Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Abordagem Fuzzy para Detecção de Novidade em Fluxo Contínuo de Dados

Tiago Pinho da Silva

Orientadora: Profa. Dra. Heloisa de Arruda Camargo

São Carlos, SP, Brasil

Abril, 2018

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Abordagem Fuzzy para Detecção de Novidade em Fluxo Contínuo de Dados

Tiago Pinho da Silva

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial.
Orientadora: Profa. Dra. Heloisa de Arruda Camargo

São Carlos, SP, Brasil

Abril, 2018

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Tiago Pinho da Silva, realizada em 25/04/2018:

Profa. Dra. Heloisa de Arruda Camargo
UFSCar

Prof. Dr. Ricardo Cerri
UFSCar

Profa. Dra. Elaine Ribeiro de Faria Paiva
UFU

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Elaine Ribeiro de Faria Paiva e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

Profa. Dra. Heloisa de Arruda Camargo

Agradecimentos

Agradecimentos a minha família que tem me apoiado em todos as minha decisões, em especial a minha mãe pelos sacrifícios feitos para que eu alcançasse este objetivo. Agradeço também minha namorada Marília pelas escapadelas para momentos gastronômicos e reparadores e também pelas palavras de conforto, que me acolhiam em momentos difíceis. Agradecimentos ao CIG, novos e antigos membros, aos que deixaram e deixarão um legado incrível e me serviram de inspirações por toda vida. Em especial aos que me aturaram e contribuíram para meu conhecimento Pri, Jerson, Dudinha, Maykonsuel, Mari, Su, Léo e Jorge. Não poderia também deixar de agradecer aos meu amigos Gean, Kenju, Jorel, Danileira, Eduardinho e Erick pelas conversas nem sempre de auto nível e companheirismo, que aliviaram os obstáculos ao longo do caminho. Em qualquer caminho que escolhemos, a presença (mesmo geograficamente distante) de pessoas como vocês faz a aventura mais divertida. Por fim, e não menos importante, agradeço a minha orientadora pelos ótimos conselhos e por acreditar no meu trabalho.

Agradecimentos aos melhores amigos que a vida poderia me dar, e que me proporcionaram aventuras internacionais, transcendentais e culturais: Zen, Aliseira, Luminha, Lulinha, Neyzin, Ramonzito, Gust e Cris vocês provocam a mente e desafiam a revisão de mim e do meu ambiente.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

The wind rises, we must try to live.
Paul Valéry,

Resumo

Nos últimos anos, presencia-se o advento de sistemas capazes de gerar uma imensa quantidade de dados em um curto espaço de tempo e aplicações podem ser encontradas em áreas como por exemplo: redes de sensores, mercado financeiro, redes de computadores, redes sociais entre outros. Sistemas como esses produzem dados incessantemente, criando, assim, um Fluxo Contínuo de Dados (FCD) que pode apresentar tamanho infinito além de poder sofrer mudanças em sua distribuição estatística de acordo com o tempo. Estes FCDs podem ser utilizados como fontes para a aquisição de conhecimento por métodos de aprendizado de máquina, como classificação, agrupamento e mineração de padrões frequentes. Entretanto, a natureza infinita e mutável destes conjuntos de dados pode causar essencialmente o surgimento de novos conceitos, que são exemplos que diferem significativamente dos conceitos aprendidos pelo modelo. Ocorrências deste comportamento em aplicações do mundo real podem ser, por exemplo, fraudes em cartões de créditos ou intrusões em redes de computadores. Desta forma, a tarefa de detecção destes exemplos, conhecida como detecção de novidade se destaca como um importante tópico de pesquisa. De modo geral, métodos clássicos para detecção de novidade não são capazes de lidar com as particularidades existentes em FCDs. Assim, diferentes abordagens vêm sendo propostas com o intuito de criar modelos adaptáveis e que possam realizar esta tarefa. No entanto, a característica de mudança de FCDs geram dificuldades no processo de aprendizagem, encorajando a busca por aprendizado flexível. A integração de conceitos da teoria de conjuntos *fuzzy* é uma forma oportuna de tornar o aprendizado em FCD mais adaptável a imprecisão dos dados. Recentemente, tem surgido propostas de modelos para aprendizado de máquina em FCD baseadas em teoria de conjuntos *fuzzy* com o objetivo de colaborar para a representação de imprecisão e adaptabilidade do conhecimento aprendido em FCDs. Entretanto, no contexto de detecção de novidade as abordagens propostas são poucas e limitam-se a domínios específicos. Este trabalho apresenta duas propostas de abordagens flexíveis para detecção de novidade em FCD, investigando técnicas e modelos de aprendizado de máquina em FCD baseados em teoria de conjuntos *fuzzy*. A análise dos resultados produzidos demonstra que as propostas propiciam melhorias na tarefa de detecção de novidade, facilitando a identificação de dados discrepantes através da representação e tratamento de imprecisão nos dados.

Palavras-chaves: fluxos contínuo de dados, detecção de novidade, teoria de conjuntos *fuzzy*, computação flexível.

Abstract

In recent years, we have witnessed the advent of computational systems capable of generating an immense amount of data in a short time period. These applications can be found in areas such as sensor networks, financial markets and computer networks. Systems that produce data incessantly, creating a continuous Data Stream (DS), can be infinite in size and can mutate in its statistical distribution over time. These DS can be used as sources for the automatic acquisition of useful knowledge by machine learning methods. However, the infinite and mutable nature of these data sets can essentially cause new concepts to emerge, which are examples that differ significantly from the examples learned by the model. Occurrences of this behavior in real-world applications may be credit card fraud or computer network intrusions. In this way, the task of detecting these examples, known as novelty detection, stands out as an important research topic. In general, classical methods for detecting novelty are not able to deal with the particularities of DS. Thus, different approaches have been proposed in order to create adaptable models that can accomplish this task. However, the unpredictable characteristics of DS's create difficulties in the learning process, encouraging the search for a more flexible learning. The integration of fuzzy set theory concepts is a timely way of making DS learning more adaptable to imprecisions. Recently, there have been proposals for machine learning models in DS based on fuzzy sets theory with the objective of collaborating for the flexibility and adaptability of the knowledge learned in DS's. Nonetheless, in the context of novelty detection the proposed approaches are few and limited to the domains of study. This paper presents a proposal for a fuzzy approach to detecting novelty in DS investigating techniques for detection of novelty in DS and machine learning models in DS based on fuzzy set theory. The analysis of the results, showed that the proposals favor the novelty detection task, facilitating the identification of discrepant data through the representation and treatment of imprecise data.

Keywords: data streams, novelty detection, fuzzy sets theory, soft computing.

Lista de Ilustrações

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

AM	Aprendizado de Máquina
FCD	Fluxo Contínuo de Dados
FCM	*Fuzzy C-Means*
DN	Detecção de Novidade
OFCM	*Online Fuzzy C-Means*
SPFCM	*Single Pass Fuzzy C-Means*
TCF	*Temporal Cluster Feature*
WFCM	*Weighted Fuzzy C-Means*

Sumário

Introdução

No aprendizado de máquina investiga-se métodos computacionais que sejam capazes de adquirir conhecimento de forma automática (MITCHELL, 1997). Dessa forma, por meio desses métodos, sistemas computacionais podem aprender e otimizar seu desempenho de forma a torná-lo mais preciso. Assim, o aprendizado de máquina cumpre um papel importante em um grande número de aplicações transformando dados em informações para a tomada de decisão em diversos ambientes.

De modo geral, a maioria das pesquisas no contexto de aprendizado de máquina consideram que os dados que serão utilizados para o processo de aprendizagem encontram-se disponíveis a *priori* e em uma quantidade limitada (PIMENTEL et al., 2014; MARKOU; SINGH, 2003). Entretanto, nos últimos anos, está sendo presenciado o advento de sistemas capazes de gerar uma imensa quantidade de dados em um curto espaço de tempo e aplicações podem ser encontradas em áreas como por exemplo: redes de sensores, mercado financeiro, redes de computadores, redes sociais entre outros. Segundo Gama (2010), sistemas como esses produzem dados incessantemente, criando assim, um Fluxo Contínuo de Dados (FCD) que pode apresentar tamanho infinito além de poder sofrer mudanças em sua distribuição ao longo do tempo. Estes FCDs podem ser utilizados como fontes para a aquisição de conhecimento por métodos de aprendizado de máquina, que abordam as tarefas de classificação, agrupamento e mineração de padrões frequentes. Entretanto, devido à natureza potencialmente infinita e mutável destes conjuntos de dados, métodos clássicos não são capazes de lidar com estas particularidades, exigindo o desenvolvimento de técnicas que sejam capazes de extrair conhecimento respeitando estas características (GAMA, 2010).

Portanto, por não apresentar tamanho definido, o armazenamento dos FCDs em memória é inviável (BABCOCK et al., 2002), logo é necessário que se processe os dados poucas vezes durante a aprendizagem. Para contornar este problema, algoritmos de aprendizado de máquina incrementais vem sendo utilizados (HULTEN; SPENCER; DOMINGOS, 2001a; LEITE; COSTA; GOMIDE, 2010a; DOMINGOS; HULTEN, 2000). Porém, dada a natureza mutável de FCD, estes algoritmos não são suficientes, pois os conceitos

conhecidos pelo modelo podem mudar ao longo do tempo ou sofrer evoluções (FARIA et al., 2016a). Nesse caso, são necessários mecanismos capazes de detectar mudanças nos conceitos conhecidos e evoluções de novos conceitos, para atualizar o modelo de decisão (FARIA et al., 2016a).

Dentre as possíveis tarefas para aquisição de conhecimento por meio de FCD, detecção de novidade será o principal foco de estudo deste projeto. Nesta tarefa, assume-se que exemplos não rotulados provenientes do FCD chegarão ao longo do tempo de forma ordenada. Portanto, o modelo de detecção de novidade deve identificar exemplos recebidos que diferem significativamente do conjunto de conceitos conhecidos em tempo real (FARIA et al., 2016a). Dessa forma, estes exemplos podem corresponder a novidades representadas por mudanças de conceito, definidas como alterações na distribuição de conceitos conhecidos, evoluções de conceito, caracterizadas pela emergência de novos conceitos ao longo do tempo, ou simplesmente ruídos, que por sua vez não devem ser levados em consideração por um modelo de decisão. Nesse cenário, faz-se necessário a utilização de métodos que sejam capazes de detectar o surgimento destas alterações e manter o modelo de decisão coerente com os novos conceitos, ou mudanças nos conceitos conhecidos que podem surgir ao longo do tempo. Portanto, modelos de detecção de novidade em FCD requerem constantes atualizações para que sua performance mantenha-se estável ao longo do tempo em ambientes dinâmicos.

Outra característica presente em grande parte das aplicações de FCD está relacionada à presença de imprecisão nos dados, que pode ocorrer, por exemplo, devido a ambientes geradores não estacionários que podem propiciar mudanças ou evoluções de conceito ao longo do tempo. Em cenários como esses, nem sempre é possível a definição de um limite de decisão exato para que um modelo de detecção de novidade possa identificar corretamente dados discrepantes, muitas vezes devido a constantes alterações sofridas por esses tipos de dados. Portanto, nessas circunstâncias a imprecisão nos dados deve ser levada em consideração para a obtenção de modelos de detecção de novidade em FCD mais estáveis e precisos. Conforme será abordado ao longo deste projeto, o presente trabalho visa contribuir com soluções nesta direção.

Motivação e Justificativa

Recentemente, o aprendizado de máquina em FCDs vem ganhando cada vez mais popularidade, devido ao grande volume de dados que podem ser gerados em alta velocidade por sistemas computacionais, por exemplo, em mineração de *clicks* na *web* (MARIN et al., 2013), medida de consumo de energia (SILVA et al., 2011; ZHANG et al., 2012) e fraude de cartão de crédito (WU; LI; HU, 2012). Neste tópico de pesquisa, detecção de novidade é uma importante tarefa a ser investigada por conta da volatilidade na dis-

tribuição dos dados inerente a muitos domínios de FCD, que pode trazer instabilidade aos resultados obtidos por um modelo de aprendizado de máquina, caso este não seja atualizado com as devidas mudanças.

Dentre as diferentes técnicas propostas para DN em FCD, a utilização de sumários que comportam estatísticas a respeito de um grupo de dados relacionados denominados micro-grupos vem obtendo interessantes resultados (FARIA et al., 2016b; AL-KHATEEB et al., 2012a; MASUD et al., 2010; MASUD et al., 2011; FARID et al., 2013; FARID; RAHMAN, 2012; HAQUE; KHAN; BARON, 2016; ABDALLAH et al., 2016). Os principais motivos do emprego de micro-grupos é a sua característica incremental o que os torna adequados para domínios de FCD, e o caráter descritivo destas estruturas que pode facilitar na tarefa de DN. Entretanto, estes trabalhos apresentam uma abordagem baseada em teoria de conjuntos clássicos, o que pode não representar corretamente a realidade de algumas aplicações de FCD que apresentam imprecisões de dados, ocorridas por exemplo, por ambientes geradores não-estacionários, e que podem causar instabilidade e resultados inconsistentes em modelos de detecção de novidade (DESHPANDE et al., 2004; KANAGAL; DESHPANDE, 2008; TRAN et al., 2009; KUROSE et al., 2006). Portanto, essa característica encoraja o estudo e o desenvolvimento de técnicas baseadas em teoria de conjuntos *fuzzy*, que possam ser utilizadas na tarefa de DN em FCD de forma a trazer uma maior flexibilidade aos modelos de aprendizado de máquina, através da representações flexíveis, obtidas por meio de medidas de pertinência *fuzzy*.

Ressonante com este problema, diversos trabalhos que empregam teoria de conjuntos *fuzzy* para a representação de imprecisão, de forma a obter uma maior flexibilidade em métodos de aprendizado de máquina em FCD foram propostos e evidenciam as vantagens do uso (LOPES; CAMARGO, 2017; LEITE; COSTA; GOMIDE, 2010b; ISAZADEH; MAHAN; PEDRYCZ, 2016). Na tarefa de detecção de novidade em FCD Lemos, Caminhas e Gomide (2013a) sugerem uma abordagem para detecção e diagnóstico adaptativo de falhas em fluxos de operações de processos. Outro método proposto por Angelov, Ramezani e Zhou (2008) apresenta uma solução tanto para o problema da detecção de novidades quanto para o rastreamento de objetos em fluxos de vídeos. Já no trabalho de Angelov e Zhou (2008) os autores propõem uma abordagem para a classificação em FCDs que baseia-se em um sistema *fuzzy* de regras (FRB) evolutivo do tipo *Takagi-Sugeno* que ajusta-se ao surgimento de novidades.

Embora existam métodos que associam a teoria de conjuntos *fuzzy* com a tarefa de detecção de novidade em FCDs, estes são poucos e em sua maioria são destinados à um domínio específico. O que estimula o desenvolvimentos de métodos *fuzzy* que abordem a tarefa de DN em FCDs de forma mais genérica.

Hipótese e Objetivo

Buscando o desenvolvimento de modelos de detecção de novidade em FCD que possam contribuir com resultados de alta precisão em ambientes de natureza não estacionária, esta proposta teve como objetivo investigar, propor e avaliar métodos capazes de detectar novidades em FCDs, por meio de representações mais flexíveis com o uso de teoria de conjuntos *fuzzy*. Tendo em vista o objetivo principal deste projeto, pode-se declarar a hipótese central:

- Por meio do uso de soluções baseadas em técnicas de agrupamento e teoria de conjuntos *fuzzy*, algoritmos de detecção de novidade multiclasse em FCD podem apresentar limites de decisão flexíveis às mudanças e evoluções de conceito que podem ocorrer ao longo do tempo, produzindo resultados comparáveis, ou melhores, do que os métodos de detecção de novidade baseados em teoria de conjuntos clássicos.

Organização do Texto de Qualificação

Capítulo 1 - Detecção de Novidade: descreve conceitos gerais a respeito de detecção de novidade em aprendizado de máquina clássico, com foco especial em técnicas desenvolvidas para abordar esta tarefa.

Capítulo 2 - Aprendizado em Fluxo Contínuo de Dados: apresenta uma contextualização quanto a FCDs, caracterizando o domínio, conceitos gerais sobre essa área e outras noções que fundamentaram o desenvolvimento de métodos para o aprendizado neste domínio. Neste capítulo é traçada uma visão geral do estado-da-arte da área de aprendizado em FCDs, com atenção particular aos métodos que realizam aprendizado por agrupamento e agrupamento *fuzzy*.

Capítulo 3 - Detecção de Novidade em Fluxos de Dados: apresenta uma contextualização quanto a tarefa de detecção de novidade em FCDs, descrevendo conceitos gerais que fundamentaram o desenvolvimento de métodos para a tarefa de detecção de novidade em FCDs. Posteriormente é traçada uma visão geral do estado-da-arte de abordagens para a detecção de novidade em FCDs .

Capítulo 4 - Abordagens *Fuzzy* para Detecção de Novidade Multiclasse em Fluxo Contínuo de Dados: retoma a contextualização e motivação para o trabalho, além do detalhamento geral das propostas e explicação das estratégias originais desenvolvidas para a tarefa de DN em FCDs.

Capítulo 5 - Metodologia para Validação: expõe a construção de experimentos, descrevendo as ferramentas, os conjuntos de dados e os algoritmos aplicados, além das métricas de avaliação empregadas.

Capítulo 6 - Resultados e Análises dos Experimentos: apresenta comentários e análises sobre os resultados obtidos pelos experimentos, baseados em valores das métricas avaliadas, identificando características dos algoritmos utilizados.

Parte I

Revisão Bibliográfica

1

Detecção de Novidade

A capacidade de aprender é uma característica fundamental para um comportamento inteligente, portanto, Aprendizado de Máquina (AM) apresenta-se como uma área de pesquisa de extrema importância em inteligência artificial. Envolve estudos de métodos computacionais para aquisição de novos conhecimentos, novas habilidades e novas maneiras de organizar o conhecimento já existente. Sendo assim, sob determinadas condições, é possível criar modelos computacionais capazes de aprender e melhorar por meio da observação, e existem diversos métodos que podem ser utilizados para realizar esse aprendizado, como aprendizado por hábito, instrução, dedução, analogia e indução (MICHALSKI; CARBONELL; MITCHELL, 2013).

Dentre os métodos mencionados, o aprendizado indutivo é um dos mais utilizados no contexto de AM, e caracteriza-se como uma forma de inferência lógica que permite obter conclusões universais partindo de um conjunto particular de premissas, também chamado conjunto de treinamento. No que diz respeito à AM, pode-se considerar essas premissas como exemplos ou instâncias previamente observados, que podem ser representados por um vetor n-dimensional composto por valores relacionados ao domínio dos exemplos (atributos). Entretanto, é importante ressaltar que as hipóteses geradas através da inferência indutiva podem ou não preservar a verdade. Mesmo assim, a inferência indutiva é um dos principais métodos utilizados para derivar conhecimento novo e predizer eventos futuros (MITCHELL, 1997). Este tipo de aprendizado pode ser dividido em supervisionado e não-supervisionado.

