluir gradualmente ou abruptamente e o sistema de DN precisa atualizar o modelo de decisão para tratar esses diferentes tipos de evolução a fim de representar o cenário atual do FCD.

Outra questão importante é a definição automática dos principais parâmetros dos algoritmos. Em geral, os algoritmos propostos na literatura requerem um conjunto de parâmetros de entrada do usuário, tais como, número de classificadores, número de grupos, parâmetros relacionados à validação dos grupos, tamanho da janela, limiar para separar novidades de extensões, etc. A maioria desses parâmetros são muito importantes para a construção dos classificadores e tem impacto direto no desempenho dos mesmos. Além

disso, esses parâmetros podem variar de acordo com a base de dados, o que dificulta sua definição pelo usuário.

Além disso, o desenvolvimento de novas propostas para lidar com atributos preditivos cujo valor não é numérico, como categórico, ordinal, hierárquico e estruturas de dados complexas, é um importante aspecto a ser tratado. Várias propostas para DN são baseadas em técnicas de agrupamento, as quais podem lidar somente com atributos numéricos. Além disso, a abundância de dados gerados a partir de diferentes fontes de dados pode gerar estruturas de dados complexas, as quais não podem ser mineradas pela maioria dos algoritmos da literatura.

Novas bases de dados públicas, reais e artificias, apresentando diferentes alternativas de configuração de dados, podem ser muito úteis para medir o desempenho das técnicas existentes e de novas técnicas para DN.

3.10 Considerações Finais

Esse capítulo apresentou uma discussão dos principais trabalhos existentes na literatura para DN em FCDs. Ele também apresentou os principais desafios da área e as principais questões pouco tratadas pelos algoritmos da literatura.

O próximo capítulo apresenta um novo algoritmo para DN em FCDs. Esse algoritmo trata a DN como uma tarefa de classificação multiclasse.

CAPÍTULO

4

MINAS

4.1 Introdução

O problema de DN tem sido apresentado na literatura como um problema de classificação de uma classe, cujo objetivo é distinguir os exemplos da classe *normal* dos exemplos da classe *não normal*. Assim, a fase de aprendizado é baseada em exemplos de uma única classe, a classe normal. Na fase de aplicação, o fluxo de exemplos não rotulados pode ser classificado como pertencente ao conceito normal ou como não normal.

No entanto, a DN pode ser vista como uma tarefa muito mais geral do que a classificação com uma classe. Em problemas envolvendo DN, o conceito normal pode ser composto por diferentes classes e novas classes podem aparecem ao longo do tempo. Para isso, o modelo de decisão não pode ser estático, mas deve evoluir a fim de representar as classes emergentes. Assim, tratar a DN sob uma perspectiva de múltiplas classes é uma proposta mais realista para o problema.

Os FCDs representam um importante cenário no qual as técnicas de DN são importantes e podem ser aplicadas. No contexto de FCDs, novos conceitos podem aparecer, conceitos antigos podem desaparecer e conceitos conhecidos podem mudar (Spinosa et al., 2009). Uma vez que os conceitos estão em constante mudança, a aplicação de técnicas de DN em FCDs representa um importante desafio a ser tratado (Gama, 2010).

Neste capítulo será apresentado um novo algoritmo, o MINAS (MultI-class learNing Algorithm for data Streams) – Algoritmo de aprendizado multiclasse para FCDs, que trata a DN em FCDs sob a perspectiva de uma tarefa multiclasse. MINAS apresenta as seguintes contribuições:

- O Modelo de decisão usado para representar o conceito conhecido trata problemas envolvendo múltiplas classes;
- Uso de um conjunto coeso e representativo de exemplos, não explicados pelo modelo de decisão atual, para a aprendizagem de novos conceitos ou extensão dos conceitos conhecidos, tornando o modelo de decisão dinâmico;
- Detecção de diferentes padrões-novidade e indução de um modelo de decisão que representa um cenários multiclasse, no qual o número de classes não é fixo e nem previamente conhecido;
- Uso de um único modelo de decisão representando todo o conhecimento aprendido até o momento atual, composto pelas classes aprendidas na fase de treinamento *offline*, e pelos padrões-novidade detectados automaticamente de maneira *online*;
- Na presença de ruídos ou *outliers*, exemplos isolados não são considerados um novo padrão-novidade, uma vez que um padrão-novidade é composto por um grupo coeso e representativo de exemplos.

As próximas seções apresentam os detalhes do algoritmo proposto. Muitas das seções apresentadas neste capítulo são baseadas no trabalho (Faria et al., 2013b).

4.2 Motivação para criação do algoritmo MINAS

Ainda que existam diferentes algoritmos na literatura para DN em FCDs, nenhum deles reúne todas as características necessárias para tratar o problema adequadamente e atender às exigências formalizadas na Seção 3.4. Esses algoritmos podem ser classificados em três categorias básicas:

- Os que consideram a DN como uma tarefa binária e atualizam o modelo de decisão usando *feedback* externo;
- Os que consideram a DN como uma tarefa binária e atualizam o modelo de decisão sem *feedback* externo;
- Os que consideram a tarefa de DN como multiclasse, permitindo, no entanto, o aparecimento de apenas uma nova classe por vez, e atualizam o modelo de decisão com *feedback* externo.

Os algoritmos da primeira categoria consideram que uma novidade é formada pela presença de um único exemplo não explicado pelo sistema, enquanto que a segunda e terceira categorias usam grupos de exemplos *desconhecidos* para identificar um padrão-novidade.

Outro ponto a ser destacado é que todos os algoritmos tendem a utilizar modelos de decisão distintos para o que é conhecido e o que é novidade. Em particular, criam um modelo de decisão para as classes conhecidas e um para as novidades. Além disso, poucos algoritmos propõem estratégias para tratar contextos recorrentes.

A ideia de desenvolver mais um algoritmo para DN em FCD, o MINAS, surge da necessidade de criar um algoritmo que reúna todas essas características e que possua um desempenho comparável aos algoritmos já propostos na literatura. Portanto, assim como muitos dos algoritmos da literatura, o MINAS:

- Possui duas fases, nomeadas *offline* e *online*.
- A fase *offline* representa cada classe por um conjunto de exemplos.
- Marca como *desconhecidos* e coloca em uma memória temporária os exemplos não explicados pelo modelo de decisão.
- Usa grupos de exemplos *desconhecidos* para criar padrões-novidade.

Diferente dos algoritmos encontrados na literatura, o MINAS propõe:

- Um modelo de decisão integrado, que represente as classes conhecidas do problema e os padrões-novidades não-rotulados. O modelo de decisão é criado a partir de dados rotulados de diferentes classes e atualizado a partir de exemplos não rotulados.
- Um procedimento para DN que consegue identificar diferentes padrões-novidade ao longo do FCD.
- Um procedimento para DN que consegue identificar extensões dos conceitos conhecidos, e não só extensões da classe normal, diferenciando-as de padrões-novidade.
- O uso de uma memória *sleep* que permite tratar o problema de contextos recorrentes.
- A remoção dos elementos da memória temporária quando esses não são usados para formar novos micro-grupos, por meio da associação de um marcador de tempo a cada elemento.
- A validação de novos micro-grupos usando a medida de validação interna silhueta.
- A identificação de padrões-novidades, formados por conjuntos de micro-grupos, que diferenciam-se de outros padrões-novidade ou de classes conhecidas por apresentarem distância a esses maior que um limiar.
- Diferentes estratégias para cálculo do limiar de forma automática.

4.3 Visão Geral do algoritmo MINAS

O algoritmo MINAS é composto por duas fase: *offline* e *online*. As figuras 4.1 e 4.2 ilustram as fases *offline* e *online* do algoritmo MINAS. Na fase *offline*, um modelo de decisão é criado utilizando os conceitos conhecidos. Nessa fase, MINAS recebe um conjunto de dados rotulados contendo exemplos de uma ou mais classes. A fase *offline* é supervisionada e executada somente uma vez. Cada classe do problema é representada por um conjunto de micro-grupos.

A ideia de criar um classificador composto por um conjunto de micro-grupos, cada um contendo um sumário estatístico dos dados, é que esses micro-grupos podem evoluir ao longo do tempo. Assim, um micro-grupo pode ser eliminado, novos micro-grupos podem ser criados ou micro-grupos existentes podem ser atualizados. Se algoritmos de classificação para cenários estáticos são usados nessa fase, o processo de atualização do modelo de decisão fica prejudicado, uma vez que um novo classificador deve ser treinado sempre que houver mudanças no FCDs.

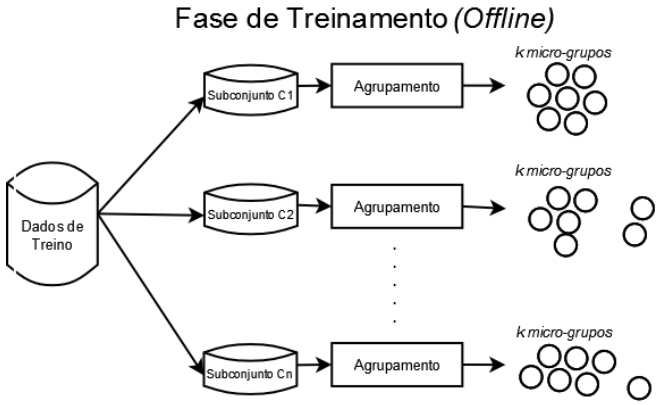


Figura 4.1: Visão geral da fase *offline*.

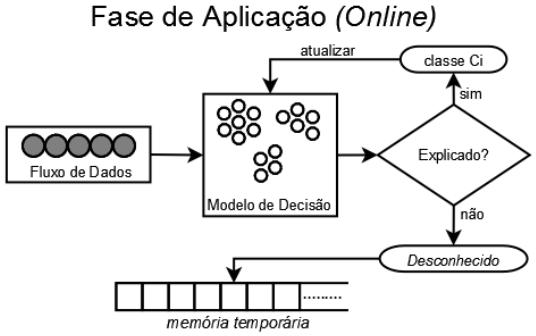


Figura 4.2: Visão geral da fase *online*.

Na fase *online*, MINAS recebe novos exemplos não rotulados e classifica-os de acordo com o modelo de decisão atual. Os exemplos classificados pelo modelo de decisão podem

ser usados para atualizá-lo. No entanto, alguns exemplos não são explicados pelo modelo de decisão atual, sendo então marcados com o perfil *desconhecido*. Exemplos *desconhecidos* são armazenados em uma *memória temporária* para análise futura. De tempos em tempos, os exemplos marcados como *desconhecidos* são agrupados, gerando novos micro-grupos. Os micro-grupos obtidos são validados a fim de descartar os não coesos ou não representativos. Os micro-grupos válidos são então analisados a fim de decidir se eles representam extensões dos conceitos já aprendidos ou padrões-novidade. Nos dois casos, esses micro-grupos são adicionados ao modelo de decisão atual e usados para classificar novos exemplos.

No entanto, além de incrementar o modelo de decisão pela adição de novos micro-grupos ou pela atualização dos micro-grupos existentes, o algoritmo MINAS inclui um mecanismo para descartar micro-grupos que se tornaram obsoletos e não representam mais o cenário atual do FCD. Além disso, o algoritmo MINAS também inclui um mecanismo para limpar a *memória temporária* a fim de eliminar os exemplos marcados como *desconhecidos* que não foram usados na criação de novos micro-grupos.

4.4 Fase de Aprendizado Supervisionada

A fase de aprendizado supervisionada (ou fase *offline* ou fase de treinamento) é executada apenas uma vez e representa o aprendizado dos conceitos conhecidos (ver Figura 4.1). O Algoritmo 4.1 ilustra os detalhes dessa fase. A fase *offline* recebe como entrada um conjunto de dados rotulados contendo exemplos de diferentes classes. Esse conjunto de dados rotulados é dividido em subconjuntos, cada um representando uma das classes do problema. Um algoritmo de agrupamento é então executado para cada subconjunto, produzindo k micro-grupos para representar cada uma das classes do problema. Micro-grupos são uma estrutura compacta para representação dos dados, apresentada na Seção 2.4.2. Neste trabalho, cada micro-grupo é representado por quatro informações: número de exemplos do grupo (N), soma linear dos N exemplos do grupo (LS), soma quadrada dos N exemplos do grupo (SS) e marcador de tempo da chegada do último exemplo no micro-grupo. Usando essas quatro informações, é possível calcular medidas como o centróide de um grupo e o seu raio. Além disso, esses micro-grupos possuem as propriedades de incrementalidade, que permite a inserção de novos exemplos no micro-grupo, e aditividade, que permite unir dois micro-grupos, somando seus componentes.

A fim de encontrar k micro-grupos representando cada uma das classes que compõem o conjunto de treinamento, o algoritmo MINAS pode usar dois algoritmos de agrupamento de dados, K-Means e CluStream. O K-Means é um dos algoritmos clássicos de AM. É um algoritmo particional baseado na ideia de protótipos, que visa minimizar a distância quadrada de cada exemplo ao seu centróide. Apesar do K-Means ter um baixo custo computacional, ele apresenta algumas desvantagens, tais como, ser sensível a *outliers*,

ser sensível à escolha dos centróides iniciais e poder gerar grupos vazios. Além disso, ele não é adequado para grandes bases de dados.

Uma alternativa ao uso do K-Means para a geração dos micro-grupos iniciais é usar a fase *online* do algoritmo CluStream, descrito na Seção 2.4.2, que mantém incrementalmente um sumário dos dados, representado por um conjunto de micro-grupos. Assim, na fase *offline* do algoritmo MINAs, a fase *online* do algoritmo Clustream é executada nos dados rotulados de cada uma das classes conhecidas do problema. O algoritmo CluStream foi desenvolvido para trabalhar com FCDs, apresenta baixa complexidade computacional, possui tratamento para *outliers*, não gera grupos vazios e é um algoritmo incremental.

Depois de executar o algoritmo de agrupamento, cada micro-grupo é representado pelas três componentes de sumarização (N, LS e SS), por um indicador de tempo t, que representa o marcador de tempo do último exemplo inserido no micro-grupo, e por um rótulo, que indica a qual classe esse micro-grupo está associado. Assim, a fronteira de decisão de cada classe é estabelecida pela união dos k micro-grupos que a representam. O modelo de decisão inicial é composto pela união dos $k-$micro-grupos obtidos para cada classe.

O número de micro-grupos usados para representar uma classe é um fator importante a ser considerado. No algoritmo K-Means, este valor deve ser fornecido como entrada para o algoritmo. No entanto, técnicas como o OMRk (Naldi et al., 2011) buscam automaticamente identificar o melhor valor de k para a execução do K-Means. No caso do algoritmo CluStream, os autores sugerem que o número de micro-grupos deve ser bem maior que o número de grupos presentes nos dados. No entanto, esse número está diretamente ligado ao espaço disponível em memória para armazená-los. Contudo, um número muito grande de micro-grupos pode ser ineficiente em termos de busca e armazenamento. Por outro lado, um número maior de micro-grupos permite representar melhor dados de diferentes formatos.

É importante notar que, usando o sumário estatístico dos dados, pode-se calcular o centroide e o limite máximo de cada grupo (aqui chamado de raio). Com essas informações é possível dizer se um novo exemplo pode ser classificado usando um dado micro-grupo, ou seja, se sua distância ao centróide é menor que o raio do micro-grupo. O raio de cada micro-grupo é calculado, como proposto no algoritmo CluStream, como o desvio padrão da distância dos exemplos do grupo ao centróide do grupo multiplicado por um fator fat.

Por último, é importante destacar que o modelo de decisão gerado na fase *offline* possibilita sua evolução ao longo do tempo, por meio da inserção de novos micro-grupos, remoção de micro-grupos antigos ou atualização de micro-grupos existentes, o que permite uma atualização com baixo custo computacional, sem necessidade de reconstrução total do modelo de decisão. Este é um fator muito importante ao se trabalhar com FCDs. Além disso, a estrutura proposta permite que o modelo de decisão seja atualizado mesmo quando os rótulos dos novos exemplos não estejam disponíveis, ou seja, sem *feedback*

externo.

Algoritmo 4.1: MINAS: Algoritmo para a fase de aprendizado *offline*

Entrada: k, $alg\_Agrupamento$, $ConjuntoTreino$

```
1 Modelo ← ∅;
2 para cada (classe C_i ∈ ConjuntoTreino) faça
3 |   ModeloTmp ← Clustering(ConjuntoTreino_{Classe=C_i}, k, alg_Agrupamento);
4 |   para cada (micro-grupo micro ∈ ModeloTmp) faça
5 |   |   micro.rotulo ← C_i;
6 |   fim
7 |   Modelo ← Modelo ∪ ModeloTmp;
8 fim
9 retorna Modelo
```

4.5 Fase de Aplicação

A fase de aplicação (ou fase *online*) é composta de três operações: classificar novos exemplos, detectar padrões-novidade e atualizar o modelo de decisão. Assim, ela recebe uma sequência de exemplos não rotulados, gerados continuamente ao longo do FCD, um por vez (ver Figura 4.2). O Algoritmo 4.2 ilustra os detalhes dessa fase. Para cada novo exemplo não rotulado, MINAS verifica se ele pode ser explicado pelo modelo de decisão atual. Para isso, a distância $Dist$ entre esse novo exemplo e o centróide do micro-grupo mais próximo a ele é calculada, usando uma medida de dissimilaridade. Nos experimentos realizados, foi utilizada a distância Euclidiana. Se a distância $Dist$ é menor que o raio do micro-grupo, então o exemplo é explicado por esse micro-grupo e classificado usando o rótulo associado a ele. Quando um exemplo é explicado por um dos micro-grupos, o indicador de tempo t do micro-grupo é atualizado com o valor do marcador de tempo associado ao exemplo inserido. Além disso, o sumário estatístico desse micro-grupo pode ser atualizado a fim de refletir o novo exemplo. Para isso, usando a propriedade da aditividade, o exemplo pode ser inserido nesse micro-grupo. Foram testadas duas abordagens, uma que atualiza o micro-grupo depois que um elemento é inserido nele e uma abordagem que não atualiza o micro-grupo. Uma comparação entre essas duas abordagens é apresentada no Capítulo 6.

Caso um novo exemplo não seja explicado por nenhum dos micro-grupos que compõem o modelo de decisão atual, ele é marcado com o perfil *desconhecido* e movido para uma *memória temporária* para análise futura. Os exemplos *desconhecidos* são usados para modelar novos micro-grupos que podem representar padrões-novidade ou extensões dos conceitos conhecidos. Classificar um exemplo como *desconhecido* significa dizer que o modelo de decisão atual não tem informação suficiente para classificá-lo. Uma dentre três situações pode ocorrer:

- O exemplo é um ruído ou *outlier* e não pode ser explicado por nenhum dos micro-grupos atuais, portanto, o melhor é eliminá-lo;
- O exemplo representa uma mudança em um dos conceitos já aprendidos;
- O exemplo é um padrão-novidade não presente no modelo de decisão atual. Nos dois últimos cenários, é necessário ter um conjunto significativo e representativo de exemplos marcados como *desconhecidos* que serão usados para atualizar o modelo de decisão de maneira não-supervisionada.

A cada vez que uma janela de dados é processada, o modelo de decisão deve esquecer os micro-grupos que não representam mais o cenário atual do fluxo. Para isso, os micro-grupos que não receberam exemplos nessa janela devem ser movidos para a memória *sleep*. Os detalhes sobre esse processo são descritos na Seção 4.8.

É importante notar que o MINAS trabalha com um único modelo de decisão, composto por diferentes classes aprendidas de forma *offline* e por um conjunto de padrões-novidade ou extensões, aprendidas de maneira *online*, ambos representados por um conjunto de micro-grupos. Assim, classificar um novo exemplo significa atribuir-lhe um rótulo obtido a partir das classes aprendidas *offline* ou a partir dos padrões-novidade ou extensões aprendidos de forma *online*.

Algoritmo 4.2: MINAS: Algoritmo para a fase de aplicação

Entrada: $Modelo$, FCD, T, $NumMinExamples$, ts, P

$MemTmp \leftarrow \emptyset$;
$MemSleep \leftarrow \emptyset$;
para cada *($exemplo \in FCD$)* **faça**
  ($Dist$,$micro$) $\leftarrow$ *micro-mais-proximo*($exemplo$,$Modelo$);
  se *($Dist <$ raio($micro$)* **então**
    $exemplo.classe \leftarrow micro.rotulo$;
    atualizar-micro($micro$,$exemplo$);
  fim
  senão
    $exemplo.classe \leftarrow desconhecido$;
    $MemTmp \leftarrow MemTmp \cup exemplo$;
    se *($|MemTemp| \geq NumMinExemplos$)* **então**
      $Modelo \leftarrow$ *deteccao-novidade*($Modelo$,$MemTmp$, T);
    fim
  fim
  $TempoAtual \leftarrow exemplo.T$;
  se *($TempoAtual$ mod $TamJanela$ == 0)* **então**
    $Modelo \leftarrow$ *mover-micro-grupos-mem-sleep*($Modelo$,$MemSleep$,P);
    $MemTmp \leftarrow$ *remover-exemplos-antigos*($MemTmp$, ts);
  fim
fim

4.6 Detecção de Padrões-novidade e Extensões

Toda vez que um exemplo é marcado como *desconhecido*, é necessário verificar se há um número mínimo de exemplos na memória temporária ($NumMinExemplos$). Caso haja, o algoritmo MINAS executa uma procedimento para DN de forma não-supervisionada, ilustrado na Figura 4.3. O Algoritmo 4.3 mostra os detalhes desse processo. A primeira etapa do procedimento de DN aplica um algoritmo de agrupamento aos exemplos da *memória temporária*, produzindo um conjunto de k novos micro-grupos. Novamente, os algoritmos K-Means ou CluStream podem ser usados nesse processo. Cada um dos novos micro-grupos é avaliado a fim de identificar se ele representa um grupo válido.

Um novo micro-grupo é considerado válido se a sua largura da silhueta é maior que 0. A Equação 4.1 representa a fórmula usada para calcular a largura da silhueta simplificada Vendramin et al. (2010) de um novo micro-grupo. Nessa equação, b representa a distância Euclidiana entre o centróide do novo micro-grupo e o centróide do seu micro-grupo mais próximo e a representa o desvio padrão das distâncias Euclidiana entre os elementos do novo micro-grupo e o centróide do novo micro-grupo. A largura da silhueta produz um valor entre 0 e 1.

$$Silhueta = \frac{b - a}{max(b,a)} \tag{4.1}$$

Além de verificar se um novo micro-grupo é valido usando a sua silhueta, também é necessário verificar se o micro-grupo é representativo. Para isso, ele deve ter um número mínimo de exemplos. Esse número mínimo de exemplos é determinado por um parâmetro de entrada do usuário. Assim, se o micro-grupo é coeso e representativo, significa que ele é válido e, portanto, deve ser inserido no modelo de decisão atual.

Antes de adicionar o novo micro-grupo válido ao modelo de decisão, é necessário decidir se ele representa uma extensão de um conceito já aprendido ou um padrão-novidade. Assim, o próximo passo é encontrar a distância $Dist$ entre o centróide do novo micro-grupo m e o centróide do seu micro-grupo mais próximo mp, dentre todos aqueles que compõem o modelo de decisão atual. Se essa distância é menor que um limiar T, então o novo micro-grupo é rotulado como uma extensão. Seu rótulo é o mesmo rótulo do micro-grupo mp. Caso contrário, o micro-grupo é rotulado como um padrão-novidade, sendo criado um novo rótulo sequencialmente $\mathbf{PN}_1, \mathbf{PN}_2, \mathbf{PN}_3, ...$ Em ambos os casos, o algoritmo atualiza o modelo de decisão e incorpora este novo micro-grupo. Uma discussão sobre como escolher o valor do limiar T ocorre na Seção 4.7.

Se um micro-grupo é inválido, os exemplos que o compõem continuarão na *memória temporária* e serão usados futuramente em novos processos de agrupamento. No entanto, se estes exemplos que não foram usados para gerar grupos válidos permanecem por muito tempo na *memória temporária*, eles devem ser removidos. A estratégia para eliminação

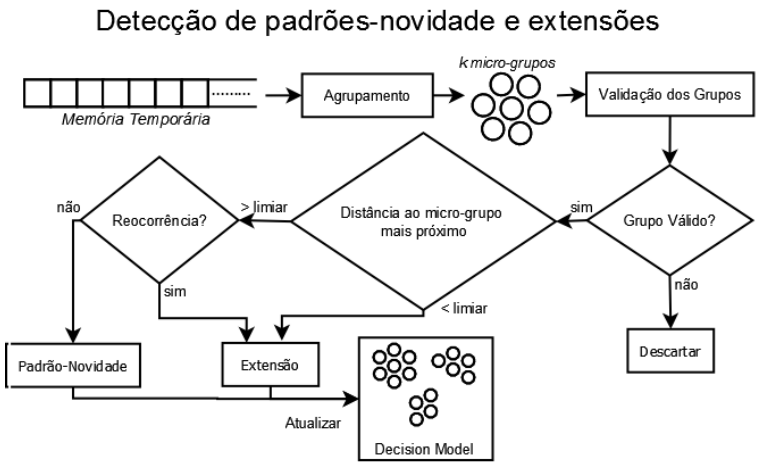


Figura 4.3: Visão geral do processo de detecção de padrões-novidade e extensões.

destes exemplos da *memória temporária* é discutida na Seção 4.9

É importante notar que o MINAS consegue distinguir entre diferentes padrões-novidade aprendidos ao longo do tempo. Além disso, um padrão-novidade pode ser formado por um micro-grupo ou por um conjunto de micro-grupos. No entanto, os padrões-novidade encontrados sem *feedback* não estão associados diretamente às classes do problema. Assim, quando uma nova classe aparece no FCD, ela pode ser identificada por um ou mais padrões-novidade. Somente um especialista de domínio poderá dizer se os padrões-novidade encontrados pelo algoritmo representam de fato uma nova classe do problema e se diferentes padrões-novidade podem estar relacionados a uma mesma classe.

4.7 Escolhendo o Valor do Limiar T

Neste trabalho foram utilizadas duas estratégias para selecionar o melhor valor do limiar T que distingue padrões-novidade de extensões. A primeira delas usa um valor absoluto, fornecido pelo usuário. No entanto, um valor que consiga discriminar entre os diferentes padrões-novidade e as extensões das classes conhecidas pode variar dependendo do conjunto de dados. Além disso, é difícil encontrar um único valor que consiga separar padrões-novidade de extensões. Em geral, valores pequenos para T produzem muitos padrões-novidade e poucas extensões. Por outra lado, valores grandes para T produzem muitas extensões e poucas novidades. Assim, essa estratégia não se mostrou eficiente.

A segunda estratégia é baseada na coesão entre os exemplos de um micro-grupo e visa encontrar um valor para T automaticamente. Em um micro-grupo com alta coesão, pequenos valores de T podem ser usados para distinguir entre extensão e novidade. A medida de coesão empregada aqui é o desvio padrão das distâncias entre os exemplos e o centróide. O limiar T usado é definido com um fator f dessa medida de coesão. O valor do fator f usado foi 1.1, definido de forma empírica. Duas outras estratégias foram usadas

Algoritmo 4.3: MINAS: Algoritmo para a detecção de padrões-novidade ou extensões

```
   Entrada: Modelo, MemTmp, T
1  ModeloTmp ← Clustering(MemTmp, k, algoritmo);
2  para cada micro-grupo micro in ModeloTemp faça
3      se CriterioValidacao(micro) então
4          (Dist,microP) ← Micro-mais-proximo(micro,Modelo);
5          se Dist ≤ T então
6              micro.rotulo ← microP.rotulo;
7          fim
8          senão
9              (Dist,microP) ← Micro-mais-proximo(micro,MemSleep);
10             se Dist ≤ T então
11                 micro.rotulo ← microP.rotulo;
12             fim
13             senão
14                 micro.rotulo ← novo rotulo;
15             fim
16         fim
17         Modelo ← Modelo ∪ micro;
18     fim
19 fim
20 retorna Modelo;
```

para calcular o valor do limiar, mas obtiveram piores resultados. Elas serão apresentadas no Capítulo 6.

4.8 Esquecimento do Passado e Tratamento de Recorrência

Na fase *online*, o modelo de decisão é atualizado de duas formas. Na primeira, ao classificar um exemplo usando um micro-grupo, esse deve ter seu sumário estatístico atualizado. Na segunda, ao obter um novo micro-grupo válido a partir de um conjunto de exemplos *desconhecidos*, esse micro-grupo é analisado como sendo uma extensão ou um padrão-novidade e deve ser inserido ao modelo de decisão. Além de atualizar o modelo de decisão pela inserção/atualização de micro-grupos, é necessário esquecer micro-grupos antigos, que podem estar não contribuindo mais para a recente atividade do FCD. Assim, cada micro-grupo armazena uma componente que representa o marcador de tempo associado ao último exemplo que foi classificado nesse micro-grupo. Os micro-grupos que não recebem novos exemplos por um dado período de tempo P são movidos para uma memória *sleep*, permitindo ao modelo esquecer micro-grupos que se tornaram antigos.

Quando um novo micro-grupo válido é detectado pelo MINAS, a memória *sleep* é consultada para decidir entre uma extensão ou um padrão-novidade. Se a distância entre o

centróide do novo micro-grupo e o centróide de um dos micro-grupos da memória *sleep* é menor que o limiar T, então o novo micro-grupo representa uma extensão de um conceito conhecido. Isto pode indicar a recorrência de um conceito que estava adormecido. Caso contrário, o novo micro-grupo é reconhecido como um padrão-novidade. Quando uma extensão é identificada, o micro-grupo que estava adormecido na memória *sleep* e retirado da mesma e volta a fazer parte do modelo de decisão atual.

4.9 Tratamento de Ruídos e *Outliers*

Um importante ponto a ser destacado em sistemas para detecção de novidade é o tratamento de ruídos e *outliers*, os quais muitas vezes podem ser confundidos com o aparecimento de um novo conceito. Quando um novo micro-grupo é obtido, MINAS aplica um critério de validação a esse grupo para eliminar aqueles que não apresentem um critério mínimo de coesão ou não apresentem um número mínimo de exemplos. Micro-grupos que possuem poucos exemplos e/ou em que esses são esparsos, são candidatos a serem removidos, pois podem indicar a presença de ruídos/*outliers*. Os critério de validação dos micro-grupos foram discutidos na Seção 4.6.