Aprendizado Supervisionado

No aprendizado supervisionado é fornecida uma referência do objetivo a ser alcançado, isto é, um conjunto de treinamento composto por exemplos com atributos de entrada e de saída (nominal ou numérico). Por meio desses conjuntos, o algoritmo de aprendizado de máquina extrai a representação do conhecimento para que este seja capaz de produzir saídas corretas para novas entradas não apresentadas antes, denominadas

conjunto de teste. Portanto, o aprendizado supervisionado pode ser empregado em duas principais sub-categorias:

- **Classificação:** tarefa onde se atribui um rótulo (classe) a uma entrada, por exemplo, a classificação do tipo de flor dado suas características de sépala e pétala, a classificação de gênero de uma pessoa dado o mapeamento de imagem da mesma e a classificação do sentimento contido em um comentário.
- **Regressão:** usada quando o valor que está sendo previsto segue um espectro contínuo. Sistemas de regressão poderiam ser usados, por exemplo, para responder às perguntas: "Quanto custa?" ou "Quantos existem?".

Aprendizado Não-Supervisionado

No aprendizado não supervisionado, também conhecido como aprendizado por observação e descoberta, a tarefa do algoritmo é identificar padrões ou relacionamentos entre exemplos não rotulados,ou seja, exemplos que não possuem o atributo de saída especificado. Nesse caso, é possível utilizar algoritmos de aprendizado para descobrir padrões nos dados a partir de alguma caracterização de regularidade, sendo esses padrões denominados agrupamento (DECKER; FOCARDI, 1995; MCCALLUM; NIGAM; UNGAR, 2000). Exemplos contidos em um mesmo grupo são mais similares, segundo alguma medida de similaridade, do que aqueles contidos em grupos diferentes.

Detecção de Novidade

Em diversas aplicações reais que envolvem aprendizado de máquina existe a ocorrência de instâncias ou conjunto delas que diferem do conjunto de conceitos aprendidos de forma significante. Uma vez que este apresenta-se como um importante problema na literatura (PIMENTEL et al., 2014; AGGARWAL, 2013; MILJKOVIĆ, 2010; MARKOU; SINGH, 2003), a detecção deste tipo de instâncias denominada detecção de novidade (DN), vem recebendo cada vez mais atenção do meio científico, e diversos trabalhos podem ser encontrados (MILJKOVIĆ, 2010; PIMENTEL et al., 2014; FARIA et al., 2016a).

Detecção de novidade pode ser encontrada na literatura como sinônimo de detecção de *outliers* ou detecção de anomalias. Estes termos foram originados de diferentes domínios de aplicações onde o conceito de detecção de novidade pode variar. Consequentemente, não existe uma definição universal aceita para DN (PIMENTEL et al., 2014). Com o intuito de simplificar o entendimento deste capítulo consideraremos DN a partir do seguinte conceito:

- A detecção de novidade pode ser definida como a tarefa de reconhecer que os exemplos do conjunto de teste diferem em algum aspecto dos exemplos disponíveis du-

rante o treinamento (PIMENTEL et al., 2014).

Em diversas aplicações reais, enquanto que podemos considerar haver um conjunto de dados representativo de eventos considerados normais (ex. transações normais de cartões de crédito), instâncias associadas à eventos anormais (ex. transações fraudulentas de cartões de crédito) encontram-se com baixa frequência ou inexistentes (MARKOU; SINGH, 2003; PIMENTEL et al., 2014). Em casos como esses, a DN faz-se necessária, afim de identificar tais instâncias e dar o tratamento correto a elas.

A tarefa de DN apresenta-se como um problema difícil e desafiador e aplicações podem ser encontrada em diferentes problemas do mundo real, dentre elas podemos destacar: detecção de intrusos em redes de computadores (COULL et al., 2003), detecção de fraude em cartões de crédito e celulares (PATCHA; PARK, 2007), filtro de *spam* (HAYAT et al., 2010), detecção de massas em imagens de mamografia (TARASSENKO et al., 1995) e mineração de texto (LI; CROFT, 2006).

Para o melhor entendimento deste trabalho, este capítulo apresenta uma revisão sobre DN baseada nos artigos de Pimentel et al. (2014), Markou e Singh (2003) ,Miljković (2010) e Aggarwal (2013). Nas próximas seções são apresentados conceitos importantes em DN (Seção 1.1), assim como técnicas e métodos em DN (Seção 1.2) e como são aplicados os conceitos de teoria de conjuntos *fuzzy* no contexto de DN (Seção 1.3).

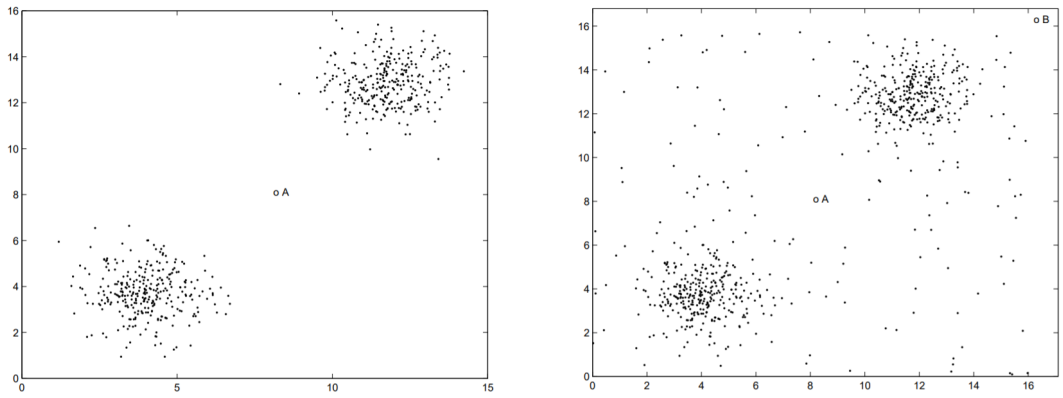
1.1 Conceitos Relacionados à DN

Essa seção apresenta os principais conceitos para o entendimento da tarefa de DN, abordando definições de ruídos, anomalia e *outlier* (Subseção 1.1.1), tipos de supervisão envolvidos (Subseção 1.1.2), formas de representar dados de saída (Subseção 1.1.3) e métodos de avaliação (Subseção 1.1.4), segundo a interpretação de DN seguida neste trabalho.

1.1.1 Definições de Ruídos, *Outliers* e Anomalias

Na literatura os termos anomalia e *outliers* vem sendo utilizados de forma indiferente (PIMENTEL et al., 2014), por possuírem definições aproximadas. Desse modo, com o objetivo de simplificar o entendimento deste trabalho consideraremos *outliers*, ruídos e anomalias a partir das definições abordadas por Aggarwal (2013), que estabelece *outliers* como sendo quaisquer pontos que discordem significativamente da distribuição dos demais dados. Ocorrem geralmente quando a fonte geradora dos dados comporta-se de forma incomum, e portanto podem conter informações úteis sobre características anormais que impactam no processo de geração de dados. Assim, *outliers* podem ser categorizados como ruídos ou anomalias.

O termo anomalia refere-se a um tipo especial de *outlier* que representa uma informação nova de interesse para o modelo. Dessa forma, no mundo real, anomalias podem ser vistas como intrusos em redes de computadores, fraudes em cartões de créditos entre outros. No entanto, existem dados denominados ruídos que da mesma forma como as anomalias, diferem dos demais dados, porém, não são relevantes e portanto devem ser descartados pois interferem nos resultados gerados pelo modelo. Para exemplificar a diferença entre os dois termos a Figura 1.1 apresenta duas distribuições de dados, uma sem ruído (Figura 1.1a) e outra com ruído (Figura 1.1b). A primeira distribuição de dados possui dois principais grupos e um ponto representado pela letra A identificado como uma anomalia. Já a segundo distribuição, além de possuir os mesmo principais grupos e dois pontos identificados como anomalia representados pelas letras A e B, possui dados irrelevantes que discordam significativamente das estruturas apresentadas pelos dois principais grupos, o que torna a identificação de anomalias complexa, pois na presença de ruídos, anomalias podem ser confundida como dados irrelevantes por parecer serem originadas de distribuições aleatórias.



(a) Distribuição de dados sem ruído (b) Distribuição de dados com ruído

Figura 1.1 – Distinção entre anomalias e ruídos (AGGARWAL, 2013)

As definições descritas acima são bastante corelacionados, e geralmente o julgamento do que se constitui um desvio suficiente para que um dado possa ser considerado *outlier* é de caráter subjetivo. Portanto, a distinção semântica entre ruídos e anomalias é de responsabilidade do analista, e a detecção destes tipos de dados é de fundamental importância pois pode ter um grande efeito na análise dos dados (GOGOI et al., 2011). Algumas das causas de *outliers* são: atividade maliciosa, erro de instrumentação, mudanças no ambiente e erro humano (CHANDOLA; BANERJEE; KUMAR, 2009).

Sobre estas circunstâncias, novidade podem ser definida como uma anomalia, ou seja, uma informação discrepante que é de interesse para o modelo de decisão, por apresentar relevância na solução do problema. Dessa forma, informações como transações fraudulentas em sistemas de cartão de crédito, e intrusões em redes de computadores são exemplos de novidades.

1.1.2 Tipos de Supervisão em DN

Os tipos de supervisão aplicadas na tarefa de DN são categorizados da seguinte maneira: aprendizado supervisionado, semissupervisionado e não supervisionado (MILJKOVIĆ, 2010; AGGARWAL, 2013). No aprendizado supervisionado em DN assume-se a disponibilidade de forma representativa do conjuntos de instâncias rotuladas que representem eventos normais, e conjuntos de instâncias que representem eventos anormais (ex. *outliers*). Dessa forma, a partir dos exemplos disponíveis é possível criar representações de dados que consigam identificar novidades durante a fase de uso (AGGARWAL, 2013).

Entretanto, nem sempre podemos considerar que as instâncias disponíveis são representativas, pois eventos considerados anormais geralmente são raros e podem demandar um alto custo de rotulação (MILJKOVIĆ, 2010). No aprendizado semi-supervisionado em DN exemplos de eventos normais ou eventos anormais estão disponíveis a *priori* não necessariamente de forma representativa. Assim, métodos são necessários para identificar exemplos que diferem significativamente do conjunto de dados disponível (MUÑOZ-MARÍ et al., 2010; TAX; JUSZCZAK, 2003; WANG; NESKOVIC; COOPER, 2005). Já no aprendizado não supervisionado nenhuma informação é dada a respeito dos rótulos das instâncias, cabendo ao método de DN identificar exemplos de eventos normais e anormais (ESKIN et al., 2002; HELLER et al., 2003; GOH; CHANG; LI, 2005).

1.1.3 Mecanismos de Saída

Tipicamente, em se tratando da representação do resultado de saída de métodos para DN existem duas categorias: (MILJKOVIĆ, 2010; AGGARWAL, 2013)

- **Pontuação**: nesta abordagem o método associa uma pontuação relativa a novidade para cada instância de teste. Dessa forma, esta pontuação também pode ser usada para ordenar as instâncias de acordo com suas tendências a serem novidade. Esta é uma forma de saída muito genérica que não fornece um resumo conciso da quantidade de exemplos que devem ser considerados novidade.

- **Rotulação binária**: nesta técnica, diferentemente das abordagens baseadas em pontuação, instâncias de teste são classificadas como pertencente a eventos normais ou como novidade. Esta abordagem apresenta menos informações do que o mecanismo de pontuação. Porém, em algumas aplicações apenas os resultados de decisão são necessários (AGGARWAL, 2013).

Métodos baseados na abordagem de pontuação podem ser adaptados para rotulação binária aplicando-se um limiar nos resultados de pontuação, de forma a decidir se um determinado exemplo do teste pode ser classificado como novidade. Os limiares

geralmente são escolhidos de acordo com a distribuição estatística das pontuações (AGGARWAL, 2013).

1.1.4 Métodos de avaliação

Com relação à eficácia, técnicas de DN podem ser avaliadas conforme sua taxa de acerto referente a instâncias classificadas como novidade e também de acordo com quantas instâncias são incorretamente classificadas como novidade, ou seja, falsos positivos. Para representar o desempenho do método baseado na taxa de falsos positivos e instâncias corretamente classificadas, curvas ROC (*Receiver operating characteristic*) geralmente são usadas (HANLEY; MCNEIL, 1982). Dessa maneira, técnicas de DN devem ter como objetivo uma alta taxa de detecção, mantendo baixa a taxa de falso positivo (PIMENTEL et al., 2014; AGGARWAL, 2013).

No que diz respeito a eficiência das abordagens de DN, avaliações relacionadas ao custo computacional e a complexidade de tempo e de espaço geralmente são empregadas. Por conseguinte, espera-se que técnicas de DN sejam eficientes e escaláveis para conjuntos de dados grandes e de alta dimensão. Além disso, dependendo da tarefa de DN, a quantidade de memória utilizada pelo método geralmente é considerada uma métrica de avaliação de desempenho importante (PIMENTEL et al., 2014).

1.2 Técnicas para Detecção de Novidade

Nesta seção serão abordadas técnicas de DN segundo a categorização de Pimentel et al. (2014). Serão apresentadas: técnicas probabilísticas, baseadas em distância, baseadas em reconstrução, baseadas em classificação e técnicas de teoria da informação. A Figura 1.2 representa a taxonomia destas técnicas.

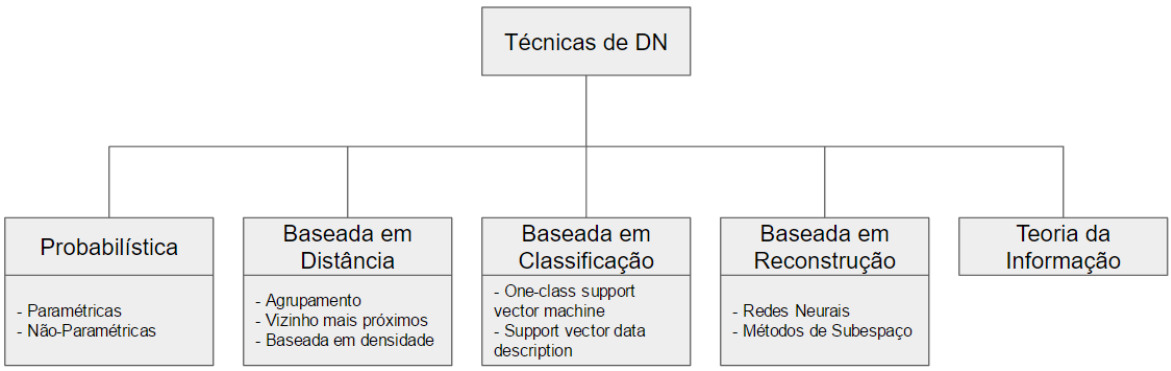


Figura 1.2 – Taxomia de técnicas para DN (PIMENTEL et al., 2014)

1.2.1 Probabilísticas

Segundo Pimentel et al. (2014) abordagens probabilísticas para DN fundamentam-se no cálculo da função de densidade de probabilidade (FDP) generativa dos dados (FILEV; TSENG, 2006b; FILEV; TSENG, 2006a; SONG et al., 2007). Dessa forma, a distribuição encontrada é utilizada para definir os limites da normalidade no espaço de dados e verificar se uma amostra de teste pertence à mesma distribuição. Portanto, assume-se que dados do conjunto de treino são gerados a partir de alguma distribuição de probabilidade que, por sua vez, é estimada a partir dos próprios dados de treinamento. Esta distribuição é utilizada para representar um modelo de normalidade. Assim, um limite de novidade pode então ser definido usando a distribuição encontrada, de modo que tenha uma interpretação probabilística. (AGGARWAL, 2013).

Estas abordagens podem ser categorizadas como paramétricas e não-paramétricas conforme a distribuição dos dados. Nas paramétricas atribui-se que dados de treino possam ter sidos gerados por uma distribuição de probabilidade (ex. Gaussiana)(FILEV; TSENG, 2006b; FILEV; TSENG, 2006a). Por outro lado, nas não-paramétricas não é feita qualquer suposição sobre a distribuição estatística dos dados, sendo mais flexível (KAPOOR et al., 2010; KEMMLER et al., 2013; KIM; LEE, 2006). Um dos métodos mais simples que utiliza este tipo de abordagem é o histograma que exibe de forma gráfica frequências relacionadas aos dados (CHANDOLA; BANERJEE; KUMAR, 2007).

Uma das vantagens das abordagens probabilísticas é que funcionam para qualquer tipo de dado, desde que seja utilizado o modelo generativo adequado. Por exemplo, se os dados constituem-se apenas de dados categóricos então uma distribuição discreta de Bernoulli pode ser aplicada. Portanto, técnicas probabilísticas fornecem um modelo genérico aplicável a qualquer tipo de dado, o que não é o caso em muitas outras técnicas. Por outro lado, isso também pode ser considerado uma desvantagem, pois nem sempre é apropriado que um conjunto de dados seja modelado de acordo com uma distribuição estatística (AGGARWAL, 2013).

1.2.2 Baseadas em Distância

As técnicas baseadas em distância funcionam de forma parecida aos métodos probabilísticos que utilizam o cálculo do FDP (PIMENTEL et al., 2014). Entretanto, esses métodos dependem de métricas de distância para calcular a medida de similaridade entre dois pontos de dados (CHANDOLA; BORIAH; KUMAR, 2008; CHAWLA; SUN, 2006), e de acordo com Aggarwal (2013) podem ser divididos em três métodos: agrupamento, baseados em densidade e vizinhos mais próximos.

Nos métodos de agrupamento e baseados em densidade, uma região densa de dados é encontrada e exemplos distantes desta região são classificados como novidade. A

principal diferença entre esses dois métodos é que métodos de agrupamento segmentam o conjunto de dados enquanto que os métodos baseados em densidade segmentam o espaço de dados (AGGARWAL, 2013). Já nos métodos de vizinhos mais próximos assume-se que dados considerados normais possuem muitos exemplos próximos, enquanto que novidades possuem vizinhos mais distantes.

Métodos baseados nos vizinhos mais próximos, costumam ser robustos com relação a ruídos e têm seu desempenho melhorado quanto mais exemplos rotulados forem disponíveis, além de serem implementados com facilidade. Por outro lado, métodos de agrupamento são capazes de serem usados de forma *online* atualizando-se com a chegada de novas instâncias. Uma das desvantagens destes métodos está na necessidade da escolha da métrica de distância que será aplicada, assim como a escolha de hiper-parâmetros (ex. número de agrupamentos, número de vizinhos próximos) (PIMENTEL et al., 2014).

1.2.3 Baseadas em Reconstrução

Os métodos baseados em reconstrução são utilizados para modelar uma representação do conjunto de dados a partir dos dados de treino. Assim, quando os dados do teste são apresentados, o erro de reconstrução, definido como a distância entre o vetor do exemplo de teste e a saída do modelo, é tido como uma pontuação de novidade.

Segundo Pimentel et al. (2014) redes neurais e métodos baseados em subespaço podem ser utilizados dessa maneira (CHEN et al., 2002; WU; WANG; LEE, 2010; MCBAIN; TIMUSK, 2011). Uma das vantagens de métodos baseados em redes neurais esta na extração autônoma de características, uma vez que a maioria dos métodos clássicos de aprendizado de máquina necessitam que este processo seja feito de forma manual. Entretanto, redes neurais possuem um alto custo computacional e são muito suscetíveis a superestimação dos exemplos de treinamento, o que pode tornar o modelo excessivamente complexo.

Um dos métodos baseado em subespaço mais utilizado é a Analise dos Componentes Principais (ACP), definido como um método utilizado para reduzir a dimensão do problema em componentes não correlacionados que são combinações lineares das variáveis originais (JOLLIFFE, 2002). Esta técnica pode ser utilizada em DN através da construção de um modelo a partir da distribuição dos dados de treino no espaço transformado. Métodos baseados em ACP conseguem lidar muito bem com grandes conjuntos de dados, entretanto, apresentam dificuldades para modelar estruturas não lineares.

1.2.4 Baseadas em Classificadores

Nesta abordagem, um classificador é construído a partir dos dados de treinamento que podem conter ou não exemplos de instâncias consideradas novidades. O modelo re-

sultante delimita um domínio de decisão onde os exemplos não contidos neste domínio são classificados como novidade.

Support Vector Machines (SVM) é um dos classificadores mais utilizados nesta abordagem pois delimita uma superfície de decisão entre diferentes classes. Este método foi proposto para a classificação de padrões binários de dados linearmente separáveis, utilizando um hiperplano para maximizar a margem de separação entre duas classes (SUYKENS; VANDEWALLE, 1999). Na tarefa de DN o classificador SVM é usado para delimitar a área que corresponde aos dados de treino. Assim, exemplos não contidos nesta área são considerados novidade (LIU; LIU; CHEN, 2010; LE et al., 2011).

Uma das principais vantagens deste método é a utilização de funções *kernel*, o que traz mais flexibilidade para o modelo, mapeando dados para dimenões mais elevadas para casos não linearmente separáveis. Entretanto, essas funções precisam ser escolhidas apropriadamente assim como os hiper-parâmetros do método.

Outro método denominando *Support Vector Data Descripition* (SVDD) (TAX; DUIN, 2004), define um limite de novidade como sendo uma hiperesfera que envolve a maioria dos dados de treino (normais), portanto, novidades são definidas como dados não contidos nos limites da hiperesfera. Similarmente, ao método SVM, SVDD também utiliza funções *kernels* para definir o limite da hiperesfera. Portanto, o que torna o método sensível a escolha destas funções.

1.2.5 Técnicas de teoria da informação

Nesta abordagem os métodos calculam o conteúdo de informações de um conjunto de dados usando métricas de teoria da informação (ex. entropia). Esses métodos assumem que uma novidade pode alterar significativamente o conteúdo da informação do conjunto de dados (KEOGH et al., 2007; FILIPPONE; SANGUINETTI, 2009). Normalmente as métricas são calculadas usando todo o conjunto de dados de treino. Então, subconjuntos de dados são eliminados do conjunto de treino e novamente são calculadas métricas, o subconjunto cuja eliminação induz a maior diferença na métrica encontrada é considerado como novidade.

De acordo com Pimentel et al. (2014) abordagens de teoria da informação para a DN geralmente não fazem quaisquer pressupostos sobre a distribuição dos dados. Entretanto, exigem uma métrica suficientemente sensível para detectar os efeitos de novidade no conjunto de dados. A principal desvantagem com este tipo de técnica está na seleção da métrica, além disso, métricas de teoria da informação podem detectar a presença de novidade somente se houver um número significativamente grande das mesmas presentes no conjunto de dados. Além disso, essas técnicas também são computacionalmente dispendiosas, embora tenham sido propostas aproximações para lidar com esse problema.

1.3 Teoria de Conjuntos *fuzzy* em DN

Durante os últimos anos, teoria de conjuntos *fuzzy* foi aplicada a uma grande quantidade de métodos de aprendizado de máquina e o seu uso no suporte à decisão facilitou o tratamento de dados incertos através da representação dos mesmos (CINTRA; MONARD; CAMARGO, 2013; BEZDEK; EHRLICH; FULL, 1984; LOPES; CAMARGO, 2017). No contexto de DN, em muitas aplicações a quantidade de dados disponíveis a priori, representantes de eventos considerados normais, nem sempre é o suficiente para delimitar um limite exato de decisão entre dados normais e anormais. Dessa maneira, incertezas de dados acabam se tornando uma característica importante nesse tipo de tarefa. Nessas circunstâncias, técnicas de DN demandam métodos capazes de lidar com essas incertezas de forma mais flexível. Assim, tentando adicionar os benefícios adquiridos por meio de teoria de conjuntos *fuzzy* em técnicas de DN, diferentes abordagens vem sendo propostas.

Filev e Tseng (2006b), Filev e Tseng (2006a) descrevem um algoritmo em tempo real para modelagem e detecção de falhas em processos maquinários. Nesta proposta utiliza-se os métodos de agrupamento *fuzzy* e modelos de misturas gaussianas para criar agrupamentos que possam representar os modos de operação principais das máquinas. De acordo com os autores, a falta de limites bem definidos dentre os principais modos de operação e a transição gradual entre eles foram as razões para considerar a utilização de teoria de conjuntos *fuzzy*.

Wang (2009) apresenta uma abordagem híbrida para identificação de *outilers* e segmentação de mercado em sistemas de gerenciamento de relacionamento com clientes, que incorpora dois métodos baseados em *kernel*, utilizando conceitos do algoritmo *possibilistic c-means* (KRISHNAPURAM; KELLER, 1993) para detecção de *outliers*, e conceitos do *fuzzy c-menas* (BEZDEK; EHRLICH; FULL, 1984) para segmentação de clientes. Nesse método medidas de pertinências baseadas no algoritmo *possibilistic c-means* são utilizadas para definir um limite não linear para identificação de *outlier*.

Gómez, González e Dasgupta (2003) propõem um método com limites flexíveis entre dados normais e anormais, por meio da classificação utilizando regras *fuzzy*. A viabilidade deste modelo é demonstrada no problema de detecção de falhas em sistemas de refrigeração (TAYLOR; CORNE, 2003).

Métodos baseados em teoria de conjuntos *fuzzy* proporcionam mais flexibilidade à tarefa de DN, e muitos problemas do mundo real são melhor representados por teoria de conjuntos *fuzzy*, pois geralmente há incertezas envolvidas. Portanto, métodos que utilizam conceitos *fuzzy* podem tratar de uma maneira mais realista a tarefa de DN.

1.4 Considerações Finais

A detecção de novidades é uma tarefa de aprendizado de máquina em que um modelo é criado a partir dos dados de treino, para a identificação de exemplos que diferem significantemente da maioria. Dessa maneira, o principal objetivo da detecção de novidade é examinar se um sistema desvia substancialmente do seu estado de normalidade. Os métodos de detecção de novidades são particularmente adequados para aplicações onde a maioria dos dados está disponível a partir da operação normal do sistema, e as falhas são raras e podem resolver problemas quando novos padrões de dados são raros (ou em alguns casos completamente ausentes) durante o treinamento. Existe uma ampla gama de métodos de detecção de novidades e alguns foram mencionados nesta breve revisão. A escolha do método apropriado depende do tipo de recursos de entrada, disponibilidade de dados de treinamento rotulados, conhecimento do domínio da aplicação entre outros.

2

Aprendizado em Fluxo Contínuo de Dados

Nos últimos anos a prática e a pesquisa na área de Aprendizado de Máquina têm se concentrado no aprendizado em *batch*. Este tipo de aprendizado caracteriza-se pela disponibilidade total dos dados durante o treinamento do algoritmo, que por sua vez processa estes dados diversas vezes se necessário. Por conseguinte, um único modelo de aprendizado é gerado e não sofre reformulações com a chegada de novos dados. Segundo Gama (2010), algoritmos baseados nesta abordagem não são capazes de lidar com dados gerados continuamente em ambientes dinâmicos, pois neste tipo de ambiente, é necessário que o algoritmo assimile de forma incremental novos dados ao modelo de aprendizado. Outra característica importante diz respeito à mudança na distribuição dos dados nestes ambientes, pois em muitas aplicações do mundo real, o processo de geração de dados não é estacionário. Portanto, o algoritmo deve possuir mecanismos que possam detectar estas mudanças de forma rápida e eficiente, de modo a manter o modelo sempre atualizado.

Levando-se em consideração as restrições apresentadas no aprendizado *batch* e o crescente aumento de aplicações capazes de gerar um grande volume de dados continuamente, um recente campo de pesquisa tem ganhado cada vez mais importância: *Aprendizado em Fluxo Contínuo de Dados*. Nesta área, algoritmos lidam com dados que chegam de forma incessante, e de maneira potencialmente infinita cuja distribuição estatística pode mudar ao longo do tempo, intitulados Fluxo Contínuo de Dados(FCD) (BABCOCK et al., 2002; GAMA, 2010).

Gama (2010) descreve FCD como um processo estocástico onde eventos ocorrem continuamente e independentemente. Outra definição é dada por Nguyen, Woon e Ng (2015), que definem um FCD como uma sequência de dados ou amostras: $FCD = \{x_1, x_2, ..., x_i, ...\}$, onde x_i é o *i-ésimo* dado a chegar. Portanto, FCD possuem dados transientes potencialmente infinitos e que podem sofrer mudanças com o tempo, diferentemente dos conjuntos de dados tradicionais que podem ser armazenados em memória. Estas características precisam ser levadas em consideração pelo algoritmo de AM, que de acordo com Aggarwal (2011) deve obedecer às seguintes restrições:

- **Passada única**: Diferentemente da mineração de dados tradicional, onde um método pode ler o mesmo conjunto de dados muitas vezes, cada amostra no fluxo deve ser examinada o mínimo de vezes possível, e não pode haver recuperação de dados já processados. A principal razão é que múltiplas leituras de um mesmo dado torna-se um processo custoso, e pode resultar em modelos desatualizados ao fim do treinamento. Entretanto, esta restrição pode ser suavizada de forma que um algoritmo possa acessar exemplos a curto prazo.

- **Resposta em tempo real**: Algumas das aplicações que geram FCDs exigem que resultados sejam entregues em tempo real, devido principalmente a características críticas que permeiam estas aplicações, como por exemplo mercado de ações, monitoramento de pacientes em hospitais, entre outros.

- **Limite de memória**: Uma das principais motivações da abordagem de fluxo de dados é o processamento de uma quantidade de dados maior que a memória disponível. Em FCD a extensa quantidade de dados que chegam é potencialmente infinita, sendo inviável armazenar todos os dados de um FCD. Desse modo, como pode-se apenas computar e armazenar um pequeno resumo do fluxo, resultados aproximados são aceitáveis.

- **Detecção de mudanças de conceito**: Mudanças de conceito referem-se a situações onde a distribuição estatística dos dados muda de acordo com o tempo. Dessa forma, algoritmos de mineração devem detectar tais alterações para estar sempre atualizados.

O aprendizado em FCD apresenta-se como uma tarefa difícil e desafiadora devido às restrições impostas, e aplicações podem ser encontradas em áreas, como monitoramento de tráfego de rede (AGGARWAL; YU, 2008; YU et al., 2009; ZHANG et al., 2012; BREVE; ZHAO, 2013), redes de sensores (GAMA; GABER, 2007; PAN; YANG; PAN, 2007; ZHANG et al., 2012; BOUCHACHIA; VANARET, 2014), detecção de falhas (LEMOS; CAMINHAS; GOMIDE, 2013a), mineração de *clicks* na *web* (MARIN et al., 2013), medida de consumo de energia (SILVA et al., 2011; ZHANG et al., 2012), fraude de cartão de crédito (WU; LI; HU, 2012), mineração de textos da *web* (FDEZ-RIVEROLA et al., 2007; CHENG et al., 2011; KMIECIAK; STEFANOWSKI, 2011; NAHAR et al., 2014), rastreamento visual (LIU; ZHOU, 2014) e registros de supermercados (YOGITA; TOSHNIWAL, 2013).

Este capítulo apresenta uma revisão sobre aprendizado de máquina em FCD baseada nos artigos de Gama (2010), Aggarwal (2014) e Nguyen, Woon e Ng (2015) destacando conceitos importantes de classificação (Seção 2.1) e agrupamento (Seção 2.2) em FCD.

2.1 Classificação em Fluxos Contínuos de Dados

A classificação é uma tarefa de aprendizado de máquina, na qual um classificador é treinado sobre um conjunto de exemplos rotulados de forma que as classes de exemplos possam ser preditas. Ainda seguindo o formato de aprendizado em *batch* muitos classificadores foram propostos com o intuito de solucionar esta tarefa (SUYKENS; VANDEWALLE, 1999; QUINLAN, 1986). No entanto, problemas em FCD são dinâmicos e classificadores baseados em aprendizado por *batch* não conseguem lidar com este tipo de problema. Logo, precisam ser revisados para atender as características presentes em FCD Gama (2010).

Conforme (LING; LING-JUN; LI, 2009), classificação em FCD consiste em um processo em que o objetivo é desenvolver um classificador a partir de uma sequência de instâncias constituídas de atributos discretos ou contínuos e uma classe cujo valor é discreto, que seja capaz de prever, com alta acurácia, a classe dos exemplos desconhecidos, que estão constantemente chegando na forma de FCD. Além disso uma das principais características da classificação em FCD é a exigência de uma única varredura nos dados.

Existem principalmente duas abordagens para classificação em FCDs: incremental (uma-fase) e *online-offline* (duas-fase) (NGUYEN; WOON; NG, 2015). A primeira funciona com apenas um classificador que sofre atualizações de forma incremental para se adaptar às mudanças no FCD. Esta abordagem, em muitos casos, exige modificações complexas na estrutura dos classificadores e recursos computacionais elevados, mas tem a vantagem de produzir resultados instantaneamente (AGGARWAL, 2014). Propostas baseadas neste tipo de abordagem são Domingos e Hulten (2000), Hulten, Spencer e Domingos (2001b) e Leite, Costa e Gomide (2010b).

Em contraste, a abordagem *online-offline* divide o processo de classificação em duas fases (NGUYEN; WOON; NG, 2015). Durante a primeira fase (*online*), os dados que chegam do fluxo são processados em tempo real e atualizados em sumários. Na fase *offline*, a classificação de dados não rotulados ocorre, com base nos sumários armazenados, sempre que um usuário requisitar. Uma das vantagens desse método é o controle sobre o processo de classificação, dando ao usuário mais opções para analise com menor demanda por recursos computacionais. Além disso, a fase *online* é independente do passo de classificação e pode continuar a processar dados de treinamento. Uma proposta que segue a abordagem *online-offline* é o algoritmo de classificação *On-Demand* (AGGARWAL et al., 2004).

2.1.1 Classificação *Fuzzy* em FCD

O surgimento de mineração em FCD impulsionou a busca de métodos para criar modelos mais dinâmicos e flexíveis, que melhor possam lidar com as mudanças nos dados. Assim, tentando resolver esta questão, os modelos de mineração em FCD que integram

teoria de conjuntos *fuzzy* surgiram recentemente, com resultados promissores.

Em Leite, Costa e Gomide (2010b), os autores propõem um sistema *fuzzy* evolutivo baseado em uma estrutura de rede neural granular modelada a partir de um FCD *fuzzy*. De acordo com os autores, seus resultados sugerem que sua abordagem pode lidar com dados *fuzzy* com êxito e supera outras abordagens em termos de precisão, transparência e compacidade. Além disso, Hashemi e Yang (2009) introduziu uma árvore de decisão flexível (FlexDT) com base na teoria dos conjuntos *fuzzy*, para a classificação de FCD. Este algoritmo integra a árvore de decisão (VFDT) (HULTEN; SPENCER; DOMINGOS, 2001a) com lógica *fuzzy*. A abordagem FlexDT foi comparada com algoritmos de classificação de FCD existentes usando diferentes FCDs e conforme os autores, os resultados sugerem que a FlexDT oferece benefícios significativos para a classificação de FCD em cenários do mundo real, especialmente onde mudança de conceito, ruído e valores faltantes coexistem. Além disso, na medida em que sabemos, este trabalho pode ser considerado o primeiro a trazer a lógica *fuzzy* ao problema de classificação de FCD. Isazadeh, Mahan e Pedrycz (2016) apresentam uma melhoria no algoritmo FlexDT chamado MFlecDT, possibilitando o algoritmo ter mais de duas partições *fuzzy*. Este algoritmo foi comparado aos algoritmos da árvore de decisão para a classificação de FCD e obteve um melhor desempenho de acordo com os autores.

Embora exibam bons resultados, os algoritmos baseados em teoria de conjuntos *fuzzy* mencionados são todos incrementais e, como dito anteriormente, podem consumir mais tempo e precisam de mais recursos computacionais do que os algoritmos baseados na abordagem *online-offline*.

2.2 Agrupamento em Fluxos Contínuos de Dados

Em aprendizado de máquina, é comum verificar a falta de informação de classe, muitas vezes por conta da natureza do domínio ou pela dificuldade em rotular exemplos. Dessa maneira, a tarefa de agrupamento caracteriza-se pela descoberta de padrões nos dados a partir de alguma caracterização de regularidade, sendo esses padrões denominados agrupamentos (DECKER; FOCARDI, 1995; MCCALLUM; NIGAM; UNGAR, 2000). Exemplos contidos em um mesmo grupo são mais similares, segundo alguma medida de similaridade, do que aqueles contidos em grupos diferentes.

Métodos de agrupamento baseado em *batch* consideram que todos os exemplos estão disponíveis na memória, o que não é uma realidade para os domínios de FCD, pois como dito anteriormente o armazenamento é inviável devido ao seu tamanho potencialmente infinito. Nessas circunstâncias, uma das soluções para contornar este problema é a criação de sumários ou sinopses da informação encontrada nos exemplos.

2.2.1 Estruturas de Sumarização de Exemplos

Diferentes técnicas tem sido desenvolvidas para a sumarização e armazenamento da informação contida em FCD (GAMA; GABER, 2007; NGUYEN; WOON; NG, 2015). As abordagens mais frequentemente utilizadas para a definição de estruturas de sumarização incluem:

- **Arranjo de Protótipos**: consiste em um conjunto de protótipos (medóides, centróides, representantes, etc.) que sumarizam a partição dos dados (GUHA et al., 2000; O'CALLAGHAN et al., 2002).
- **Grades de Dados**: particiona o espaço n-dimensional de atributos em células de densidade. O processo de manutenção da estrutura de grade mapeia novos exemplos às células do espaço de atributos (CAO et al., 2006).
- **Vetor de Atributos** conservam estatísticas a respeito de um grupo de dados, de forma que possam ser calculadas informações importantes (centróide, desvio padrão). Esta técnica será abordada com mais detalhes na Subseção 2.2.2, pois seu entendimento é indispensável para a proposta que será apresentada.

Métodos baseados em arranjo de protótipos geralmente consideram que os FCD estão divididos em partes (*chunks*) que são utilizadas para o cálculo dos protótipos. Apesar de simples esta abordagem não representa da melhor forma o comportamento do FCD ao longo do tempo, pois não se consegue identificar os momentos exatos onde ocorrem mudanças no FCD (GAMA; SEBASTIÃO; RODRIGUES, 2013). Já métodos baseados em grades de dados, possuem um tempo de processamento rápido e são independentes do número de exemplos. Entretanto, são dependentes do número de células definidas e os formatos de agrupamento estão limitados a união de células retangulares.

2.2.2 Vetor de Atributos

Zhang, Ramakrishnan e Livny (1996) introduziram pela primeira vez a utilização de vetores de atributos para a sumarização de grandes conjuntos de dados, devido a natureza potencialmente infinita desses tipos de conjuntos. A estrutura de sumarização proposta para representar um grupo de exemplos, possui três estatísticas: N a cardinalidade do grupo, LS a soma linear dos exemplos pertencentes ao grupo e SS a soma quadrática dos exemplos do grupo. Os componentes LS e SS são estruturas k-dimensionais onde k é definido de acordo com o dimensão dos exemplos. Segundo Aggarwal et al. (2003a) este vetor possui propriedades incrementais, sendo possível a inserção de um novo exemplo para a atualização das estatísticas. Além disso este vetores possuem a propriedade de serem mesclados para criar um terceiro vetor com a soma dos componentes dos vetores

mesclados. Uma das vantagens desse tipo de estrutura é a fácil manutenção, e flexibilidade de utilização em diferentes métodos de agrupamentos.