Quando um micro-grupo é removido por não apresentar as características mínimas para se tornar uma extensão ou novidade, seus exemplos continuam na *memória temporária* para análise futura. No entanto, se um exemplo está na *memória temporária* por um longo período do tempo, ele deve ser removido. A razão para isso é que o exemplo já participou de pelo menos um processo de agrupamento e o seu micro-grupo não atendeu a um critério de validade, portanto é um candidato a ruído ou *outlier*. Ou então, o exemplo está há muito tempo na *memória temporária* e pode ser que ele não contribua mais para as características atuais do FCD. Assim, é necessário o desenvolvimento de um mecanismo que retire da *memória temporária* exemplos antigos.

A cada vez que um novo exemplo é adicionado na *memória temporária*, o marcador de tempo t associado a esse exemplo é também armazenado. Elementos que possuem um marcador de tempo muito antigo, em relação ao marcador de tempo atual, devem ser eliminados. Assim, todo vez que um novo exemplo é adicionado à *memória temporária*, esta é verificada a fim de eliminar exemplos que possuam a diferença entre marcador de tempo atual e o marcador de tempo do exemplo menor que um limiar ts. O parâmetro ts é definido pelo usuário e discutido na Capitulo 6.

4.10 Análise de Complexidade

A complexidade algorítmica é um importante fator a ser considerado no desenvolvimento de algoritmos para FCDs. Uma das exigências para se trabalhar com FCDs é que, quando um novo exemplo do fluxo é processado, uma única varredura seja feita nos dados. Além

disso, também não há espaço suficiente em memória para armazenar todos os exemplos, sendo possível armazenar apenas um sumário dos dados.

No algoritmo MINAS, a fase *offline* é executada de maneira *batch*, sendo que, nesse caso, o algoritmo usado pode fazer várias varreduras nos dados. Nessa fase, um algoritmo de agrupamento é executado para cada uma das c classes do problema. Como resultado, k micro-grupos são gerados para cada classe. Para criar os k micro-grupos, duas sugestões de algoritmos de agrupamento de dados foram apresentadas, CluStream e K-Means.

Usando o K-Means tem-se a seguinte complexidade para cada classe $O(k \times N \times d \times v)$, sendo k o número de micro-grupos, N o número de exemplos a ser agrupados, ou seja, o número de exemplos do conjunto de treinamento de uma dada classe, d a dimensionalidade dos dados, e v o número de iterações do K-Means.

Usando o algoritmo CluStream, k micro-grupos também são produzidos para cada uma das c classes do problema. Inicialmente, o Clustream executa o K-Means em uma pequena porção inicial dos dados (nomeada $InitNumber$), para inicializar os micro-grupos. Em seguida, ele atribui cada um dos demais exemplos a um dos micro-grupos, ou cria um novo micro-grupo para representar o novo exemplo. Assim, a complexidade para executar o K-Means no conjunto que contém $InitNumber$ exemplos é $O(k \times InitNumber \times d \times v)$. A complexidade para inserir um novo exemplo requer a busca pelo micro-grupo mais próximo, que é da ordem de $O(k \times d)$. Se o micro-grupo pode receber o exemplo, ele é inserido, atualizando o sumário estatístico, caso contrário, os dois micro-grupos mais próximo são encontrados, $O(k^2 \times d)$, e unidos, o que é da ordem de O(1).

Na fase *online*, cada vez que um novo exemplo é lido, o micro-grupo mais próximo a ele é encontrado. Para isso, cada um dos q micro-grupos presente na memória deve ser consultado, o que tem complexidade $O(q \times d)$. É importante destacar que q é o número de micro-grupos na memória em um dado momento, composto pelos micro-grupos aprendidos na fase *offline*, k micro-grupos para cada uma das c classes, mais os micro-grupos aprendidos de forma *online*, que representam extensões ou padrões-novidade. No entanto, desse conjunto devem ser subtraídos os micro-grupos esquecidos ao longo do fluxo, que são movidos para a memória *sleep*.

A fim de identificar padrões-novidade, de tempos em tempos os exemplos da memória temporária, marcados com o perfil desconhecido, são agrupados usando os algoritmos K-Means ou CluStream, cuja complexidade já foi discutida. A fim de identificar se um novo micro-grupo é uma extensão ou um padrão novidade, o micro-grupo mais próximo a ele é encontrado, o que tem complexidade $O(q \times d)$. Além disso, os micro-grupos que ficam muito tempo sem receber novos exemplos são movidos para a memória *sleep*. Essa tarefa verifica o indicador de tempo de cada micro-grupo, que é da ordem de $O(q \times d)$.

É importante notar que o número de micro-grupos presentes na memória é um fator importante para análise de complexidade. Se, por um lado, usar uma grande quantidade de micro-grupos permite uma maior separabilidade entre as classes e a representação das

classes com diferentes formatos, por outro lado, a classificação de novos exemplos pode ser computacionalmente mais custosa. Além disso, o número de micro-grupos na memória está diretamente ligado à capacidade de armazenamento da mesma.

4.11 Considerações Finais

Esse capítulo apresentou o algoritmo MINAS, desenvolvido para DN em cenários envolvendo múltiplas classes em FCD. MINAS cria um modelo de decisão na fase *offline*, composto por um conjunto de micro-grupos, que representa cada uma das classes conhecidas do problema. Na fase *online*, MINAS classifica novos exemplos em uma das classes conhecidas ou com o perfil *desconhecido*. Grupos de exemplos *desconhecidos* são usados para modelar extensões dos conceitos conhecidos ou padrões-novidades. Em ambos os casos, o modelo de decisão é atualizado.

Além do desenvolvimento de novas técnicas para o problema de DN em FCD, também é importante destacar como avaliá-las. Para cenários nos quais a DN é tratada como uma tarefa de classificação com uma classe, há medidas de avaliação disponíveis na literatura. No entanto, quando a DN é tratada como um problema envolvendo múltiplas classes em um cenário que evolui ao longo do tempo, há poucas metodologias e medidas na literatura que consigam avaliar esse tipo de algoritmo.

O próximo capítulo apresenta uma nova metodologia proposta para avaliar algoritmos para DN em problemas multiclasse em FCD.

CAPÍTULO

5

Metodologia de Avaliação

5.1 Introdução

DN é uma técnica útil para sistemas de aprendizagem, especialmente em FCD, no qual conceitos podem aparecer, desaparecer ou evoluir ao longo do tempo. Há diferentes trabalhos investigando o uso de novas técnicas e algoritmos para DN em FCD (Al-Khateeb et al., 2012a), (Masud et al., 2011a), (Hayat e Hashemi, 2010), (Farid et al., 2013) . Entretanto, não há um consenso em como avaliar o desempenho desses algoritmos, em particular em problemas envolvendo múltiplas classes.

Neste trabalho, além de propor um novo algoritmo para DN, o MINAS, uma nova metodologia para avaliar algoritmos de DN multiclasse em cenários envolvendo FCD é proposta. Essa metologia é capaz de lidar com:

- Tarefa de DN usando exemplos não rotulados, que gera padrões-novidade sem uma associação com as classes do problema, e no qual uma classe novidade pode ser composta por um ou mais padrões-novidade;
- Matriz de confusão que é incrementada ao longo do tempo, adicionando linhas (classes novidade) ou colunas (padrões-novidade detectados pelo algoritmo);
- Matriz de confusão que contém uma coluna representando os exemplos *desconhecidos*, aqueles não explicados pelo modelo de decisão atual;
- Representação dos resultados obtidos a partir do uso de medidas de avaliação ao longo do tempo.

Este capítulo apresenta uma nova metologia para associar os padrões-novidade, detectados pelo algoritmo usando exemplos não rotulados, às classes do problema, sendo que uma classe pode ser composta por um conjunto de padrões-novidade. Muitas das seções apresentadas neste capítulo são baseadas no trabalho (Faria et al., 2013a).

5.2 Contextualização do Problema

A maioria dos trabalhos da literatura sobre DN tratam-na como uma tarefa de classificação com uma classe, cujo objetivo é discriminar entre exemplos da classe normal dos exemplos da classe não-normal (Marsland et al., 2002), (Markou e Singh, 2003a), (Clifton et al., 2006), (Ahmed e Coates, 2007), (Spinosa et al., 2009). Esses trabalhos usam medidas de classificação binárias para avaliar o desempenho de algoritmos para DN.

Trabalhos recentes têm tratado DN como um problema de classificação multiclasse, no qual o conceito normal pode ser composto por diferentes classes e novas classes podem surgir ao longo do tempo (Masud et al., 2011a), (Farid e Rahman, 2012), (Al-Khateeb et al., 2012a). Entretanto, a maioria dessas técnicas usa medidas de classificação binárias para avaliar seus classificadores (Masud et al., 2011a) (Farid e Rahman, 2012), (Al-Khateeb et al., 2012a), como a porcentagem de exemplos das novas classes classificados incorretamente como classes conhecidas ($MNew$) e a porcentagem de exemplos das classes conhecidas falsamente identificados como novas classes ($FNew$). Entretanto, essas medidas não podem expressar o problema adequadamente, especialmente quando diferentes padrões-novidade aparecem ao longo do tempo.

Em DN, é comum o desenvolvimento de classificadores com a opção de rótulo *desconhecido* (Spinosa et al., 2009), (Hayat e Hashemi, 2010), (Masud et al., 2011a), (Al-Khateeb et al., 2012a). Os exemplos *desconhecidos* são usados para modelar padrões-novidade ou extensões dos conceitos conhecidos . Assim, a presença destes exemplos na matriz de confusão deve ser considerada na avaliação de algoritmos para DN.

É válido ressaltar que em tarefas de DN em FCD a matriz de confusão é incremental, pois o número de linhas e colunas não é fixo. Além disso, a matriz de confusão não é quadrada, o que indica que o número de padrões-novidade, detectados por um algoritmo, não é necessariamente igual ao número de classes novidade.

Neste trabalho, uma nova metodologia de avaliação para problemas de DN envolvendo FCD multiclasse é proposta. A metodologia proposta também pode ser usada em problemas de DN binários, uma vez que esses podem ser considerados um caso específico de problemas multiclasse. Ela também pode ser usada em cenários nos quais a atualização do modelo de decisão é feita tanto com *feedback* externo quanto sem *feedback*.

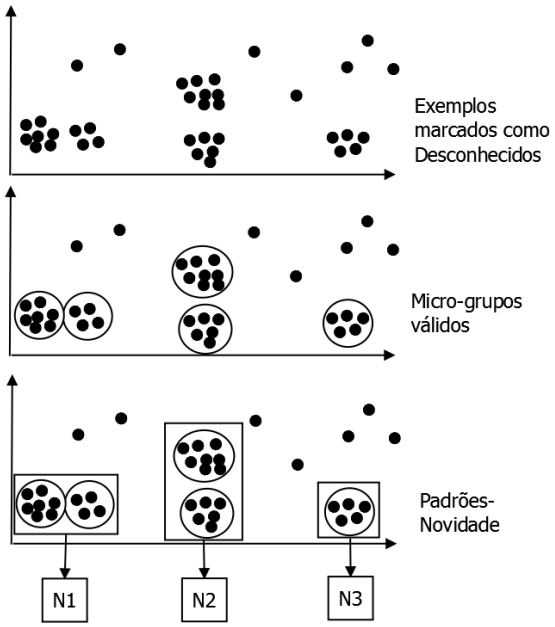


Figura 5.1: Tarefa para DN a partir de exemplos *desconhecidos* proposta pelo MINAS.

5.3 Formalização do Problema

Esta seção apresenta uma formalização para o problema de DN multiclasse em FCD. A formalização aqui apresentada é diferente do que tem sido apresentado na literatura, por acrescentar outros importantes aspectos do processo, como matriz incremental, classificadores com opção de rótulo *desconhecido* e definição do conceito novidade formado por diferentes padrões-novidade.

Os algoritmos para DN em FCD objetivam identificar padrões-novidade a partir de exemplos não rotulados. Em geral, a maioria dos algoritmos da literatura usam classificadores com opção de rótulo *desconhecido*, no qual exemplos não explicados pelo modelo de decisão são marcados temporariamente como *desconhecidos*. Esses exemplos são usados para modelar padrões-novidade que serão usados para classificar novos exemplos.

O algoritmo MINAS, proposto neste trabalho, aplica uma técnica de agrupamento nos exemplos desconhecidos, produzindo novos micro-grupos. Os micro-grupos são avaliados e aqueles considerados inválidos são descartados. Para os micro-grupos válidos, é necessário decidir se eles representam uma extensão ou um padrão-novidade, de acordo com a distância entre o novo micro-grupo e o micro-grupo mais próximo a ele do modelo de decisão. Assim, um padrão-novidade pode ser composto por um ou mais micro-grupos. Na Figura 5.1 é ilustrada a tarefa de DN pelo algoritmo MINAS.

Como a tarefa de DN é realizada a partir de exemplos não rotulados, os padrões-novidade detectados pelos algoritmos não possuem uma associação direta com as classes do problema, sendo que uma classe do problema pode ser representada por um mais padrões-novidade pelo algoritmo de DN.

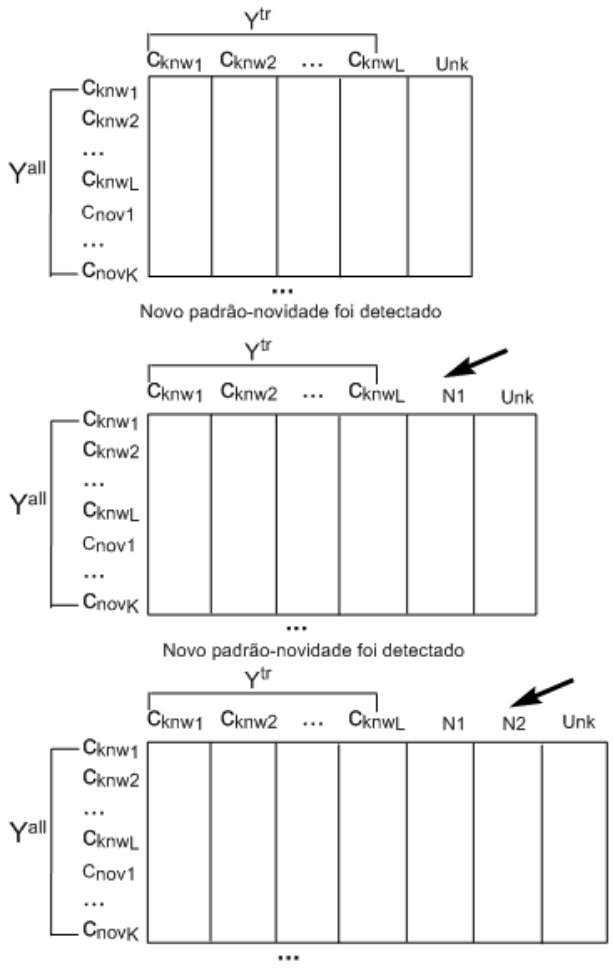


Figura 5.2: Evolução da matriz de confusão ao longo do tempo no algoritmo MINAS.

Além disso, a matriz de confusão gerada pelos algoritmos de DN não é quadrada e o número de colunas aumenta sempre que um novo padrão-novidade é detectado. Na Figura 5.2 é mostrado, um exemplo de matriz de confusão gerada pelo algoritmo MINAS. Cada linha da matriz de confusão representa uma classe do problema, classes conhecidas ($C_{knw_1}, C_{knw_2},..., C_{knw_L}$) e classes novidades($C_{nov_1}, C_{nov_2},..., C_{nov_K}$), e cada coluna representa uma das classes preditas pelo algoritmo. As colunas $C_{knw_1}, C_{knw_2},..., C_{knw_L}$ correspondem às classes aprendidas durante a fase *offline* e as colunas N_1,N_2,... correspondem aos padrões-novidade aprendidos na fase *online*. A última coluna corresponde aos exemplos marcados com o perfil *desconhecido*. Os padrões-novidade são sequencialmente rotulados como N_1, N_2, ...

É importante destacar que os padrões-novidade detectados pelo algoritmo não possuem uma relação direta com as classes do problema. Uma dada classe do problema pode ser associada a um ou mais padrões-novidade e uma classe particular pode não ser detectada pelo algoritmo.

A fim de usar a matriz de confusão mostrada na Figura 5.2, cinco requisitos devem ser considerados:

1. Uma classe pode ser representada por dois ou mais padrões-novidade, assim é possível ter mais padrões-novidade do que classes do problema;

2. O algoritmo pode detectar menos padrões-novidade do que o número de classes novidade. Isso acontece se o algoritmo não conseguiu distinguir os exemplos de todas as classes novidade;

3. Presença de exemplos não explicados pelo modelo de decisão atual e rotulados pelo algoritmo como *desconhecido*;

4. Cenário multiclasse. O cálculo de medidas de acurácia e erro deve considerar as diferentes classes aprendidas nas fases *offline* e *online*, sendo que esse cenário é mais difícil de ser tratado do que simplesmente distinguir entre conceitos normal e novidade;

5. Representação da matriz de confusão pode variar ao longo do tempo.

Para lidar com os dois primeiros requisitos, é necessário decidir como associar padrões-novidade a classes do problema. A Seção 5.5.1 explica a abordagem proposta neste trabalho para atender a esses requisitos. Uma alternativa para lidar com o terceiro requisito é discutido na Seção 5.5.2. A abordagem para tratar o quarto requisito é proposta na Seção 5.5.3. Por último, a Seção 5.5.4 descreve como o último requisito pode ser tratado. A próxima seção apresenta como os trabalhos da literatura relacionados a DN têm sido avaliados.

5.4 Trabalhos Relacionados

5.4.1 Algoritmos e Avaliação de DN em FCD

Muitos algoritmos têm sido propostos para lidar com DN em FCD. Entretanto, pouca atenção tem sido dada para uma adequada avaliação desses algoritmos.

OLINDDA (Spinosa et al., 2008) é um algoritmo que considera a DN como uma tarefa de classificação com uma única classe. Três submodelos são criados (normal, extensão e novidade) e usados para classificar novos exemplos. O classificador usado no OLINDDA trabalha com a opção de rótulo *desconhecido*. Para a avaliação do OLINDDA, os autores analisam como os exemplos pertencentes às diferentes classes do problema são classificados:

- Pertencentes ao conceito normal;
- Pertencentes ao conceito extensão;
- Pertencentes ao conceito novidade ou *desconhecido*.

Outras medidas são usadas, como a porcentagem de exemplos de novas classes classificados incorretamente como conceito normal ($MNew$) e a porcentagem de exemplos do conceito normal incorretamente classificados como conceito novidade ou extensão ($FNew$) (ver equações 3.11 e 3.12, na seção 3.7.3).

A matriz de confusão gerada pelo OLINDDA é mostrada na Figura 5.3. A primeira linha representa os exemplos do conceito normal ou classe negativa, os quais estão associados à classe C_1. O bloco de linhas seguintes é nomeado "outras classes" ou "classes positivas", representadas pelas classes C_2 até C_M, sendo M o número de classes observadas ao longo do FCD. As células representam quantos exemplos de cada classe foram classificados como pertencentes à classe negativa (normal) ou classe positiva (extensão, novidade ou *desconhecido*). Assim, a matriz de confusão representa um problema de classificação binário, com exemplos negativos (conceito normal) e exemplos positivos (outros conceitos). É importante observar que OLINDDA, não distingue entre diferentes classes que compõem o modelo normal, nem entre os diferentes padrões-novidade que compõem o modelo novidade.

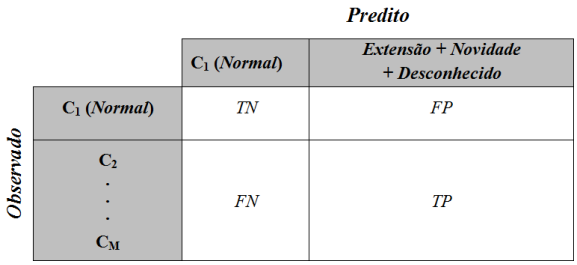


Figura 5.3: Matriz de Confusão do OLINDDA e DETECTNOD.

O algoritmo DETECNOD (Hayat e Hashemi, 2010) também considera a DN uma tarefa de classificação com uma classe e o classificador usado trabalha com a opção de rótulo *desconhecido*. Assim como OLINDDA, DETECNOD não diferencia entre os diferentes padrões-novidade aprendidos *online* nem entre as diferentes classes que compõem o modelo normal. A estratégia de avaliação, bem como a matriz de confusão gerada, são similares às utilizadas no OLINDDA.

ECSMiner (Masud et al., 2011a), MCM (Masud et al., 2010a) e CLAM (Al-Khateeb et al., 2012a) são algoritmos que consideram que a DN é uma tarefa de classificação multiclasse. Os classificadores usados nesses algoritmos possuem a opção de rótulo *desconhecido*. O conceito conhecido do problema é composto por diferentes classes, assim como diferentes novas classes podem surgir ao longo do FCD. Todos esses algoritmos supõem que, para a atualização do modelo de decisão, todos os rótulos verdadeiros dos exemplos estarão disponíveis. Além disso, eles supõem que, em cada bloco de dados, somente uma classe novidade aparece por vez. Se mais que uma classe aparecer no mesmo bloco de dados, os exemplos serão simplesmente rotulados como novidade, não sendo possível separar entre os diferentes padrões-novidade que compõem a novidade.

A matriz de confusão obtida pelo ECSMiner, MCM e CLAM é ilustrada na Figura 5.4. Nessa figura, as linhas correspondem às classes observadas ao longo do FCD e as colunas às classes preditas pelo algoritmo. As classes C_1 até C_M representam as classes conhecidas do problema, que foram aprendidas na fase inicial de treinamento ou na fase de atualização *online* do modelo, executada ao longo do FCD, quando o rótulo verdadeiro dos novos exemplos é obtido. O conceito conhecido do problema é composto por uma ou mais classes. As classes C_{M+1} e C_{M+2} representam duas classes novidade observadas no bloco de dados atual. O algoritmo associa um novo exemplo a uma das classes conhecidas, para o qual um classificador já foi treinado, ou rotula-o com o perfil *desconhecido*. Grupos de exemplos *desconhecidos* formam uma novidade. A diagonal principal da matriz representa os acertos nas classes conhecidas (Tp). Para os exemplos das novas classes, um acerto significa classificá-lo como novidade. Nesta matriz de confusão, FN e FP seguem a mesma definição dada anteriormente, e FE é o erro de classificação nas classes conhecidas.

Observado \ *Predito*	C_1	C_2	...	C_M	*Novidade*
C_1	*TN*	*FE*	*FE*	*FE*	*FP*
C_2	*FE*	*TN*	*FE*	*FE*	*FP*
⋮	⋮	⋮	⋮	⋮	⋮
C_M	*FE*	*FE*	*FE*	*TN*	*FP*
C_{M+1}	*FN*	*FN*	*FN*	*FN*	*TP*
C_{M+2}	*FN*	*FN*	*FN*	*FN*	*TP*

Figura 5.4: Matriz de Confusão do ECSMiner, MCM e CLAM

O algoritmo ECSMiner, MCM e CLAM, bem como os algoritmos propostos em (Farid e Rahman, 2012), (Farid et al., 2013) (Masud et al., 2010a) usam medidas de avaliação como $Mnew$, $Fnew$ e a porcentagem total de erros de classificação – Err – (ver Equação 5.1). Entretanto, estas medidas não conseguem avaliar apropriadamente cenários onde o conceito novidade é composto por diferentes classes.

$$Err = \frac{(FP + FN + FE) * 100}{N} \tag{5.1}$$

5.4.2 Medidas de Avaliação para Classificação com Opção de Rejeição

Em classificação com opção de rejeição (Pillai et al., 2011), (Nadeem et al., 2010), (Marrocco et al., 2007), (Tax e Duin, 2008), um exemplo é rejeitado se a sua classe não pode

ser predita com confiança (Nadeem et al., 2010). Neste caso, é considerado melhor rejeitar o exemplo do que cometer um erro de classificação. Para essas situações, as taxas de acurácia e erro podem ser calculadas considerando todos os exemplos ou apenas os exemplos aceitos pelo classificador. Em ambos os casos, as seguintes propriedades podem ser verificadas (Hanczar e Dougherty, 2008).

$$p(aceitacao) = 1 - p(rejeicao) \tag{5.2}$$

$$p(f(x) = y) + p(f(x) \neq y) + p(rejeicao) = 1 \tag{5.3}$$

$$p(f(x) = y|aceitacao) + p(f(x) \neq y|aceitacao) = 1 \tag{5.4}$$

Na Equação 5.2, a taxa de rejeição é a probabilidade de um dado classificador rejeitar um novo exemplo e a taxa de aceitação é a probabilidade de um dado classificador aceitar o exemplo. As taxas de aceitação e rejeição são complementares. A Equação 5.3 considera que os exemplos rejeitados não são nem erros, nem acertos, mas são computados separadamente. A Equação 5.4 apresenta a probabilidade de cometer um erro de classificação, dado que o classificador tenha aceitado o exemplo.

Em Chow (1970), o autor define o compromisso entre a taxa de rejeição e a taxa de erro, considerando que a taxa de erro decresce monotonicamente com o aumento da taxa de rejeição. O uso das curvas rejeição-acurácia (ARCs, do inglês, *Accuracy Rejection Curves*) para comparar o desempenho preditivo de classificadores considerando diferentes taxas de rejeição é investigado em (Nadeem et al., 2010). ARCs resultam da plotagem da acurácia de um classificador como uma função da sua taxa de rejeição, ambos variando entre 0 e 1 (100 %). Todas as ARCs tem uma acurácia de 100% para uma rejeição de 100% (convergência no ponto (1,1)). O primeiro ponto tem coordenadas em (0,*a*), onde *a* corresponde a acurácia do classificador quando ele não rejeita qualquer exemplo. Em Marrocco et al. (2007), curvas de erro-rejeição são usadas para comparar diferentes métodos para classificação com opção de rejeição.

5.4.3 Medidas de Avaliação para Problemas Multiclasse

Diversos problemas de classificação em FCD envolvem mais que duas classes. Apesar dos estudos em DN em FCD não terem explorado o uso de medidas de avaliação multiclasse, essa questão já foi bastante estudada em contextos *batch*.

Uma proposta para avaliar o desempenho preditivo de um classificador multiclasse é dividir a matriz de confusão original $M \times M$ em M matrizes binárias um-contra-todos, uma por classe (ver Figura 5.5). Para cada classe C_i, TP_i é o número de exemplo da classe C_i corretamente classificados, FP_i é o número de exemplos da classe $C_j(j =$

$1,..., M, j \neq i$) incorretamente classificados na classe C_i, FN_i é o número de exemplos da classe C_i incorretamente classificados na classe C_j, e TN_i é o número de exemplos da classe $C_j (j = 1,..., M, j \neq i)$ corretamente classificados na classe C_j.

Uma medida aplicada para problemas de classificação multiclasse, definido na Equação 5.8, é o erro combinado (*CER - Combined Error Rate*), o qual é dado pela média ponderada de falsos positivos e falsos negativos por classe (ver Equação 5.7) (Rosenberg, 2009). Nessas Equações, $\#ExC_i$ representa o número de exemplos da classe C_i e $\#Ex$ representa o número total de exemplos. Outra medida de erro usada é o $Erro\_Medio$, que calcula a média dos erros cometidos por cada classe (ver Equação 5.9) (Sokolova e Lapalme, 2009).

Uma medida de acerto usada para avaliar o desempenho preditivo de um algoritmo de classificação é a F-measure, definida como a média harmônica ponderada entre a precisão e revocação (ver equação 5.6). Essa medida pode ser adaptada para tarefas de classificação multiclasse, como proposto em (Özgür et al., 2005). Nesse caso, a média da F1-measure (F-measure com $\alpha = 1$) para cada classe é calculada.

As medidas apresentadas nesta seção podem ser usadas para avaliar algoritmos de classificação multiclasse e são adequadas para tratar classes desbalanceadas. Entretanto, elas não levam em consideração os exemplos marcados com o perfil *desconhecido*.

		Predito	
		$\mathbf{C_i}$	$\mathbf{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_M}$
Observado	$\mathbf{C_i}$	TP_i	FN_i
	$\mathbf{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_M}$	FP_i	TN_i

Figura 5.5: Matriz de confusão para classificação multiclasse.

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \qquad \rho_i = \frac{TP_i}{TP_i + FN_i} \tag{5.5}$$

$$F_i = \frac{(1+\alpha)\pi_i\rho_i}{\alpha\pi_i + \rho_i} \qquad F - macro = \frac{\sum_{i=1}^{M} F_i}{M} \tag{5.6}$$

$$FPR_i = \frac{FP_i}{FP_i + TN_i} \qquad FNR_i = \frac{FN_i}{FN_i + TP_i} \tag{5.7}$$

$$CER = \frac{\sum_{i=1}^{M} \frac{\#ExC_i}{\#Ex} FPR_i + \sum_{i=1}^{M} \frac{\#ExC_i}{\#Ex} FNR_i}{2} \tag{5.8}$$

$$Erro\_Medio = \frac{\sum_{i=1}^{M} \frac{FP_i + FN_i}{FP_i + FN_i + TP_i + TN_i}}{M} \tag{5.9}$$

5.5 Metodologia Proposta

A seguir, será apresentada a metodologia usada para avaliar a matriz de confusão gerada em problemas envolvendo DN em FCDs.

5.5.1 DN: uma Matriz de Confusão Retangular

Considerando que a tarefa de DN usa dados não rotulados, os padrões-novidade detectados pelo algoritmo não possuem uma ligação direta com as classes do problema. Assim, para calcular medidas como acurácia e erro, é necessário associar os padrões-novidade detectados pelo algoritmo às classes do problema. Nesse processo, todo padrão-novidade deve ser associado a uma classe e uma classe pode ser associada a um ou mais padrões-novidade. Esse problema pode ser formalizado como um grafo bipartido.

Definição 5.1. Grafo Bipartido: Um grafo pode ser representado por $G(V,E)$, onde V é o conjunto de vértices e E é o conjunto de arestas no grafo. Um grafo $G(V,E)$ é bipartido em dois conjuntos de vértices X e Y, se $V = X \cup Y$ com $X \cap Y = \emptyset$ e cada aresta em E tem um ponto final em X e um ponto final em Y. Se, para cada $v_i \in X$, $v_j \in Y$, $\{v_i,v_j\} \in E$, o grafo é nomeado **completo**.