A característica incremental dessas estruturas permite computar e manter estatísticas simples sobre FCDs (GAMA; GABER, 2007). Por exemplo, pode-se obter o exemplo médio de um FCD, para isso precisa-se apenas manter o número de exemplos processados (n) e a soma dos mesmos ($\sum_{i=1}^{n} x_i$), onde x_i representa o i-ésimo exemplo do FCD. Dessa forma, com a chegada de um novo exemplo, a média pode ser atualizada de forma incremental. De maneira semelhante outras estatísticas, tais como o desvio padrão, também pode ser calculadas.

No cenário de FCD, um dos métodos que se baseiam nesta abordagem é o *CluStream* (AGGARWAL et al., 2003a). Neste método, estruturas nomeadas micro-grupos são definidas de acordo com o vetor de atributos composto pelos componentes (LS, SS, N), proposto por Zhang, Ramakrishnan e Livny (1996), adicionando mais dois componentes: a soma de *timestamps* ($CF1^t$) e a soma quadrática de *timestamps* ($CF2^t$), onde *timestamps* são definidos como o momento de chegada de um exemplo. Estes dois novos componentes foram incluídos para possibilitar o tratamento de aspectos temporais de FCD. Este algoritmo será melhor detalhado na Subseção 2.2.3.

2.2.3 Técnicas de Agrupamento em Fluxo de Dados

Existem diferentes abordagens de agrupamento em FCD, algumas são baseadas em blocos (*chunks*) e permitem o processamento do FCD por partes (GUHA et al., 2000; O'CALLAGHAN et al., 2002), porém os modelos criados a partir desta abordagem não representam as mudanças ocorridas ao longo dos FCDs de maneira adequada. Outras abordagens existentes na literatura podem ser categorizadas em incrementais e *online-offline* (NGUYEN; WOON; NG, 2015). Entretanto, muitas delas baseiam-se no *framework online-offline* proposto primeiramente por Aggarwal et al. (2003b). Este *framework* pode ser definido em duas etapas, onde na primeira etapa conhecida como fase *online* sumariza-se os dados, para que na fase *offline* possam ser agrupados sempre que houver requisição de um usuário. Apesar de considerar um único modelo geral para a aprendizagem, essas abordagens se diferenciam quanto aos mecanismos adotados para cada uma das fases do *framework*. Dentre os algoritmos de agrupamento para FCD destacam-se principalmente: DenStream (CAO et al., 2006), D-Stream (CHEN; TU, 2007), CluTree (KRANEN et al., 2011) e CluStream (AGGARWAL et al., 2003a) que será abordado com maior detalhe a seguir, para mehor compreensão da proposta deste trabalho.

CluStream

CluStream (AGGARWAL et al., 2003a) é um algoritmo de agrupamento para FCD que utiliza vetores de características (micro-grupos) durante sua fase *online* e o agrupa-

mento *k-means* para encontrar macro-grupos durante a fase *offline*. A ideia principal é sumarizar o FCD em micro-grupos com o objetivo de identificar padrões no fluxo que possam ser agrupados de acordo com alguma métrica de similaridade.

A estrutura de micro-grupos deste método é fundamentada na proposta de Zhang, Ramakrishnan e Livny (1996) de vetores de atributos, entretanto, define cinco estatísticas $(\overline{CF2^e}, \overline{CF1^e}, CF2^t, CF1^t, n)$, onde $\overline{CF2^e}$ é a soma quadrática dos exemplos do grupo, $\overline{CF1^e}$ é a soma linear dos exemplos que pertencem ao grupo, $CF2^t$ é a soma quadrática de *timestamps*, $CF1^t$ é a soma linear de *timestamps* e n é a cardinalidade do micro-grupo.

O método inicia aplicando o algoritmo de agrupamento *kmeans* em uma quantidade de pontos inicias do FCD escolhida pelo usuário, os grupos resultantes são então sumarizados em micro-grupos para que a fase *online* possa ser iniciada. Durante a fase *online* a distância de cada novo exemplo é verificada para cada micro-grupo e o mais próximo, cujo exemplo se encontra dentro de seu raio, é atualizado com o novo exemplo. Caso o novo exemplo esteja fora do raio de todos os micro-grupos existentes, é necessária a criação de um novo micro-grupo para esse exemplo.

Para criar um novo micro-grupo é necessário que ocorra uma remoção de um micro-grupo ou uma junção de dois outros. A remoção é feita caso o micro-grupo que possui o menor valor do m-ésimo (variável definida pelo usuário) percentil de uma gaussiana, cujo desvio padrão e a média são calculadas por meio dos valores de estatísticas desse micro-grupo, for menor que um limiar estabelecido pelo usuário. Caso contrário esse micro-grupo é mesclado com outro mais próximo.

Ao longo do tempo os estados dos micro-grupos são salvos em uma estrutura piramidal que permite a recuperação dos mesmos em diversos espaços de tempo. A fase *offline* ocorre a partir da requisição do usuário, permitindo a flexibilidade de explorar o FCD em diferentes momentos. Nesta fase o algoritmo *kmeans* é aplicado a fim de analisar o FCD e detectar agrupamentos.

Este método apresenta uma ótima estrutura para a mineração de dados em FCD, entretanto, não é robusta a ruídos. Além disso, é muito dependente da forma como é inicializado e da quantidade de pontos iniciais a ser utilizado.

Agrupamento *Fuzzy* em FCD

A extensão *fuzzy* de métodos de aprendizado tem beneficiado diversas tarefas em AM, por tornar o processo de aprendizado mais flexível, incorporando características definidas pela teoria de conjuntos *fuzzy* e lógica *fuzzy*. Com o sucesso de métodos *fuzzy* em AM clássico e o aspecto mutável dos domínios de FCD, a busca por estratégias que façam uso de conceitos *fuzzy* torna-se relevante, e pode contribuir para o aspecto adaptativo esperado para o aprendizado no contexto de FCD.

Diversos métodos que incorporam conceitos *fuzzy* são utilizados para agrupamento em FCD. *Single-Pass Fuzzy C-Means* (SPFCM) (HORE; HALL; GOLDGOF, 2007b) é uma extensão do algoritmo não-supervisionado FCM (BEZDEK; EHRLICH; FULL, 1984) cujo objetivo é o agrupamento de conjuntos de dados onde a grande dimensionalidade impede o seu armazenamento total em memória. No entanto, é uma técnica que considera que a distribuição dos dados é estática, o que deixa de lado a característica mutável do contexto de FCD.

Outros métodos baseados em SPFCM buscaram abranger a escolha de algoritmo de agrupamento ponderado (HORE; HALL; GOLDGOF, 2007a) utilizando métodos *fuzzy*, como o Gustafson-Kessel (GUSTAFSON; KESSEL, 1978) e o *Possibilistic Fuzzy C Means* (KRISHNAPURAM; KELLER, 1996). Também foram propostos métodos semissupervisionados baseados no SPFCM (MAGDY; BASSIOUNY, 2010), e algoritmos que tratavam problemas relacionados a desvio de conceito (MOSTAFAVI; AMIRI, 2012).

Além disso, os métodos propostos por Hore et al. (2008), Labroche (2014), Zhu et al. (2014) e Li et al. (2016) também realizam o agrupamento dos exemplos em FCD, mas utilizam abordagens diferentes do SPFCM.

FuzzStream

Recentemente, um método de agrupamento em FCD baseado no *framework (online-offline)* e algoritmo CluStream proposto por Lopes e Camargo (2017) integra teoria de conjuntos *fuzzy* na suas duas fases, servindo como base para a proposta deste trabalho. Sua estrutura de sumarização baseada no vetor de atributos *Temporal Cluster Feature* (ZHOU et al., 2008) e nomeada *Fuzzy Micro-Cluster* (FMiC) é definida como um vetor $(\overline{CF2^e}, \overline{CF1^e}, t, M)$, onde M é a cardinalidade escalar do micro-grupo, definida como a soma das pertinências μ_j de todos os n exemplos e_j que possuem pertinência maior que zero no micro-grupo (Equação 2.1) (PEDRYCZ; GOMIDE, 1998, p. 13), $\overline{CF2^e}$ é a soma quadrática dos exemplos ponderada pelos valores de pertinência μ_j (Equação 2.2), $\overline{CF1^e}$ é a soma linear dos exemplos ponderada pelos valores de pertinência μ_j, onde e_j e μ_j são respectivamente o j-ésimo exemplo e seu valor de pertinência no micro-grupo e t é o *timestamp* do exemplo mais atual.

$$M = \sum_{j=1}^{n} \mu_j \tag{2.1}$$

$$\overline{CF2^e} = \sum_{j=1}^{n} \mu_j e_j^2 \tag{2.2}$$

$$\overline{CF1^e} = \sum_{j=1}^{n} \mu_j e_j \tag{2.3}$$

No processo de manutenção da estrutura de FMiC que se da na etapa *online* com a chegada de novos exemplos e_j, processa-se cada novo exemplo um a um e previamente verifica-se a quantidade de micro-grupos existentes. Caso esta quantidade seja menor que um número mínimo escolhido pelo usuário, cria-se um novo micro-grupo *fuzzy* para o exemplo que esta sendo processado.

Caso contrário, é calculada a pertinência do exemplo e_j para todos os FMiC disponíveis, como é feito no algoritmo *Fuzzy C-Means* (BEZDEK; EHRLICH; FULL, 1984). Se o valor máximo dentre as pertinências obtidas (u_j^{max}) for menor que um limiar θ definido pelo usuário, então cria-se um novo FMiC; caso contrário, o exemplo e_j será adicionado a todas as estrutura de FMiC, ponderado pelo seu valor de pertinência em cada FMiC.

A criação de um novo FMiC ocorre caso o valor máximo de pertinência do exemplo seja menor que θ, inicia verificando se não há espaço disponível para armazenamento do novo FMiC, nesse caso o FMiC com menor *timestamp* t é removido do conjunto de FMiC, para a adição do novo FMiC a partir do exemplo e_j, com pertinência $\mu_j = 1$. Caso contrário, então nenhum FMiC é removido e um novo é criado a partir do exemplo e_j, com pertinência $\mu_j = 1$.

Após verificar se um novo FMiC deve ou não ser criado, executa-se o passo de mescla de FMiC que tem o objetivo de reduzir a estrutura de sumarização por meio da junção de FMiCs muito próximos. Entretanto este processo é realizado de forma que a quantidade de FMiC no conjunto não seja menor que o mínimo de FMiC definido pelo usuário. Neste passo, primeiramente é calculada a distância euclidiana entre todos os FMiC disponíveis, e depois calculado a pertinência de cada FMiC para os $minFMiC$ mais próximos, onde $minFMiC$ é um valor também definido pelo usuário que representa a quantidade de FMiCs que poderão ser mesclados com o FMiC sendo processado. Então cria-se uma lista de FMiC que podem ser mesclados, cujo valor máximo de pertinência seja maior ou igual a um limiar Θ definido.

Para cada $FMiC_i$ na lista de mescla, verifica-se qual o FMiC mais próximo, se este ainda não foi mesclado com outro, então os FMiC são mesclados pela soma das componentes $\overline{CF2^e}, \overline{CF1^e}, M$ e a componente t é atualizada de acordo com o FMiC mais atual entre os mesclados. Os FMiC mesclados são removidos da lista para mescla. O processo é repetido até que a lista de mescla esteja vazia.

O passo *offline* consiste em transformar cada FMiC em exemplos ponderados para serem aplicados no método de agrupamento *Weighted Fuzzy C-Means* (WFCM). Para cada FMiC na estrutura de sumarização, obtém-se um protótipo dividindo $\overline{CF1^e}$ por M, e o peso do protótipo é dado por M. O conjunto de protótipos ponderados é dado como entrada para o algoritmo de agrupamento WFCM, que gera uma partição que pode ser generalizada para esse momento específico no fluxo processado. A partir desta partição, macro-agrupamentos podem ser obtidos.

Segundo Lopes e Camargo (2017) os métodos *CluStream* e *FuzzStream* reconhecem concentrações de ruído mais densas, entretanto, o método FuzzStream é notadamente mais robusto para dados escassos e ruidosos, e em casos de suaves sobreposições entre agrupamentos, FuzzStream apresenta uma melhor eficácia na identificação de agrupamentos.

2.3 Considerações Finais

No aprendizado em FCD os métodos devem ser capazes de tratar os aspectos característicos de FCD. Abordagens baseadas em aprendizado supervisionado apresentam bons resultados, mas dependem de um conjunto rotulado, o que pode ser uma premissa não realista para domínios onde a velocidade de chegada e dimensionalidade dos exemplos é imprevisível.

A teoria de conjuntos *fuzzy* pode colaborar para um aprendizado mais flexível dentro dos domínios de FCD. Apesar de existirem algumas propostas, no agrupamento a maioria baseia-se na divisão em *chunks* de exemplos e não por processamento incremental. No próximo capítulo serão apresentados conceitos e algoritmos relacionados a tarefa de DN em FCD.

3

Detecção de Novidade em Fluxos Contínuos de Dados

Como explicado no Capítulo 1 a tarefa de Detecção de Novidade (DN) compreende identificar que um dado exemplo não rotulado, ou coleção deles, encontra-se divergente dos demais conceitos (subconjunto de um domínio) aprendidos (FARIA et al., 2016a). Este, apresenta-se como um importante problema/tarefa a ser tratado em AM e possui consideráveis estudos em contextos *batch* (PIMENTEL et al., 2014; MARKOU; SINGH, 2003). Entretanto, recentemente com o advento da mineração em FCD a tarefa de DN surge com novos desafios a serem tratados (FARIA et al., 2016a; GAMA, 2010). Para facilitar o entendimento deste capítulo definiremos a tarefa de DN em FCDs da seguinte maneira: torna possível reconhecer um conceito novidade, que pode indicar o surgimento de novos conceitos, uma mudança ocorrida nos conceitos conhecidos ou a presença de ruído (GAMA, 2010).

No cenário de mineração em FCDs exemplos são gerados constantemente. Dessa forma, FCDs podem ter tamanhos potencialmente ilimitados além de decorrerem em alta velocidade e possuírem distribuição mutável ao longo do tempo, o que motiva a possibilidade do surgimento de novos conceitos e o desaparecimento de conceitos já aprendidos. Portanto, ressonante com estas características, a tarefa de DN em FCDs apresenta diferentes desafios que incluem (GAMA, 2010; FARIA et al., 2016a):

- *Desvio de Conceito:* apresentam-se como uma extensão dos conceitos já aprendidos. Dessa forma esta particularidade dificulta a distinção de novos conceitos a partir do modelo;
- *Ruído ou outlier:* apresentam a possibilidade de novos conceitos, porém devem ser analisados para que não sejam confundidos;
- *Conceitos recorrentes:* são conceitos aprendidos, que podem ocorrer esporadicamente em intervalos de tempos espaçados e portanto, podem ser confundidos com o aparecimento de um novo conceito, caso o modelo o esqueça;

- *Evolução de conceitos:* esta característica define-se como o aumento do número de classes do problema ao longo do tempo.

Dentre as diversas aplicações de DN em FCDs encontradas na literatura, podem ser citadas: Detecção de Intrusos (COULL et al., 2003), Detecção de Falhas (ZHANG et al., 2006), Diagnósticos Médicos (PERNER, 2009), Detecção de Regiões de Interesse em Imagens (SINGH; MARKOU, 2004), Detecção de Fraudes (WANG et al., 2003), Detecção do Tipo de Cobertura Florestal (MASUD et al., 2011), Filtros de Spam (HAYAT et al., 2010) e Classificação de Texto (LI; CROFT, 2006).

Para o melhor entendimento deste trabalho, este capítulo apresenta uma revisão sobre DN em FCDs baseada nos artigos de Faria et al. (2016a) e Gama (2010), destacando conceitos importantes da tarefa de DN em FCDs (Seção 3.1), assim como técnicas e métodos propostos (Subseção 3.2.1) e como são aplicados os conceitos de teoria *fuzzy* neste contexto (Subseção 3.2.2).

3.1 Mudança de Conceito e Evolução de Conceito

Em AM o termo conceito pode ser definido como uma representação de um subconjunto de dados formada através da combinação de atributos e regras, as quais são compostas por conjunções de restrições que qualificarão quais exemplos pertencem ao conceito. Nesse contexto, o aprendizado por conceito caracteriza-se pela inferência de uma função através dos exemplos do conjunto de treinamento que mapeia entradas em conceitos (MITCHELL, 1997).

Situacionalmente em aplicações do domínio de FCDs, a distribuição dos dados que representam os conceitos aprendidos pelo modelo pode sofrer mudanças ao longo do tempo como consequência à natureza não estacionária do ambiente gerador (WIDMER; KUBAT, 1996). Por exemplo, em uma tarefa de classificação, as classes podem apresentar alterações com o tempo. Assim, no tempo t valores de atributos utilizados para identificar um determinada classe, podem não ser válidos no tempo $t + k$. Portanto, um modelo de classificação gerado no tempo t não será tão eficiente no tempo $t + k$ se não for atualizado com as mudanças ocorridas no período de tempo k. Em aprendizado em FCDs, esse tipo de alteração na distribuição dos dados de conceitos conhecidos pelo modelo ao longo do tempo é denominada desvio de conceito.

Outro problema referente a natureza mutável presente em FCDs é o surgimento de novos conceitos, ou evolução de conceito (FARIA et al., 2016a). Ainda referente a tarefa de classificação, por exemplo, a evolução de conceito caracteriza-se pelo surgimento de classes não observadas a *priori*, e dessa forma, os conceitos aprendidos pelo modelo em um tempo t podem não mais representar o estado do conjunto de dados no tempo $t + k$,

o que pode interferir nos resultados gerados pelo modelo de classificação, caso o modelo não seja atualizado com os novos conceitos que surgiram no período de tempo k.

A detecção destas alterações nas distribuição dos dados torna-se importante pois observações antigas, que refletem um comportamento passado, podem ser irrelevantes para o modelo de aprendizado, interferindo nos resultados obtidos pelos modelos de AM, portanto, o algoritmo deve esquecer essas informações (GAMA, 2010). Essas mudanças também podem ser categorizadas de acordo com a velocidade que elas acontecem (GAMA, 2010):

- **Repentinas**: Apresentam-se abrupta e irreversivelmente nos dados de exemplo;
- **Incrementais ou Graduais**: Ocorrem de maneira lenta em relação ao tempo;
- **Recorrentes**: Mudanças temporárias que geralmente são revertidas com o tempo;
- **Ruidosas**: Constituem-se de ruídos e não devem ser consideradas mudanças.

3.2 Visão Geral de DN em FCDs

Segundo Faria et al. (2016a), a maioria dos métodos para DN em FCDs são caracterizados por duas etapas, conhecidas como *offline* e *online*. Na primeira etapa, denominada offline, é pressuposto a disponibilidade de um conjunto de exemplos rotulados, ou conceitos conhecidos, que são utilizados para a indução de um modelo de decisão inicial. Durante a fase online, para cada exemplo não rotulado do FCD, verifica-se se o modelo pode explicá-lo, caso contrário este exemplo é classificado como potencial ruído, novidade ou extensão de um conceito conhecido. As Figura 3.1 e Figura 3.2 representam estas etapas.