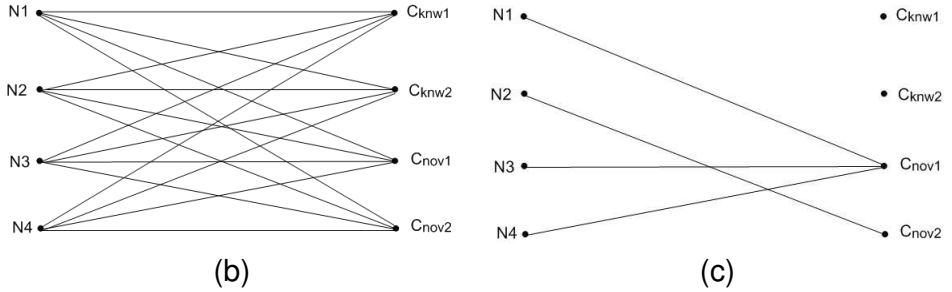
Definição 5.2. Grafo Bipartido Ponderado: Um grafo bipartido $G(V,E)$ é ponderado se cada aresta $\{v_i,v_j\} \in E$ tem um peso associado $w_{ij} \geq 0$.

No contexto de DN em cenários multiclasse, X representa os padrões-novidade preditos pelo algoritmo, Y representa as classes do problema, e w_{ij} é o número de exemplos da classe j classificados na classe i. Nas figuras 5.6 (a) e 5.6 (b) é mostrado um exemplo de matriz de confusão e o seu correspondente grafo bipartido. O peso associado a cada aresta é omitido para simplificar a figura.

O objetivo é encontrar um subgrafo bipartido $G'(V,E')$, no qual cada vértice em X tem grau um. Nesse caso, $|E'|$, o número de arestas no grafo G', é igual a $|X|$, número de

	C_{knw1}	C_{knw2}	N1	N2	N3	N4	Desconhecido
C_{knw1}	6000	2900	0	800	50	0	100
C_{knw2}	2000	5200	700	0	30	0	50
C_{nov1}	300	450	1950	100	100	50	50
C_{nov2}	200	300	1900	1400	100	20	100

(a)



(b) (c)

Figura 5.6: Exemplo de uma matriz de confusão e o seu correspondente gráfico bipartido; (a) Matriz de confusão; (b) Grafo bipartido correspondente e (c) Subgrafo bipartido resultante, representando a associação entre padrões-novidade e classes do problema.

elementos em X. Assim, é necessário associar cada padrão-novidade a uma das classes do problema. Entretanto, há muitas maneiras para calcular esse novo subgrafo G'. Para cada padrão-novidade $x_i \in X$, há $|Y|$ diferentes maneiras possíveis de fazer uma associação entre x_i e uma classe do problema $y_i \in Y$, onde $|Y|$ é o número de classes reais do problema. O número de possíveis combinações para associar cada elemento em X a um elemento em Y é portanto $|Y|^{|X|}$.

Definição 5.3. ***Matching***: Um *matching* para um grafo bipartido $G(V,E)$ é um subconjunto $M \subseteq E$, tal que todo vértice em V tem no máximo uma aresta de M incidindo sobre ele.

Definição 5.4. ***Matching* Perfeito**: Um *matching* perfeito para um grafo bipartido $G(V,E)$ é um subconjunto $M \subseteq E$, tal que todo vértice em V tem exatamente uma aresta de M incidindo nele.

A abordagem proposta neste trabalho para resolver o problema é baseado no método Húngaro (Kuhn, 1955), um algoritmo combinatorial para resolver problemas de associações. Entretanto, ele não pode ser diretamente usado porque, em teoria dos grafos, o *matching* é uma correspondência um-para-um (ver definição 5.3). Em problemas de *matching* em DN, uma classe do problema pode ser representada por um ou mais padrões-novidade. Em termos de grafo bipartido, o objetivo é encontrar o *matching* perfeito com custo mínimo. Para superar essa restrição, é escolhido, para cada elemento $x \in X$, a aresta w_{ij} com o maior peso, indicando que o elemento x_i está associado com o elemento y_j. Em caso de empate, qualquer um deles pode ser escolhido. Na Figura 5.6 (c) é mostrado o subgrafo

G', resultante da associação entre padrões-novidade e classes do problema para a matriz de confusão mostrada na Figura 5.6 (a).

É importante observar que todo elemento em X tem um correspondente em Y, mas a recíproca não é verdadeira. A Figura 5.6 (c) pode ser descrita da seguinte forma. O algoritmo proposto associou três padrões-novidade à classe C_{nov_1} e um padrão-novidade à classe C_{nov_2}. Para as classes induzidas na fase *offline* (C_{knw_1} e C_{knw_2}), o algoritmo não associou nenhuma novidade para representá-las. Assim, se as classes aprendidas na fase *offline* evoluíram ao longo do tempo, o classificador identificou essas mudanças como extensão dos conceitos conhecidos, ao invés de padrões-novidade.

5.5.2 Tratamento para os Exemplos com o Perfil *Desconhecido*

Uma importante questão a ser tratada é a presença de exemplos marcados com o perfil *desconhecido*, presentes na matriz de confusão. Neste trabalho, os exemplos *desconhecidos* não são considerados nem acertos, nem erros, são computados separadamente. É importante destacar que, neste trabalho, será considerado que $ACC + Err + UnkR = 1$, onde ACC é a taxa de exemplos corretamente classificados, Err é a taxa de exemplos incorretamente classificados, e $UnkR$ é a taxa de exemplos classificados com o perfil *desconhecido*.

Assim como proposto pelos classificadores com rejeição (Nadeem et al., 2010), este trabalho usa a abordagem de computar medidas como erro e acurácia usando somente exemplos explicados pelo modelo. Assim, ACC_{Exp} + Err_{Exp} = 1, onde ACC_{Exp} e Err_{Exp} são as taxas de acurácia e erro, respectivamente, considerando somente os exemplos explicados pelo modelo. Assim, a medida FN_i é o número de exemplos da classe i incorretamente classificados em outra classe, excluindo os exemplos classificados como desconhecidos.

A fim de verificar como o número de exemplos desconhecidos variam ao longo do tempo, a taxa de exemplos desconhecidos é calculada para cada classe, e então, a médias dessas taxas é calculada, de acordo com a Equação 5.10.

$$UnkR = \frac{\sum_{i=1}^{M} \frac{\#Unk_i}{\#ExC_i}}{M} \tag{5.10}$$

5.5.3 Adaptação das Medidas de Avaliação Multiclasses

Após fazer a associação entre padrões-novidade a classes do problema e calcular a taxa de exemplos *desconhecidos*, as medidas de avaliação encontradas na literatura, como por exemplo o CER (ver Equação 5.8) e o $Erro\_Medio$ (ver Equação 5.9) podem ser usadas para expressar os erros de classificação de um algoritmo de aprendizado, considerando apenas os exemplos classificados pelo sistema, os exemplos que não foram marcados

como *desconhecidos*. Assim, nessas equações $\#ExC_i$ representa o número de exemplos da classe C_i, $\#Ex$ representa o número total de exemplos e FPR_i e FNR'_i representam a taxa de falsos positivos e a taxa de falsos negativos para classe C_i, respectivamente, todos calculados desconsiderando os exemplos *desconhecidos*.

É importante destacar que como a tarefa de DN é considera neste trabalho como multi-classe, faz-se necessário o uso de medidas de avaliação também multiclasse.

5.5.4 Avaliação ao Longo do Tempo

Em DN em FCD, assim que novos dados chegam ao longo do tempo e novas classes aparecem, desaparecem ou evoluem, não é suficiente calcular somente uma matriz de confusão ao fim do conjunto de teste. É necessário avaliar as diferentes matrizes de confusão geradas ao longo do tempo e verificar como o classificador adapta-se a cenários não-estacionários.

A matriz de confusão, ilustrada na Figura 5.2, pode ser facilmente mantida incrementalmente. Sempre que um novo exemplo chega, a matriz de confusão é incrementada e as medidas de avaliação são atualizadas. Entretanto, em termos de custo computacional, não é interessante atualizar essas medidas cada vez que um novo exemplo chega. Assim, as medidas de avaliação são calculadas após um dado período de tempo, mas a matriz de confusão é incrementada toda vez que um novo exemplo chega.

Um possível modo de verificar o comportamento do classificador ao longo do tempo é construir um gráfico 2D, no qual o eixo X representa o tempo de chegada dos dados e Y representa os valores das medidas de avaliação. Nesse gráfico, é importante plotar uma medida representando a taxa de exemplos *desconhecidos* em comparação com uma ou mais medidas de acurácia ou erro. Adicionalmente, é importante destacar nesse gráfico os pontos nos quais houve detecção de novos conceitos pelo algoritmo.

5.6 Estudo da Aplicação da Metodologia Proposta em Cenários-problema

A aplicação das medidas de avaliação da literatura na matriz de confusão apresentada na Figura 5.2 pode produzir resultados indesejados. Dois cenários, nos quais resultados indesejados podem ser obtidos, são descritos a seguir.

O primeiro cenário, ilustrado na Figura 5.7, mostra uma matriz de confusão de um classificador que resulta em acurácia elevada e erro de classificação baixo se a coluna *desconhecido* não é considerada. Entretanto, para calcular medidas como $Fnew$, $Mnew$ e Err, é necessário decidir se os exemplos *desconhecidos* são considerados acertos ou erros. Se os exemplos *desconhecidos* são considerados erros, o desempenho preditivo do

classificador decai. O tratamento de exemplos *desconhecidos* tem um efeito direto no cálculo de medidas de avaliação. Este trabalho considera que a classificação de um exemplo *desconhecido* não deve ser considerado um erro ou acerto. Assim, este trabalho propõe fazer a análise preditiva do desempenho do classificador baseado na matriz de confusão dividida em duas partes, com o desempenho avaliado separadamente em cada uma dessas partes. Na primeira parte, a medida de avaliação é calculada para toda a matriz sem a coluna *desconhecidos*. Na segunda parte, o desempenho preditivo dos exemplos *desconhecidos* é medido considerando todas as classes.

	C_{knw1}	C_{knw2}	N1	N2	Desconhecido
C_{knw1}	7500	0	0	0	2500
C_{knw2}	0	7000	0	0	1000
C_{nov1}	100	100	1300	0	1500
C_{nov2}	50	50	100	2800	1000

Figura 5.7: Detecção de múltiplos padrões-novidade: Cenário 1.

O segundo cenário problema é mostrado na Figura 5.8. Nesse caso, o algoritmo identifica quatro padrões-novidade. Considerando as medidas de desempenho da literatura, os exemplos das classes C_{nov1} e C_{nov2} classificados como C_{unk1} e C_{unk2} serão computados como erro, assim como os exemplos das classes C_{knw1} e C_{knw2} classificados como $N1$, $N2$, $N3$ e $N4$. Entretanto, o número de erros é subestimado. Por exemplo, 1.450 exemplos da classe C_{nov1} foram classificados como $N1$ e 1.000 exemplos da classe C_{nov2} foram classificados como $N1$. É necessário decidir se $N1$ deve ser associado com C_{nov1} ou C_{nov2}. De acordo com a metodologia proposta neste trabalho, para associar padrões-novidades às classes do problema, o padrão-novidade $N1$ será associado à classe C_{nov1}. Assim, os exemplos das classes C_{knw1}, C_{knw2} e C_{nov2} classificados como $N1$ serão considerados incorretamente classificados.

	C_{knw1}	C_{knw2}	N1	N2	N3	N4	Desconhecido
C_{knw1}	6000	2900	500	500	0	0	100
C_{knw2}	200	500	400	550	100	0	50
C_{nov1}	500	1000	1450	0	500	300	50
C_{nov2}	400	1000	1000	1500	50	50	100

Figura 5.8: Detecção de múltiplos padrões-novidade: Cenário 2.

5.7 Considerações Finais

Esse capítulo apresentou uma nova metodologia para avaliar a matriz de confusão gerada pelos algoritmos de DN em FCDs. A matriz de confusão gerada não é quadrada, o número de colunas aumenta ao longo do tempo, a matriz contém colunas que representam os padrões-novidade detectados na fase *online*, que não estão associados às classes do problema e a matriz contém uma coluna com os exemplos rotulados como *desconhecidos*. A metodologia proposta sugere associar os padrões-novidade às classes do problema,

tornando assim a matriz quadrada, o uso de medidas de classificação multiclasse e o cálculo de medidas para avaliar a presença de exemplos marcados como desconhecidos.

O próximo capítulo apresenta os experimentos realizados usando o algoritmo MINAS, comparando-o com algoritmos encontrados na literatura. Finalmente, serão apresentados e discutidos os valores usados para cada um dos parâmetros dos algoritmos.

CAPÍTULO

6

Experimentos

6.1 Introdução

Nos capítulos anteriores foram apresentados o algoritmo MINAS e uma nova metodologia de avaliação para DN em FCDs. Este capítulo descreve os detalhes sobre os experimentos realizados neste trabalho. Primeiramente, serão apresentadas as bases de dados artificiais e reais usadas. A seguir, serão detalhadas as configurações dos algoritmos usados nos experimentos. Em seguida, serão apresentados os experimentos, bem como uma discussão sobre os resultados obtidos. Por último, será mostrado um cenário real para a aplicação do algoritmo MINAS, que consiste no reconhecimento de atividades humanas usando acelerômetros. Os principais trabalhos da literatura que lidam com essa tarefa, bem como os resultados obtidos usando o algoritmo MINAS são também mostrados.

6.2 Bases de Dados e Pré-processamentos

Para os experimentos realizados neste capítulo foram utilizadas bases de dados artificiais e reais. A Tabela 6.1 mostra um resumo das principais características das bases de dados utilizadas nos experimentos.

6.2.1 Bases de Dados Artificiais

Três bases de dados artificias foram usadas nos experimentos, MOA, SynD e SynEDC. As bases SynD e SynEDC foram escolhidas pois são utilizadas em alguns trabalhos da

Tabela 6.1: Resumo das bases de dados usadas nos experimentos.

Base de Dados	Atributos	Exemplos	Classes	Classes no Treino
MOA	4	100.000	4	2
SynD	10	250.000	2	2
SynEDC	40	400.000	20	7
KDD	34	490.000	5	2
CoverType	54	581.000	7	3

literatura. A base MOA foi criada a fim simular características dos FCD, como mudança de conceito e aparecimento/desparecimento das classes do problema. A base de dados MOA pode ser gerada usando a ferramenta MOA, disponível em: http://moa.cms.waikato.ac.nz/. As bases de dados artificiais SynD e SynEDC foram usadas nos trabalhos de Masud et al. (2011a), Al-Khateeb et al. (2012a) e Al-Khateeb et al. (2012b) e estão disponíveis em: http://dml.utdallas.edu/Mehedy/index_files/Page675.html. A seguir, essas três bases serão detalhadas.

- **MOA**:
  Essa base, aqui nomeada MOA, foi criada usando um gerador de funções de base radial, disponível na ferramenta MOA – *Massive Online Analysis* – (Bifet et al., 2010). Um número fixo de centróides aleatórios foi gerado, cada um com uma posição aleatória e um rótulo de classe também aleatório. Todos os centros possuem o mesmo desvio padrão inicial (isto é, o raio), o qual pode variar ao longo do tempo. Isso cria uma hiperesfera com distribuição normal dos exemplos ao redor de cada centro com densidades que podem variar. Mudanças de conceito são introduzidas movendo os centróides com velocidade constante. O gerador foi configurado para introduzir no máximo duas novas classes ao longo do fluxo. A cada 30.000 exemplos ocorre um evento de aparecimento/desaparecimento de uma classe e os grupos são movidos com uma distância de 0,01 a cada 1.500 exemplos. A base de dados gerada tem 100.000 exemplos.

- **SynD - *Synthetic data with only concept-drift*:**
  Essa base simula somente mudança de conceito, sem presença de novas classes. A base foi gerada a partir de hiperplanos que se movem, sendo composta de apenas duas classes. Cada exemplo é um vetor de 10 dimensões, sendo que o valor de cada atributo está no intervalo $[0,1]$. Ruído é introduzido na base de forma aleatória, mudando o rótulo de 5% dos exemplos. A base possui 250.000 instâncias

- **SynEDC - *Synthetic data with concept-drift and novel-class*:**
  Essa base contém mudança de conceito e surgimento de novas classes ao longo do FCD. A base é composta de 20 classes, 40 atributos e 400.000 instâncias. Os exemplos pertencentes a cada classe são gerados usando uma distribuição gaussiana com diferentes médias (-5.0 a +5.0) e variâncias por classes (0.5 a 6). Além

disso, a distribuição de probabilidade varia ao longo do tempo, assim algumas classes aparecem e desaparecem ao longo do tempo. A fim de inserir mudança de conceito, os valores médios de uma certa porcentagem dos atributos são alterados constantemente. Os valores de todos os atributos estão no intervalo $[0,1]$.

6.2.2 Bases de Dados Reais

As bases de dados reais usadas neste trabalho estão disponíveis no repositório de dados UCI (Frank e Asuncion, 2010). Essas bases foram escolhidas, pois elas são usadas na maioria dos trabalhos que envolvem DN em FCD.

- **Base de Dados KDD Cup 99 - Detecção de Intrusão:**
  A base de dados KDDCup 99 para detecção de intrusos em uma rede de computadores é um importante conjunto de dados para experimentos em FCD. Ela corresponde a um problema real de detecção automática e em tempo real de ciberataques. A base de dados usadas nos experimentos corresponde a versão 10%, contendo 490.000 exemplos, que é uma versão resumida da base original. Cada exemplo representa a informação sobre uma conexão na rede, classificado como normal ou como um dos 22 diferentes tipos de ataque, que podem ser condensados em 4 classes. Cada exemplo dessa base de dados contém 42 atributos. Somente os 34 atributos numéricos dessa base de dados foram usados, sendo que todos foram re-escalados para o intervalo $[0,1]$.

- **Base de Dados *Covertype* - Cobertura de Floresta:**
  Esta base tem como objetivo predizer o tipo de cobertura de floresta a partir de informações cartográficas como elevação, declive, tipo de solo, etc.. A base de dados contém informação sobre sete diferentes tipos de cobertura de florestas e possui cerca de 581.000 exemplos. Cada exemplo é composto por 54 atributos numéricos. Todos os atributos foram usados e re-escalados para o intervalo $[0,1]$.

6.2.3 Técnica de Amostragem

Não há uma padrão na literatura sobre a técnica de amostragem usada em experimentos envolvendo DN em FCD. A técnica de amostragem de dados utilizada neste trabalho usa 10% da base de dados na fase de treinamento (*offline*) e o restante dos dados para teste. A ordem dos exemplos não foi alterada, sendo mantida a ordem da base de dados original.

O conjunto de treinamento da base de dados MOA (10% da base original) contém exemplos de duas classes do problema e o conjunto de teste contém exemplos das 4 classes. Tanto o conjunto de treinamento quanto o de teste da base SynD contém exemplos das duas classes do problema. O conjunto de treinamento da base SynEDC contém

exemplos de 6 classes do problema e o conjunto de teste contém exemplos de todas as 20 classes do problema.

Para criar o conjunto de treinamento da base KDD, 10% dos dados da base foram usados, mas somente os exemplos das classes normal e um dos ataques (*dos*) foram mantidos. Assim, o conjunto de treinamento contém exemplos de 2 classes do problema e o conjunto de teste contém exemplos das 5 classes do problema.

Para criar o conjunto de treinamento da base Covertype, 10% dos dados da base foram usados, mas somente os exemplos de 3 classes forma mantidos. Essa base de dados será aqui nomeada de Covertype-V1. Uma outra versão da base de dados de cobertura de floresta também foi usado e será aqui nomeada por Covertype-V2. Essa base é a mesma Covertype, no entanto, os dados são rearranjados de modo que em cada janela ocorram no máximo 3 e no mínimo 2 classes. Essa base foi usada nos experimentos realizados por Masud et al. (2011a) e está disponível em: http://dml.utdallas.edu/Mehedy/index_files/Page675.html.

6.3 Algoritmos Utilizados e suas Configurações

Para os experimentos, quatro algoritmos para DN em FCDs foram utilizados. São eles: MINAS, o algoritmo proposto nesta tese, OLINDDA[1] (Spinosa et al., 2009), ECSMiner[2] (Masud et al., 2011a) e CLAM[3] (Al-Khateeb et al., 2012a). Para a execução de cada um desses algoritmos uma série de parâmetros devem ser configurados. A Tabela 6.2 sumariza os parâmetros de cada um dos algoritmos.

Foram propostas modificações nos algoritmos originais usados nos experimentos, gerando assim novas versões dos algoritmos. Tanto as versões originais dos algoritmos, quanto as versões modificadas utilizam os mesmos parâmetros. Na Tabela 6.3 é mostrado um resumo dos nomes dos algoritmos usados nos experimentos com uma descrição sumária de suas características. A seguir serão descritos os principais aspectos de cada um dos algoritmos bem como suas versões modificadas são apresentados.

6.3.1 ECSMiner – versão com *feedback* (ECSMiner-CF)

Essa é a versão original do algoritmo ECSMiner, descrita no Capítulo 3, que supõe que os rótulos de todos os exemplos estarão disponíveis. Para a execução do ECSMiner, dois classificadores são utilizados, KNN ou árvores de decisão. Experimentos com os dois classificadores foram realizados. Os parâmetros do algoritmo foram configurados, em sua maioria, conforme descrito em (Masud et al., 2011a).

[1] Agradecimentos ao Eduardo Spinosa por fornecer o código fonte.
[2] O código executável encontra-se disponível em: http://dml.utdallas.edu/Mehedy/index files/Page675.html.
[3] Agradecimentos aos autores por fornecerem o código executável.

Tabela 6.2: Resumo dos parâmetros a serem configurados para cada algoritmo.

Algoritmo	Parâmetros
ECSMiner	Tamanho da janela de dados Número de janelas de dados na fase *offline* Tc, tempo máximo para rotular exemplo Tl, tempo máximo para obter o rótulo verdadeiro Número de classificadores do comitê Número de exemplos para executar o processo de DN q, usado na detecção de novidade Número de grupos por janela de dados Parâmetros do classificador (KNN ou árvore)
CLAM	Mesmos do ECSMiner
OLINDDA	Tamanho da base de treinamento Tamanho da memória temporária Número de exemplos no grupo Critério de validação dos grupos Parâmetros do método de agrupamento (K-Means)
MINAS	Tamanho da base de treinamento Número de exemplos para executar agrupamento (NroExDN) Número de exemplos no grupo (NroExGrp) Tamanho da janela de dados de esquecimento (TamJan) Limiar (Lim) Parâmetros do método de agrupamento (K-Means ou CluStream)

Tabela 6.3: Algoritmos usados nos experimentos.

Algoritmo	Descrição
ECSMiner-CF	Versão original: utiliza *feedback*, todos os exemplos são rotulados
ECSMiner-SF	Versão modificada: não utiliza *feedback*, somente os exemplos da base de treinamento são rotulados
CLAM-CF	Versão original: utiliza *feedback*, todos os exemplos são rotulados
CLAM-SF	Versão modificada: não utiliza *feedback*, somente os exemplos da base de treinamento são rotulados
OLINDDA	Versão original
MINAS-SF	Versão original: não utiliza *feedback*, somente os exemplos da base de treinamento são rotulados
MINAS-AA	Versão modificada: solicita ao usuário o rótulo de alguns exemplos do FCD

- Tamanho da janela: usado para definir o tamanho da janela de dados a ser usada na fase de treinamento inicial e no treinamento ao longo do FCD. Indica o número de exemplos que devem ser rotulados para que um novo classificador seja treinando. Nos experimentos são usadas janelas de tamanho 2.000.

- Número de janelas de treinamento: determina o número de janelas usadas na fase de treinamento *offline*. Nos experimentos descritos em (Masud et al., 2011a), os autores usam apenas 3 janelas. No entanto, para ficar coerente com a metodologia proposta neste trabalho, que usa 10% da base para treinamento, o número de janelas é definido de modo que este valor multiplicado pelo tamanho da janela seja igual ou próximo a 10% dos dados.

- Tc: tempo máximo permitido para classificar um novo dado, ou seja, um dado deve ser classificado pelo sistema no máximo até Tc unidades de tempo após sua chegada. O valor usado nos experimentos é 400.

- Tl: tempo máximo para rotular um novo dado, ou seja, todos os dados serão rotulados em até Tl unidades de tempo após sua chegada. É importante notar que $Tc < Tl$ deve ser verificado. O valor usados nos experimentos é 1000.

- Número de classificadores: define o número de classificadores do comitê. O valor usado nos experimentos é 6 classificadores.

- Número de exemplos para execução do processo de DN ($MinPts$): quantos exemplos (no mínimo) devem estar na memória temporária para que um processo de DN seja executado, na fase *online*. O valor usado é 50.

- Número de grupos: identifica o número de grupos criados por janela de dados de treinamento. O valor usado é 50.

- Parâmetro q: uma nova classe é detectada, se no mínimo q exemplos com o perfil *desconhecido* têm valor positivo para a medida q-NSC. O valor usado é o mesmo de $MinPts$.

- Parâmetros do classificador KNN: Valor do $K = 50$, na fase *offline*. Na fase *online*, esse valor é automaticamente determinado pelo algoritmo.

- Parâmetros do algoritmo de indução de árvores de decisão: são usados os parâmetros padrões do Weka.

6.3.2 ECSMiner – versão sem *feedback* (ECSMiner-CF)

Neste trabalho, foi criada uma nova versão do algoritmo ECSMiner, que supõe que os rótulos dos exemplos, na fase *online*, nunca estarão disponíveis, ou seja, o modelo de decisão deve ser atualizado sem *feedback* do usuário. Nessa nova versão, a fase *offline* não foi modificada. Nela, um comitê de classificadores é criado, a partir de dados rotulados. Na fase *online*, os exemplos são inicialmente rotulados usando o comitê de classificadores ou são marcados como *desconhecidos*, caso não sejam explicados pelo comitê. Grupos de exemplos *desconhecidos* são agrupados formando padrões-novidade. A partir de então, os exemplos são classificados usando inicialmente o comitê de classificadores. Caso não sejam explicados pelo comitê, o modelo formado pelo conjunto de padrões-novidade detectados ao longo do FCD é usado. Os valores dos parâmetros usados pelo ECSMiner-SF são os mesmos do ECSMiner-CF.

6.3.3 CLAM – versão com *feedback* (CLAM-CF)

Essa é a versão original do algoritmo CLAM, descrita no Capítulo 3, que supõe que os rótulos de todos os exemplos estarão disponíveis. O CLAM propõe o uso de apenas um classificador, o KNN. Os parâmetros do algoritmo CLAM são os mesmos do algoritmo ECSMiner. As configurações dos parâmetros do CLAM são as mesmas usadas para o ECSMiner, descritas anteriormente. A única exceção é o parâmetro número de classificadores, uma vez que no CLAM, um comitê de classificadores é criado para cada classe do problema. O número de classificadores utilizado é 3, conforme sugerido em (Al-Khateeb et al., 2012a).

6.3.4 CLAM – versão sem *feedback* (CLAM-SF)

Assim como para o ECSMiner, uma nova versão sem *feedback* foi também criada para o CLAM. Nessa nova versão, a fase *offline* não foi modificada. Nela, um comitê de classificadores é criado, para cada uma das classes encontradas na base de treinamento. A fase *online* é a mesma descrita no ECSMiner - versão sem *feedback*. Os parâmetros usados na sua configuração são os mesmos do CLAM.

6.3.5 OLINDDA

Para a execução do OLINDDA, diferentes algoritmos de agrupamento podem ser usados. Neste trabalho optou-se por usar apenas o algoritmo K-Means. Os parâmetros do algoritmo OLINDDA foram configurados, em sua maioria, conforme descrito em (Spinosa et al., 2009). São eles:

- Tamanho da base de treinamento: número de exemplos que compõem a base de treinamento, usada na fase *offline*. Conforme descrito na Seção 6.2, foram usados 10% da base de dados para treinamento.
- Tamanho da memória temporária: número máximo de exemplos permitidos na memória temporária. O tamanho da memória usado é 200.
- Número de exemplos no grupo: número mínimo de exemplos em um grupo para que este seja considerado válido. Foi usado o valor 3.
- Critério de validação de grupos: OLINDDA propõe três critérios para validação de grupos. O critério de validação usado foi a distância média dos exemplos ao centróide.
- Parâmetros do K-Means - número de grupos: Nos experimentos, o valor usado é 50, na fase *offline*. Na fase *online*, esse valor é automaticamente determinado pelo algoritmo.

6.3.6 MINAS – versão sem *feedback* (MINAS-SF)

Essa é a versão original do algoritmo MINAS, proposto nesta tese. Para execução do MINAS, dois algoritmos de agrupamento podem ser utilizados, K-Means ou CluStream. Experimentos com os dois algoritmos foram realizados. Os parâmetros a serem configurados no MINAS são descritos a seguir.

- Tamanho da base de treinamento: número de exemplos que compõem a base de treinamento, usada na fase *offline*. Conforme descrito na Seção 6.2, foram usados 10% da base de dados para treinamento.
- Número de exemplos para execução do processo de detecção de novidade (NroExDN): quantos exemplos devem estar na memória temporária para que o processo de detecção de novidade seja executado na fase *online*.
- Número de exemplos no grupo (NroExGrp): número mínimo de exemplos em um grupo para que este seja considerado válido.
- Tamanho da janela de esquecimento (TamJan): usado para eliminar exemplos da memória temporária que não formaram grupos válidos e para colocar na memória *sleep* os grupos que não receberam novos exemplos na última janela de dados.
- Limiar (Lim): usado para definir se um novo grupo válido é uma extensão ou uma novidade.
- Parâmetros do K-Means: número de grupos (K).
- Parâmetros do CluStream: número de micro-grupos (K) e número inicial de exemplos usados para executar o K-Means, responsável por inicializar os micro-grupos.