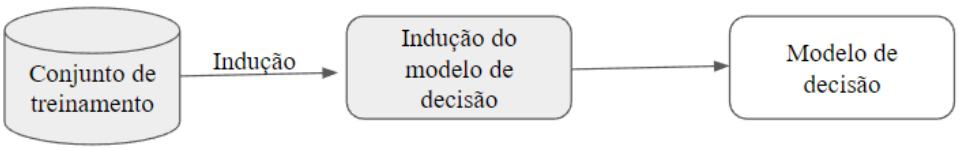


Figura 3.1 – Fase *offline*: criação do modelo de decisão a partir de dados rotulados (FARIA et al., 2016a)

3.2.1 Métodos de DN em FCDs

As primeiras propostas para a tarefa de DN em FCDs na literatura supõem que um conjunto de dados disponíveis a priori representam uma única classe, chamada de classe normal ou conceito normal. Dessa forma, por meio dos exemplos disponíveis induz-se um

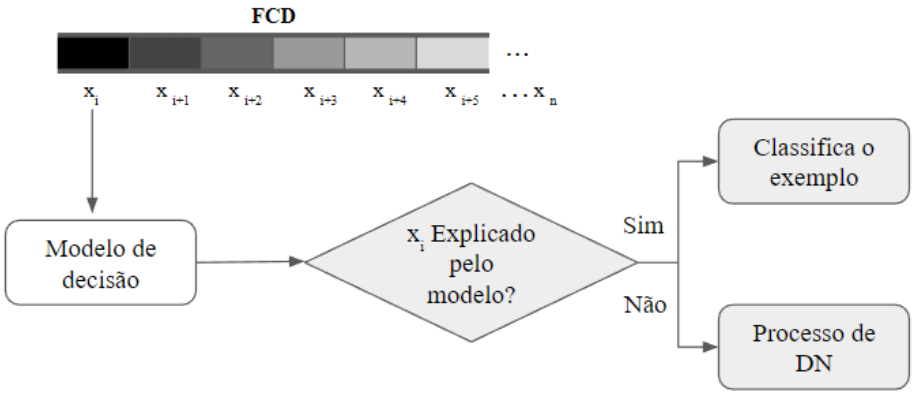


Figura 3.2 – Fase *online*: identificação potencial ruído, novidade ou extensão de um conceito conhecido (FARIA et al., 2016a)

modelo de decisão, nomeado de modelo normal, e dados que não podem ser explicado por esse modelo são considerados novidade. Essa abordagem denominada binária é utilizada por algoritmos como Hayat e Hashemi (2010), Spinosa et al. (2009), Albertini e Mello (2007), Rusiecki (2012), Tan, Ting e Liu (2011), Krawczyk (2013), que serão detalhados na Subseção 3.2.1.1.

Recentemente, houve o surgimento de diferentes trabalhos na literatura que tratam a tarefa de DN em FCD em um contexto multi-classe: Farid et al. (2013), Farid e Rahman (2012), Masud et al. (2010), Masud et al. (2011), Al-Khateeb et al. (2012b), Al-Khateeb et al. (2012a), Faria et al. (2016b), Abdallah et al. (2016), Haque, Khan e Baron (2016), estes métodos serão detalhados na Subseção 3.2.1.2. Segundo Faria et al. (2016a), na tarefa de DN multi-classe em FCD o conjunto de treinamento D_t é composto por exemplos x_i e cada exemplo possui uma saída y_i, onde $y_i \in Y = (C_1, C_2, .., C_n)$, C_i representa a i-ésima classe conhecida e n o número total de classes observadas. Dessa forma, assim que novos exemplo chegam através do FCD, novas classes podem surgir expandindo o conjunto de classes conhecidas $Y' = (C_1, C_2, .., C_n, CN_1, CN_2.., CN_k)$, onde CN_i representa a i-ésima classe não identificada a *priori* e k a quantidade de classes novas que surgiram.

3.2.1.1 Abordagens Binárias

Em Spinosa et al. (2009) os autores propõem o algoritmo OLINDDA que na etapa *offline* agrupa o conjunto de exemplos disponíveis em grupos que representem a classe normal. Os grupos são definidos pela utilização de um algoritmo de agrupamento e são representados por um centro e um raio. Estes grupos são definidos como hiperesferas e representam o conceito normal e cujo centro é calculado como o sendo o centróide do grupo e o raio é a distância ao elemento do grupo mais distantes do centro.

Durante a fase *online* essas hiperesferas são utilizadas para detectar possíveis extensões dos conceitos existentes ou conceitos novidades. Dessa forma, um novo exemplo

é classificado como normal se estiver dentro de alguma das hiperesferas. Caso contrário, o exemplo é armazenado em um *buffer* temporário, denominado memória de curto prazo, para que futuramente verifique-se a qual classe o modelo deve rotulá-los: extensão de um conceito conhecido ou conceito novidade.

Hayat e Hashemi (2010) propõem o algoritmo DETECTNOD *(DiscrETE Cosine Transform based NOvelty and Drift detection)* que na fase *offline* calcula grupos por meio do agrupamento de dados inicialmente disponíveis. A partir desses, sub-grupos são calculados utilizando um algoritmo de agrupamento em cada grupo. Uma etapa de interpolação dos sub-grupos é executada para que possuam a mesma quantidade de exemplos, seguida de uma etapa final que aplica uma transformada discreta de cosseno nos sub-grupos, utilizando os coeficientes gerados para representá-los com o intuito de diminuir o uso de memória. Já na fase *online* o FCD é divido em dois blocos e cada novo exemplo é processado individualmente, assim, uma pontuação de novidade é calcula levando-se em consideração o sub-grupo mais próximo do exemplo. Esta pontuação é calculada indicando o quanto ele é desconhecido, se essa pontuação for maior que um limiar definido pelo usuário, o exemplo é rotulado como desconhecido. Ao fim de cada bloco os exemplos rotulados como desconhecidos são agrupados para que verifique-se se constituem uma extensão de um conceito conhecido ou um conceito novidade.

Em Albertini e Mello (2007), os autores apresentam o algoritmo SONDE *(Self-Organizing Novelty Detection)* caracterizado por uma rede neural não supervisionada e que originalmente não foi proposto para o domínio de FCDs. Entretanto de acordo com (FARIA et al., 2016a) pode ser facilmente usada em FCDs. Esta rede é composta por uma camada de entrada, uma camada competitiva e uma camada de saída, denominada BMU (*Best Matching Unit*), onde o neurônio vencedor é usado pra representar um novo exemplo recebido. Neste modelo cada neurônio é representado por um centróide, um raio médio e um nível de similaridade para reconhecer novos exemplos. Portanto, entradas similares são sempre associadas a um mesmo neurônio. Os exemplos usados para o treinamento da rede neural representam o conceito normal. Durante a fase *online* a rede classifica um novo exemplo identificando o neurônio que melhor possa representá-lo. 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.

Rusiecki (2012), apresenta duas redes neurais *feedforward* para a tarefa de DN. Enquanto uma é treinada utilizando a função de erro robusta a ruídos LMLS *Least Mean Log Square* que introduz uma função que descreve a influência que *outliers* pontencialmente podem ter durante o treino da rede, a outra utiliza o algoritmo *Levenberg-Marquardt*

(HAGAN; MENHAJ, 1994) durante o treino. Assim os exemplos usados no treinamento das redes representam o conceito normal. Na etapa *online* assim que um novo exemplo é submetido às duas redes, verifica-se a diferença entre os resultados produzidos pelas duas redes. Caso seja menor que um limiar, então o exemplo é classificado como pertencente ao conceito normal, de outra forma, ele é rotulado como novidade.

No trabalho de Krawczyk (2013), os autores propõem o algoritmo WOCSVM *(Weighted One-Class Support Vector Machine)* (BICEGO; FIGUEIREDO, 2009), para representar o conceito normal, uma característica desse algoritmo é a adição de um peso a cada exemplo do conjunto de treinamento. 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.

Tan, Ting e Liu (2011) propõem um *ensemble* de árvores de decisão denominadas *Half-Space-Trees* (HS-Tree) para representar a classe normal. Na fase *online* calcula-se uma pontuação para cada novo exemplo não rotulado para cada árvore, usando a massa r (uma informação calculada na fase de treinamento para cada nó). A soma das pontuações do exemplo, obtidas em cada árvore, define uma pontuação de novidade, que é usada para decidir se um novo exemplo é novidade ou não.

3.2.1.2 Abordagens Multi-Classe

Al-Khateeb et al. (2012a) propõem um ensemble de classificadores baseado em *classes* que utilizam o conceito de micro-grupos, nomeado CLAM. Durante a fase *offline* a inicialização de cada classificador é similar ao algoritmo de classificação em FCDs *On-Demand* (AGGARWAL et al., 2004), onde para cada classe conhecida do problema são criados micro-grupos baseados nos exemplos rotulados disponíveis, cada micro-grupo é sumarizado pelo seu raio, centro e número de elementos que possui. O conjunto de micro-grupos referente a uma classe representa um classificador e ao todo são criados L classificadores, sendo que L representa o número de classes conhecidas pelo modelo. Na etapa *online* quando o CLAM recebe um novo exemplo, cada classificador verifica se o exemplo é desconhecido. Um exemplo é considerado desconhecido por um classificador se ele não esta dentro de nenhum dos micro-grupos pertencentes ao classificador. Se todos os classificadores não conseguirem rotular um novo exemplo então este é rotulado como desconhecido. Caso contrário, o exemplo é classificado em uma das classes conhecidas do problema, verificando qual o micro-grupo mais próximo dentre todos os micro-grupos de ensemble de classificadores. Para detecção de novidade, o algoritmo CLAM agrupa os exemplos classificados como desconhecidos e calcula uma medida interna de validação do agrupamento, denominada q-NSC, que avalia a coesão e separação por meio da comparação com grupos existentes no *essemble* de classificadores. CLAM pressupõe uma

segmentação do fluxo de dados entre *chunks* com exemplos rotulados e não rotulados, um *chunk* é uma porção contínua de dados do FCD, sendo o seu tamanho definido pelo usuário. Nesse cenário, o algoritmo sofre alterações somente com a chegada de exemplos rotulados, o que não é ideal em problemas de FCDs. Ademais, o método não é capaz de distinguir entre diferentes padrões-novidade no mesmo *chunk*.

No trabalho de Masud et al. (2011), um *ensemble* de classificadores denominado ESCMiner é proposto, baseados em árvores de decisão. Em respeito a árvores de decisão, cada uma é induzida usando os exemplos de um *chunk* do FCD cujo tamanho é definido pelo usuário e após a construção das árvores de decisão, um agrupamento é realizado nos nó folha de cada árvore, o qual é usado para identificar exemplos desconhecidos. Os autores também propõem um *ensemble* de classificadores *KNN*. Durante a fase *online* verifica-se se o exemplo pode ser classificado por algum dos classificadores. Caso contrário este exemplo é marcado como desconhecido. A classificação de um exemplo é definida através do voto da maioria dos classificadores. Na detecção de novidade ESCMiner se comporta de forma similar ao método CLAM. Entretanto, similarmente ao método de Al-Khateeb et al. (2012a), ESCMiner pressupõe dados rotulados ao longo do FCD para a atualização do modelo de decisão, além de não ser capaz de detectar mais de um padrão novidade.

Semelhante ao método proposto por Al-Khateeb et al. (2012a) o algoritmo MCM (MASUD et al., 2010), propõe um *ensemble* de classificadores KNN formados por K micro-grupos, obtidos através de *chunks* do FCD de tamanhos iguais usando o algoritmo K-Means, sendo que cada micro-grupo é representado por uma hiperesfera composta pelo centróide, raio calculado como a distância do centróide para o exemplos mais distante do micro-grupo e a frequência de ocorrência de cada classe. Na fase *online* um novo exemplo é classificado como desconhecido se não se encontra dentro do raio de nenhum dos micro-grupos existentes, e então armazenado em um *buffer* para futuras análises. Entretanto, os raios de cada micro-grupo são somados a um limiar definido pelo usuário que determina uma região ao redor do micro-grupo denominado espaço *slack*, cuja intenção é adicionar exemplos disposto nas proximidades do perímetro de micro-grupos que podem ser categorizados como potenciais extensões de conceitos conhecidos. Portanto, se um exemplo está fora do raio da hiperesfera, mas dentro do espaço slack, ele é considerado pertencente a essa hiperesfera. Esse método utiliza o Coeficiente Gini para diferenciar extensões de classe de evoluções de conceito e também para a validação de uma classe novidade, juntamente com a validação de agrupamentos q-NSC. Apesar de pressupor dados rotulados ao longo do FCD, esse método consegue distinguir diferentes classes novidade, por meio de componentes conectados do grafo que representa os grupos de exemplos marcados como novidade.

O trabalho de Farid et al. (2013) propõe uma extensão do trabalho de Farid e

Rahman (2012) por meio do uso de um *ensemble* de três árvores de decisão. Na etapa *offline* o método classifica cada exemplo do conjunto de treinamento utilizando o algoritmo Naive Bayes e a probabilidade resultante de cada exemplo é associada ao exemplo como seu peso inicial. A primeira árvore é construída usando um conjunto de dados obtido a partir de uma técnica de seleção de recomposição. Para construir as outras duas árvores, utiliza-se os exemplos que possuem os maiores pesos, sendo que o peso dos exemplos é atualizado, de acordo como sua classificação, realizada usando a primeira árvore de decisão. Assim como o trabalho de Masud et al. (2011) cada nó folha da árvore é agrupado e além disso, é computada 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. Durante a fase *online* este método usa uma estratégia similar ao de Masud et al. (2011), 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. Se um exemplo é classificado como desconhecido, então é colocado em memória temporária e o valor 1 é atribuído ao *flag* novidade. Caso o número de exemplos classificados por um nó folha aumente ou diminua, esse exemplo pertence a uma classe novidade. De maneira parecida aos métodos Al-Khateeb et al. (2012a), Masud et al. (2011), esse método pressupõe dados rotulados durante o FCD e não é capaz de distinguir diferentes pardrões novidade.

Abdallah et al. (2016) propõem um método denominado AnyNovel, que na fase *offline* define estruturas denominadas BLM *(Baseline Learning Model)*, A estrutura BLM é composta por estatísticas divididas em três categorias: grupos, sub-grupos e holísticas. Estatísticas relacionadas a grupos são calculadas por meio do resultado do agrupamento dos dados de treino disponíveis, são elas: centróide, número de sub-grupos, número de exemplos no grupo, raio e distância máxima entre quaisquer 2 pares de sub-grupos. Já as estatísticas de sub-grupos são determinadas após executar um algoritmo de agrupamento para cada grupo, são elas: centróide, número de exemplos no sub-grupo e raio. Estatísticas holísticas são mais genéricas e relacionadas a estrutura do agrupamento, são elas: Número de grupos, centróide de todos os exemplos de treino e número total de exemplos de treino. Durante a fase *online* dois principais componentes denominados CVC *(Cohesion Validation Component)* e *Observer* são utilizados para identificar novidade no FCD. O componente CVC, agrega dados dependentes que representam um mesmo conceito e aqueles que não podem ser explicados são armazenados em um *buffer* temporário, já o *Observer* monitora a evolução do FCD analisando a coesão e separabilidade dos dados agregados no *buffer* temporário e de acordo com os resultados gerados pelo *observer* a estrutura *BLM* é atualizada. Esse método adota métodos de active learnign para rotular dados incertos ao longo do FCD, e detecta apenas um novo conceito por vez.

A abordagem proposta por Haque, Khan e Baron (2016) mantém um *ensemble* de classificadores baseados em cluster, similar ao proposto por Al-Khateeb et al.

(2012a).Nneste método cada classificador é treinado em um *chunk* de dados parcialmente rotulado do FCD. Esta abordagem assume o surgimento de uma única classe novidade de cada vez. Dessa forma, qualquer novo exemplo que esteja fora do limite de decisão do *ensemble* de classificadores é identificado como um *outlier*. O método interpreta a presença de um número suficientemente grande de *outliers* com forte coesão entre eles como o surgimento de uma nova classe.

Outro método que lida com a tarefa de DN em um contexto multi-classe é o algoritmo MINAS proposto por Faria et al. (2016b). Na sua fase *offline* esse algoritmo cria micro-grupos associados às classes conhecidas, podendo utilizar algoritmos tradicionais de agrupamento como *K-Means* ou algoritmos de agrupamentos para FCDs como o Clustream (AGGARWAL et al., 2003b). Já na fase *online* este método verifica se um novo exemplo esta dentro do raio de algum dos micro-grupos presentes no modelo. Se estiver, o exemplo é classificado com o rotulo do micro-grupo mais próximo e o atualiza. Caso contrário, o exemplo é guardado em um *buffer* temporário até que haja exemplos suficientes para que se possa agrupá-los e calcular a silhueta do agrupamento e a densidade para definir se o agrupamento é valido. Se o valor de silhueta for menor que zero ou sua densidade for baixa este agrupamento é descartado. Caso contrário, verifica-se a distância deste agrupamento para o micro-grupo mais próximo, se esta distância for menor que um limiar definido pelo usuário, ou definido automaticamente, este agrupamento é definido como uma extensão de um conceito já conhecido. Senão, investiga-se se este agrupamento pode ser um conceito recorrente, ou seja, um conceito que deixou de existir porém que retornou após um período de tempo. Se for verdade, o agrupamento é atualizado no modelo com o rótulo do conceito recorrente. Caso contrário, este agrupamento é definido como um padrão novidade, que é definido como um padrão identificado em exemplos previamente desconhecidos, e é atualizado no modelo.

3.2.2 Abordagens Fuzzy

No contexto de teoria de conjuntos *fuzzy*, os métodos desenvolvidos para a tarefa de DN apresentam-se em pouca quantidade na literatura. Entretanto, alguns métodos foram desenvolvidos para a tarefa de DN em domínios específicos.

O trabalho de Lemos, Caminhas e Gomide (2013a) sugere uma abordagem para detecção e diagnóstico adaptativo de falhas. A abordagem proposta detecta novos modos de operação de um processo, como alterações e falhas do ponto de operação, e incorpora informações sobre os modos de operação em um classificador evolutivo *fuzzy* usado para o diagnóstico. A abordagem depende de um procedimento de agrupamento incremental para gerar regras *fuzzy* que descrevem novos estados operacionais detectados. O classificador executa o diagnóstico de forma adaptável e, uma vez que cada novo modo de operação detectado é aprendido e incorporado no classificador, ele é capaz de identificar o mesmo

modo de operação na próxima vez que ocorrer.

Em Angelov, Ramezani e Zhou (2008) os autores apresentam métodos tanto para o problema da detecção de novidades quanto para o rastreamento de objetos em fluxos de vídeo. Essas abordagens se baseiam em técnicas recursivas, que utilizam funções kernel denominadas Cauchy para calcular a estimativa de densidade recursiva para a detecção de novidades visuais e sistema neuro-*fuzzy* Takagi-Sugeno (eTS) para o rastreamento do objeto detectado.

No trabalho de Angelov e Zhou (2008) os autores propõem uma abordagem para a classificação em FCDs que baseia-se em um sistema fuzzy de regras (FRB) evolutivo de tipo Takagi-Sugeno. O método proposto, chamado eClass *(evolving classifier)*, inclui diferentes arquiteturas e métodos de aprendizagem *online* e uma propriedade importante do eClass é que ele pode começar a aprender do zero. Dessa forma regras *fuzzy* não precisam ser pré-especificadas e também o método se ajusta ao surgimento de novas classes (conceitos novidades). Entretanto, este método necessita do rótulo verdadeiro dos exemplos para identificação de novos conceitos.

3.3 Considerações Finais

Esse capítulo apresentou uma discussão sobre a tarefa de DN em FCD e os principais trabalhos existentes na literatura. Ele também apresentou abordagens *fuzzy* para DN em FCD que ainda são pouco tratadas na literatura. Os próximos capítulos apresentam os resultados parciais obtidos em estudos preliminares e a proposta deste trabalho.

Parte II

Proposta e Experimentos Preliminares

4

Abordagens *Fuzzy* para Detecção de Novidade em Fluxo Contínuo de Dados

FCDs são conjuntos que diferem dos convencionais em aspectos como: os exemplos chegam de maneira contínua e constante; não há controle sobre a ordem na qual os exemplos chegam para serem processados; pode haver mudança na distribuição dos exemplos gerados.

Abordagens de aprendizado de máquina clássico não são capazes de tratar as particularidades do contexto de FCDs, em que um exemplo processado deve ser descartado ou arquivado, não podendo ser recuperado de forma simples. No aprendizado clássico, assume-se que o conjunto de exemplos está disponível, é finito e a distribuição dos exemplos é estática.

Desponta, então, o interesse na investigação de propostas para que aprendizado seja possível em domínios de FCDs, considerando suas peculiaridades. As abordagens são baseadas em metodologias já consagradas de aprendizado clássico e incluem mecanismos capazes de tratar os aspectos característicos de FCD.

Entre tarefas de Aprendizado em FCD, esta proposta destaca a DN multiclasse, especificamente as que exploram o *Framework Offline-Online* (FARIA et al., 2016a). Essas abordagens realizam o aprendizado em duas fases: na fase *offline* exemplos rotulados disponíveis *a priori* são utilizados para a indução do modelo inicial, na fase *online* o modelo de DN sofre atualizações de acordo com predições de exemplos não rotulados provenientes do FCD. Nesse *framework* a fase *online* é contínua e constante, enquanto a fase *offline* é executada somente uma vez. Métodos baseados no *Framework Offline-Online* produzem resultados de forma incremental, atualizando o modelo a cada novo exemplo do FCD. Ademais, apresenta uma abordagem realística considerando diferentes domínios de FCD.

Dentre as peculiaridades de FCDs, as características imprevisíveis desses domínios geram dificuldades no processo de aprendizagem, deste modo a busca por aprendizado flexível é oportuna. A integração de conceitos da teoria de conjuntos *fuzzy* é uma maneira

de tornar o Aprendizado em FCDs mais adaptável e tolerante à imprecisão. Recentemente surgiram algumas propostas com o objetivo de desenvolver modelos de DN flexíveis em Fluxo de Dados por meio de técnicas *fuzzy*.

As abordagens mais representativas nesse contexto, propostas por Lemos, Caminhas e Gomide (2013b), Angelov, Ramezani e Zhou (2008) e derivadas, realizam a tarefa de DN considerando Fluxo de Dados em domínios específicos, ou não consideram a DN multiclasse. Esses trabalhos sugerem a DN em processos maquinários, a fim de detectar falhas, e na detecção de novidades visuais em vídeos. Por outro lado, a proposta desse trabalho aborda a tarefa de DN multiclasse de modo genérico, sem limitações ou vantagens impostas por domínios específicos.

Seguindo linha diferenciada, o eClass (ANGELOV; ZHOU, 2008) realiza a classificação e DN por meio de regras *fuzzy*. Entretanto esse algoritmo considera a disponibilidade de rótulos verdadeiros para que possa se adaptar a mudanças e evoluções de conceito.

Nesse contexto, formula-se a hipótese de que uma abordagem *fuzzy* para DN multiclasse em FCD, baseado no *Framework Offline-Online*, e na teoria de conjuntos *fuzzy* com o objetivo de tornar a aprendizagem mais adaptável às imprecisões inerentes ao domínio de FCD, produz resultados comparáveis, ou melhores às técnicas já existentes.

Detecção de Novidade *Fuzzy* baseada no *Framework Offline-Online*

O *Framework Offline-Online* (FARIA et al., 2016a) divide a DN em duas fases. A fase *online* tenta reduzir o tamanho do problema, transformando, de maneira incremental, um conjunto de exemplos de tamanho indefinido em um modelo de decisão representativo do FCD, por exemplo, árvores de decisão ou micro-grupos. Entretanto é necessário a criação de um modelo de decisão inicial a partir de dados previamente rotulados (fase *offline*). A vantagem de abordagens baseadas nesse *framework* é a possibilidade do modelo de decisão ser atualizado a cada exemplo processado, acarretando a geração de conhecimento mais adaptável a mudanças no conjunto de exemplos.

No trabalho original de Faria et al. (2016b), a fase *online* é definida por um modelo de decisão baseado em micro-grupos, a manutenção dos mesmos e detecção de novidade por meio desses. Na fase *offline* podem ser empregados algoritmos como *k-means* (HARTIGAN; WONG, 1979) ou *Clustream* (AGGARWAL et al., 2003b), que agrupam exemplos de mesma classe, afim de definir um modelo de decisão baseado em micro-grupos que possa representar as classes conhecidas *a priori*.

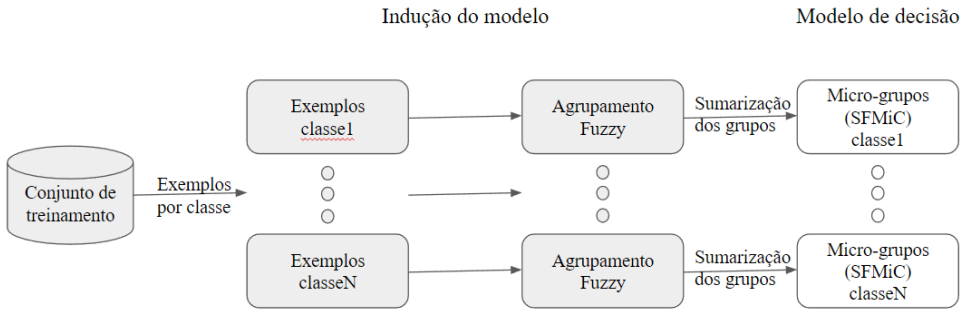
Ressonante com a proposta de Faria et al. (2016b), objetivo deste trabalho é desenvolver uma proposta de generalização do *Framework Offline-Online*, investigando técnicas baseadas na teoria de conjuntos *fuzzy* para as fases *online* e *offline*, que utilizem modelos

de decisão baseados em micro-grupos, e algoritmos de agrupamento na fase *offline*.

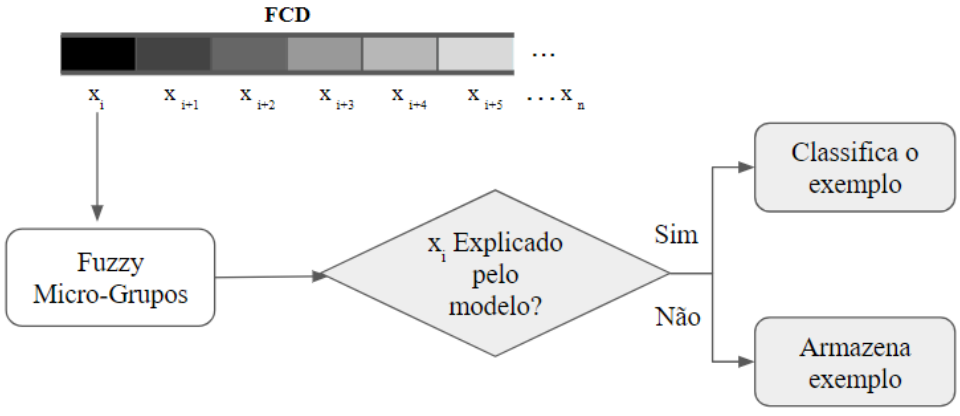
Deste modo, uma ideia inicial e simples foi a alteração do método aplicado na fase *offline* para uma abordagem de agrupamento *fuzzy*. Características relevantes para a técnica escolhida devem ser: produção rápida de resultados e habilidade de produzir agrupamentos flexíveis. Um método que contempla esses atributos é o *Fuzzy C-Means* (FCM) (BEZDEK; EHRLICH; FULL, 1984).

Entretanto, em domínios de FCDs não-estacionários podem existir constantes mudanças na distribuição dos dados, o que torna um modelo de decisão inicial, produzido na fase *offline*, desatualizado com o tempo, gerando dessa forma resultados incoerentes com o estado atual do FCD. Infere-se por meio disso, que o maior ganho quanto ao conhecimento adquirido pode estar, justamente, na fase *online* por meio da definição do modelo de decisão, sua manutenção e processo de DN, fase principal e contribuição maior deste trabalho.

Para a definição do *Framework Offline-Online Fuzzy* foram consideradas, de maneira individual, técnicas *fuzzy* para aplicação em ambas as fases *online* e *offline*, por meio de um modelo de decisão *fuzzy* baseado em micro-grupos (*online*) (Figura 4.1b), e fase de agrupamento (*offline*) que utiliza o método FCM (Figura 4.1a).



(a) Fase *Offline*



(b) Fase *Online*

Figura 4.1 – Visão geral da proposta de generalização *Fuzzy* para o *Framework Offline-Online*

Foram definidos quatro processos para a fase *online fuzzy*, que incluem a definição das estatísticas do micro-grupos *fuzzy*, classificação por meio destes, sua manutenção, e detecção de novidade. As Seções 4.1 e 4.2 detalham as propostas *Fuzzy Multiclass Novelty Detector for Data Streams* (SILVA et al., 2018) e *Possibilistic Fuzzy Multiclass Novelty Detector for Data Streams* para a generalização do *Framework Offline-Online*.

4.1 *Fuzzy Multiclass Novelty Detector for Data Streams*

FuzzND (Fuzzy multiclass Novelty Detector for data streams) (SILVA et al., 2018) é uma abordagem para a DN multiclasse em FCDs, definida como uma generalização *fuzzy* do método *MINAS* apresentado em (FARIA et al., 2016b). Portanto, *FuzzND* separa o processo de aprendizagem em duas fases: *offline* e *online*. Na primeira, um modelo de decisão é criado a partir de uma porção de exemplos rotulados a priori. Mais tarde, durante a fase *online*, novos exemplos não rotulados do FCD são classificados, de forma incremental, em uma das classes conhecidas do modelo ou como desconhecidos. Intermitentemente, os exemplos desconhecidos são agrupados para procurar grupos coesos que serão incorporados no modelo de decisão como padrões novidade.

Buscando por um processo de aprendizado mais flexível, essa proposta utiliza o algoritmo de agrupamento Fuzzy C-Means (FCM) (BEZDEK; EHRLICH; FULL, 1984) para criar o modelo de decisão inicial composto por um conjunto de micro-grupos *fuzzy* denominados SFMiC *(Supervised Fuzzy Micro-Cluster)*, definido em (SILVA et al., 2017). Na fase *online*, o método faz uso dessas estruturas para classificar novos exemplos entre as classes conhecidas ou como um exemplo discrepante, ou seja, não pertencente a nenhuma das classes conhecidas. Diferentemente de sua versão original proposta em (SILVA et al., 2017), a estrutura SFMiC foi adaptada para incluir estatísticas que permitem calcular a similaridade *fuzzy* entre dois micro-grupos (XIONG; CHAN; TAN, 2004), por meio da adição do componente estatístico SSD^e, explicado mais tarde.

Definição 4.1. ***Supervised Fuzzy Micro-Cluster*** **(SFMiC)** Um *Supervised Fuzzy Micro-Cluster* (SFMiC) com parâmetro de fuzzificação m, para um conjunto de exemplos $e_1, ..., e_n$ d-dimensionais, com valores de pertinência $\mu_1, ..., \mu_n$,*timestamps* $t_1, ..., t_n$ e centróide c é definido como o vetor $(M, \overline{CF1^e}, SSD^e, t, class\_id)$, onde M é a soma linear das pertinências dos exemplos no micro-grupo (Equação 2.1), $\overline{CF1^e}$ é a soma linear dos n exemplos x_j ponderados por suas pertinências (Equação 2.3), SSD^e é a soma quadrática das distâncias dos exemplos para o protótipo do micro-grupo elevadas a um fator de fuzzificação m, e ponderadas pelas pertinências de cada exemplo (Equação 4.1), t é o tempo de chegada do exemplo mais atual associado ao SFMiC e $class\_id$ é a classe associada ao micro-grupo.

$$\overline{SSD^e} = \sum_{j=1}^{n} \mu_j^m |e_j - c|^2 \tag{4.1}$$

Enquanto o algoritmo *MINAS* classifica exemplos do FCD baseado apenas na distância para os micro-grupos, na abordagem proposta os graus de pertinência são considerados, a fim de identificar o conjunto de micro-grupos que melhor represente o exemplo, e classificá-lo com o rótulo desses micro-grupos. Além disso, o algoritmo também permite a criação de uma superfície de decisão mais flexível, tornando o modelo mais tolerante a imprecisão.

Fase *Offline*

Quanto ao método proposto, Algoritmo 1 descreve a fase *offline*, que requer como entrada o FCD DS, o parâmetro de fuzzificação m do algoritmo FCM, o número de grupos por classe k_class e o conjunto de exemplos rotulados e utilizados para calcular os primeiros micro-grupos $init\_points$.

Algoritmo 1 FuzzND - Fase *Offline* Baseada em (FARIA et al., 2016b)

Requer $DS, m, k\_class, init\_points$
1: $model \leftarrow \emptyset$
2: **para cada** *class* $C_i \in init\_points$ **faça**
3: $\quad class\_clusters \leftarrow \text{FCM}(init\_points_{class=C_i}, m, k\_class)$
4: $\quad class\_SFMiC \leftarrow \text{SUMMARIZE}(class\_clusters)$
5: $\quad model \leftarrow model \cup class\_SFMiC$
6: **fim para**

Em Algoritmo 1, as linhas 3 e 4 sinalizam os métodos que foram revisados para obedecer conceitos de teoria de conjuntos *fuzzy*. No início, para cada classe existente, o conjunto de pontos correspondentes é dado como entrada para o algoritmo de agrupamento FCM (Passo 3). Os grupos de cada conjunto retornados pelo FCM são armazenados na variável $class\_cluster$ e mais tarde sumarizados em estruturas de micro-grupos *fuzzy* supervisionadas denominadas SFMiCs (Passo 4). O modelo de decisão é definido como o conjunto de SFMICs encontrados para cada classe conhecida (Passo 6).

Fase *Online*

Durante a fase *online* do algoritmo, a classificação é realizada para cada exemplo do FCD, e o processo de DN ocorre de maneira eventual. O Algoritmo 2 descreve a fase *online*, onde os parâmetros de entrada $init\_\theta$ e $init\_\theta_{class}$ são respectivamente os limiares iniciais de classificação e processamento dos exemplos. O parâmetro $\theta_{adapter}$ corresponde a um limiar de adaptação referente a etapa de classificação. Exemplos rotulados como desconhecidos são armazenados em uma memória de curto prazo ($short\_memory$). T é

a quantidade mínima de exemplos desconhecidos na $short\_memory$ para que o procedimento de detecção de novidade seja executado. O Parâmetro P é um limiar de tempo em relação ao esquecimento de SFMiCs mais antigos e ts é um limite de tempo correspondente à remoção de exemplos desconhecidos mais antigos da $short_memory$. Finalmente, parâmetro max_mic_{class} corresponde ao máximo de micro-grupos permitidos por classe.

Algoritmo 2 FuzzND - Fase *Online*

Requer $DS, init\_\theta, init\_\theta_{class}, \theta_{adapt}, m, T, P, ts, max\_mic_{class}$
1: $short\_memory \leftarrow \emptyset$
2: $all\_comp_{max} \leftarrow init\_\theta$
3: $all\_memb_{max} \leftarrow init\_\theta_{class}$
4: **enquanto** !ISEMPTY(DS) **faça**
5: $x \leftarrow$ NEXT(DS)
6: $all\_membership \leftarrow$ MEMBERSHIP($x, model, m$)
7: $all\_comp \leftarrow \emptyset$
8: **para cada** *class* $C_i \in model$ **faça**
9: $class\_compatibility \leftarrow$ SUM($all\_membership_{class=C_i}$)
10: $all\_comp \leftarrow all\_comp \cup class\_compatibility$
11: **fim para**
12: $(max\_class, max\_comp) \leftarrow$ MAX(all_comp)
13: **se** $max\_comp \geq$ MEAN(all_comp_{max}) $- \theta_{adapt}$ **então**
14: $all\_comp_{max} \leftarrow all\_comp_{max} \cup max\_comp$
15: $x.class \leftarrow max\_class$
16: $membership \leftarrow$ MEMBERSHIP($x, model_{class=max\_class}, m$)
17: $max\_memb_{class} \leftarrow$ MAX($membership$)
18: **se** $max\_memb_{class} \geq$ MEAN(all_memb_{max}) **então**
19: $all\_memb_{max} \leftarrow all\_memb_{max} \cup max\_memb$
20: UPDATE($model_{class=max\_class}, x$)
21: **senão**
22: CREATE_SFMIC($model, x, max\_class$)
23: **se** $|model_{class=max\_class}| > max\_mic_{class}$ **então**
24: $model \leftarrow$ REMOVE_OLDEST_SFMIC($model_{class=max\_class}$)
25: **fim se**
26: **fim se**
27: **senão**
28: $x.class \leftarrow unknown$
29: $short\_memory \leftarrow short\_memory \cup x$
30: **se** $|short\_memory| \geq T$ **então**
31: $model \leftarrow$ NOVELTY_DETECTION($short\_memory, model$)
32: **fim se**
33: **fim se**
34: $current\_time \leftarrow x.time$
35: $model \leftarrow$ REMOVE_OLD_SFMICS($model, P$)
36: $short\_memory \leftarrow$ REMOVE_OLD_UNKNOWN($short\_memory, ts$)
37: **fim enquanto**

Em Algoritmo 2, os novos desenvolvimentos introduzem conceitos *fuzzy* em relação ao passo de classificação e manutenção dos SFMiCs.

Para cada exemplo x que chega do fluxo, o algoritmo calcula a pertinência de x para todos os SFMiCs presentes no modelo (Passo 6). Depois, as pertinências relativas a SFMiCs de mesma classe são somadas, resultando em um valor de correspondência de x para cada classe, a qual denominamos compatibilidade de classe (Passos 8-11).

Esses valores serão usados para decidir se x receberá o rótulo de uma classe existente ou será classificado como desconhecido. Este processo é feito verificando se a maior compatibilidade de classe é superior ou igual a média das compatibilidades de classe máxima de todos os exemplos anteriores rotulados como pertencendo a uma classe conhecida até o momento (armazenadas em all_comp_{max}), menos um limite de adaptação $\theta_{adapter}$ (Passo 13). O parâmetro $theta_{adapter}$ garante que os valores máximos de compatibilidade de classe ligeiramente abaixo da média também podem passar nessa comparação. Quando o primeiro exemplo estiver sendo processado, seu maior valor de compatibilidade de classe será comparado com o valor do parâmetro inicial $init\_\theta$.

Se x for rotulado como pertencente a uma classe existente C_i, ou seja, sua compatibilidade de classe máxima é maior ou igual a média das compatibilidades máximas de exemplos anteriores, esse valor é adicionada a all_comp_{max} para atualizar a média das compatibilidades de classe máxima anteriores (Passos 14-15). Depois disso, o algoritmo verifica se x é uma extensão de C_i ao verificar a pertinência máxima entre x e os SFMiCs pertencentes à classe C_i (Passos 16-17). Uma extensão de classe é composta por exemplos pertencentes a uma classe conhecida, no entanto, fora do limite de decisão da classe definido pelo modelo (FARIA et al., 2016b). Esse exemplos podem indicar uma mudança de conceito.