A fim de verificar a influência no desempenho do algoritmo MINAS-SF, a partir de variações nos valores dos parâmetros, diferentes configurações foram testadas, as quais descritas a seguir. Além disso, diferentes estratégias foram avaliadas a fim de tornar a configuração dos parâmetros o mais automática possível. As configurações usadas nos experimentos são mostradas na Tabela 6.4.

- **Configuração 1 (C1):**
  Nessa versão, foi usado o algoritmo CluStream tanto na fase *offline* quanto *online*, com 100 micro-grupos (parâmetro K). Na fase *offline*, para inicializar os micro-grupos do CluStream, inicialmente é executado o K-Means pegando os primeiros 10*K elementos. Na fase *online* o CluStream é executado, sem essa inicialização. A cada 2.000 exemplos marcados com o perfil *desconhecido*, o processo de DN é executado. Um grupo válido deve ter no mínimo NroExMempTmp/K exemplos. Nessa

configuração, quando um elemento é classificado usando um micro-grupo, o raio e o centróide do mesmo não são atualizados. Para separar as novidades das extensões, três propostas são usadas para variar o limiar, aqui nomeadas de VL1, VL2, VL3, respectivamente. Essas propostas são detalhadas a seguir:

- **VL1:** Nessa técnica, quando um novo micro-grupo válido é identificado, o seu micro-grupo mais próximo é encontrado M_{prox}. O valor do limiar usado é o desvio padrão D das distâncias Euclidiana entre os elementos do micro-grupo M_{prox} ao seu centróide, multiplicado por um fator t, com $t = 1.1$. Se a distância entre o centróide do novo micro-grupo e M_{prox} for menor que D, o novo micro-grupo é considerado uma extensão. Caso contrário, uma novidade.
- **VL2:** Nessa técnica, quando um novo micro-grupo válido é identificado, o seu micro-grupo mais próximo é encontrado M_{prox}. A maior distância Euclidiana D_{max} entre o centróide do micro-grupo M_{prox} e o centróide de um micro-grupo que pertence à mesma classe que M_{prox} é encontrada. Caso o micro-grupo M_{prox} seja o único micro-grupo de uma classe, o valor de D_{max} é o desvio padrão das distâncias Euclidiana entre os elementos do micro-grupo M_{prox} e o seu centróide, multiplicado por um fator t, sendo $t = 1.1$. Se a distância entre o centróide do novo micro-grupo e M_{prox} for menor que D_{max}, o novo micro-grupo é considerado uma extensão. Caso contrário, uma novidade.
- **VL3:** Essa técnica é semelhante ao VL2, mas ao invés da distância máxima, a distância média foi usada.

É importante observar que quando um micro-grupo é considerado uma extensão, ele está próximo a um outro micro-grupo que já possui rótulo. Este rótulo pode ser uma classe conhecida, ou um número sequencial que identifique o padrão-novidade.

- **Configuração 2(C2):**
  Essa é a primeira versão do algoritmo MINAS, que foi publicada na conferência SAC (Faria et al., 2013b). Algumas diferenças entre a versão atual e essa primeira são:
  - Para classificar um novo exemplo, não era escolhido o micro-grupo mais próximo, mas o primeiro micro-grupo tal que a distância do elemento ao centróide fosse menor que o seu raio;
  - Os elementos nunca eram removidos da memória temporária;
  - Na fase *offline*, inicialmente era executado o algoritmo CluStream. Em seguida, os micro-grupos gerados eram macro-agrupados usando o algoritmo K-Means e a técnica OMRk, que escolhe o número de grupos automaticamente;
  - Na fase *online*, o algoritmo K-Means era sempre usado;

Tabela 6.4: Configurações do algoritmo MINAS.

	Configuração 1		
Parâmetros	**VL-1**	**VL-2**	**VL-3**
NroExDN	2.000	2.000	2.000
NroExGrp	$NroExMemTmp$/K	$NroExMemTmp$/K	$NroExMemTmp$/K
TamJan	2* NroExDN	2* NroExDN	2* NroExDN
Lim	VL-1	VL-2	VL-3
Algoritmo	CluStream	Clustream	CluStream
K	100	100	100
Atualizar Centro/Raio	Não	Não	Não

Parâmetros	**Configuração 2**	**Configuração 3**	**Configuração 4**
NroExDN	NroExGr* K	2.000	50
NroExGr	3	$NroExMemTmp$/K	30
TamJan	1.000-10.000	2* NroExDN	2 * NroExDN
Lim	VL-1	VL-1	VL-1
Algoritmo	Clustream+KMeans	CluStream	CluStream
K	100 (Fase Offline)	100	50
Atualizar Centro/Raio	Não	Sim	Não

$NroExMemTmp$ é o número atual de exemplos na memória temporária.

  - O número de grupos usados na fase *online* era determinando usando a mesma estratégia do algoritmo OLINDDA;
  - O critério usado para validar um micro-grupo era verificar se o valor da medida de coesão do grupo era no mínimo a coesão do grupo menos coeso do modelo normal (mesma estratégia do OLINDDA). O tamanho da janela usada para esquecimento era 1.000 nas bases artificias e 10.000 nas bases reais.

- **Configuração 3(C3):**
  Essa configuração é idêntica à configuração 1, exceto que atualiza o raio e o centro de um micro-grupo quando ele é usado para classificar um novo exemplo.

- **Configuração 4(C4):**
  Essa configuração foi criada para testar o comportamento do algoritmo MINAS-SF quando a tarefa de a DN é executada com uma frequência menor, isso é, NroExDN = 50. Também foi usado um número menor de micro-grupos, apenas 50 micro-grupos. Nessa configuração, um grupo válido deve ter no mínimo 30 exemplos.

6.3.7 MINAS – versão com *aprendizado ativo* (MINAS-AA)

Essa é uma versão modificada do algoritmo MINAS usando aprendizado ativo (do inglês, *active learning*). A versão original do algoritmo MINAS foi desenvolvida pensando em um cenário no qual o rótulo dos exemplos não estará disponível na fase *online*, e portanto, usa um método de atualização do modelo de decisão sem *feedback*. No entanto, em alguns cenários, pode acontecer que o rótulo de alguns exemplos esteja disponível depois de um certo tempo, ou que um especialista possa rotular alguns exemplos do FCD. Pensando nestes cenários, uma nova versão do algoritmo MINAS foi desenvolvida, aqui chamada de

MINAS-AA. Os mesmos parâmetros configurados na versão original do MINAS devem ser configurados na versão com aprendizado ativo.

O MINAS-AA é executado da mesma forma que o MINAS original, com apenas pequenas alterações. Ela supõe que a cada Tr exemplos lidos, será executada uma rotina, que rotula cada um dos novos grupos (e não cada um dos exemplos) criados neste intervalo de tempo. No intervalo Tr, grupos extensão ou padrões-novidade podem ter sido criados. A rotina irá solicitar ao especialista de domínio o rótulo de cada grupo, fornecendo como exemplo a ser rotulado o centro do grupo. Note que nessa versão do MINAS com *feedback* não será necessário treinar um novo classificador, simplesmente será necessário fazer um atualização do rótulo do grupo. Além disso, se o especialista de domínio não souber o rótulo do grupo ou se não houver informação suficiente para rotulá-lo, o grupo continua existindo com o seu rótulo antigo, sem prejudicar o processo de classificação.

Nos experimentos realizados, como não havia a presença de um especialista de domínio, foi feita uma simulação do comportamento do mesmo. Para isso, foi considerado que o rótulo de um novo grupo é o rótulo da maioria dos exemplos que o compõe.

6.3.8 Observações sobre as Diferenças entre os Algoritmos

Antes de mostrar os resultados, é importante destacar as principais diferenças entre os algoritmos para facilitar o entendimento dos resultados obtidos.

Inicialmente, é importante destacar que os algoritmos MINAS, na sua versão original (MINAS-SF), e OLINDDA não fazem uso de *feedback* na fase *online*, ou seja, somente os exemplos da fase de treinamento (*offline*) são rotulados. Por outro lado, ECSMiner e CLAM, nas suas versões originais, ECSMiner-CF e CLAM-CF, respectivamente, supõem que após um atraso de Tl unidades de tempo o rótulo verdadeiro de todos os exemplos estará disponível. Assim, MINAS-SF e OLINDDA atualizam o modelo de decisão sem *feedback*, isso é, de forma não-supervisionada. Por outro lado, ECSMiner-CF e CLAM-CF atualizam o modelo de decisão substituindo um classificador do comitê por um novo classificador, treinado com os dados rotulados mais recentes. Assim, é esperado que ECSMiner-CF e CLAM-CF possuam melhor desempenho que MINAS-SF e OLINDDA.

O segundo ponto a ser destacado é que o algoritmo OLINDDA não usa o rótulo dos exemplos para criar o modelo de decisão na fase *offline*, pois ele supõe que todos os exemplos da base de treinamento pertencem à uma única classe, a classe normal. Por outro lado, MINAS, ECSMiner e CLAM, nas versões originais e modificadas, são algoritmos que supõem que a base de treinamento pode conter exemplos de diferentes classes e, portanto, usam a informação das classes na criação do modelo de decisão.

O terceiro ponto a ser destacado é que os quatro algoritmos usados nos experimentos utilizam um processo de DN baseado em agrupamento, no entanto há diferenças sobre o que será considerado um padrão-novidade. Para o MINAS, o processo de DN foi descrito

na Seção 4.6. Nesse processo, um padrão-novidade pode ser composto por um ou mais grupos. Um padrão-novidade é identificado quando um novo grupo válido é criado e a distância ao seu grupo mais próximo é maior que um limiar. Para o OLINDDA, cada vez que um novo grupo é considerado válido, e está fora da macro-hiperesfera criada na fase *offline*, ele é rotulado pelo sistema como pertencente ao conceito novidade, e então será aqui considerado como um padrão-novidade. Para ECSMiner e CLAM, cada vez que o processo de detecção de novidade é executado, um conjunto de grupos é gerado. Esse conjunto de grupos é avaliado para identificar se eles formam ou não uma novidade. Se uma novidade for detectada, esse conjunto de grupos será aqui considerado um padrão-novidade.

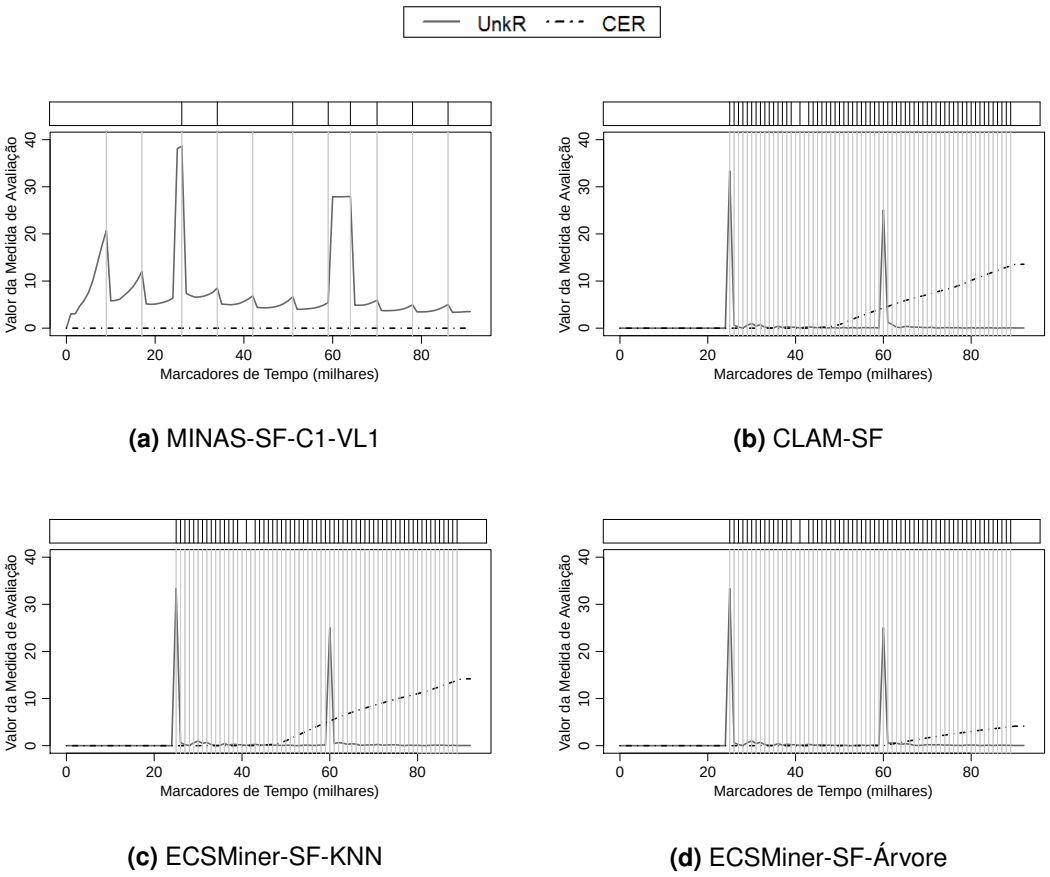
Por último, ECSMiner e CLAM impõem uma restrição sobre o tempo para classificar um exemplo. Em até Tc unidades de tempo, um exemplo marcado inicialmente como *desconhecido*, deve ser rotulado. Essa exigência não é imposta pelos algoritmos MINAS e OLINDDA. Exemplos marcados como *desconhecidos* ficam na memória temporária para formarem grupos válidos. Se esses exemplos formam grupos válidos, eles são movidos na matriz de confusão da coluna *desconhecidos* para a respectiva coluna na qual foram classificados. Se os exemplos não formam um padrão-novidade, eles são descartados da memória temporária e contabilizados na matriz confusão como *desconhecidos*.

6.4 Avaliação Usando a Metodologia Proposta e os Algoritmos sem *Feedback*

O primeiro conjunto de experimentos foi realizado usando as bases de dados apresentadas na Seção 6.2 e a metodologia de avaliação proposta neste trabalho. As medidas de avaliação usadas foram o CER (Equação 5.8) e $UnkR$ (Equação 5.10). Os algoritmos comparados foram CLAM-SF, ECSMiner-SF e MINAS-SF. Para o algoritmo ECSMiner-SF foram usados dois diferentes classificadores, árvores de decisão (ECSMiner-SF-Árvore) e KNN (ECSMiner-SF-KNN).

Nas figuras 6.1, 6.2, 6.3, 6.4, 6.6, 6.7 são mostrados os resultados dos experimentos usando os algoritmos CLAM-SF, ECSMiner-SF e MINAS-SF-C1-VL1 (configuração 1, valor do limiar VL1), e as bases de dados MOA, SynD, SynEDC, KDD, Covertype-V1 e Covertype-V2, respectivamente. Nessas figuras, as linhas verticais cinza representam os marcadores de tempo nos quais o algoritmo identificou um padrão-novidade. Na horizontal, a linha contínua representa a medida $UnkR$ e a linha pontilhada a medida CER.

Analisando os resultados dos experimentos usando a base MOA (ver Figura 6.1), pode-se perceber que a medida $UnkR$ apresentou dois picos nos quatro algoritmos testados. Esses picos indicam os marcadores de tempo no qual os exemplos das duas classes novidade começam a aparecer. Após cada pico, uma novidade é encontrada (linha vertical),



(a) MINAS-SF-C1-VL1

(b) CLAM-SF

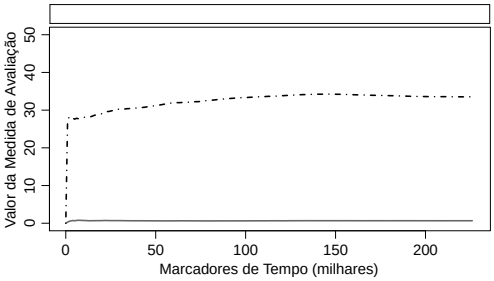
(c) ECSMiner-SF-KNN

(d) ECSMiner-SF-Árvore

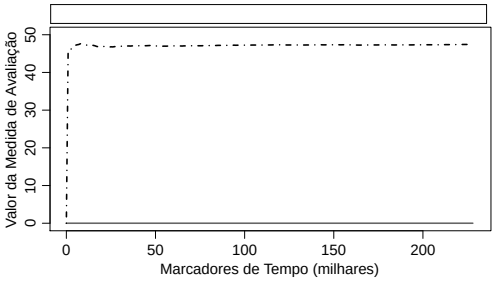
Figura 6.1: Desempenho usando a base MOA.

e a $UnkR$ diminui, indicando que aqueles exemplos formam um padrão-novidade. Considerando a $UnkR$ obtida pelo MINAS-SF, antes, depois e entre os dois picos principais, aparecem picos menores. Esse picos se referem às mudanças de conceito nas classes. Essas mudanças não são inicialmente explicadas pelo modelo de decisão, mas, a seguir, formam-se grupos que são identificados como extensões dos conceitos conhecidos. Note que ECSMiner-SF e CLAM-SF identificaram muito mais novidades que o MINAS-SF. Isso pode ser explicado porque, no MINAS, um padrão-novidade pode ser composto por um ou mais grupos que estão dentro de um limiar. Assim, novos grupos válidos podem ser identificados como extensões dos conceitos conhecidos, e não como novos padrões-novidade. É também importante ressaltar que todos os algoritmos mantiveram a taxa CER igual a zero.

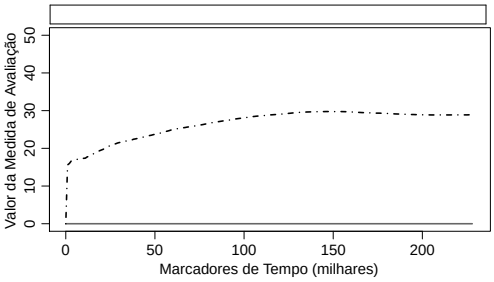
Para a base de dados SynD (ver Figura 6.2), nenhum dos quatro algoritmos utilizados detectou algum padrão-novidade. De fato, essa base não apresenta novas classes na fase *online*. Somente exemplos das classes conhecidas estão disponíveis na fase *online*. Para essa base, pode-se perceber que o algoritmo ECSMiner-SF-Árvore apresentou os melhores resultados. Uma possível conclusão é que o modelo de decisão baseado em árvores é melhor nessa base do que o modelo baseado em grupos, usado pelos três outros algoritmos. Como pouquíssimos exemplos foram classificados como *desconhecido*,



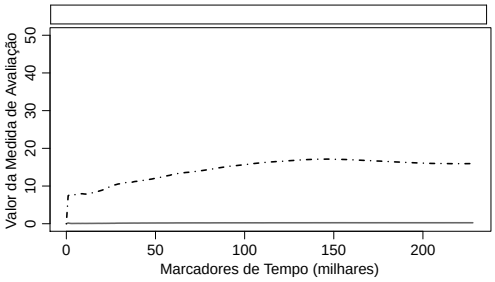
(a) MINAS-SF-C1-VL1



(b) CLAM-SF

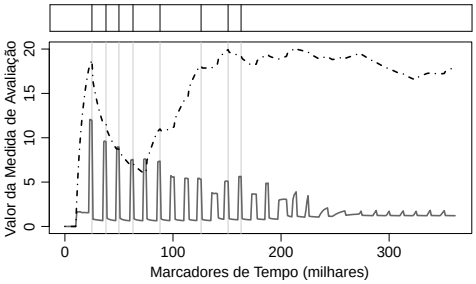


(c) ECSMiner-SF-KNN

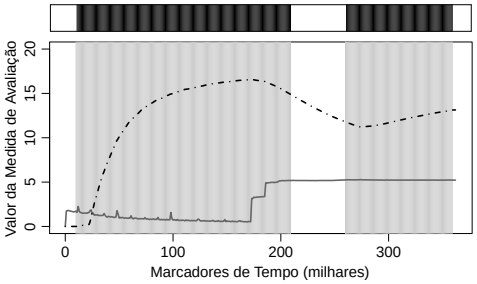


(d) ECSMiner-SF-Árvore

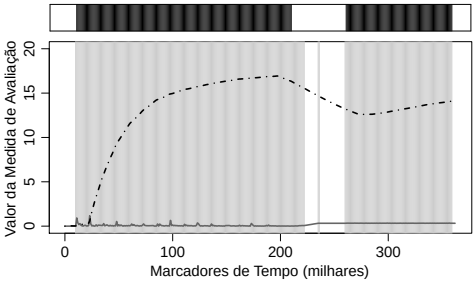
Figura 6.2: Desempenho usando a base SynD.



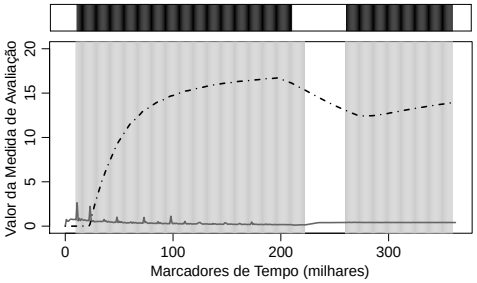
(a) MINAS-SF-C1-VL1



(b) CLAM-SF

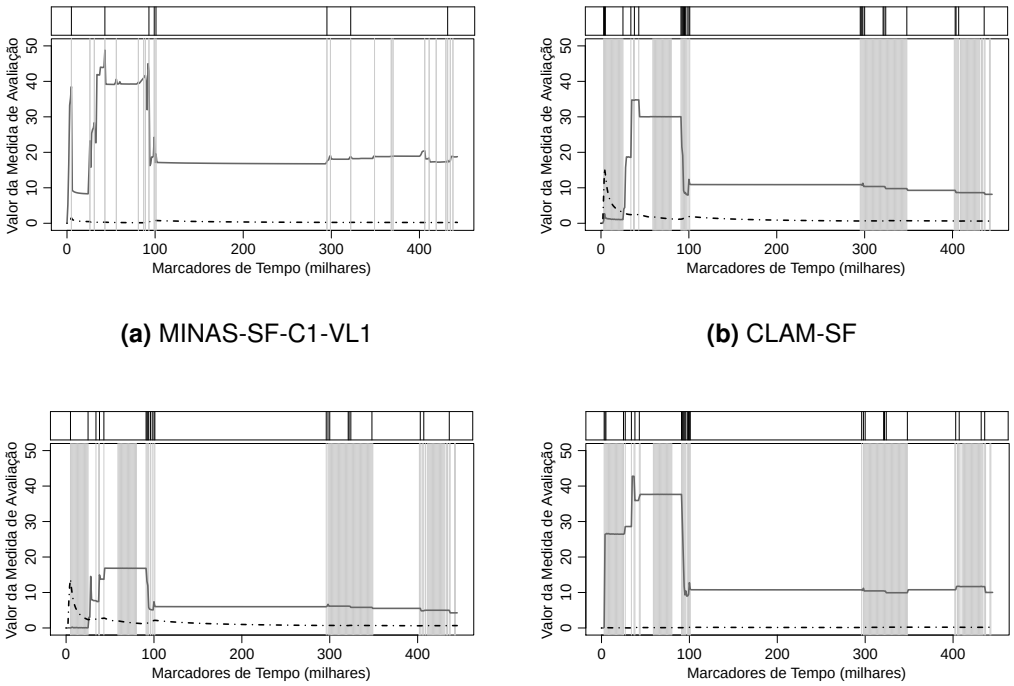


(c) ECSMiner-SF-KNN



(d) ECSMiner-SF-Árvore

Figura 6.3: Desempenho usando a base SynEDC.



(a) MINAS-SF-C1-VL1

(b) CLAM-SF

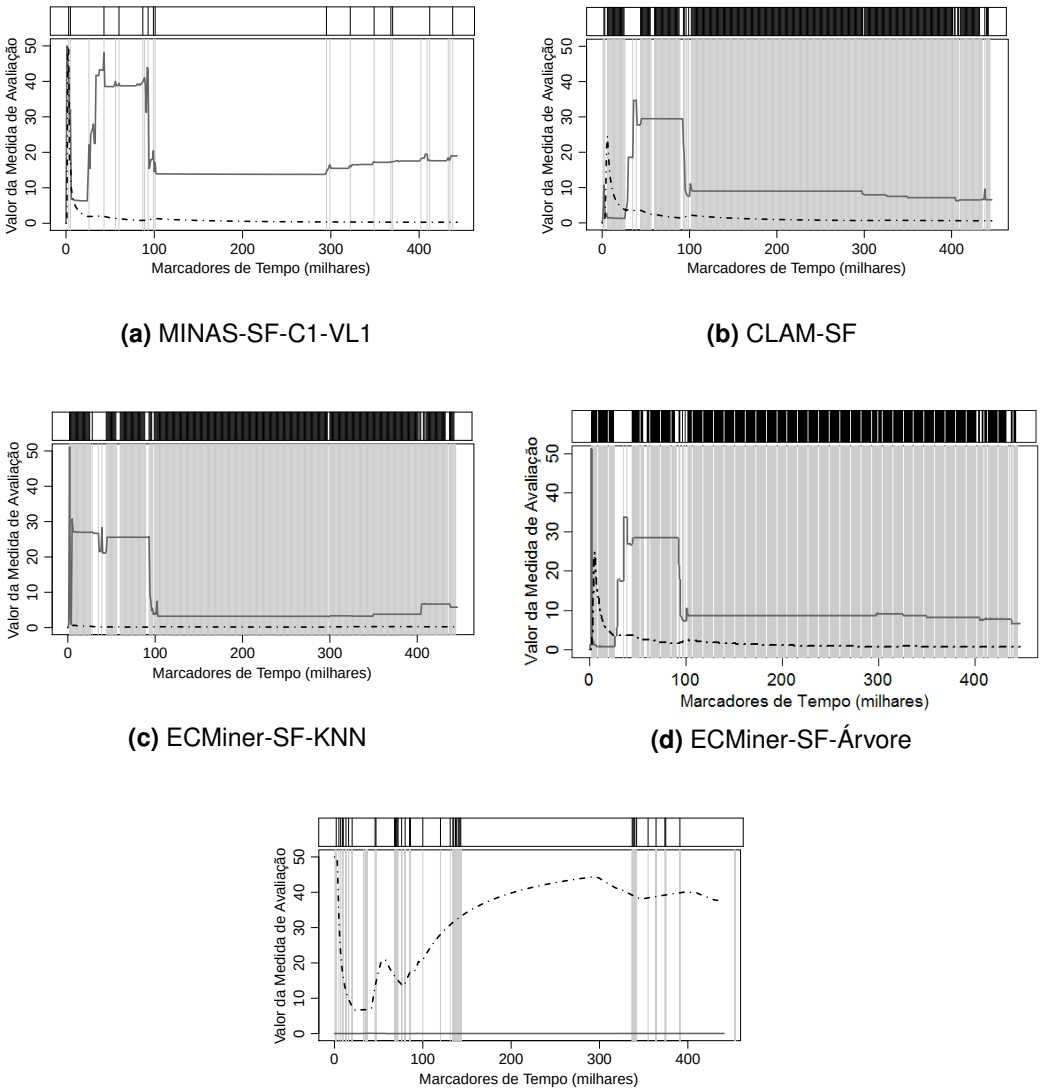
(c) ECMiner-SF-KNN

(d) ECMiner-SF-Árvore

Figura 6.4: Desempenho usando a base KDD.

conforme mostrado pela $UnkR$ baixa, pode-se supor que os exemplos foram classificados pelo modelo de decisão, mas não na classe correta. Uma possível conclusão é que o modelo de decisão gerado na fase *offline* não é adequado. O algoritmo CLAM-SF, que usa um comitê de classificadores para cada uma das classes do problema, apresentou as taxas mais altas de CER.

Os resultados obtidos nos experimentos com a base SynEDC são mostrados na Figura 6.3. O algoritmo MINAS-SF apresentou picos de $UnkR$ cada vez que exemplos de uma nova classe chegavam ao longo do FCD. Até o marcador de tempo 100.000, quase todo pico de UnR é seguido pela identificação de um padrão-novidade, decrescendo assim o valor do CER. No entanto, a partir desse ponto, picos de $UnkR$ aparecem, mas nem sempre há identificação de padrões-novidade. A explicação para esse fato é que os padrões-novidade foram incorretamente identificados como extensões dos conceitos conhecidos, aumentando assim o valor do CER. Nesse caso, o valor do limiar usado não foi capaz de separar as novidades das extensões. Os demais algoritmos apresentam comportamento semelhante. Como o processo de DN nesses algoritmos é executado com alta frequência, quando pelo menos 50 exemplos foram marcados como *desconhecidos*, o número de padrões-novidade gerados é alto, ou seja, há muitas linhas verticais no gráfico. Além disso, como nesses algoritmos não há identificação de extensões dos conceitos conhecidos, os exemplos das classes novidade foram sempre considerados como padrões-novidade, justificando assim um valor mais baixo para CER do que o MINAS-SF.



(a) MINAS-SF-C1-VL1

(b) CLAM-SF

(c) ECMiner-SF-KNN

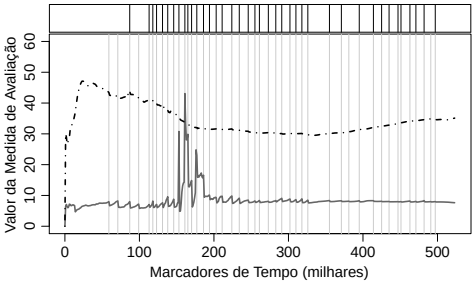
(d) ECMiner-SF-Árvore

(e) OLINDDA

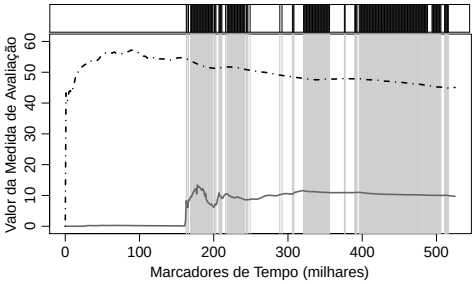
Figura 6.5: Desempenho usando a base KDD, com conjunto de treinamento contendo somente exemplos da classe normal.

Considerando a base KDD, os algoritmos MINAS, ECSMiner e CLAM apresentaram comportamento semelhante, com baixa valor para a medida CER ao longo do FCD, conforme mostrado na Figura 6.4. O algoritmo MINAS apresenta os maiores valores para a $UnkR$. Uma possível explicação para esse fato é que o critério para validar um novo grupo é muito restritivo ou que os exemplos *desconhecidos* são removido da memória temporária muito precocemente. A medida CER foi baixa durante todo o fluxo, para todos os algoritmos, especialmente para o MINAS-SF e o ECSMiner-SF-Árvore.

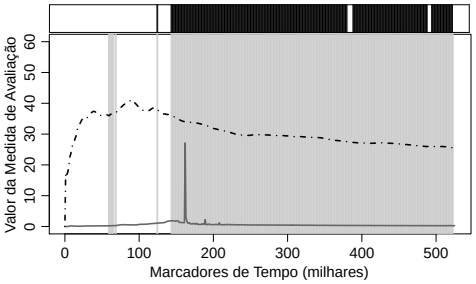
Uma segunda versão da base KDD, contendo apenas exemplos da classe normal no conjunto de treinamento, foi utilizada. O objetivo de utilizar essa base é avaliar o desempenho do algoritmo OLINDDA em comparação com os algoritmos MINAS, ECSMiner e



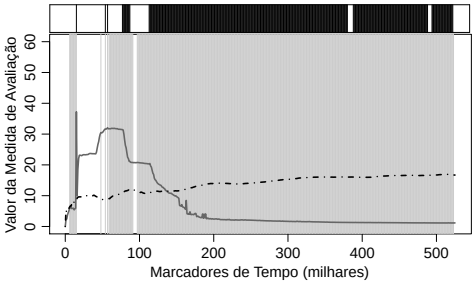
(a) MINAS-SF-C1-VL1



(b) CLAM-SF

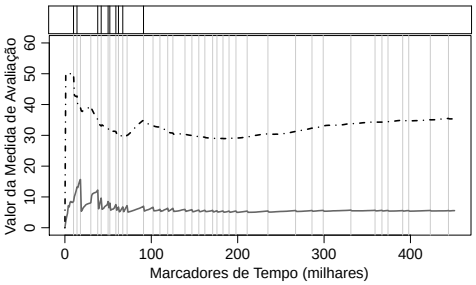


(c) ECSMiner-SF-KNN

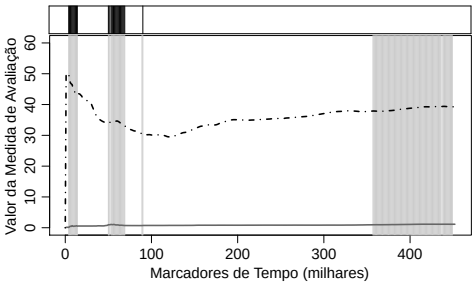


(d) ECSMiner-SF-Árvore

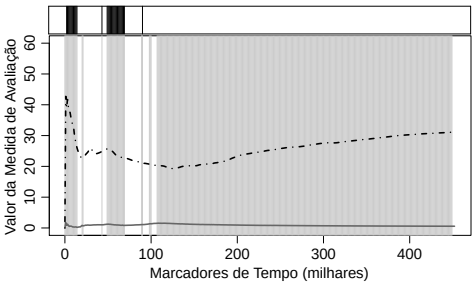
Figura 6.6: Desempenho usando a base Covertype-V1.



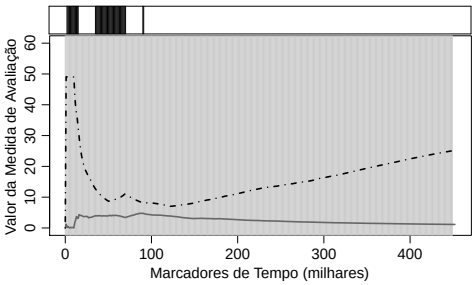
(a) MINAS-SF-C1-VL1



(b) CLAM-SF



(c) ECSMiner-SF-KNN



(d) ECSMiner-SF-Árvore

Figura 6.7: Desempenho usando a base Covertype-V2.

CLAM. O algoritmo OLINDDA supõe que todos os exemplos do conjunto de treinamento pertencem à mesma classe. Os resultados desses experimentos são mostrados na Figura 6.5. Pode-se observar que os algoritmos MINAS, ECSMiner e CLAM apresentam comportamentos semelhantes em relação à taxa CER. No entanto, os algoritmos CLAM e ECSMiner identificaram muito mais padrões-novidades que o MINAS. A explicação para isso é que o MINAS coloca no mesmo padrão-novidade grupos válidos que estão próximos, dentro de um dado limiar. Já o CLAM e ECSMiner, a cada vez que executam um processo para DN, podem identificar um novo padrão-novidade. O algoritmo OLINDDA obteve os maiores valores para a taxa CER e apresentou poucos exemplos marcados como *desconhecidos*, o que pode ser visto pela valor da medida $UnkR$ próxima de zero ao longo do FCD. Uma possível explicação é que o modelo de decisão criado na fase *offline* foi muito geral, classificando de forma incorreta os exemplos das classes novidade. Uma outra execução do algoritmo OLINDDA foi realizada, usando apenas 6.000 exemplos da classe normal na fase treinamento e o restante dos exemplos na fase *online*. Nessa execução, o algoritmo OLINDDA teve um desempenho bem melhor, com baixo valor para a media CER.

Os resultados dos experimentos com a base Covertype-V1 são ilustrados na Figura 6.6. Os algoritmos MINAS-SF, CLAM-SF e ECSMiner-SF-KNN apresentam comportamento semelhante. O valor da taxa CER é alto no começo do FCD e diminui um pouco ao longo do tempo. Ainda que o algoritmo MINAS-SF tenha identificado vários picos de $UnkR$, ele manteve alto valor para a medida CER. O algoritmo ECSMiner-V2-Árvore mostrou o menor valor para a medida CER, indicando que o modelo de decisão baseado em uma árvore de decisão apresenta melhores resultados para essa base.

O trabalho de Masud et al. (2011a) propõe o uso de uma versão modificada da base Covertype, aqui chamada de Covertype-V2. Os resultados dos experimentos com essa base são mostrados na Figura 6.7. Os resultados obtidos com os algoritmos MINAS, CLAM-SF e ECSMiner-SF-KNN são semelhantes. Ainda que o MINAS tenha encontrado alguns picos de $UnkR$, esses não foram capazes de melhorar o desempenho do algoritmo em relação aos demais competidores. Novamente, o ECSMiner-V2-Árvore apresentou os melhores resultados.

6.5 Comparação entre a Metodologia Proposta e a Metologia Empregada na Literatura

A metodologia usada na literatura para avaliar os algoritmos de DN em FCD considera o problema uma tarefa de classificação binária, composta pelas classes normal e novidade. Nesse cenário, um erro é computado quando:

- Exemplos das classes conhecidas são classificados como novidade (FP - falso po-

sitivo);

- Exemplos das classes novidades são classificados nas classes conhecidas (FN-falso negativo).

Para a avaliação, são usadas medidas como $MNew$ e $FNew$, descritas na Seção 3.7.3, nas equações 3.11 e 3.12. Os trabalhos (Masud et al., 2011a), (Al-Khateeb et al., 2012a), (Farid et al., 2013) também propõem o uso da medida Err, descrita na Seção 3.7.3, na Equação 3.13. Essa medida considera, além do número de FP e os FN, o número de FE. O número de FE são computados quando exemplos das classes conhecidas são classificados incorretamente, considerando que o conceito normal é composto por mais de uma classe.

Na metodologia usada na literatura, todos os padrões-novidades encontrados ao longo do FCD são sumarizados, formando uma única coluna na matriz de confusão, que representa todos os exemplos classificados como novidade. Assim, as medidas de avaliação não levam em consideração que as diferentes classes novidades podem estar associadas a diferentes padrões-novidade.

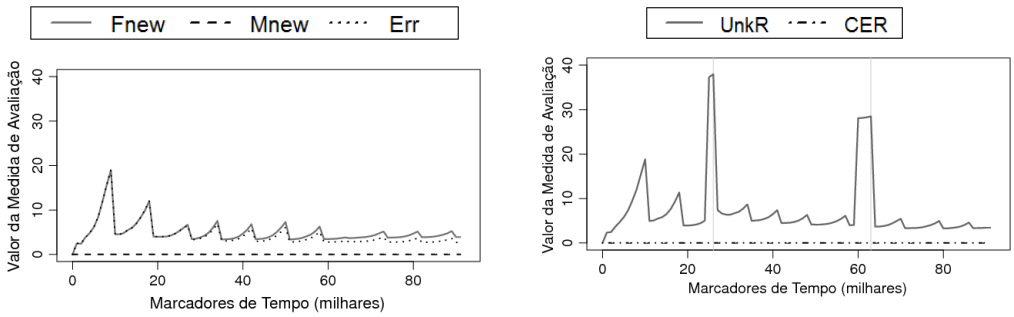
Outro ponto a ser destacado na metodologia de avaliação usada na literatura é que não há um tratamento especial para avaliar os exemplos marcados com o perfil *desconhecido*. Nos trabalhos de Masud et al. (2011a) e Al-Khateeb et al. (2012a), os exemplos marcados como *desconhecidos* não são contabilizados na matriz de confusão. Como após Tc unidades de tempo um exemplo deve ser classificado, com um rótulo diferente de *desconhecido*, ele será contabilizado na matriz de confusão somente neste momento. Já o trabalho de Spinosa et al. (2009) considera os exemplos *desconhecidos* como falsos positivos (FP), se eles pertencem às classes conhecidas do problema. Neste trabalho, nos experimentos realizados usando a metodologia da literatura, será usada a mesma estratégia do OLINDDA, os exemplos *desconhecidos* serão considerados falsos positivos (FP), se eles pertencem às classes conhecidas do problema. Além disso, diferente do OLINDDA, que considera erros os exemplos da classe normal classificados como extensão, aqui eles serão computados como acertos.

Também é importante destacar que as medidas de avaliação usadas na literatura não são para problemas multiclasse. Isso ocorre porque a maioria dos algoritmos supõe que o problema de DN é um problema de classificação binário, composto pela classe normal e pela classe novidade. Além disso, não há nenhuma ponderação em relação a classes desbalanceadas.

A fim de comparar a metodologia de avaliação proposta neste trabalho com a metodologia usada na literatura, alguns experimentos foram executados, e os resultados obtidos são mostrados nas figuras 6.8, 6.9 e 6.10.

O primeiro experimento foi executado usando o algoritmo MINAS-SF e a base de dados MOA. Foi usado o MINAS-SF, configuração 1 - VL1, com uma pequena modificação, o valor

do parâmetro t escolhido é o que melhor separa as extensões das novidades. Esse experimento tem como objetivo mostrar a importância de se analisar os exemplos desconhecidos, separadamente das medidas de desempenho como erro ou acurácia. Os resultados são mostrados na Figura 6.8. Usando a metodologia da literatura, é possível observar que, na presença de mudança de conceito nas classes conhecidas do problemas, MINAS-SF inicialmente classifica esses exemplos como desconhecidos, aumentando assim o valor da medida $FNew$. A seguir, esses exemplos desconhecidos são usados para modelar extensões dos conceitos conhecidos, decrementando assim a medida $FNew$. Nesse caso, pode-se notar que o $FNew$ e o Err são determinados somente pelos exemplos *desconhecidos*. Seria mais fácil entender o comportamento do sistema se os exemplos *desconhecidos* fossem computados separadamente. Usando a metodologia proposta neste trabalho, pode-se concluir que o classificador não cometeu qualquer erro, pois a medida CER mantém o valor zero ao longo do FCD. Além disso, as variações na medida $UnkR$ ajudam a entender o comportamento do algoritmo. É possível notar que a taxa $UnkR$ aumenta na presença de exemplos das classes novidades ou na presença de mudança de conceito. Também é possível notar que somente dois padrões-novidade foram encontrados, representando as duas classes novidade presentes na base de teste.



(a) Met. Literatura **(b)** Met. Proposta

Figura 6.8: Comparação entre as metodologias de avaliação usando a base MOA.

O segundo experimento foi executado usando o algoritmo MINAS-SF e a base de dados SynEDC. Esse experimento tem como objetivo mostrar as diferenças entre os resultados obtidos associando ou não os padrões-novidade às classes do problema. Os resultados são mostrados na Figura 6.9. Nessa figura, a linha contínua representa o valor da medida Err considerando a metodologia da literatura e a linha pontilhada o valor da medida Err considerando a metodologia proposta. A Tabela 6.5 mostra a matriz de confusão gerada pelo algoritmo MINAS-SF, configuração 1 - VL1, usando a base SynEDC. Nessa matriz, as linhas representam as classes reais do problema e as colunas as classes preditas pelo algoritmo. As classes $\mathbf{C}_1$ até $\mathbf{C}_7$ são usadas no treinamento e as demais só aparecem no conjunto de teste. As colunas $\mathbf{PN}_1$ até $\mathbf{PN}_9$ representam os padrões-novidade detectados pelo algoritmo. A coluna $\mathbf{Unk}$ representa os exemplos classificados como desconhecidos,

Tabela 6.5: Matriz de confusão final usando o algoritmo MINAS-SF e a base SynEDC.

	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$	$\mathbf{C_4}$	$\mathbf{C_5}$	$\mathbf{C_6}$	$\mathbf{C_7}$	$\mathbf{PN_1}$	$\mathbf{PN_2}$	$\mathbf{PN_3}$	$\mathbf{PN_4}$	$\mathbf{PN_5}$	$\mathbf{PN_6}$	$\mathbf{PN_7}$	$\mathbf{PN_8}$	$\mathbf{PN_9}$	**Desc.**
$\mathbf{C_1}$	11966	0	0	0	0	497	0	0	0	0	0	0	0	261	67	61	96
$\mathbf{C_2}$	0	0	0	0	0	80	0	0	0	0	0	0	0	22	11873	1	456
$\mathbf{C_3}$	845	0	17764	2592	147	0	0	0	0	0	0	0	0	0	1594	0	590
$\mathbf{C_4}$	44	0	0	12712	6	0	0	0	0	0	0	0	0	0	3428	0	220
$\mathbf{C_5}$	0	0	0	0	13749	0	0	0	0	0	0	0	0	109	12	0	82
$\mathbf{C_6}$	0	0	0	0	0	12142	0	0	0	0	0	0	0	338	0	73	93
$\mathbf{C_7}$	0	0	0	0	0	0	0	0	0	0	0	0	0	12412	20	15	93
$\mathbf{C_8}$	12722	0	2003	1	7	0	0	9216	0	0	0	0	0	0	10	0	1159
$\mathbf{C_9}$	2	0	131	1	2	0	0	0	24821	0	0	0	0	0	5	0	0
$\mathbf{C_{10}}$	0	0	219	226	0	0	0	0	0	24221	0	0	0	0	0	0	190
$\mathbf{C_{11}}$	0	0	184	0	0	0	0	0	1	121	24562	0	0	0	0	0	176
$\mathbf{C_{12}}$	0	0	0	0	0	0	0	691	1	2	1	24031	0	0	0	0	171
$\mathbf{C_{13}}$	0	0	0	0	0	0	0	0	0	0	24345	1	0	0	0	0	165
$\mathbf{C_{14}}$	0	0	0	0	0	0	0	0	0	138	49	24	22872	0	0	0	181
$\mathbf{C_{15}}$	0	0	0	0	0	0	0	0	0	0	572	0	19619	0	0	0	174
$\mathbf{C_{16}}$	0	0	0	0	0	0	0	0	0	0	16	0	12493	0	0	0	91
$\mathbf{C_{17}}$	0	0	0	0	0	0	0	0	0	0	187	0	303	11755	0	0	85
$\mathbf{C_{18}}$	0	0	0	0	0	0	0	0	0	0	0	0	247	12057	0	0	279
$\mathbf{C_{19}}$	0	0	0	0	0	0	0	0	0	0	0	0	207	201	12058	0	78
$\mathbf{C_{20}}$	0	0	0	0	0	0	0	0	0	0	0	0	1	28	0	12350	87

que não foram usados na geração de padrões-novidade.

Observando a Tabela 6.5, pode-se perceber que a maioria dos exemplos das classes conhecidas foram corretamente classificados. Para as classes novidade $\mathbf{C_9}$ até $\mathbf{C_{12}}$, o algoritmo conseguiu identificar um padrão-novidade e classificar a maioria dos exemplos dessas classes novidade em um padrão-novidade. No entanto, os exemplos das classes novidade $\mathbf{C_{14}}$ até $\mathbf{C_{18}}$ são classificados, em sua maioria, nos padrões-novidade $\mathbf{PN_6}$ e $\mathbf{PN_7}$. Usando a metodologia da literatura, os padrões-novidade $\mathbf{PN_1}$ até $\mathbf{PN_9}$ são unidos em uma única coluna, chamada novidade. Os exemplos das classes novidade classificados em algum desses padrões-novidade são computados como acerto. No entanto, a metodologia proposta neste trabalho considera associar cada padrão-novidade a uma classe do problema e em seguida calcular a medida de erro. Assim, usando a metodologia proposta neste trabalho, o padrão-novidade $\mathbf{PN_6}$ será associado à classe $\mathbf{C_{14}}$. Os exemplos das classes $\mathbf{C_{15}}$ até a $\mathbf{C_{19}}$ classificados no $\mathbf{PN_6}$ são computados como erro. Isso explica o maior valor da medida Err encontrado usando a metodologia da literatura.

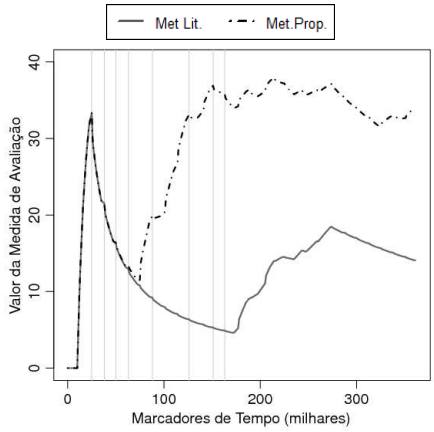


Figura 6.9: Comparação entre as metodologias de avaliação usando a base SynEDC e o algoritmo MINAS-SF.

O terceiro experimento visa mostrar as diferenças entre o uso de medidas de erro por classe, medidas ponderadas pelo número de exemplos e medidas que calculam o erro global. Para isso, o algoritmo ECSMiner-CF e a base KDD foram usados. A Tabela 6.6 mostra a matriz de confusão gerada para esse cenário. Nessa tabela, as linhas representam as classes do problema e as colunas as classes preditas pelo algoritmo. A base de treinamento contém apenas exemplos das classes $\mathbf{C}_1$ e $\mathbf{C}_2$. As classes $\mathbf{C}_3$, $\mathbf{C}_4$ e $\mathbf{C}_5$ aparecem nas colunas da matriz, pois o algoritmo usa *feedback* e quando há elementos rotulados suficientes, um novo classificador é treinado. As colunas $\mathbf{PN}_2$ e $\mathbf{PN}_4$ representam o conjunto de padrões-novidade associados às classe $\mathbf{C}_2$ e $\mathbf{C}_4$, respectivamente. A criação desses padrões-novidade aconteceu antes que o rótulo verdadeiro dos exemplos estivesse disponível e um novo classificador fosse treinado. A coluna $\mathbf{Unk}$ representa os exemplos marcados com o perfil *desconhecido*. A base de dados KDD ilustra um cenário envolvendo classes desbalanceadas. Há muitos exemplos das classes $\mathbf{C}_1$ e $\mathbf{C}_2$, que correspondem ao acesso normal e ao ataque *dos*, respectivamente, e poucos exemplos dos outros três tipos de ataque.

Nesse terceiro experimento, três medidas de avaliação foram utilizadas, Err (Equação 3.13), CER (Equação 5.8) e $Erro\_Medio$ (Equação 5.9). Os resultados são mostrados na Figura 6.10. É possível perceber que a medida Err apresenta os maiores valores dentre as três medidas de erro. Isso acontece porque ela contabiliza o erro global sem se preocupar com o número de exemplos na classe. Já a medida CER aplica uma ponderação na taxa de FP e taxa de FN. Os exemplos das classes novidades $\mathbf{C}_3$, $\mathbf{C}_4$ e $\mathbf{C}_5$ foram, em sua maioria, classificados incorretamente. No entanto, eles representam poucos exemplos em relação ao número total de exemplos da matriz de confusão, o que gera valores baixo para a medida CER. Os menores valores são encontrados usando a medida $Erro\_Medio$. Essa medida não faz nenhuma ponderação em relação ao número de exemplos, mas calcula a média de erro por classe. Pode-se concluir que a medida de avaliação usada depende das características do problema e do que se pretende avaliar. Características, como classes desbalanceadas ou balanceadas e penalização maior para os exemplos das classes novidade em relação às classes conhecidas podem nortear a escolha da medida de avaliação.

Tabela 6.6: Matriz de confusão final usando o algoritmo ECSMiner-CF e a base KDD.

	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	$\mathbf{C}_5$	$\mathbf{PN-C}_2$	$\mathbf{PN-C}_4$	$\mathbf{Desc.}$
$\mathbf{C}_1$	51662	66	5374	95	22	0	0	72
$\mathbf{C}_2$	1078	357369	20638	89	19	1932	0	299
$\mathbf{C}_3$	792	3	1	259	0	0	0	10
$\mathbf{C}_4$	784	8	2033	244	0	4	609	82
$\mathbf{C}_5$	18	0	0	13	8	1	0	7

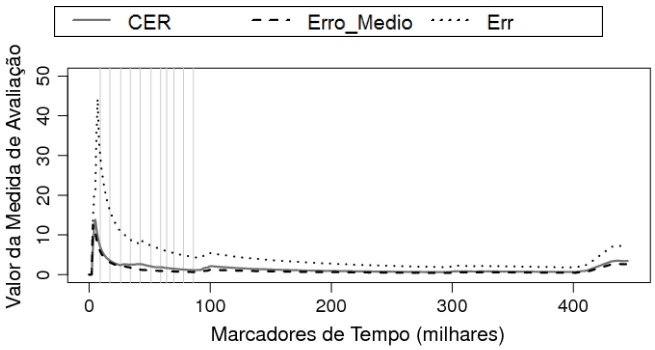
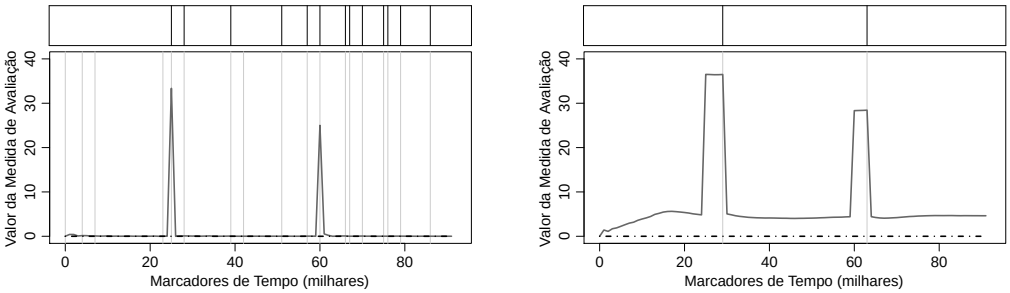


Figura 6.10: Comparação entre as medidas de erro usando a base KDD, a metodologia proposta e o algoritmo ECSMiner-CF.

6.6 Avaliação MINAS *sem Feedback* - Variações nas Configurações e no Cálculo do Limiar

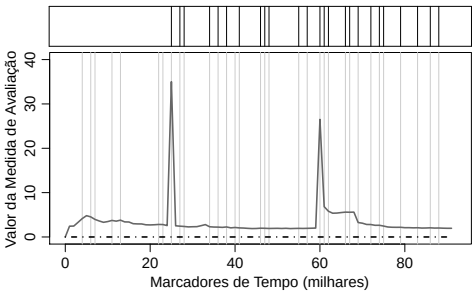
Essa seção apresenta experimentos com variações nos valores dos parâmetros a serem configurados no MINAS. As bases usadas foram MOA, SynEDC, KDD e Covertype. A base SynD não será apresentada nos resultados, pois não houve modificações em relação à versão MINAS-SF-CF1-VL1 já apresentada. Os resultados desses experimentos são mostrados nas Figuras 6.11, 6.12, 6.13 e 6.14. O algoritmo utilizado nos experimentos é o MINAS-SF com as configurações 2, 3 e 4, aqui nomeadas por MINAS-SF-C2, MINAS-SF-C3 e MINAS-SF-C4, respectivamente, e as variações no limiar 2 e 3 (ver Seção 6.3.6) usando a configuração 1, aqui nomeadas de MINAS-SF-C1-VL2 e MINAS-SF-C1-VL3, respectivamente.

Usando a base MOA (ver Figura 6.11), nota-se que cada configuração do algoritmo MINAS apresenta um comportamento distinto em relação à medida $UnkR$. No entanto, a medida CER mantém o valor zero ao longo de todo o FCD em todos os casos. No MINAS-SF-C2, como o agrupamento é executado com muita frequência e o número mínimo de exemplos para validar um grupo é pequeno, as mudanças nos conceitos conhecidos do problema são rapidamente identificadas como extensões, sem aumentar o número de exemplos *desconhecidos*. Um cenário parecido é apresentado no MINAS-SF-C4. No entanto, nesse último, o agrupamento é executado com uma frequência um pouco menor, além de que o número mínimo de exemplos exigido para validar um grupo é maior. O MINAS-SF-C3 apresenta um comportamento bem diferente, no qual extensões não são identificadas, mas apenas padrões-novidades. Isso acontece porque o modelo de decisão é constantemente atualizado a cada vez que um novo micro-grupo classifica um exemplo (atualização do centro e raio). Assim, o modelo de decisão consegue lidar com as mudanças nos conceitos de forma constante, sem gerar extensões dos conceitos conhecidos. O MINAS-SF-C1-VL2 e MINAS-SF-C1-VL3 apresenta os mesmos resultados, os quais são

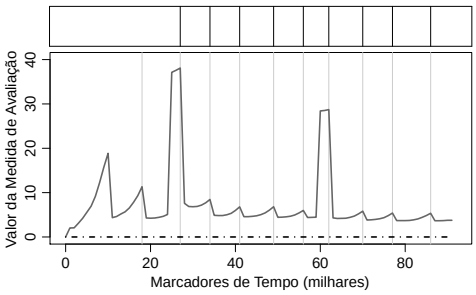


(a) MINAS-SF-C2

(b) MINAS-SF-C3



(c) MINAS-SF-C4

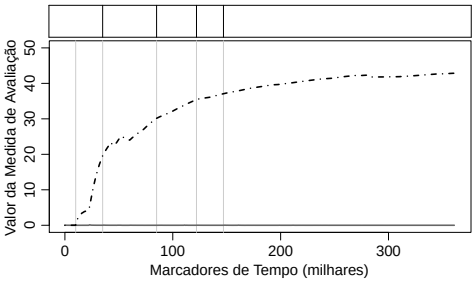


(d) MINAS-SF-C1 VL2/VL3

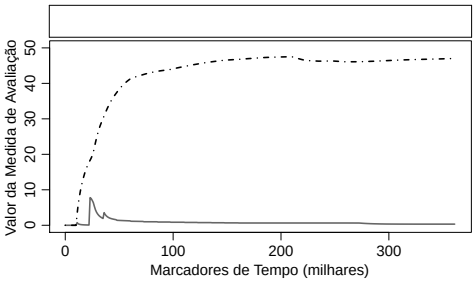
Figura 6.11: Desempenho do algoritmo MINAS-SF usando diferentes configurações e a base MOA.

bem parecidos com os da execução MINAS-SF-C1-VL1, já apresentados (ver Figura 6.1a). Isso mostra que mudanças no limiar não trouxeram mudanças no comportamento do algoritmo.

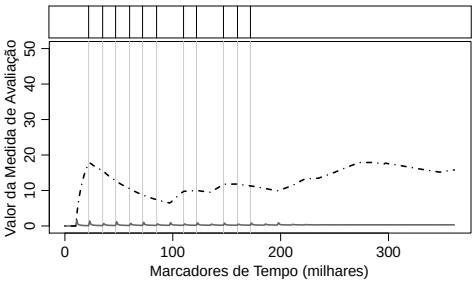
Em relação à base SynEDC (ver Figura 6.12), pode-se notar que o MINAS-SF-C4 obteve o melhor desempenho. Atualizar o centro e o raio a cada vez que um exemplo é classificado, adotado pelo MINAS-SF-C3, embora seja uma boa estratégia para a base MOA, não é uma boa estratégia para a base SynEDC. Uma possível explicação é que nessa base os grupos estão muito próximos, e há sobreposições. Assim, quando o MINAS-SF-C3 classifica incorretamente um elemento em uma classe e o correspondente grupo é atualizado, o formato do grupo é alterado de forma não desejada. Isso pode contribuir para que outros elementos sejam incorretamente classificados nesse grupo. No caso do MINAS-SF-C2, os altos valores para a media CER podem ser explicados pela criação de um modelo de decisão não muito adequado aos dados. O valor de K na fase *offline*, obtido pelo MINAS-SF-C2, usando a técnica OMRk, é igual a 2, criando grupos muito grandes e talvez com muita sobreposições. Outro ponto importante a ser destacado é que nas configurações MINAS-SF-C2 e MINAS-SF-C3 poucas novidades foram encontradas, além de que a medida $UnkR$ permanece próxima de zero na maior parte do FCD. Isso significa que o modelo de decisão criado é muito generalista, classificando incorretamente os exemplos das classes novidades nas classes conhecidas, ao invés de classificá-los como *desconhe-*



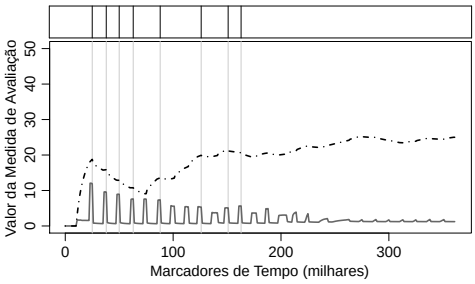
(a) MINAS-SF-C2



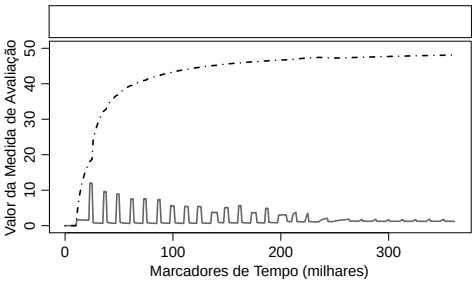
(b) MINAS-SF-C3



(c) MINAS-SF-C4



(d) MINAS-SF-C1-VL2

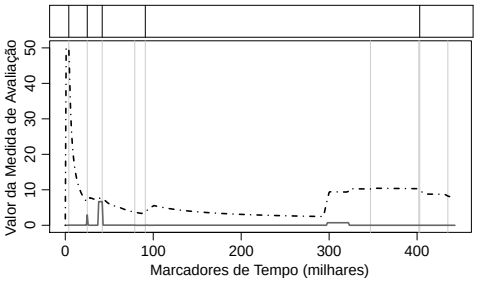


(e) MINAS-SF-C1-VL3

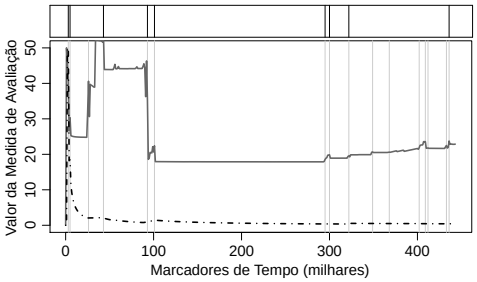
Figura 6.12: Desempenho do algoritmo MINAS-SF usando diferentes configurações e a base SynEDC.

cido. Com relação às mudanças no limiar, pode-se perceber que a abordagem VL3 não foi adequada. Ainda que o algoritmo tenha identificado picos de $UnkR$ cada vez que uma nova classe aparece, os novos grupos foram identificados como extensões dos conceitos conhecidos e não como padrões-novidade. Já o MINAS-SF-C1-VL2 apresenta resultados bem parecidos com o MINAS-SF-C1-VL1 (ver Figura 6.3a).