Se a pertinência máxima de x para os SFMiCs da classe C_i for maior que a média dos graus de pertinência máxima de todos os exemplos anteriores atualizados em SFMiCs de classes conhecidas até o momento, armazenados em all_memb_{max}, x não é uma extensão de C_i (Passo 18). Nesse caso, x é associado aos SFMiCs da classe C_i e sua pertinência máxima ao SFMiC de C_i é adicionada a all_memb_{max} para atualizar a média das máximas pertinências de exemplos anteriores (Passos 19-20). Caso contrário, x corresponde a uma extensão da classe C_i e um novo SFMiC é criado para esta classe com x como o protótipo (Passo 22). Posteriormente é verificado se a quantidade total de micro-grupos da classe C_i é maior do que a permitida max_mic_{class}, se afirmativo, então o micro-grupo dessa classe com menor valor de t é removido (Passos 23-24).

Enquanto o algoritmo *MINAS* verifica a extensão de classe apenas na etapa de detecção de novidade, em *FuzzND* o uso de pertinências permite a descoberta dessas extensões de classe durante a etapa de classificação, adaptando assim o modelo de forma incremental.

Quando um exemplo é classificado como desconhecido, o *FuzzND* funciona como o algoritmo *MINAS*, adicionando o exemplo a uma memória de curto prazo $short\_memory$ e verificando se o passo de detecção de novidade será executado ou não (Passos 27- 32).

Após essas etapas, *FuzzND* remove SFMiCs que não foram atualizados no último intervalo de tempo P e deleta exemplos desconhecidos anteriores a $ts - current\_time$ unidades de tempo da memória de curto prazo (Passos 35-36).

Detecção de Novidade

Sempre que um novo exemplo for rotulado com o perfil desconhecido, o *FuzzND* verifica se há um número mínimo de exemplos na memória de curto prazo ($short\_memory$). Em caso afirmativo, o *FuzzND* executa um procedimento de detecção de novidades de forma não-supervisionada que requer como entrada a estrutura $short\_memory$ contendo os exemplos desconhecidos, o parâmetro de fuzzificação m e o número de grupos k_short.

Algoritmo 3 FuzzND - Passo Detecção de Novidade

Requer $short\_memory, model, m, k\_short$
1: $temp\_clusters \leftarrow$ FCM($short\_memory, m, k\_short$)
2: **para cada** $cluster\ temp\_c_i \in temp\_cluster$ **faça**
3: **se** VALIDATE($temp\_c_i$) **então**
4: $SFMiC_{NP} \leftarrow$ SUMARIZING($temp\_c_i$)
5: **para cada** $SFMiC_{class=NP\#}\ NP_i \in model$ **faça**
6: $FR \leftarrow$ SIMILARITY($NP_i, temp\_c_i$)
7: $all\_FR \leftarrow all\_FR \cup FR$
8: **fim para**
9: $(label_{max}, max\_fr) \leftarrow$ MAX(all_FR)
10: **se** $max\_fr \geq \phi$ **então**
11: $SFMiC_{NP}.label \leftarrow label_{max}$
12: $model \leftarrow SFMiC_{NP}$
13: **senão**
14: $label_{new} \leftarrow$ NEW_NP_LABEL($model$)
15: $SFMiC_{NP}.label \leftarrow label_{new}$
16: $model \leftarrow SFMiC_{NP}$
17: **fim se**
18: **fim se**
19: **fim para**

Em Algoritmo 3 o primeiro passo é a aplicação do algoritmo de agrupamento FCM (BEZDEK; EHRLICH; FULL, 1984) nos exemplos presentes na memória de curto prazo ($short\_memory$), produzindo k_short grupos (Passo 1). O *FuzzND* avalia cada um dos grupos para decidir se são válidos, verificando se o coeficiente de silhueta *fuzzy* (CAMPELLO; HRUSCHKA, 2006) é maior que 0 e se o grupo possui um número representativo de exemplos (Passo 3).

Sempre que um grupo é validado, o próximo passo é sumarizá-lo e verificar se será atribuído um novo rótulo gerado pelo método (Padrão Novidade), ou o rótulo do Padrão Novidade (PN) mais próximo presente no modelo (Extensão de PN). Os novos rótulos são definidos como PN#id, onde #id é um número único que representa o PN. Para atribuir um rótulo a um novo grupo válido, o *FuzzND* determina a similaridade *fuzzy* FR (Equação 4.3) entre o grupo válido e cada um dos outros PN_i existentes no modelo (Passos 5-8). Se a similaridade máxima entre o novo grupo e os outros PNs for maior do que um limiar ϕ escolhido pelo usuário, o novo grupo compreende uma extensão de um PN existente, portanto, recebe o mesmo rótulo do PN ao qual ele possui a máxima similaridade (Passos 11 -12). Caso contrário, um novo rótulo é criado e associado ao grupo, que por sua vez constitui-se um novo PN (Etapas 14-16). Se um grupo não for

validado, ele será descartado e seus exemplos permanecerão na memória de curto prazo ($short\_memory$) até o modelo decidir removê-los.

$$dispersão_i = \sqrt{\frac{SSD^e}{N}} \tag{4.2}$$

$$FR_{i,j} = \frac{dispersão_i + dispersão_j}{|centróide_i - centróide_j|} \tag{4.3}$$

Diferente do algoritmo *MINAS*, o *FuzzND* busca gerar uma quantidade de novos rótulos mais próximos do número de novas classes total, executando a etapa de avaliação de similaridade para cada novo NP. Este processo é feito para facilitar a etapa de classificação ao tornar um rótulo PN mais representativo. Além disso, a geração de mínimo possível de rótulos PN, evita que os resultados finais se tornem muito complexos para entender.

4.2 *Possibilistic Fuzzy Multiclass Novelty Detector for Data Streams*

O FCM (BEZDEK; EHRLICH; FULL, 1984) é um dos algoritmos de agrupamento *fuzzy* mais populares, e base para a desenvolvimento do método *FuzzND*, pois utiliza-o durante a fase *offline*, além de herdar o mesmo cálculo de pertinência durante a fase *online*. O algoritmo FCM associa pertinências para um determinado exemplo x_k, que são inversamente relacionadas a distância relativa de x_k para os protótipos existentes no modelo. Supondo que o número de protótipos seja igual a 2, se x_k encontra-se equidistante de ambos os protótipos, a pertinência de x_k em cada grupo será igual a 0.5, independente do valor da distância de x_k para os dois protótipos. Essa condição pode ocorrer para quaisquer outros exemplos do conjunto de dados, devido a restrição do método que impõe que a somatória das pertinências de x_k para todos os protótipos no modelo seja igual a 1. Além do mais, quanto maior a quantidade de protótipos mais fragmentada se tornam as pertinências. Por exemplo, sob as mesmas circunstâncias entretanto com 4 protótipos, um exemplo x_k apresentará pertinências iguais de 0.25 a cada um dos 4 grupos. Cria-se então um problema onde *outliers* longe mas equidistantes, ou pelos menos com distâncias aproximadas, das estruturas centrais dos grupos presentes no modelo, serão dadas pertinências similares para cada grupo. Na verdade parece mais natural que para esse exemplos sejam associadas pertinências muito baixas, ou até nenhuma (0).

Dirigindo-se a este problema, Pal et al. (2005) propõem um algoritmo denominado *Possibilistic C-Means* (PCM) que relaxa a restrição do somatório das pertinências impostas pelo método FCM, ou seja, o somatório dos valores de pertinência de x_k aos grupos não necessariamente é igual a 1. Entretanto, os autores sugerem que esses valores

calculados pelo método sejam interpretados como a tipicidade de x_k relativa aos grupos no modelo, ao contrário de sua pertinência. Entretanto, este método é muito sensível a sua inicialização e escolhas de parâmetros (PAL et al., 2005).

Pal et al. (2005), propõem mais tarde uma extensão do PCM, denominada *Possibilistic Fuzzy C-Means* (PFCM), que utiliza os cálculos de pertinências do FCM e tipicidades do PCM, afim de criar modelos menos sensíveis a inicialização e escolhas de parâmetros, e que possam descrever melhor *outliers*. Tendo como base essa proposta, foi desenvolvida uma extensão do método *FuzzND* com alterações na fase *online*, especificamente na definição dos componentes dos micro-grupos e classificação de novos exemplos, para que o método possa melhor descrever *outliers* por meio do conceito de tipicidade aliado a pertinências. De maneira a comportar estatísticas para o cálculo da tipicidade, foi proposta a estrutura *Supervised Possibilistic Fuzzy Micro-Cluster* (SPFMiC).

Definição 4.2. ***Supervised Possibilistic Fuzzy Micro-Cluster*** **(SPFMiC)**: Um *Supervised Possibilistic Fuzzy Micro-Cluster* (SPFMiC) com parâmetros de fuzzificação m e tipicidade n, para um conjunto de exemplos $e_1, ..., e_n$ d-dimensionais, com valores de pertinência $\mu_1, ..., \mu_n$, valores de tipicidade $\gamma_1, ..., \gamma_n$, *timestamps* $t_1, ..., t_n$ e centróide c é definido como o vetor $(M^e, T^e, \overline{CF1^e_\mu}, \overline{CF1^e_t}, SSD^e, N, t, class\_id)$, onde M^e é a soma linear das pertinências dos exemplos no micro-grupo elevadas a m (Equação 4.4), T^e é a soma linear das tipicidades dos exemplos no micro-grupo elevadas a n (Equação 4.5), $\overline{CF1^e_\mu}$ é a soma linear dos exemplos pertencentes ao micro-grupo ponderados por suas pertinências (Equação 2.3), $\overline{CF1^e_\gamma}$ é a soma linear dos exemplos pertencentes ao micro-grupo ponderados por suas tipicidades (Equação 4.6), SSD^e é a soma das distâncias dos exemplos para o protótipo do micro-grupo, elevadas a m e ponderadas pelas pertinências de cada exemplo (Equação 4.1), N é a quantidade de exemplos pertencentes ao micro-grupo, t é o tempo de chegada do exemplo mais atual associado ao SFMiC e $class\ _id$ é a classe associada ao micro-grupo.

$$M^e = \sum_{j=1}^{N} \mu_j^m \tag{4.4}$$

$$T^e = \sum_{j=1}^{N} \gamma j^n \tag{4.5}$$

$$\overline{CF1^e_\gamma} = \sum_{j=1}^{N} \gamma_j e_j \tag{4.6}$$

Fase *Online*

Durante a fase *online* do algoritmo, a classificação é realizada para cada exemplo do FCD, e o processo de DN ocorre de maneira eventual como no método *FuzzND* Al-

goritmo 3. Algoritmo 4 descreve a fase *online*, onde o parâmetros de entrada $init\_\theta$ é um limiar inicial de classificação e processamento dos exemplos. Os parâmetros $\theta_{adapter}$ e θ_{class} correspondem a limiares de adaptação referente a etapa de classificação. Exemplos rotulados como desconhecidos são armazenados em uma memória de curto prazo ($short\_memory$). T é a quantidade mínima de exemplos desconhecidos na $short\_memory$ para o procedimento de detecção de novidade ser executado. Parâmetro P é um limiar de tempo em relação ao esquecimento de SFMiCs mais antigos e ts é um limite de tempo correspondente à remoção de exemplos desconhecidos mais antigos da $short\_memory$. max_mic_{class} corresponde ao máximo de micro-grupos por classe. Os parâmetros α, β, K e n são necessários para o cálculo de tipicidade Equação 4.9, além de também serem utilizados para a definição do centróide dos micro-grupos Equação 4.7. Finalmente, dec_new é um fator de decaimento da componente SSD^e utilizado na criação de novos micro-grupos que representam extensões de classes.

$$centroide_i = \frac{\alpha * \overline{CF1_e^x} + \beta * \overline{CF1_t^x}}{\alpha * M^e + \beta * T^n} \tag{4.7}$$

$$\gamma_i = K * \frac{SSD^e}{M^e} \tag{4.8}$$

$$tipicidade_i = \frac{1}{1 + (\frac{\beta}{\gamma_i} * |x_k - centróide_i|^2)^{1/(n-1)}} \tag{4.9}$$

Em Algoritmo 4, para cada exemplo x que chega do fluxo, o algoritmo calcula a pertinência e tipicidade de x para todos os SPFMiCs presentes no modelo (Passos 5-6). Os valores de tipicidade serão usados para decidir se x receberá o rótulo de uma classe existente ou será classificado como desconhecido. Este processo é feito verificando se a maior tipicidade é superior ou igual a média das tipicidades máximas de todos os exemplos anteriores rotulados como pertencendo a uma classe conhecida até o momento (armazenadas em all_tip_{max}), menos um limiar de adaptação θ_{adapt} cuja função é assegurar que exemplos ligeiramente abaixo da média possam ser classificados com os rótulos conhecidos (Passo 8). Quando o primeiro exemplo estiver sendo processado, seu maior valor de tipicidade será comparado com o valor do parâmetro inicial $init\_\theta$.

Se x for rotulado como pertencente a uma classe existente C_i, ou seja, sua tipicidade máxima é maior ou igual a média das tipicidades máximas de exemplos anteriores menos um limiar de adaptação θ_{adapt}, sua tipicidade máxima é adicionada a all_tip_{max} para atualizar a média das tipicidades máxima anteriores (Passos 9), e x é associado aos SPFMiCs da classe C_i (Passos 11). Caso contrário, se x for maior ou igual a média de tipicidades máximas anteriores menos um limiar de adaptação θ_{class} (Passo 12), x corresponde a uma potencial extensão da classe C_i e um novo SPFMiC é criado para esta

Algoritmo 4 PFuzzND - Fase *Online*

Requer $DS, init\_\theta, \theta_{adapt}, \theta_{class}, m, T, P, ts, max\_mic_{class}, \alpha, \beta, K, n, dec_new$
1: $short\_memory \leftarrow \emptyset$
2: $all\_tip_{max} \leftarrow init\_\theta$
3: **enquanto** !ISEMPTY(DS) **faça**
4: $x \leftarrow$ NEXT(DS)
5: $all\_membership \leftarrow$ MEMBERSHIP($x, model, m$)
6: $all\_tipicallities \leftarrow$ TIPICALLITY($x, model, n, k, \beta$)
7: $(max\_class, max\_tip) \leftarrow$ MAX($all\_tipicallities$)
8: **se** $max\_tip \geq$ MEAN(all_tip_{max}) $- \theta_{adapt}$ **então**
9: $all\_tip_{max} \leftarrow all\_tip_{max} \cup max\_tip$
10: $x.class \leftarrow max\_class$
11: UPDATE($model_{class=max\_class}, x, all\_membership, all\_tipicallities$)
12: **senão se** $max\_tip \geq$ MEAN(all_tip_{max}) $- \theta_{class}$ **então**
13: CREATE_SFMIC($model, x, max\_class, dec_new$)
14: **se** $|model_{class=max\_class}| > max\_mic_{class}$ **então**
15: $model \leftarrow$ REMOVE_OLDEST_SFMIC($model_{class=max\_class}$)
16: **fim se**
17: **senão**
18: $x.class \leftarrow unknown$
19: $short\_memory \leftarrow short\_memory \cup x$
20: **se** $|short\_memory| \geq T$ **então**
21: $model \leftarrow$ NOVELTY_DETECTION($short\_memory, model$)
22: **fim se**
23: **fim se**
24: $current\_time \leftarrow x.time$
25: $model \leftarrow$ REMOVE_OLD_SFMIC($model, P$)
26: $short\_memory \leftarrow$ REMOVE_OLD_UNKNOWN($short\_memory, ts$)
27: **fim enquanto**

classe com x como o protótipo (Passo 13). Entretanto, de forma a manter as propriedades referentes ao cálculo de tipicidade e permitir que exemplos próximos do novo micro-grupo sejam associados a ele, as estatísticas SSD^e e M^e do mesmo são inicializadas com os valores referentes do micro-grupo de mesma classe mais próximo multiplicado por um valor de decaimento *dec_new*. Posteriormente é verificado se a quantidade total de micro-grupos da classe C_i é maior do que a permitida max_mic_{class}, se afirmativo, então o micro-grupo dessa classe com menor valor de t é removido (Passos 14-16).

Diferentemente do método *FuzzND* que utiliza compatibilidade de classes por meio de pertinência *fuzzy* para classificar novos exemplos a fim identificar extensões de classes e *outliers*. Em *PFuzzND* o uso de tipicidades permite uma melhor descrição destes tipos de exemplos durante a etapa de classificação, além de tornar o método mais simples. De modo geral, os valores de tipicidade não são fragmentados com o aumento de micro-grupos no modelo. Além disso, os valores gerados, para um determinado novo exemplo, são baseados na distância média dos exemplos associados aos micro-grupos e suas pertinências, esta característica pode melhor definir a área de ação de um micro-grupo, quando comparado a pertinências, que são baseadas apenas nas distâncias do novo exemplo, para os centróide dos micro-grupos existentes no modelo.

Quando um exemplo é classificado como desconhecido, o *PFuzzND* funciona como

os algoritmos *MINAS* e *FuzzND*, adicionando o exemplo a uma memória de curto prazo *short_memory* e decidindo se o passo de detecção de novidade será executado ou não (Passos 17- 23).

Após essas etapas, *PFuzzND* remove SFMiCs que não foram atualizados no último intervalo de tempo P e deleta exemplos desconhecidos anteriores a $ts - current\_time$ unidades de tempo da memória de curto prazo (Passos 25-26).

4.3 Considerações Finais

Este capítulo apresentou uma proposta de generalização *fuzzy* para a DN em Fluxo de Dados, baseada no *Framework Offline-Online*. A fase *offline* aplica o algoritmo FCM aos exemplos rotulados disponíveis, enquanto que a fase *online* utiliza micro-grupos com conceitos *fuzzy* para a abstração, classificação de novos exemplos e DN.

A principal contribuição deste trabalho são as estratégias originais *Fuzzy Novelty Detector for data streams* (FuzzND) e sua extensão *Possibiliztic Fuzzy Novelty Detector for data streams* (PFuzzND). Ambas as propostas utilizam abstração *fuzzy* e realizam a manutenção de micro-grupos, classificação de novos exemplos e DN a partir de conceitos *fuzzy*.

Julga-se que a utilização das estratégias *fuzzy* implica na aquisição de conhecimento mais adaptado e tolerante a imprecisões dentro dos domínios de FCDs. O Capítulo 5 descreve a metodologia para validação da hipótese, detalhando ferramentas utilizadas, algoritmos comparados, conjuntos de exemplos e métricas de avaliação aplicadas.

5

Metodologia para Validação

A avaliação das abordagens apresentadas no Capítulo 4 foi efetivada de acordo com a execução de uma série de experimentos. Esses experimentos foram definidos considerando ferramentas de software disponíveis, conjuntos de exemplos sintéticos e reais utilizados na literatura, para tornar possível a comparação entre as novas propostas *FuzzND*, *PFuzzND* e a abordagem tida como base *MINAS*.

Este capítulo descreve os conjuntos de exemplos, sintéticos e reais, utilizados para os experimentos, os algoritmos empregados, a configuração de parâmetros para cada experimento e as métricas aplicadas para validação da proposta. Os resultados dos experimentos são apresentados e comentados no Capítulo 6.