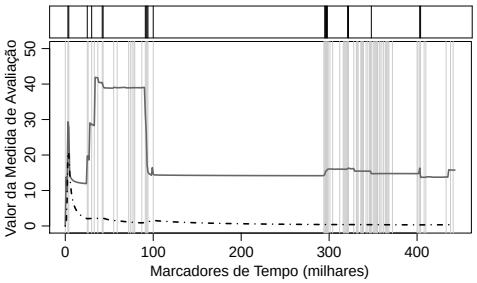
Na base KDD (ver Figura 6.13), é possível notar que a medida CER apresenta um comportamento semelhante no MINAS-SF-C2, MINAS-SF-C3 e MINAS-SF-C4, exceto pelo fato de que o erro aumenta no fim do FCD no MINAS-SF-C2. No MINAS-SF-C2, o baixo número de exemplos classificados como *desconhecido* pode ser explicado pelo modelo de decisão generalista, criado na fase *offline*, assim como na base SynEDC. No MINAS-SF-



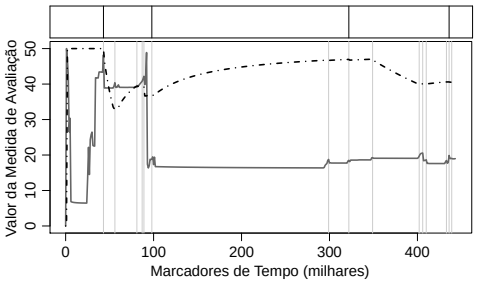
(a) MINAS-SF-C2



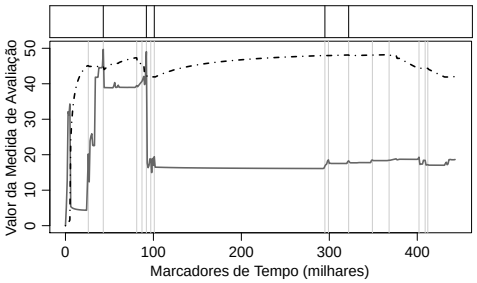
(b) MINAS-SF-C3



(c) MINAS-SF-C4



(d) MINAS-SF-C1-VL2

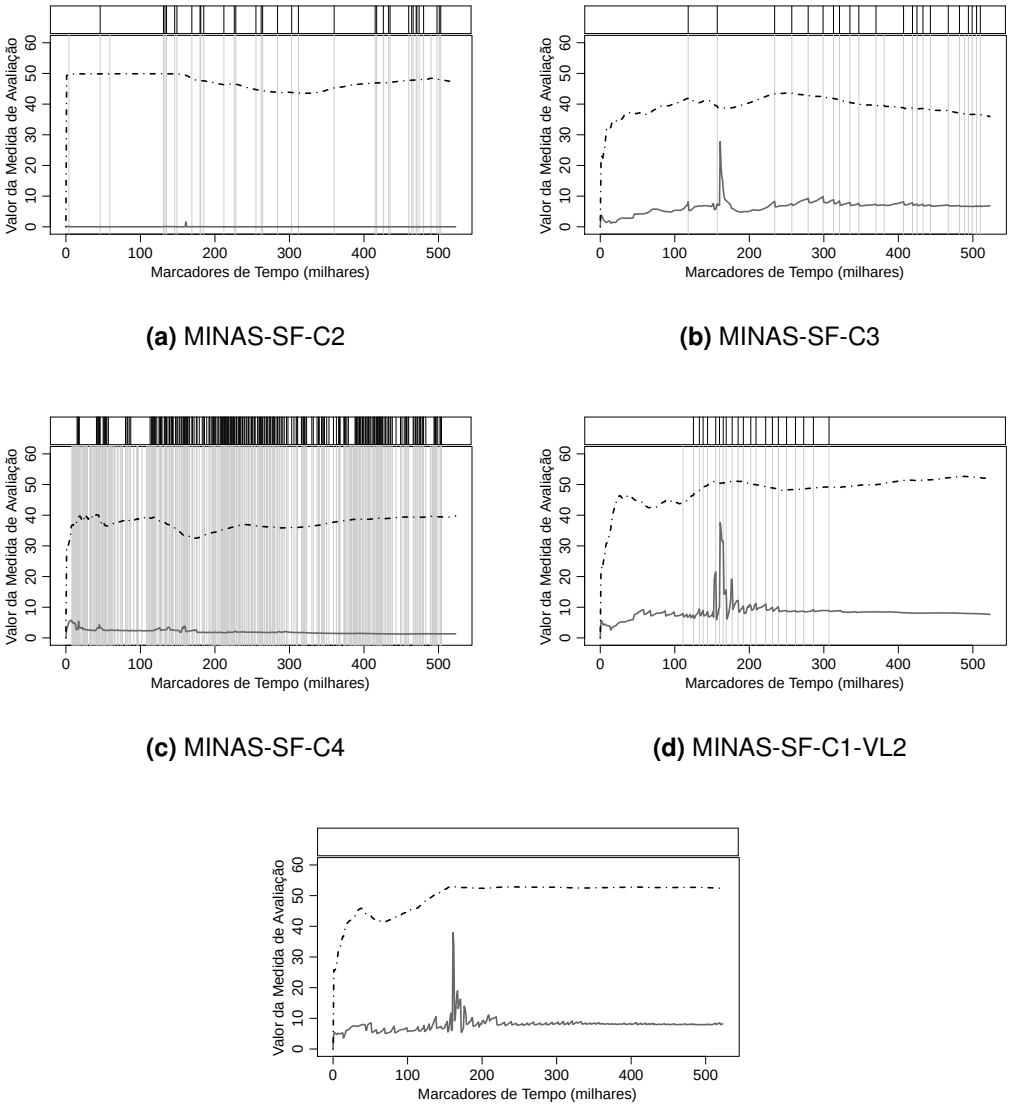


(e) MINAS-SF-C1-VL3

Figura 6.13: Desempenho do algoritmo MINAS-SF usando diferentes configurações e a base KDD.

C3, como há atualização do raio e centróide, menos padrões-novidade são identificados. Além disso, o processo de DN é executado com uma frequência menor do que no MINAS-SF-C4. Variações no cálculo do limiar apresentam os piores resultados. Nesse caso, os novos grupos identificados a partir de exemplos das classes novidades foram incorretamente classificados nas classes conhecidas, aumentando assim o valor da medida CER. Tanto o MINAS-SF-C1-VL2, quanto o MINAS-SF-C1-VL3, apresentaram resultados piores que o do MINAS-SF-C1-VL1, mostrados na Figura 6.3a.

Considerando a base Covertype-V1 (ver Figura 6.14), o MINAS-SF-CF2 apresentou as maiores taxa de CER e os menores valores para a medida $UnkR$. Mesmo com variações nas configurações dos parâmetros e variações no limiar, o valor da medida CER é



(a) MINAS-SF-C2

(b) MINAS-SF-C3

(c) MINAS-SF-C4

(d) MINAS-SF-C1-VL2

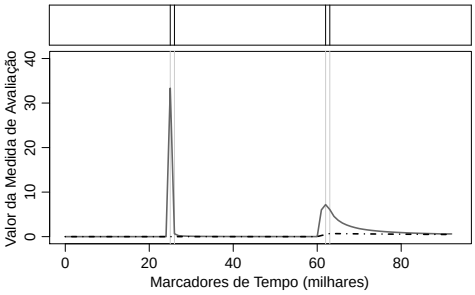
(e) Covertype-VL3

Figura 6.14: Desempenho do algoritmo MINAS-SF usando diferentes configurações e a base Covertype-V1.

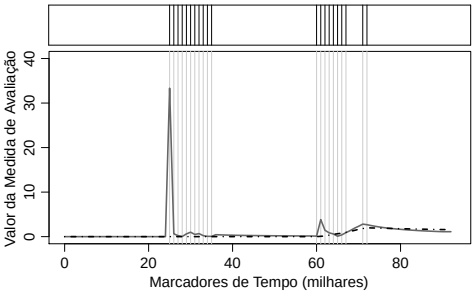
elevado. Os melhores resultados são encontrados nas versões MINAS-SF-C3 e MINAS-SF-C4, assim como para o MINAS-SF-C1-VL1, mostrados na Figura 6.6a.

6.7 Avaliação Usando os Algoritmos com *Feedback*

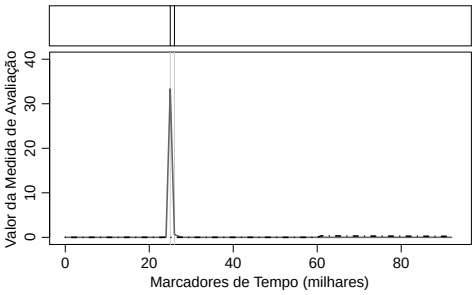
A maioria dos algoritmos encontrados na literatura para a tarefa de classificação em FCD assume que o rótulo verdadeiro de todos os exemplos do fluxo estará disponível imediatamente após sua classificação ou após um dado atraso, isto é, Tl unidades de tempo após sua chegada. Usando esses rótulos, o modelo de decisão criado pode ser constantemente atualizado, respondendo a mudanças nos conceitos alvos ou acrescentando novos



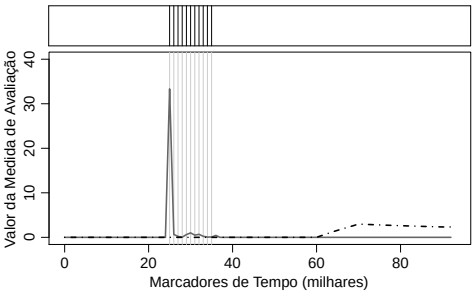
(a) CLAM-CF



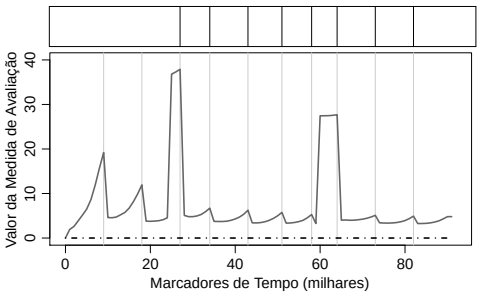
(b) CLAM-CF-V2



(c) ECSMiner-CF-KNN



(d) ECSMiner-CF-V2-KNN

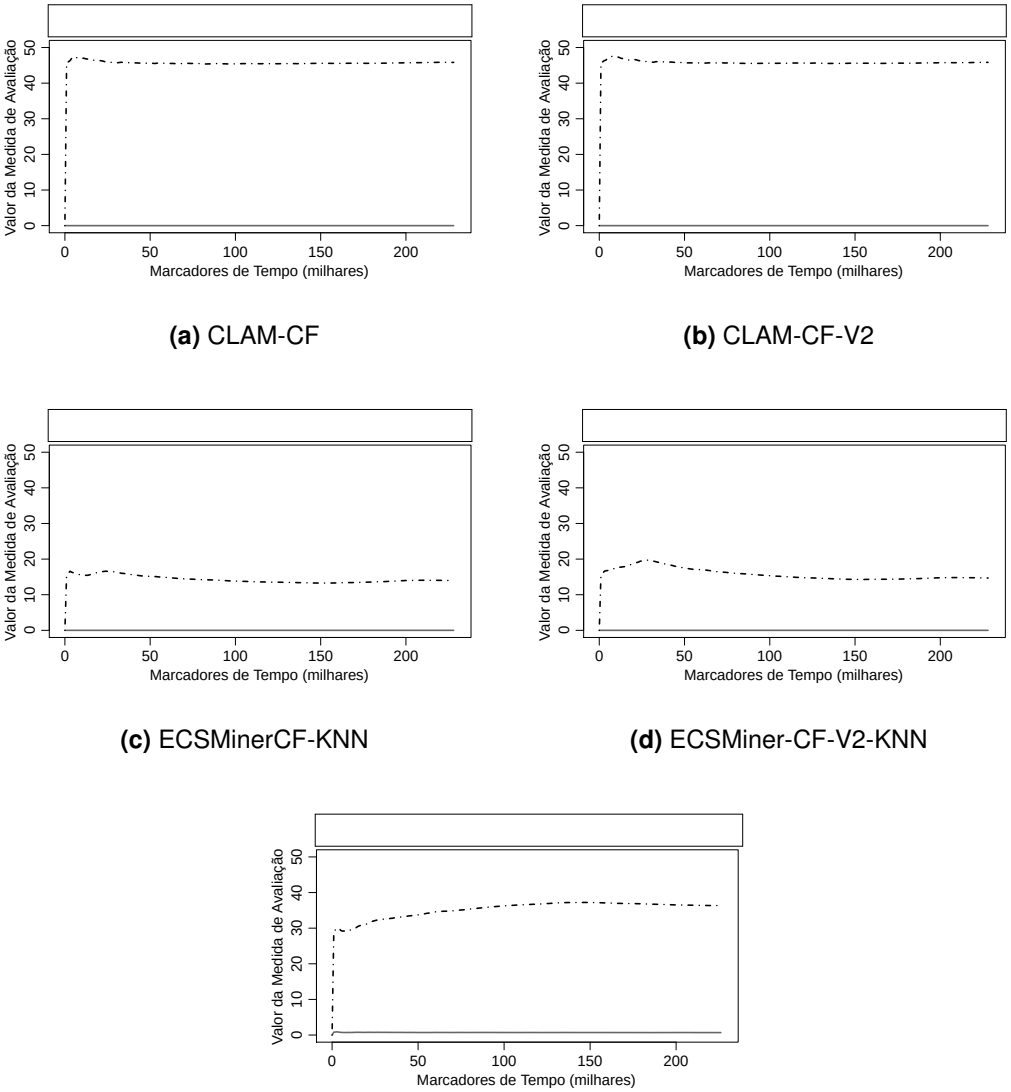


(e) MINAS-AA

Figura 6.15: Desempenho usando a base MOA.

conceitos. No entanto, sabe-se que a tarefa de rotular exemplos exige tempo e esforço, e, em muitos casos, exige a presença de um especialista de domínio. Em muitos cenários envolvendo FCD, essa tarefa torna-se impraticável. No entanto, solicitar ao especialista de domínio que rotule alguns exemplos é uma tarefa que exige menos esforço e tempo e pode contribuir para melhorar o desempenho do algoritmo.

Essa seção faz uma comparação entre os algoritmos CLAM e ECSMiner em suas versões originais, isto é usando *feedback* externo. Também é usado na comparação o algoritmo MINAS-AA, descrito na Seção 6.3.7, uma versão do algoritmo MINAS que usa aprendizado ativo. Os algoritmos utilizados nos experimentos são, MINAS-AA, CLAM-CF, ECSMiner-CF, CLAM-CF-V2, ECSMiner-CF-V2, sendo os dois últimos as versões CLAM-CF e ECMiner-CF com o parâmetro $Tl = 10.000$. As versões CLAM-CF-V2 e ECSMiner-



(a) CLAM-CF

(b) CLAM-CF-V2

(c) ECSMinerCF-KNN

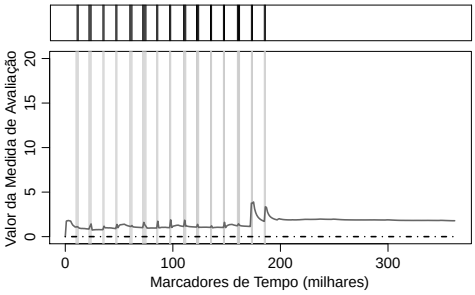
(d) ECSMiner-CF-V2-KNN

(e) MINAS-AA

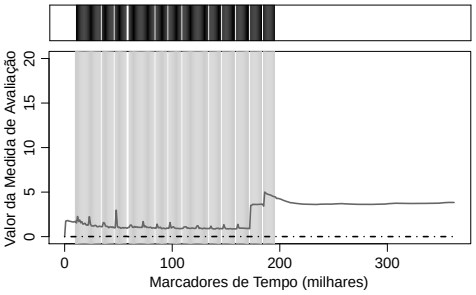
Figura 6.16: Desempenho usando a base SynD.

CF-V2 são empregadas para verificar o desempenho dos algoritmos quando os rótulos verdadeiros chegam com uma atraso maior. Os resultados dos experimentos são mostrados nas Figuras 6.15, 6.16, 6.17, 6.18 e 6.19.

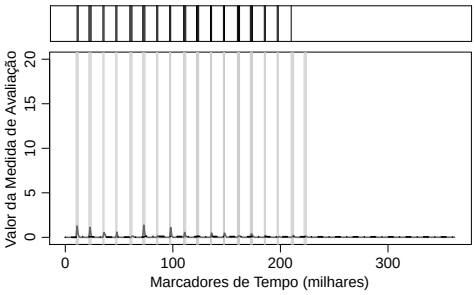
Considerando a base MOA, ver Figura 6.15, todos os algoritmos apresentaram o valor 0.00 para a medida CER na maior parte do FCD. Nas versões CLAM-CF-V2 e ECSMiner-CF-V2, nas quais o rótulo verdadeiro chega com uma atraso maior, o valor da medida CER começa a aumentar no fim do fluxo. Nota-se também que as versões CLAM-CF-V2 e ECSMiner-CF-V2 apresentaram mais novidades que as versões CLAM-CF e ECSMiner-CF. Nesse caso, como o modelo é atualizado constantemente, mas refletindo a situação do FCD de 10.000 marcadores de tempo atrás, ele não consegue classificar os modelos das classes novidade nas novas classes aprendidas, detectando assim novos padrões-



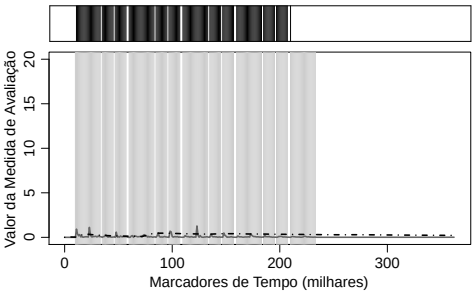
(a) CLAM-CF



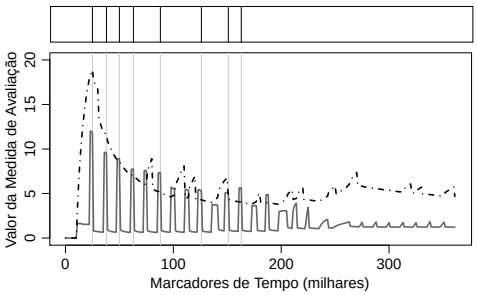
(b) CLAM-CF-V2



(c) ECSMiner-CF-KNN



(d) ECSMiner-CF-V2-KNN

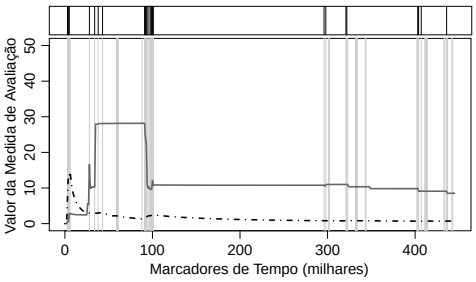


(e) MINAS-AA

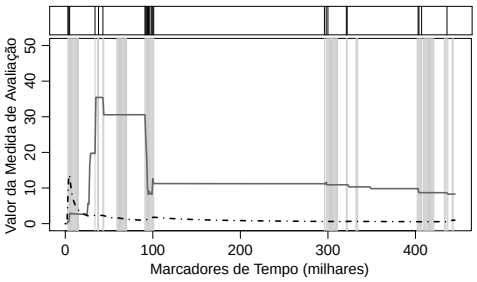
Figura 6.17: Desempenho usando a base SynEDC.

novidade. É importante destacar que o ECSMiner-CF-KNN e ECSMiner-CF-V2-KNN criaram um modelo de decisão para representar a segunda classe novidade logo após os primeiros exemplos dessa classe chegarem, o que contribuiu para que não houvesse muitos exemplos dessa classe novidade classificados como *desconhecido*. Por último, o comportamento do algoritmo MINAS com aprendizado ativo, MINAS-AA, é bem parecido com sua versão as versões do MINAS sem *feedback*, MINAS-SF, já apresentadas.

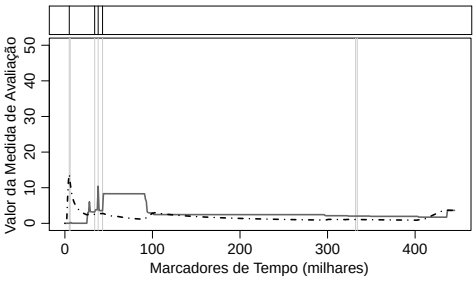
Os resultados dos experimentos com a base SynD são mostrados na Figura 6.16. Pode-se perceber que o atraso na obtenção dos rótulos verdadeiros dos exemplos não prejudicou o desempenho dos algoritmos. Também é importante verificar que o MINAS-AA obteve melhores resultados que CLAM-CF e CLAM-CF-V2. Aqui, as versões ECSMiner-CF-KNN e ECSMiner-CF-V2-KNN apresentaram os melhores resultados.



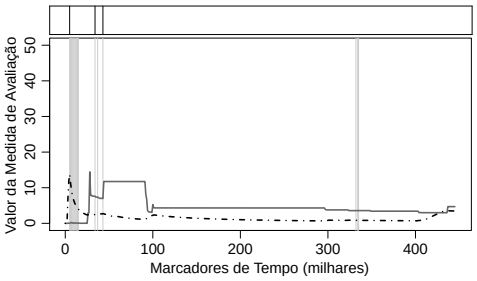
(a) CLAM-CF



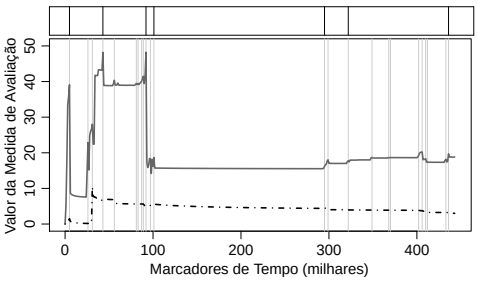
(b) CLAM-CF-V2



(c) ECSMiner-CF-KNN



(d) ECSMiner-CF-V2-KNN

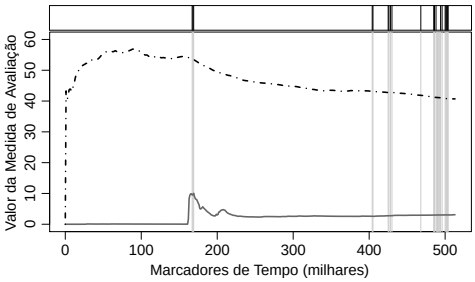


(e) MINAS-AA

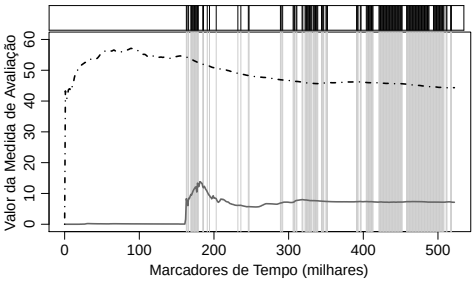
Figura 6.18: Desempenho usando a base KDD.

Os resultados dos experimentos com a base SynEDC são mostrados na Figura 6.17. Pode-se notar que o uso do *feedback* nos algoritmos CLAM e ECSMiner melhora bastante o desempenho desse algoritmos, quando comparados com a versão sem *feedback* (ver Figura 6.3). Nesses algoritmos, o atraso na chegada do rótulo verdadeiro dos exemplos provocou um aumento considerável no número de padrões-novidade, representado pelas linhas verticais no gráficos. No algoritmo MINAS, o uso do aprendizado ativo melhorou o desempenho em relação à versão sem *feedback* (ver Figura 6.3a). Essa melhora pode ser notada pela diminuição no valor da medida CER. É possível também observar que o algoritmo MINAS identificou menos padrões-novidade que os demais algoritmos.

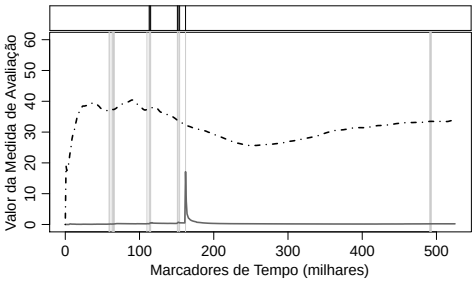
A Figura 6.18 mostra os resultados dos experimentos para a base KDD. Considerando os algoritmos CLAM e ECSMiner, não houve muitas modificações em relação à medida



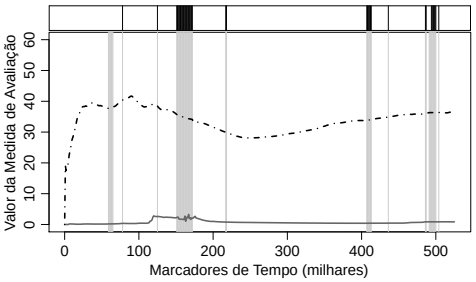
(a) CLAM-CF



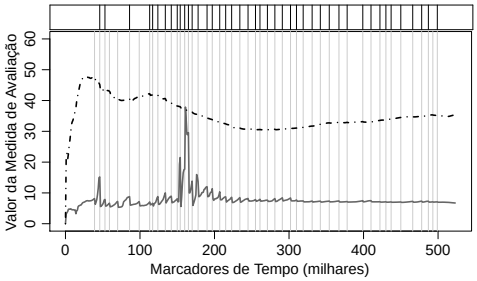
(b) CLAM-CF-V2



(c) ECSMiner-CF-KNN



(d) ECSMiner-CF-V2-KNN



(e) MINAS-AA

Figura 6.19: Desempenho usando a base Covertype-V1.

CER nas versões originais com *feedback* (CLAM-CF e ECSMiner-CF) e nas versões com *feedback* com atraso na chegada dos rótulos (CLAM-CF-V2 e ECSMiner-CF-V2). Houve apenas um pequeno aumento do valor da medida $UnkR$. Além disso os algoritmos CLAM e ECSMiner com *feedback* também apresentam poucas mudanças em relação à versão sem *feedback* (ver Figura 6.4). Considerando o algoritmo MINAS, a versão com aprendizado ativo (MINAS-AA) apresentou piores resultados para a medida de erro, CER, quando comparada com a versão sem *feedback* (MINAS-SF). A princípio, esse resultado pode parecer estranho, mas a justificativa está na forma de avaliação. Na avaliação do MINAS-SF, a cada vez que uma avaliação é realizada (a cada 1.000 marcadores de tempo), uma associação entre padrões-novidade e classes do problema é realizada, de forma a minimizar o erro do classificador. Para facilitar o entendimento, segue um exemplo. Suponha que

um padrão-novidade PN_1 contendo 50 exemplos foi criado no marcador de tempo t_1 para representar os exemplos da classe novidade C_{nov1}. No entanto, no marcador de tempo t_5 apareceram 200 exemplos da classe novidade C_{nov2} que foram incorretamente classificados como pertencentes ao padrão-novidade PN_1. Assim, usando a versão MINAS-SF, no marcador de tempo t_1, o PN_1 será associado à classe C_{nov1}, mas no marcador de tempo t_5 ele será associado à classe C_{nov2}. No entanto, usando a versão MINAS-AA, após o marcador de tempo t_1, o usuário será solicitado a dar um rótulo ao padrão-novidade PN_1 e o fará indicando que PN_1 tem como rótulo C_{nov2}. Já no marcador de tempo t_5, não haverá associação entre o PN_1 e as classes do problema, visto que um rótulo já foi dado a ele. Nesse caso, os 200 exemplos da classe C_{nov2} classificados no padrão-novidade de rótulo C_{nov1} serão contabilizados como erro.

A Figura 6.19 mostra os resultados para a base Covertype-V1. É possível perceber que as versões com *feedback* não apresentaram melhoras em relação às versões sem feedback (ver Figura 6.6). Também é possível notar que o algoritmo MINAS-AA apresenta menor taxa deerro que o algoritmo CLAM-CF e CLAM-CF-V2 e resultados semelhantes ao algoritmo ECSMiner-CF e ECSMiner-CF-V2. Assim, o uso de todos os rótulo verdadeiros para atualização do modelo não trouxe melhoras no desempenho. Pode-se notar que essa base de dados representa um problema de difícil solução, mesmo atualizando o modelo de decisão com o rótulo verdadeiro dos exemplos. Os melhores resultados para essa base foram obtidos usando o algoritmo ECSMiner e árvores de decisão (ver Figura 6.6d).

6.8 Aplicação: Detecção de Novidade em Dados de Acelerômetro

Atualmente, dispositivos móveis, por exemplo, os telefones celulares, incorporaram sensores diversos e poderosos, como sensores de GPS, de imagens, de áudio e acelerômetros (Kwapisz et al., 2011). Esses dispositivos têm tamanho reduzido, o que possibilita seu uso em diferentes posições do corpo humano. Além disso, eles possuem considerável poder computacional, podem enviar e receber dados e produzem grandes quantidades de informação em tempo real. Em especial, o uso de sensores de acelerômetro têm sido muito utilizados em tarefas de reconhecimento de atividade (Györbíró et al., 2009), (Khan et al., 2010), (Sun et al., 2010) and (Kwapisz et al., 2011).

Em computação ubíqua, reconhecer e compreender as atividades executadas por diferentes pessoas a partir de leituras de sensores é uma importante tarefa (Hao Hu et al., 2008). O reconhecimento de atividade objetiva reconhecer e predizer atividades humanas baseadas em uma série de leituras de sensores (Gil et al., 2011), (Cook et al., 2013). O reconhecimento de atividade pode ser aplicado em muitos problemas do mundo real, como monitoramento *fitness*, monitoramento de saúde, detecção de queda de idosos, compor-

tamento sensível ao contexto, automação residencial e do trabalho, etc. (Lockhart et al., 2012).

Alguns dos trabalhos envolvendo reconhecimento de atividades usam múltiplos sensores dispostos em diferentes posições do corpo (Maurer et al., 2006) (Iglesias et al., 2010), enquanto outros trabalho usam apenas um sensor de acelerômetro, em geral, disponível em telefones celulares (Ravi et al., 2005) (Lee et al., 2011). Em geral, a tarefa de reconhecimento de atividade é definida como um problema de classificação, em que as atividades humanas são divididas em um conjunto pré-definido de classes, tais como, correndo, andando, parado, descendo escadas, subindo escadas, sentando, dentre outras, cujo objetivo é classificar uma nova atividade humana em uma dessas classes. Algoritmos bem conhecidos de AM têm sido usados para a classificação, tais como, algoritmos de indução de árvores de decisão, redes neurais, SVM, KNN, etc. Além disso, a maioria desses trabalhos assume que o modelo de decisão criado na fase de treinamento é estático e o número de classes de atividades humanas é fixo, sendo previamente definido.

O reconhecimento de atividades usando acelerômetros pode ser visto como uma tarefa de classificação em FCD, uma vez que os dados chegam de forma contínua (em alta velocidade) e o fluxo de dados é infinito. Este problema apresenta como desafio, além dos requisitos de tempo e memória, o tratamento de mudanças e evolução de conceito impostas em cenários não estacionários, nos quais novas atividades humanas podem aparecer ou serem modificadas ao longo do tempo. De acordo com Iglesias et al. (2010), a tarefa de reconhecimento de atividade precisa levar em conta que o comportamento humano é muitas vezes errático e algumas vezes indivíduos comportam-se diferentemente, tornando necessário atualizar o modelo de decisão que descreve o comportamento que um indivíduo executa em uma atividade específica. Entretanto, a maioria dos estudos nesta área usa modelos de decisão estáticos.

6.8.1 Trabalhos relacionados

Há um crescente interesse no reconhecimento automático das atividades executadas por um ser humano. Muitos trabalhos foram desenvolvidos para tratar esse tema. Entretanto, a maioria deles assume que há um conjunto pré-definido de classes e não consideram que a forma de executar uma atividade pode mudar ao longo do tempo.

Em Ravi et al. (2005), o problema de reconhecimento de atividade é formulado como um problema de classificação composto por oito atividades (parado, andando, correndo, subindo escadas, descendo escadas, sentado, aspirando a casa e escovando os dentes). Os dados são coletados usando apenas um sensor de acelerômetro triaxial. Esses dados coletados são submetidos a um processo de extração de características que extrai a média, desvio padrão, energia e correlação em cada um dos três eixos, considerando janelas de tamanho 256. Essas características são os atributos usados na tarefa de classificação.

O artigo testa diferentes propostas (*bagging*, *boosting*, pluralidade de votação e *stacking*) e diferentes classificadores (tabelas de decisão, árvores de decisão, KNN, SVM e *Naive Bayes*) e escolhe a melhor técnica para o problema. No trabalho apresentando em (Lee et al., 2011), os autores propõem, além do reconhecimento de atividades humanas, um sistema para geração de informação sobre o exercício, como gasto de energia, distância percorrida e velocidade.

Em Maurer et al. (2006), os autores apresentam um sistema para identificar as atividades executadas por uma pessoa em tempo real, usando múltiplos sensores e variando a posição desses sensores no corpo. O trabalho foca em seis atividades primárias: sentando, parado, andando, subindo escadas, descendo escadas e correndo. Os valores obtidos pelos sensores de acelerômetro são divididos em pequenas janelas de tempo, sendo que cada uma delas são transformadas em atributos, usando diferentes funções características, tais como média empírica, raiz quadrada da média, desvio padrão e variância. Três diferentes métodos de classificação são comparados, árvores de decisão, KNN e *Naive Bayes*. Em Bao e Intille (2004), os autores também propõem um sistema de reconhecimento de atividade para identificar 20 atividades usando acelerômetros colocados em cinco posições diferentes do corpo do usuário.

O trabalho (Brezmes et al., 2009) usa celular convencional equipado com um único acelerômetro triaxial para reconhecer seis diferentes atividades humanas: andando, subindo escadas, descendo escadas, parado, sentado e deitado. O dispositivo móvel é usado pelo usuário sem uma orientação predefinida, para avaliar o desempenho do sistema usando o acelerômetro em diferentes orientações. Os autores também propõem usar o algoritmo KNN para classificar novos exemplos.

Em Sun et al. (2010), os autores desenvolveram um classificador SVM para reconhecer sete atividades físicas: parado, andando, correndo, andando de bicicleta, descendo escadas, subindo escadas e dirigindo. Os dados são divididos em janelas, que são então divididas em quadros. No total, 22 atributos são extraídos dos quadros de dados, com base em cinco tipos de características (média, variância, correlação, FFT e entropia no domínio da frequência), e usados na classificação.

Em Kwapisz et al. (2011), os dados colectados pelo acelerômetro são divididos em segmentos de 10 segundos, que corresponde a 200 leituras de cada um dos três eixos, e então um conjunto de características são extraídas de cada um desses segmentos. A classificação de seis atividades usa 43 atributos. Para a classificação, árvores de decisão, regressão logística e RN são usadas.

Iglesias et al. (2010) argumentam que a maioria dos modelos criados para reconhecimento de atividade não conseguem se adaptar a mudanças nos hábitos do indivíduo que desempenha a atividade. Por isso, eles propõem uma abordagem baseada em sistemas difusos que criam modelos dinâmicos que evoluem ao longo do tempo. A abordagem proposta é aplicada em sistema de reconhecimento de atividades da vida diária (*Activities of*

Daily Living (ADLs)), que incluem atividades como lavar as mãos e fazer uma chamada telefônica. Os dados usados na tarefa de classificação representam leituras de sensores coletadas em um pequeno apartamento para testes. Cada atividade é descrita por uma ou mais regras difusas, que evoluem ao longo do tempo.

Em Lopes et al. (2012) um sistema semi-supervisionado é usado para predizer quatro diferentes atividades: caminhando, correndo, sentado e ocioso. Dois diferentes classificadores foram usados *Naive Bayes* e *Hoeffding Trees*. Os novos dados rotulados pelo sistema podem ser usados para ajustar o modelo de decisão.

Apesar de muitos trabalhos terem sido criados para a tarefa de reconhecimento de atividades, poucos disponibilizam suas bases de dados originais e processadas. O trabalho de Anguita et al. (2013) cria uma base de dados de domínio público para ser usada em reconhecimento de atividade humana usando *smartphones*. A base processada contém 561 atributos, que foram extraídos para descrever cada janela de atividade, e 6 classes de atividades. O trabalho proposto por Kwapisz et al. (2011) também cria uma base de dados para reconhecimento de atividade humana. A base processada é composta por 43 atributos e 6 classes.

6.8.2 Proposta

Neste trabalho, uma nova abordagem para tratar a tarefa de reconhecimento de atividade, usando dados de sensores de acelerômetro é proposta. Para isso, algoritmos de DN para FCD são usados, a fim de reconhecer a atividade que um usuário está executando e identificar novas atividades que podem surgir ao longo do fluxo. A nova abordagem proposta apresenta importantes contribuições para a tarefa de reconhecimento de atividades.

- O reconhecimento de atividade é tratado com um problema envolvendo mudança de conceito, isso é, a forma de executar uma tarefa pode mudar ao longo do tempo. Assim, é necessário o uso de algoritmos de aprendizado incremental que possam evoluir ao longo do tempo, ajustando-se ao contexto corrente. Além disso, pode ser que o modelo de decisão inicial tenha sido criado usando uma base de dados composta por atividades de diferentes usuários. Na fase de aplicação, o sistema de classificação de atividades humanas será usado por um usuário específico, sendo necessário refinar o modelo de decisão para que ele reflita o comportamento desse usuário.

- A tarefa de reconhecimento de atividade precisa lidar com a evolução de conceitos, no qual o número de classes do problema (atividades) pode aumentar ao longo do tempo. Para isso, o algoritmo é treinado com um conjunto de classes conhecidas e, ao longo do tempo, novas classes podem ser detectadas e incorporadas ao modelo de decisão. Essa é uma importante contribuição, uma vez que montar um conjunto

de treinamento com todas as possíveis atividades executadas por um ser humano pode ser impraticável.

- Avaliação da tarefa de classificação de atividades humanas usando uma metodologia de avaliação adequada para problemas envolvendo DN multiclasse em FCD.

6.8.3 Bases de Dados e Pré-processamentos

Em geral, para a coleta de dados para problemas de reconhecimento de atividades humanas são necessários diferentes usuários, executando diferentes atividades. Um ou mais sensores de acelerômetro são usados durante a coleta dos dados. Um sensor de acelerômetro é composto pelos seguintes atributos: tempo, aceleração ao longo do eixo X, aceleração ao longo do eixo Y e aceleração ao longo do eixo Z. Em geral, os dados dos acelerômetros são coletados a cada 50ms, ou seja, há 20 coletas de dados a cada segundo.

Os trabalhos encontrados na literatura sobre reconhecimento de atividades não usam os dados brutos dos acelerômetros na tarefa de classificação. A frequência de aquisição de dados é tão rápida (20 coletas por segundo) que um exemplo, representado pelo tempo e coordenadas X, Y e Z, não contém informações suficientes para descrever cada movimento a ser considerado. Assim, os dados são divididos em janelas e um conjunto de características, como média, desvio padrão, maior valor e menor valor, são extraídas de cada janela. Essas características são os atributos a serem usados na tarefa de classificação.

Em geral, os trabalhos da literatura para reconhecimento de atividades criam suas próprias bases de dados e usam diferentes extratores de características. Há pesquisas inclusive para identificar qual o melhor conjunto de características a ser usado no processo de reconhecimento de atividades (Cui e Xu, 2013). Há trabalhos que usam apenas um acelerômetro (Kwapisz et al., 2011), (Lee et al., 2011), (Brezmes et al., 2009), enquanto outros usam informações de um acelerômetro e de um giroscópio (Anguita et al., 2013). Há também trabalhos que usam diversos acelerômetros espalhados pelo corpo do usuário (Maurer et al., 2006). Pode-se assim perceber que não há um padrão na criação das bases de dados.

Apesar de haver muitos trabalhos tratando o assunto, poucos deles disponibilizam suas bases de dados. Recentemente, alguns esforços têm sido realizados no sentido de criar bases de dados públicas que possam ser usadas em estudos comparativos de classificadores. Ainda que haja trabalhos nesse sentido, as bases de dados disponíveis são pequenas. Em geral, essas bases de dados apresentam milhões de exemplos brutos, que são os dados de um acelerômetro em três eixos. Contudo esses exemplos quando processados dão origem a uma base com apenas milhares de instâncias.

Para os experimentos, usados nessa tese, uma base de dados públicas para reconheci-

mento de atividade usando acelerômetros foi utilizada. Os detalhes da base são descritos a seguir.

- **WISDM - projeto *Wireless Sensor Data Mining*:**
  Essa base foi usada em Kwapisz et al. (2011)[4]. A base é composta por dados obtidos de 29 usuários que carregam um *smartphone* com um sistema baseado em *Android*, enquanto desempenhavam um conjunto específico de atividades. A base é composta por dados de apenas um acelerômetro e contém seis possíveis classes (atividades): caminhando, correndo lento, subindo escadas, descendo escadas, sentado e parado. A base de dados bruta contém 1.098.027 exemplos. A base processada é composta por 46 atributos e 5.424 exemplos. A base de treino usada contém 10% dos dados, e possui exemplos de 4 atividades. A base de teste contém exemplos das 6 atividades.

6.8.4 Algoritmos Usados e Configurações

Para os experimentos, foram usados os algoritmos MINAS-SF, ECSMiner-SF e CLAM-SF. Também foi utilizado um algoritmo clássico de AM para cenários estáticos, o algoritmo de indução de árvores de decisão, que é um algoritmo comumente utilizado nos trabalhos de classificação de atividades humanas usando acelerômetros.

As configurações dos algoritmos são descritas a seguir. Como as bases de dados possuem poucos exemplos em relação às bases comumente usadas em FCD, que contém centenas de milhares de exemplos, foi necessário alterar as configurações dos algoritmos descritas na Seção 6.3.

- **MINAS:** As configurações usadas são semelhantes à Configuração 1-VL1 (ver Tabela 6.4), exceto pelo fato de que o parâmetro $NroExDN$ foi configurado com o valor 20, o algoritmo utilizado foi o KMeans, com K igual a 10.

- **ECSMiner:** As configurações usadas para o ECSMiner são: Tamanho da janela = 300, Número de janelas de treinamento = 2, Tc = 80, Tl = 200, nro de classificadores = 3, número de exemplos para execução da DN = 20, número de grupos = 10 e q = 20.

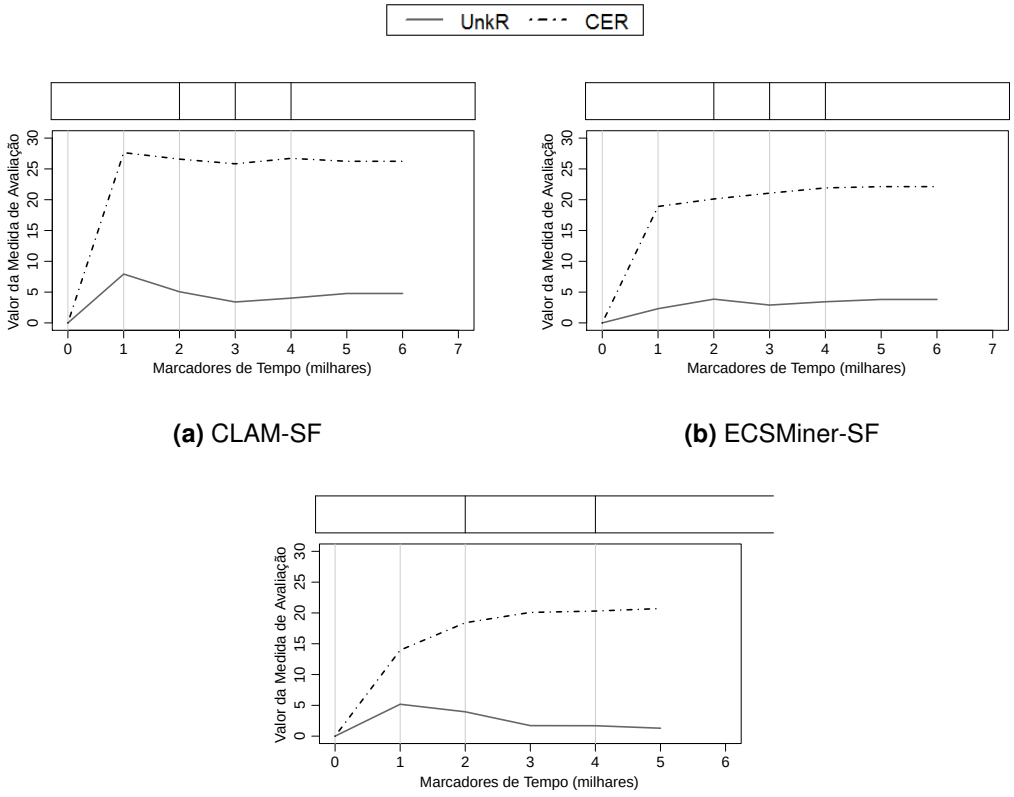
- **CLAM:** Mesmas configurações do ECSMiner.

- **Algoritmo de Indução de Árvore de decisão:** Foi utilizado o algoritmo J48, que implementa o algoritmo C4.5, disponível na ferramenta Weka (Hall et al., 2009).

[4] A base está disponível em: http://www.cis.fordham.edu/wisdm/dataset.php

6.8.5 Resultados dos Experimentos

Os resultados dos experimentos usando a base WISDM e os algoritmos CLAM-SF, ECSMiner-SF e MINAS-SF são mostrados na Figura 6.20. Pode-se perceber que o algoritmo MINAS e o ECSMiner apresentaram os menores valores para a taxa CER.



(a) CLAM-SF

(b) ECSMiner-SF

(c) MINAS-SF

Figura 6.20: Desempenho dos algoritmo usando a base WISDM.

A Tabela 6.7 mostra a matriz de confusão final obtida usando o algoritmo MINAS e usando o algoritmo J48 para indução de árvores de decisão. Pode-se perceber que, o uso de abordagens *batch* para a tarefa de classificação de atividades humanas, não produz bons resultados. Isso acontece porque o modelo de decisão é induzido a partir de um conjunto de classes fixas e se exemplos pertencentes a novas classes do problema surgem ao longo do FCD, eles serão erroneamente classificados nas classes conhecidas. Já o algoritmo MINAS consegue identificar padrões-novidade para representar as classes novidade. Na Tabela 6.7(b), $\mathbf{PN}_{nov1}$ e $\mathbf{PN}_{nov2}$ representam o conjunto de padrões-novidades detectados pelo algoritmo MINAS que foram associados à classe novidade $\mathbf{C}_{nov1}$ e $\mathbf{C}_{nov2}$ respectivamente. É importante também observar que o algoritmo MINAS, nas classes conhecidas do problema, obteve um desempenho semelhante às árvores de decisão.

Tabela 6.7: Matriz de confusão para a base WISDM.

	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$
$\mathbf{C}_1$	1194	243	176	220
$\mathbf{C}_2$	112	1327	23	6
$\mathbf{C}_3$	211	92	203	63
$\mathbf{C}_4$	220	52	106	77
$\mathbf{C}_{nov1}$	5	6	288	7
$\mathbf{C}_{nov2}$	24	28	65	129

(a) Árvore de Decisão

	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	$\mathbf{PN}_{nov1}$	$\mathbf{PN}_{nov2}$	$\mathbf{Unk}$
$\mathbf{C}_1$	1483	109	96	140	0	0	5
$\mathbf{C}_2$	73	1356	9	14	0	0	16
$\mathbf{C}_3$	283	61	108	111	0	0	6
$\mathbf{C}_4$	270	19	58	107	0	0	0
$\mathbf{C}_{nov1}$	30	0	14	9	177	71	5
$\mathbf{C}_{nov2}$	46	0	14	8	3	166	9

(b) MINAS

6.9 Considerações Finais

Esse capítulo apresentou um conjunto de experimentos usando o algoritmo MINAS e os principais algoritmos da literatura para DN em FCD. Os experimentos foram realizados utilizando a metodologia proposta neste trabalho e a metodologia proposta na literatura. Experimentos com bases de dados artificiais e reais foram executados. Por último, uma aplicação para a tarefa de DN em FCD, o reconhecimento de atividades humanas usando dados de acelerômetros, é apresentada.

Os experimentos mostram o potencial do algoritmo e das metodologias propostas, bem como os cenários nos quais eles apresentam limitações. O próximo capítulo apresenta as principais contribuições desta tese, discute as limitações das técnicas propostas e os principais trabalhos futuros a serem desenvolvidos.

CAPÍTULO

7

Conclusões

7.1 Introdução

Esse capítulo apresenta as principais conclusões, contribuições e limitações deste trabalho. Também são apresentados temas que podem ser abordados em trabalhos futuros.

7.2 Contribuições

Este trabalho apresentou um novo algoritmo para tratar DN em FCDs. As principais diferenças desse algoritmo em relação aos algoritmos da literatura são:

- Atualiza o modelo de decisão ao longo do FCD sem usar *feedback* externo ou com *feedback* parcial, isto é, apenas alguns exemplos são rotulados ao longo do FCD;
- Considera a DN como uma tarefa multiclasse, na qual diferentes padrões-novidade são criados para representar as diferentes classe novidade que podem surgir ao longo do FCD;
- Uso de um modelo de decisão único representando todo o conhecimento aprendido até o momento.

Este trabalho também propôs uma nova metodologia para avaliar a tarefa de DN em FCDs.

As principais contribuições da metodologia de avaliação são:

- Apresentar a matriz de confusão gerada para a tarefa de DN em FCD sob uma nova perspectiva que considera que diferentes classes novidades devem estar associadas a diferentes padrões-novidade.
- Propor uma estratégia para transformar a matriz de confusão retangular, na qual o número de padrões-novidades é diferente do número de classes novidade, para uma matriz de confusão quadrada. Essa estratégia consiste em associar os padrões-novidade às classes do problema.
- Considerar a avaliação dos exemplos marcados como *desconhecidos* separadamente das medidas de erro e acerto.
- Propor o uso de medidas de avaliação multiclasse, uma vez que a tarefa de DN em FCDs pode ser considerada como uma tarefa de classificação multiclasse.

As principais contribuições do trabalho são descritas a seguir.

- **Proposta de uma organização dos termos e definições da área:** A área de FCDs é relativamente recente, mas muitos trabalhos já foram publicados sobre o tema. No entanto, poucos trabalhos formalizam o problema, termos e definições ou propõem taxonomias para classificar os trabalhos da literatura. Este trabalho resultou em dois *surveys* sobre FCD. Um deles já foi publicado e trata da tarefa de agrupamento em FCD. O outro está em processo de análise e trata do problema de DN em FCD. As propostas de taxonomias da literatura classificam os trabalhos sobre DN somente sob a perspectiva do método utilizado e, em geral, só tratam o problema em cenários *batch*. No entanto, muitas outras variações precisam ser analisadas em FCDs, tais como, o uso de exemplos rotulados para atualizar o modelo de decisão, estratégias para esquecimento do passado, tratamento de ruídos, tratamento de contextos recorrentes e metodologias de avaliação.
- **Investigação da DN em FCDs como uma tarefa de classificação multiclasse:** A maioria dos trabalhos da literatura trata a DN com uma tarefa de classificação binária, cujo objetivo é distinguir os exemplos normais e não normais. No entanto, em muitos problemas o conjunto de dados de exemplos normais pode conter exemplos de diferentes classes, assim como diferentes classes novidade podem aparecer ao longo do tempo. Para isso, este trabalho apresentou uma nova abordagem para o problema, que considera que o conjunto de dados usado na fase *offline*, para o treinamento de um modelo de decisão que represente o conceito conhecido do problema, pode ser composto por uma ou mais classes. Além disso, na fase *online*, não basta que exemplos das classes novidade sejam classificados apenas como novidade. É necessário distinguir entre diferentes padrões-novidade que possam surgir para representar cada classe novidade. Essa abordagem exige uma solução bem

mais complexa do que apenas distinguir entre exemplos normais e não normais e representa um avanço em relação ao que tem sido proposto na literatura.

- **Atualização do modelo de decisão sem usar *feedback* externo:** A maioria dos trabalhos encontrados na literatura para classificação em FCD assume que o rótulo verdadeiro de todos os exemplos estará disponível para a atualização do modelo de decisão. Alguns trabalhos consideram que o rótulo estará disponível imediatamente após sua classificação; outros consideram um atraso na chegada do rótulo verdadeiro. Uma vez que rotular os exemplos é uma tarefa custosa e que demanda tempo, isso pode-se tornar impraticável. O algoritmo MINAS possibilita a atualização do modelo de decisão sem usar o rótulo verdadeiro dos exemplos, mantendo um desempenho comparável aos algoritmos da literatura. No entanto, se o rótulo verdadeiro de alguns exemplos está disponível, eles podem ser usados para melhorar o desempenho preditivo do classificador.

- **Desenvolvimento de um modelo de decisão integrado:** Muitos trabalhos da literatura dão muita ênfase na separação entre o normal e a novidade. Assim, alguns trabalhos criam modelos de decisão separados para tratar o conceito normal, novidade e extensão. O algoritmo MINAS sugere o uso de um modelo de decisão integrado, composto pelos micro-grupos aprendidos na fase de treinamento *offline*, pelas extensões dos conceitos e padrões-novidades aprendidos de forma *online*. O modelo de decisão é constantemente atualizado para refletir o conhecimento que se têm sobre o problema. As classes que em um dado momento são novidades, podem futuramente se tornar conceitos conhecidos. Além disso, alguns conceitos podem se tornar obsoletos e precisam ser esquecidos. Isso reflete a forma como o aprendizado humano funciona.

- **Nova metodologia de avaliação:** Poucos trabalhos da literatura tratam o problema de avaliação dos classificadores para DN em FCDs. Em geral, as estratégias de avaliação propostas para cenários *batch* são usadas nos cenários de FCDs. Além disso, medidas binárias de avaliação são usadas, uma vez que a tarefa de DN é considera binária. Este trabalho propôs uma nova metodologia de avaliação que trata vários aspectos da matriz de confusão gerada para cenários de FCDs. O número de colunas da matriz de confusão, que representam as classes preditas pelo algoritmo, aumentam ao longo do tempo, sempre que um novo padrão-novidade é identificado. O número de colunas da matriz não é o mesmo que o número de linhas, uma vez que uma classe pode ser representada por um ou mais padrões-novidade. Os exemplos marcados com o perfil *desconhecido* também fazem parte da matriz de confusão e devem ser avaliados. A metodologia de avaliação proposta neste trabalho trata todas essas características da matriz de confusão e pode ser usada tanto em problemas

binários, quanto multiclasse. Também pode ser usada tanto em cenários que atualizam o modelo de decisão usando o rótulo verdadeiro dos exemplos como quando o rótulo não está disponível. Este trabalho representa um primeiro passo na mudança de paradigma para avaliação da matriz de confusão gerada em FCDs. Essa é uma metodologia genérica que pode ser usada em conjunto com as medidas de avaliação comumente empregadas na literatura.

7.3 Limitações

Algumas das limitações deste trabalho são descritas a seguir.

- **Parametrização do algoritmo:** Essa é uma das limitações apresentadas pelo algoritmo MINAS, bem como pelos algoritmos da literatura para DN em FCDs. Muitos parâmetros devem ser configurados, tais como, número de grupos usado para representar cada classe, quando realizar o processo de DN, critérios relativos à validação de um novo grupo, limiar para separar um novo padrão-novidade de uma extensão e tamanho da janela da dados atual. Em geral, os valores dos parâmetros podem variar de uma base para outra. Encontrar os melhores valores de parâmetros para uma base de dados não é uma tarefa fácil. Além disso, nos experimentos pode-se notar que muitas vezes o desempenho preditivo do classificador piora consideravelmente quando os parâmetros não são configurados adequadamente. Em especial, notou-se que a escolha apropriada do limiar para separar extensões de novidades é um parâmetro crítico para o desempenho do algoritmo.

- **Modelo baseado em hiperesferas:** Em alguns dos experimentos realizado foi possível notar que o uso do modelo de decisão baseado em árvores de decisão apresentou melhores resultados do que o modelo de decisão baseado em micro-grupos hiperesféricos. No entanto, o modelo baseado em micro-grupos possibilita sua atualização constante, mesmo sem ter o rótulo verdadeiros dos exemplos. No caso do uso de árvores de decisão, isso não seria possível. Outro ponto a ser destacado é que, em alguns experimentos, notou-se que pode haver sobreposição entre os grupos criados na fase *offline*, sendo que não há um tratamento específico para essa situação. Na fase *online*, se um novo micro-grupo válido intercepta um micro-grupo existente, ele é considerado uma extensão e não um padrão-novidade.

- **Técnica para tratamento de mudança de conceito:** Uma das limitações do trabalho diz respeito às técnicas para tratamento de mudança de conceito. Em geral, evoluções graduais nos conceitos já aprendidos são tratadas atualizando o modelo de decisão constantemente, quer pela inserção de micro-grupos identificados como extensões, quer pelo esquecimento de micro-grupos não ativos. No entanto, mudan-

ças abruptas nos conceitos conhecidos nem sempre são identificadas corretamente pelo algoritmo.

- **Metodologia de avaliação:** A metodologia de avaliação apresentada neste trabalho representa uma contribuição na busca por metodologias de avaliação adequadas para os algoritmos que tratam FCD. Ela possibilita a avaliação de algoritmos para DN em FCD que produzem matrizes de confusão retangulares, que incrementam ao longo do tempo e que usam a estratégia de marcar os exemplos não explicados pelo modelo de decisão como *desconhecidos*. No entanto, a metodologia de avaliação proposta precisa ser refinada. Um dos problemas já identificados é que ela não penaliza um classificador quando muitos padrões-novidades são identificadas para representar uma classe. Quanto mais padrões-novidades um classificador cria, mas chances há de que ele consiga separar bem as classes novidades. No entanto, é preciso ponderar entre a taxa de erro versus o número de novidades criadas.

7.4 Trabalhos Futuros

Alguns dos possíveis trabalhos futuros são descritos a seguir:

- **Desenvolvimento de novas estratégias para o cálculo do limiar:** A escolha do limiar que separa extensões das novidades é um fator importante no desempenho do algoritmo MINAS. Novas estratégias para calcular automaticamente este valor devem ser estudadas.

- **Estudos sobre novas forma de uso do aprendizado ativo:** O uso de aprendizado ativo em tarefas de DN em FCDs parece ser uma boa alternativa. Solicitar o rótulo verdadeiro de apenas alguns exemplos do fluxo pode aumentar o desempenho preditivo do classificador sem exigir muito esforço na rotulação. Neste trabalho, foi usado uma técnica simples de aprendizado ativo que solicita ao usuário o rótulo do centróide de um novo micro-grupo. Outras técnicas de aprendizado ativo devem ser investigadas, a fim de melhorar o processo de escolha do exemplo ou conjunto de exemplos a serem rotulados. Também devem ser investigadas novas formas de atualizar o modelo de decisão usando essa informação.

- **Tratamento de Ruídos e *Outliers*:** O tratamento de ruídos e *outliers* presente no MINAS elimina grupos com poucos exemplos na fase *offline* após a execução do algoritmo CluStream, e valida micro-grupos novos exigindo que eles sejam coesos e possuam um número mínimo de exemplos. Novas técnicas para tratamento de ruídos e *outliers* devem ser estudadas, assim como testes mais elaborados devem ser executados a fim de identificar os cenários nos quais o algoritmo MINAS confunde o aparecimento de novas classes com ruídos e *outliers* presentes nos dados.

- **Tratamento de Contextos Recorrentes:** Contextos recorrentes são uma característica comum aos cenários envolvendo FCD. O algoritmo MINAS trata contextos recorrentes usando uma memória temporária, chamada *memória sleep*, que guarda os micro-grupos que ficam muito tempo sem classificar novos grupos. Esses micro-grupos podem se tornar ativos novamente, indicando a presença de um contexto recorrente. Testes mais elaborados devem ser realizados a fim de avaliar o desempenho do algoritmo MINAS no tratamento de cenários recorrentes.

- **Aprofundar nos estudas relativos à metodologia de avaliação:** Um primeiro passo foi dado no sentindo de propor uma nova metodologia de avaliação que atende às características presentes na matriz de confusão, gerada em problemas envolvendo DN em FCDs. Novas medidas e metodologias de avaliação para FCD precisam ser investigadas, uma vez que as medidas e metodologias existentes para cenários *batch* não podem ser diretamente aplicadas a FCD.

- **Criação de novas bases de dados experimentais:** Poucas bases de dados reais e artificiais estão disponíveis para serem usadas em experimentos envolvendo FCDs. Faltam bases que simulem contextos recorrentes, presença de ruídos, mudança de conceito e surgimento de novas classes. A criação de um repositório de dados que possa ser usado na validação de experimentos com algoritmos para FCD pode trazer importantes benefícios para a pesquisa na área.

Referências

ACKERMANN, M. R.; MÄRTENS, M.; RAUPACH, C.; SWIERKOT, K.; LAMMERSEN, C.; C., S. Streamkm++: A clustering algorithms for data streams. *ACM Journal of Experimental Algorithmics*, v. 17, n. 1, 2012.

AGGARWAL, C. C. *Data streams: Models and algorithms (advances in database systems)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

AGGARWAL, C. C. *Outlier analysis*. Springer, 2013.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. On demand classification of data streams. In: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, New York, NY, USA: ACM, 2004, p. 503–508.

AGGARWAL, C. C.; JIAWEI HAN, J. W.; YU, P. S. A framework for clustering evolving data streams. In: *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03)*, VLDB Endowment, 2003, p. 81–92.

AHMED, T.; COATES, M. Multivariate online anomaly detection using kernel recursive least squares. In: *Proceedings of the 26th IEEE International Conference on Computer Communications*, IEEE, 2007, p. 625–633.

AL-KHATEEB, T.; MASUD, M. M.; KHAN, L.; AGGARWAL, C.; HAN, J.; THURAISINGHAM, B. Stream classification with recurring and novel class detection using class-based ensemble. In: *Proceedings of the 12nd IEEE International Conference on Data Mining (ICDM '12)*, Washington, DC, USA: IEEE Computer Society, 2012a, p. 31–40.

AL-KHATEEB, T. M.; MASUD, M. M.; KHAN, L.; THURAISINGHAM, B. Cloud guided stream classification using class-based ensemble. In: *Proceedings of the 5th IEEE International Conference on Computing (CLOUD'12)*, Washington, DC, USA: IEEE Computer Society, 2012b, p. 694–701.

ALBERTINI, M. K.; MELLO, R. F. A self-organizing neural network for detecting novelties. In: *Proceedings of the 2007 ACM Symposium on Applied computing (SAC '07)*, New York, NY, USA: ACM, 2007, p. 462–466.

ANGUITA, D.; GHIO, A.; ONETO, L.; PARRA, X.; REYES-ORTIZ, J. L. A public domain dataset for human activity recognition using smartphones. In: *Proceedings of the 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'13)*, 2013, p. 437–442.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.; SANDER, J. Optics: ordering points to identify the clustering structure. *SIGMOD Record*, v. 28, p. 49–60, 1999.

AREGUI, A.; DENŒUX, T. Fusion of one-class classifiers in the belief function framework. In: *Proceedings of the 10th International Conference on Information Fusion*, IEEE, 2007, p. 1–8.

BABCOCK, B.; BABU, S.; DATAR, M.; MOTWANI, R.; WIDOM, J. Models and issues in data stream systems. In: *Proceedings of the 21 Symposium on Principles of Database Systems*, New York, NY, USA: ACM, 2002, p. 1–16.

BAO, L.; INTILLE, S. S. Activity recognition from user-annotated acceleration data. *Pervasive Computing: Lecture Notes Computer Science*, v. 3001, p. 1–17, 2004.

BARBARÁ, D. Requirements for clustering data streams. *SIGKDD Explorations*, v. 3, n. 2, p. 23–27, 2002.

BARTOSZ KRAWCZYK, M. W. Incremental learning and forgetting in one-class classifiers for data streams. In: *Proceedings of the 8th International Conference on Computer Recognition Systems (CORES' 13), Advances in Intelligent Systems and Computing Volume 226*, 2013, p. 319–328.

BICEGO, M.; FIGUEIREDO, M. A. T. Soft clustering using weighted one-class support vector machines. *Pattern Recognition*, v. 42, n. 1, p. 27–32, 2009.

BIFET, A.; HOLMES, G.; B.PFAHRINGER; KRANEN, P.; H.KREMER; JANSEN, T.; SEIDL, T. MOA: Massive online analysis, a framework for stream classification and clustering. v. 11, p. 44–50, 2010.

BIFET, A.; PFAHRINGER, B.; READ, J.; HOLMES, G. Efficient data stream classification via probabilistic adaptive windows. In: *Proceedings of the 28th ACM Symposium on Applied Computing (SAC'13)*, New York, NY, USA: ACM, 2013, p. 801–806.

BOX, G. E. P.; JENKINS, G. *Time series analysis: Forecasting and control*. 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994.

BREIMAN, L.; FRIEDMAN, J.; STONE, C. J.; OLSHEN, R. A. *Classification and Regression Trees*. 1 ed. Chapman & Hall/CRC, 1984.

BREZMES, T.; GORRICHO, J.-L.; COTRINA, J. Activity recognition from accelerometer data on a mobile phone. In: *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living (IWANN '09)*, Berlin, Heidelberg: Springer-Verlag, 2009, p. 796–799.

CAO, F.; ESTER, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: *Proceedings of the SIAM Conference on Data Mining*, SIAM, 2006, p. 328–339.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM Computing Surveys*, v. 41, n. 3, p. 15:1–15:58, 2009.

CHANG, J. H.; LEE, W. S. estwin: Online data stream mining of recent frequent itemsets by sliding window method. *Journal Informaton Science*, v. 31, n. 2, p. 76–90, 2005.

CHEN, L.; ZOU, L.; TU, L. Stream data classification using improved fisher discriminate analysis. *Journal of Computers*, v. 4, n. 3, p. 208–214, 2009.

CHEN, Y.; TU, L. Density-based clustering for real-time stream data. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 133–142, 2007.

CHOW, C. On optimum recognition error and reject tradeoff. *Information Theory, IEEE Transactions on*, v. 16, n. 1, p. 41–46, 1970.

CLIFTON, L. A.; YIN, H.; ZHANG, Y. Support vector machine in novelty detection for multi-channel combustion data. In: *Proceedings of the 3rd International Conference on Advances in Neural Networks - Volume Part III (ISNN'06)*, Berlin, Heidelberg: Springer-Verlag, 2006, p. 836–843.

COOK, D.; FEUZ, K. D.; KRISHNAN, N. C. Transfer learning for activity recognition: a survey. *Knowledge and Information Systems*, v. 36, n. 3, p. 537–556, 2013.

COULL, S.; BRANCH, J.; SZYMANSKI, B.; BREIMER, E. Intrusion detection: A bioinformatics approach. In: *Proceedings of 19th International Conference on Computer Security Applications (ACSAC 2003), Las Vegas, Nevada, USA*, Washington, DC, USA: IEEE Computer Society, 2003, p. 24–33.

CUI, J.; XU, B. Cost-effective activity recognition on mobile devices. In: *Proceedings of the 8th International Conference on Body Area Networks (BodyNets '13)*, ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, p. 90–96.

DAWID, A. P. Statistical theory: the prequential approach (with discussion). *Journal of the Royal Statistical Society A*, v. 147, p. 278–292, 1984.

DENIS, F.; GILLERON, R.; LETOUZEY, F. Learning from positive and unlabeled examples. *Theoretical Computer Science*, v. 348, n. 1, p. 70–83, 2005.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: *Proceedings of the ACM 6th International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, 2000, p. 106–113.

DRIES, A.; RÜCKERT, U. Adaptive concept drift detection. *Statistical Analysis and Data Mining*, v. 2, n. 56, p. 311–327, 2009.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, v. 22, n. 10, p. 1517–1531, 2011.

ESTER, M.; KRIEGEL, H.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, AAAI Press, 1996, p. 226–231.

FARIA, E.; GONCALVES, I.; GAMA, J.; CARVALHO, A. Evaluation methodology for multiclass novelty detection algorithms. In: *2nd Brazilian Conference on Intelligent Systems (BRACIS'13)*, IEEE Computer Society, 2013a, p. 19–25.

FARIA, E. R.; GAMA, J.; CARVALHO, A. C. P. L. F. Novelty detection algorithm for data streams multi-class problems. In: *Proceedings of the 28th Symposium on Applied Computing (ACM SAC'13)*, New York, NY, USA: ACM, 2013b, p. 795–800.

FARID, D. M.; RAHMAN, C. M. Novel class detection in concept-drifting data stream mining employing decision tree. In: *7th International Conference on Electrical Computer Engineering (ICECE' 2012)*, Tarrytown, NY, USA: Pergamon Press, Inc., 2012, p. 630–633.

FARID, D. M.; ZHANG, L.; HOSSAIN, A.; RAHMAN, C. M.; STRACHAN, R.; SEXTON, G.; DAHAL, K. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems and Applications*, v. 40, n. 15, p. 5895–5906, 2013.

FISHER, D. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, v. 2, p. 139–172, 1987.

FRANK, A.; ASUNCION, A. UCI machine learning repository. 2010. Disponível em: `http://archive.ics.uci.edu/ml`

GABER, M. M.; ZASLAVSKY, A.; KRISHNASWAMY, S. Mining data streams: a review. *SIGMOD Rec.*, v. 34, n. 2, p. 18–26, 2005.

GAMA, J. *Knowledge discovery from data streams*. CRC Press Chapman Hall., 2010.

GAMA, J.; GABER, M. M. *Learning from data streams: Processing techniques in sensor networks*. Springer, 2007.

GAMA, J.; RODRIGUES, P. P. An overview on mining data streams. In: *Foundations of Computational Intelligence - Volume 6: Data Mining*, v. 206 de *Studies in Computational Intelligence*, Berlin: Springer, p. 29–45, 2009.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Machine Learning*, v. 90, n. 3, p. 317–346, 2013.

GAMA, J. A.; ROCHA, R.; MEDAS, P. Accurate decision trees for mining high-speed data streams. In: *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, New York, NY, USA: ACM, 2003, p. 523–528.

GAUGHAN, G.; SMEATON, A. F. Finding new news: novelty detection in broadcast news. In: *Proceedings of the 2nd Asia Conference on Asia Information Retrieval Technology (AIRS'05)*, Berlin, Heidelberg: Springer-Verlag, 2005, p. 583–588.

GIL, G. B.; BERLANGA, A.; MOLINA, J. M. Physical actions architecture: Context-aware activity recognition in mobile devices. In: *User-Centric Technologies and Applications*, v. 94 de *Advances in Intelligent and Soft Computing*, Springer Berlin Heidelberg, p. 19–27, 2011.

GOGOI, P.; BHATTACHARYYA, D.; BORAH, B.; KALITA, J. K. A survey of outlier detection methods in network anomaly identification. *Computer Journal*, v. 54, n. 4, p. 570–588, 2011.

GUHA, S.; MEYERSON, A.; MISHRA, N.; MOTWANI, R.; O'CALLAGHAN, L. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, v. 15, p. 515–528, 2003.

GUHA, S.; MISHRA, N.; MOTWANI, R.; L., O. Clustering data streams. In: *Annual IEEE Symposium on Foundations of Computer Science*, Piscataway, NJ, USA: IEEE Educational Activities Department, 2000, p. 359–366.

GUHA, S.; RASTOGI, R.; SHIM, K. Cure: an efficient clustering algorithm for large databases. In: *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, ACM, 1998, p. 73–84.

GUHA, S.; RASTOGI, R.; SHIM, K. Rock: A robust clustering algorithm for categorical attributes. In: *Proceedings of the 15th International Conference on Data Engineering (ICDE'99)*, Washington, DC, USA: IEEE Computer Society, 1999, p. 512–537.

GYÖRBÍRÓ, N.; FÁBIÁN, Á.; HOMÁNYI, G. An activity recognition system for mobile phones. *Mobile Networks and Applications*, v. 14, n. 1, p. 82–91, 2009.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, v. 11, n. 1, p. 10–18, 2009.

HAN, J. *Data mining: Concepts and techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

HANCZAR, B.; DOUGHERTY, E. R. Classification with reject option in gene expression data. *Bioinformatics*, v. 24, n. 17, p. 1889–1895, 2008.

HAO HU, D.; PAN, S. J.; ZHENG, V. W.; LIU, N. N.; YANG, Q. Real world activity recognition with multiple goals. In: *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp '08)*, New York, NY, USA: ACM, 2008, p. 30–39.

HAYAT, M.; BASIRI, J.; SEYEDHOSSEIN, L.; SHAKERY, A. Content-based concept drift detection for email spam filtering. In: *Proceedings of the 5th International Symposium on Telecommunications (IST'10)*, 2010, p. 531–536.

HAYAT, M. Z.; HASHEMI, M. R. A DCT based approach for detecting novelty and concept drift in data streams. In: *International Conference on Soft Computing and Pattern Recognition (SoCPaR)*, -: IEEE, 2010, p. 373–378.

HODGE, V.; AUSTIN, J. A survey of outlier detection methodologies. *Artificial Intelligence Review*, v. 22, n. 2, p. 85–126, 2004.

HOFFMANN, H. Kernel PCA for novelty detection. *Pattern Recognition*, v. 40, n. 3, p. 863–874, 2007.

IGLESIAS, J. A.; ANGELOV, P. P.; LEDEZMA, A.; SANCHÍS, A. Human activity recognition based on evolving fuzzy systems. *International Journal of Neural Systems*, v. 20, p. 355–364, 2010.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 1988.

JUSZCZAK, P.; DUIN, R. P. W. Combining one-class classifiers to classify missing data. In: *Multiple Classifier Systems*, Lecture Notes in Computer Science, Hildeberg, Berlin: Springer, 2004, p. 92–101 (*Lecture Notes in Computer Science*,).

KARNICK, M.; AHISKALI, M.; MUHLBAIER, M.; POLIKAR, R. Learning concept drift in nonstationary environments using an ensemble of classifiers based approach. In: *IEEE International Joint Conference on Neural Networks (IJCNN'08)*, 2008, p. 3455–3462.

KARYPIS, G.; HAN, E.; KUMAR, V. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, v. 32, n. 8, p. 651–666, 1999.

KATAKIS, I.; TSOUMAKAS, G.; VLAHAVAS, I. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge Information Systems*, v. 22, n. 3, p. 371–391, 2010.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2005.

KHALILIAN, M.; MUSTAPHA, N. Data stream clustering: challenges and issues. In: *Proceedings of the International Conference MultiConference of Engineers and Computer Scientists (IMECS)*, 2010, p. 978–998.

KHAN, A. M.; LEE, Y. K.; LEE, S.; KIM, T. S. Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. In: *5th International Conference onFuture Information Technology (FutureTech-2010)*, 2010, p. 1–6.

KHAN, L.; AWAD, M.; THURAISINGHAM, B. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, v. 16, n. 4, p. 507–521, 2007.

KOLTER, J. Z.; MALOOF, M. A. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, v. 8, p. 2755–2790, 2007.

KRANEN, P.; ASSENT, I.; BALDAUF, C.; SEIDL, T. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, v. 29, n. 2, p. 249–272, 2011.

KUHN, H. W. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, v. 2, p. 83–97, 1955.

KWAPISZ, J. R.; WEISS, G. M.; MOORE, S. A. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, v. 12, n. 2, p. 74–82, 2011.

LANGLEY, P. *Learning in humans and machines: Towards an inter-disciplinary learning science - order effects in incremental learning*. Oxford, 1995.

LEE, H.; ROBERTS, S. On-line novelty detection using the kalman filter and extreme value theory. In: *Proceedings of 19th International Conference on Pattern Recognition (ICPR 2008), Tampa, Florida, USA*, IEEE, 2008, p. 1–4.

LEE, M.-W.; KHAN, A. M.; KIM, T.-S. A single tri-axial accelerometer-based real-time personal life log system capable of human activity recognition and exercise information generation. *Personal Ubiquitous Comput.*, v. 15, n. 8, p. 887–898, 2011.

LI, X. Improving novelty detection for general topics using sentence level information patterns. In: *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM '06)*, New York, NY, USA: ACM, 2006, p. 238–247.

LIU, B.; DAI, Y.; LI, X.; LEE, W. S.; YU, P. S. Building text classifiers using positive and unlabeled examples. In: *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, Washington, DC, USA: IEEE Computer Society, 2003, p. 179–186.

LLOYD, S. P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, v. 28, n. 2, p. 129–137, 1982.

LOCKHART, J. W.; PULICKAL, T.; WEISS, G. M. Applications of mobile activity recognition. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*, New York, NY, USA: ACM, 2012, p. 1054–1058.

LOPES, A.; MENDES-MOREIRA, J.; GAMA, J. Semi-supervised learning: predicting activities in android environment. In: *Ubiquitous Data Mining (UDM'2012) - Workshop in conjunction with the 20th European Conference on Artificial Intelligence - ECAI 2012*, 2012, p. 38–42.

MACQUEEN, J. B. Some methods of classification and analysis of multivariate observations. In: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, p. 281–297.

MAHDIRAJI, A. R. Clustering data stream: A survey of algorithms. *International Journal Knowledge-Based Intelligent Engineering Systems*, v. 13, p. 39–44, 2009.

MARKOU, M.; SINGH, S. Novelty detection: a review–part 1: statistical approaches. *Signal Processing*, v. 83, n. 12, p. 2481 – 2497, 2003a.

MARKOU, M.; SINGH, S. Novelty detection: a review part 2: neural network based approaches. *Signal Processing*, v. 83, n. 12, p. 2499–2521, 2003b.

MARROCCO, C.; SIMEONE, P.; TORTORELLA, F. A framework for multiclass reject in ECOC classification systems. In: *Proceedings of the 15th Scandinavian Conference on Image analysis (SCIA'07)*, Berlin, Heidelberg: Springer-Verlag, 2007, p. 313–323.

MARSLAND, S. Novelty Detection in Learning Systems. *Neural Computing Surveys*, v. 3, p. 157–195, 2003.

MARSLAND, S.; SHAPIRO, J.; NEHMZOW, U. A self-organising network that grows when required. *Neural Network*, v. 15, p. 1041–1058, 2002.

MASUD, M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Trans. on Knowl. and Data Eng.*, v. 23, n. 6, p. 859–874, 2011a.

MASUD, M. M.; AL-KHATEEB, T. M.; KHAN, L.; AGGARWAL, C.; GAO, J.; HAN, J.; THURAISINGHAM, B. Detecting recurring and novel classes in concept-drifting data streams. In: *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM '11)*, Washington, DC, USA: IEEE Computer Society, 2011b, p. 1176–1181.

MASUD, M. M.; CHEN, Q.; KHAN, L.; AGGARWAL, C. C.; GAO, J.; HAN, J.; THURAISINGHAM, B. M. Addressing concept-evolution in concept-drifting data streams. In: *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM'10)*, 2010a, p. 929–934.

MASUD, M. M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. Classification and novel class detection in data streams with active mining. In: *Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part II (PAKDD'10)*, 2010b, p. 311–324.

MAURER, U.; SMAILAGIC, A.; SIEWIOREK, D.; DEISHER, M. Activity recognition and monitoring using multiple sensors on different body positions. In: *International Workshop on Wearable and Implantable Body Sensor Networks, 2006*, 2006, p. 116–12–.

MENAHEM, E.; ROKACH, L.; ELOVICI, Y. Combining one-class classifiers via meta-learning. In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013)*, New York, NY, USA: ACM, 2013, p. 2435–2440.

MITCHELL, T. M. *Machine learning*. 1 ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

NADEEM, M. S. A.; ZUCKER, J.-D.; HANCZAR, B. Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. *Journal of Machine Learning Research - Proceedings Track*, v. 8, p. 65–81, 2010.

NALDI, M.; CAMPELLO, R.; HRUSCHKA, E.; CARVALHO, A. Efficiency issues of evolutionary k-means. *App. Soft Comp.*, v. 11, p. 1938–1952, 2011.

O'CALLAGHAN, L.; MISHRA, N.; MEYERSON, N. A.; GUHA, S.; MOTWANI, R. Streaming-data algorithms for high-quality clustering. In: *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, Washington, DC, USA: IEEE Computer Society, 2002, p. 685–694.

ÖZGÜR, A.; ÖZGÜR, L.; GÜNGÖR, T. Text categorization with class-based and corpus-based keyword selection. In: *Proceedings of the 20th International Conference on Computer and Information Sciences*, ISCIS'05, Berlin, Heidelberg: Springer-Verlag, 2005, p. 606–615 (*ISCIS'05*,).

PARK, C. H.; SHIM, H. Detection of an emerging new class using statistical hypothesis testing and density estimation. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 24, n. 1, p. 1–14, 2010.

PERDISCI, R.; GU, G.; LEE, W. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In: *Proceedings of the 6th International Conference on Data Mining (ICDM '06)*, Washington, DC, USA: IEEE Computer Society, 2006, p. 488–498.

PERNER, P. Concepts for novelty detection and handling based on a case-based reasoning process scheme. *Engineering Applications of Artificial Intelligence*, v. 22, p. 86–91, 2009.

PILLAI, I.; FUMERA, G.; ROLI, F. A classification approach with a reject option for multi-label problems. In: *Proceedings of the 16th International Conference on Image Analysis and Processing: Part I (ICIAP'11)*, Berlin, Heidelberg: Springer-Verlag, 2011, p. 98–107.

PIMENTEL, M. A.; CLIFTON, D. A.; CLIFTON, L.; TARASSENKO, L. A review of novelty detection. *Signal Processing*, v. 99, n. 1, p. 215–249, 2014.

PINTO, C. M. S. *Algoritmos incrementais para aprendizagem bayesiana*. Tese de Doutoramento, Faculdade de Economia da Universidade do Porto, Porto, Portugal, 2005.

QUINLAN, J. R. *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

RAMEZANI, R.; ANGELOV, P.; ZHOU, X. A fast approach to novelty detection in video streams using recursive density estimation. In: *Proceedings of the 4th International IEEE Conference on Intelligent Systems (IS '08)*, IEEE, 2008, p. 14–2–14–7.

RAVI, N.; DANDEKAR, N.; MYSORE, P.; LITTMAN, M. L. Activity recognition from accelerometer data. In: *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3 (IAAI'05)*, 2005, p. 1541–1546.

RIOS, G.; FILHO, R. H.; COELHO, A. L. C. An autonomic security mechanism based on novelty detection and concept drift. In: *Proceedings of the 7th International Conference on Autonomic and Autonomous Systems*, 2011.

ROKACH, L.; MAIMON, O. *Data mining with decision trees: Theory and applications*, v. 69 de *Machine Perception Artifical Intelligence*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2008.

ROSENBERG, A. *Automatic detection and classification of prosodic events*. Tese de Doutoramento, Columbia University, 2009.

RUSIECKI, A. Robust neural network for novelty detection on data streams. In: *Proceedings of the 11th International Conference on Artificial Intelligence and Soft Computing - Volume Part I (ICAISC'12)*, Berlin, Heidelberg: Springer-Verlag, 2012, p. 178–186.

SCHÖLKOPF, B.; PLATT, J. C.; SHAWE-TAYLOR, J. C.; SMOLA, A. J.; WILLIAMSON, R. C. Estimating the support of a high-dimensional distribution. *Neural Computation*, v. 13, n. 7, p. 1443–1471, 2001.

SCHÖLKOPF, B.; WILLIAMSON, R.; SMOLA, A.; TAYLOR, J. S.; PLATT, J. Support vector method for novelty detection. *Advances in Neural Information Processing Systems*, v. 12, p. 582–588, 2000.

SHYU, M.-L.; SARINNAPAKORN, K.; KURUPPU-APPUHAMILAGE, I.; CHEN, S.-C.; CHANG, L.; GOLDRING, T. Handling nominal features in anomaly intrusion detection problems. In: *Proceedings of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE '05)*, Washington, DC, USA: IEEE Computer Society, 2005, p. 55–62.

SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. P. L. F.; GAMA, J. Data stream clustering: A survey. *ACM Computing Surveys*, v. 46, n. 1, p. 13:1–13:31, 2013.

SINGH, S.; MARKOU, M. A black hole novelty detector for video analysis. *Pattern Analysis and Applications*, v. 8, n. 1, p. 102–114, 2005.

SINGH, S.; MARKOW, M. An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, v. 16, n. 4, p. 396–407, 2004.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, v. 45, n. 4, p. 427–437, 2009.

SPINOSA, E. J.; CARVALHO; GAMA, J. Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: *Proceedings of 23rd Symposium on Applied computing (SAC '08)*, New York, NY, USA: ACM, 2008, p. 976–980.

SPINOSA, E. J.; CARVALHO, A. C. P. L. F. SVMs for novel class detection in bioinformatics. In: *Proceedings of 3rd Brazilian Workshop on Bioinformatics (WOB 2004), Brasília, DF, Brazil*, 2004, p. 81–88.

SPINOSA, E. J.; CARVALHO, A. C. P. L. F.; GAMA, J. Novelty detection with application to data streams. *Intelligent Data Analysis*, v. 13, n. 3, p. 405–422, 2009.

SRIVASTAVA, A. Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques. In: *IEEE Aerospace Conference*, Institute of Electrical and Electronics Engineers (IEEE), 2006.

STEINWART, I.; CHRISTMANN, A. *Support vector machines*. 1st ed. Springer Publishing Company, Incorporated, 2008.

SUN, L.; ZHANG, D.; LI, B.; GUO, B.; LI, S. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In: *Proceedings of the 7th International Conference on Ubiquitous Intelligence and Computing - (UIC'10)*, Berlin, Heidelberg: Springer-Verlag, 2010, p. 548–562.

TAN, S. C.; TING, K. M.; LIU, T. F. Fast anomaly detection for streaming data. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume 2 (IJCAI'11)*, AAAI Press, 2011, p. 1511–1516.

TAVAKKOLI, A.; NICOLESCU, M.; BEBIS, G. A novelty detection approach for foreground region detection in videos with quasi-stationary backgrounds. In: *Proceedings of the 2nd International Symposium on Visual Computing*, Berlin, Heidelberg: Springer-Verlag, 2006.

TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A. A detailed analysis of the KDD Cup 99 data set. In: *IEEE Symposium on Computational Intelligence for Security and Defense Applications, (CISDA'09)*, Piscataway, NJ, USA: IEEE Press, 2009, p. 1–6.

TAX, D. M. J.; DUIN, R. P. W. Combining one-class classifiers. In: *Proceedings of the 2nd International Workshop on Multiple Classifier Systems (MCS '01)*, 2001, p. 299–308.

TAX, D. M. J.; DUIN, R. P. W. Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, v. 29, n. 10, p. 1565–1570, 2008.

TING, K. M.; TAN, S. C.; LIU, F. T. *Mass: A new ranking measure for anomaly detection.* Technical report fa2386-09-1-4014, Gippsland School of Information Technology, Monash University, 2009.

TSYMBAL, A. *The problem of concept drift: definitions and related work*. Technical report TCD-CS-2004-15, Computer Science Department, Trinity College, Dublin, 2004.

VAPNIK, V. N. *Statistical learning theory*. 1 ed. Wiley, 1998.

VENDRAMIN, L.; CAMPELLO, R.; HRUSCHKA, E. Relative clustering validity criteria: A comparative overview. *Stat. Anal. and Data Min.*, v. 3, p. 209–235, 2010.

WANG, H.; FAN, W.; YU, P. S.; HAN, J. Mining concept-drifting data streams using ensemble classifiers. In: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, New York, NY, USA: ACM, 2003, p. 226–235.

WANG, W.; GUAN, X.; ZHANG, X. Processing of massive audit data streams for real-time anomaly intrusion detection. *Computer Communications*, v. 31, n. 1, p. 58 – 72, 2008.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, v. 23, n. 1, p. 69–101, 1996.

YANG, Y.; ZHANG, J.; CARBONELL, J.; JIN, C. Topic-conditioned novelty detection. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, New York, NY, USA: ACM, 2002, p. 688–693.

YEUNG, D.; CHOW, C. Parzen-window network intrusion detectors. In: *Proceedings of the 16th International Conference on Pattern Recognition*, Washington, DC, USA: IEEE Computer Society, 2002, p. 385–388.

YEUNG, D.; DING, Y. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, v. 36, p. 229–243, 2003.

ZHANG, J.; YAN, Q.; ZHANG, Y.; HUANG, Z. Novel fault class detection based on novelty detection methods. In: *Intelligent Computing in Signal Processing and Pattern Recognition*, v. 345 de *Lecture Notes in Control and Information Sciences*, Berlin Heidelberg: Springer, p. 982–987, 2006.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, New York, NY, USA: ACM Press, 1996, p. 103–114.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, v. 1, n. 2, p. 141–182, 1997.