5.1 Ferramentas de Software

O aumento de interesse na pesquisa sobre AM em FD proporcionou o investimento em ferramentas de software para a aplicação das diversas técnicas e abordagens propostas, muitas disponíveis gratuitamente, como:

- *Massive On-line Analysis* (MOA) (BIFET et al., 2010), desenvolvido na Universidade de Waikato, é um *framework* de código aberto que disponibiliza implementação, em linguagem Java, de uma série de algoritmos e métricas para classificação e para agrupamento em FCDs. A ferramenta também conta com recursos para visualização dos processos de aprendizado.
- *Very Fast Machine Learning* (VFML) (HULTEN; DOMINGOS, 2003), um pacote de implementações para mineração de FCDs de alta velocidade, que utiliza as linguagens C e Python.

Embora essas ferramentas ofereçam um ambiente favorável à execução de experimentos que apliquem técnicas já implementadas pelos desenvolvedores desses softwares, a extensão e utilização de partes dessas implementações não é trivial.

O Projeto R (R Core Team, 2016) é uma plataforma que engloba a linguagem R e um ambiente no qual é possível a utilização de uma variedade de técnicas para manipulação, análise e visualização de dados, além de ser facilmente extensível (por meio de pacotes), multiplataforma e gratuito.

A linguagem R, popular tanto na indústria quanto na academia, vem sendo amplamente utilizada em análise e ciência dos dados (PIATETSKY, 2016). Essa linguagem foi adotada para a implementação, validação e avaliação da proposta de agrupamento baseado em FMiC, devido à agilidade de implementação e à facilidade para geração de conjuntos de dados, cálculo de métricas de avaliação e geração de gráficos.

Os principais pacotes R específicos utilizados para a efetivação das propostas, execução de experimentos e geração de resultados foram:

`stream` é uma alternativa à MOA que provê um *framework* para representação e processamento de FDs a fim de facilitar o desenvolvimento, teste e comparação entre algoritmos de aprendizado em FD na linguagem R, além de permitir a integração de técnicas modeladas em linguagens para as quais haja uma interface R, como C/C++, Java, Python (HAHSLER; BOLANOS; FORREST, 2016).

`streamMOA` é uma extensão do pacote `stream` que faz a interface entre algoritmos de agrupamento já disponíveis na ferramenta MOA (HAHSLER; BOLANOS; FORREST, 2015).

`e1071` pacote que contém diversas funções para mineração de dados: *Support Vector, Machines*, classificador *Naive Bayes*, agrupamento *Fuzzy*, entre outros. (MEYER et al., 2015).

5.2 Conjuntos de Exemplos

A avaliação dos métodos *FuzzND*, *PFuzzND* e *MINAS* foi feita usando conjuntos de dados sintéticos denominados MOA (FARIA et al., 2016b), RBF (Computational Intelligence Group, 2017) , SynEDC (MASUD et al., 2011) e dados reais denominados KDD Cup 99, Covertype e FcTe (BACHE; LICHMAN, 2013) detalhados em Tabela 5.1. As Colunas #Instâncias, #Instâncias *(Offline)* e #Atributos indicam o número total de exemplos, o número total de exemplos disponíveis durante a fase *offline* e a quantidade de atributos para cada conjunto de dados, respectivamente. O número total de classes e o número de classes disponíveis durante a fase *offline* são indicados nas colunas #Classes e #Classes *(Offline)*, respectivamente. #Classes (Offline) é quantidade de classes conhecidas no início da fase *online*. Nesta seção, será apresentado uma breve descrição de cada conjunto de dados seguidas de uma aproximação em duas dimensões dos respectivos conjuntos de dados rotulados a priori, produzidas por meio da técnica de redução de dimensionalidade

t-Distributed Stochastic Neighbor Embedding (t-SNE) (MAATEN; HINTON, 2008), que apresenta-se adequada para visualização de conjunto de dados com alta dimensão.

Tabela 5.1 – Conjunto de dados usados nos experimentos

Identificador	#Instâncias	#Instâncias(*Offline*)	#Atributos	#Classes	#Classes(Offline)
MOA	100.000	10.000	4	4	2
RBF	48.588	836	2	5	3
SynEDC	400.000	30.000	54	20	6
KDD99	490.000	48.791	34	5	2
CoverType	581.000	47.045	54	7	3
FcTe	480,000	30,000	54	7	2

Conjuntos de Dados Artificiais

Com o propósito de identificar diferentes aspectos da DN utilizando *FuzzND* e *PFuzzND* foram elaborados experimentos que fazem uso de conjuntos de exemplos artificiais com distribuição estática (estacionários) e com distribuição mutável (não estacionários) de acordo com o tempo.

MOA - Base de dados criada por meio de uma função geradora disponível na ferramenta MOA. Essa base apresenta grupos em formato hiperesféricos que se movimentam em uma taxa de velocidade fixa (mudança de conceito). Além do mais este conjunto de dados apresenta aparecimento e desaparecimento de classes (evolução de conceito) a cada intervalo de 30.000 exemplos.

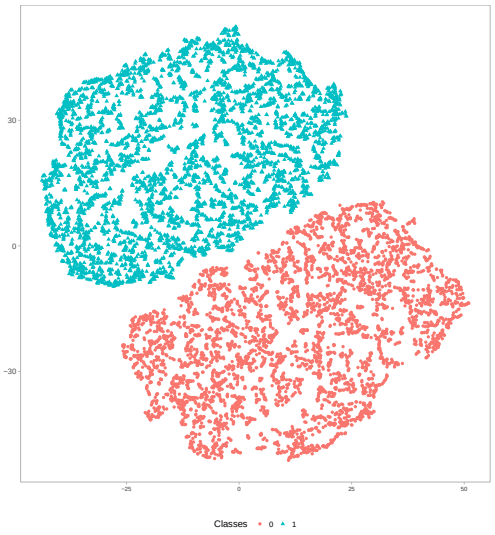


Figura 5.1 – Representação em duas dimensões dos exemplos rotulados a priori do conjunto de dados MOA

RBF - Base de dados estacionária, gerada a partir de uma função geradora de dados de base radial aleatória da ferramenta MOA, Esse conjunto de dados apresenta somente desaparecimento e aparecimento de novas classes (evolução de conceito) a cada intervalo de 10.000 exemplos.

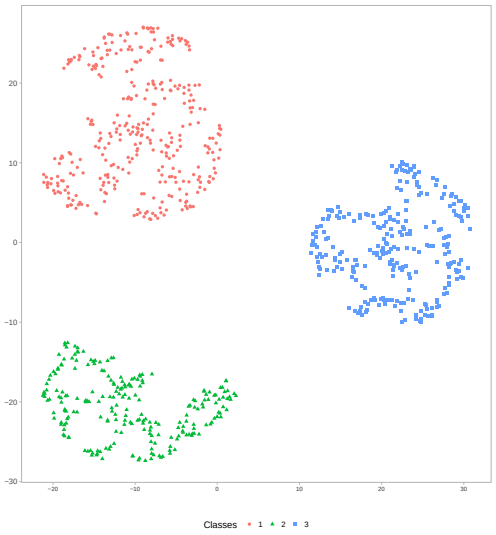


Figura 5.2 – Representação em duas dimensões dos exemplos rotulados a priori do conjunto de dados RBF

SynEDC - Base de dados não estacionária, gerada a partir de uma distribuição gaussiana com médias entre -5 e 5 e variância por classe de 0.5 a 5. Apresenta aparecimento de desaparecimento de classes (evolução de conceito), assim como mudanças na distribuição dos atributos (mudança de conceito) ao longo do tempo.

Conjuntos de Dados Reais

A aplicação de técnicas de aprendizado a conjuntos sintéticos é interessante para avaliação de características específicas, já que é possível controlar o comportamento dos conjuntos. No entanto, também é importante realizar avaliação quanto a conjuntos de exemplos de domínios reais.

Para os experimentos com exemplos de domínios reais foram escolhidos dois conjuntos disponíveis no repositório UCI (BACHE; LICHMAN, 2013). Tipicamente utilizados para avaliação de técnicas de classificação, são utilizados para avaliação de técnicas de AFD em geral, devido às dimensões (número de exemplos e atributos) elevadas.

Covertype e FcTe - associado à tarefa de classificação do tipo de cobertura florestal de uma área. O conjunto original possui 581.000 exemplos, divididos em 7 classes,

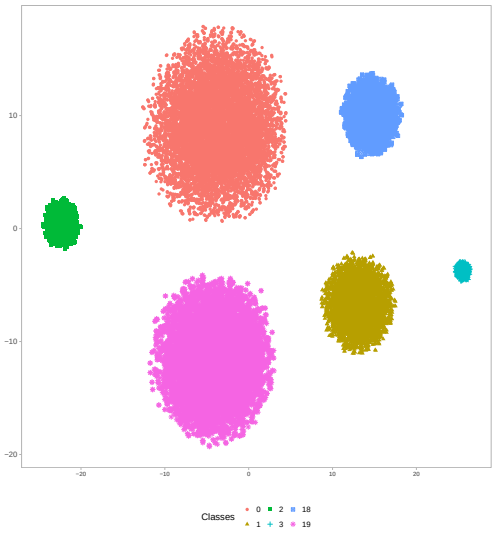
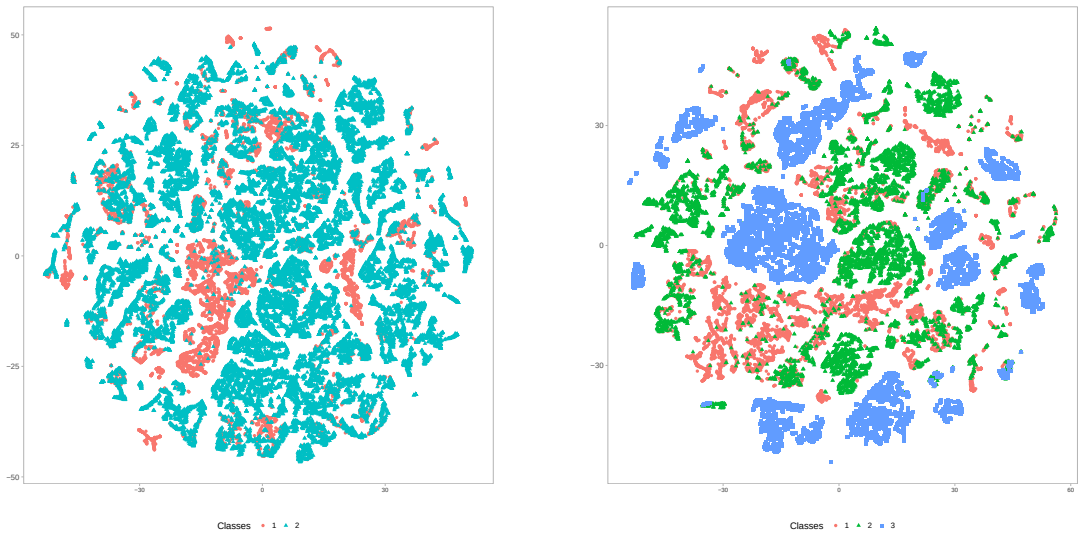


Figura 5.3 – Representação em duas dimensões dos exemplos rotulados a priori do conjunto de dados SynEDC

e representados por 54 atributos e um atributo classe (com 7 valores possíveis): 10 quantitativos e 44 binários. A principal diferença entre os conjuntos de dados Covertype e FcTe esta na ordem de desaparecimento e aparecimento das classes.



(a) CoverType (b) FcTe

Figura 5.4 – Representação em duas dimensões dos exemplos rotulados a priori dos conjuntos de dados CoverType e FcTe

KDD99 - utilizado na competição realizada em conjunto com a *KDD-99 The Fifth International conference on Knowledge Discovery and Data Mining*, onde a tarefa era construir um detector de intrusões em uma rede de computadores. O conjunto original possui 4.000.000 exemplos, descritos por 41 atributos e um atributo classe com 23 valores possíveis. Para os experimentos foi utilizado um subconjunto dos exemplos originais (versão conhecida como 10%, disponível no repositório UCI), com 490.000 exemplos. Após o pré-processamento, o conjunto teve sua dimensão reduzida para 34 atributos numéricos e um atributo classe (com 5 valores possíveis).

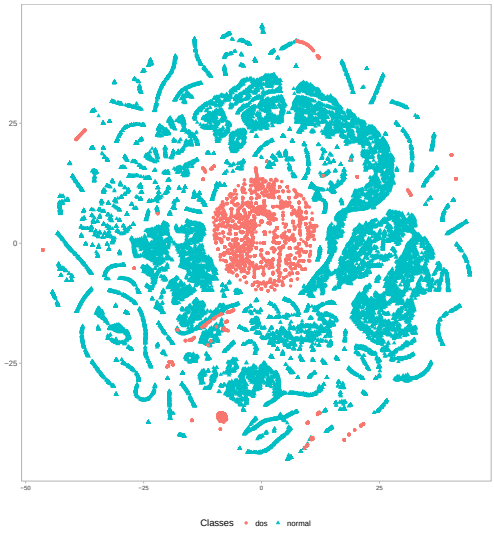


Figura 5.5 – Representação em duas dimensões dos exemplos rotulados a priori do conjunto de dados KDD99

5.3 Algoritmos Utilizados

As propostas para a tarefa de DN em FCDs *FuzzND* e *PFuzzND* são baseadas no método *MINAS*, que por conseguinte será utilizado para comparação e validação das propostas apresentadas nesse trabalho. Tabela 5.2, Tabela 5.3, Tabela 5.5 apresentam os parâmetros necessários para execução dos métodos, categorizados por fases. Além disso, apresentam uma breve descrição dos mesmos e seus respectivos valores utilizados nos experimentos. Os parâmetros indicados com * na coluna valor diferenciam-se para cada conjunto de dados. Importante destacar que os valores definidos nessa seção foram escolhidos de modo empírico, conforme diferentes análises dos resultados.

Ademais, os métodos avaliados possuem, ou podem possuir componentes aleatórios em sua inicialização e processo de DN , logo, cada experimento foi executado 10 vezes e as médias das métricas de desempenho foram calculadas e utilizadas com fins comparativos.

Tabela 5.2 – Valores de parâmetros escolhidos para o algoritmo *FuzzND*

FuzzND			
Fase		Parâmetro	Valor
Offline	m	parâmetro de fuzzificação referente ao método FCM	2
	k_class	número de grupos por classe	4
Online	m	parâmetro de fuzzificação referente ao cálculo de pertinência	2
	$init\_\theta$	limiar inicial de classificação	0,3
	$init\_\theta_{class}$	limiar inicial de identificação de extensão de classe	*
	θ_{adapt}	limiar de adaptação referente ao processo de classificação	0,1
	T	Número mínimo de exemplos desconhecidos para execução da DN	40
	P	limiar temporal de remoção de micro-grupos	500
	ts	limiar temporal de remoção de exemplos desconhecidos da memória de curto prazo	200
	$window\_size$	intervalo onde ocorre as remoções citadas acima	1000
DN	k_short	número de grupos	4
	min_weight	número mínimo de exemplos em um grupo válido	25
	m	parâmetro de fuzzificação referente ao método FCM	2
	ϕ	limiar de similaridade *fuzzy*	0,5

Os experimentos relacionados ao método *FuzzND* foram executados apenas nos conjuntos de dados sintéticos, pois devido a limitações do algoritmo não foram possíveis avaliações nos conjuntos de dados reais. Tais limitações estão associadas a características destes conjuntos, por exemplo, o algoritmo *FuzzND* assume que sempre irá existir pelo menos duas classes no modelo, para que seja calculada as compatibilidade de classes, caso exista somente uma classe, essa medida não pode ser calculada. Os conjuntos de dados KDD99 e CoverType apresentam a permanência de apenas uma classe por um longo período de tempo, o que inviabiliza a utilização do método. Além disso, a presença de classes sobrepostas interfere nos cálculos de pertinência do método, o que dificulta a classificação e detecção de outliers. Os conjuntos de dados CoverType e portanto FcTe apresentam esta característica.

Os valores do parâmetro θ_{class} foram escolhidos de acordo com os conjuntos de dados. Resultados de experimentos mostraram que valores de 0.3, 0.3 e 0.2 para os conjuntos MOA, RBF e SynEDC respectivamente, obtiveram os melhores resultados. Isso se deve ao fato de que os conjuntos de dados MOA e RBF possuem uma complexidade menor que o conjunto SynEDC, devido a alta dimensionalidade deste, o que permite a utilização de valores maiores para o parâmetro θ_{class}. Dessa maneira a escolha de um valor menor de θ_{class} permite que o método seja mais rígido na detecção de extensões de classe, e portanto, classifique mais exemplos como desconhecidos.

Referente ao método *PFuzzND*, análises de resultados determinados de maneira empírica demostraram uma sensibilidade do algoritmo em relação a diferentes conjuntos de dados. Desse modo, os parâmetros k_class, k_short, ts, min_weight e os limiares $init\_\theta$ e θ_{adapt} foram selecionados distintamente de acordo com a Tabela 5.4. A escolha desses valores está relacionada a alta dimensionalidade dos conjuntos de dados e complexidade dos mesmos. Dessa maneira, conjuntos de dados mais simples podem ser representados

Tabela 5.3 – Valores de parâmetros escolhidos para o algoritmo *PFuzzND*

PFuzzND			
Fase		Parâmetro	Valor
Offline	m	parâmetro de fuzzificação referente ao método FCM	2
	k_class	número de grupos por classe	*
Online	m	parâmetro de fuzzificação referente ao cálculo de pertinência	2
	n	parâmetro referente ao cálculo de tipicidade	2
	α	parâmetro referente ao cálculo de centróide	1
	β	parâmetro referente ao cálculo de tipicidade e centróide	1
	$init\_\theta$	limiar inicial de classificação	*
	θ_{adapt}	limiar de adaptação referente ao processo de classificação	*
	θ_{class}	limiar de adaptação referente ao processo de classificação	*
	T	Número mínimo de exemplos desconhecidos para execução da DN	40
	P	limiar temporal de remoção de micro-grupos	500
	ts	limiar temporal de remoção de exemplos desconhecidos da memória de curto prazo	*
	$window\_size$	intervalo onde ocorre as remoções citadas acima	1000
DN	k_short	número de grupos	*
	min_weight	número mínimo de exemplos em um grupo válido	*
	m	parâmetro de fuzzificação referente ao método FCM	2
	ϕ	limiar de similaridade *fuzzy*	0,5

Tabela 5.4 – Valores de parâmetros do método *PFuzzND* por conjunto de dados

	$init\_\theta$	θ_{class}	θ_{adapt}	k_class	k_short	ts	min_weight
MOA	0.95	0.00	0.90	4	4	200	25
RBF	0.95	0.00	0.90	4	4	200	25
SynEDC	0.90	0.00	0.58	8	8	200	25
KDD99	0.99	0.06	0.90	8	8	1000	10
CoverType	0.99	0.00	0.90	8	8	1000	10
FcTe	0.99	0.00	0.90	8	8	1000	10

com poucos micro-grupos, além de possuir um baixo intervalo para a definição de *outliers* pelo fato de não apresentarem uma grande quantidade de dados imprecisos (MOA e RBF). Por outro lado, em conjuntos de dados mais complexos demanda-se um número maior de micro-grupos, e por conter uma maior quantidade de imprecisões necessitam de um maior intervalo para a definição de *outliers*, como pode ser visto nos valores de parâmetros do conjunto de dados SynEDC. Entretanto, em conjuntos de dados reais, devido a sobreposição de classes e ruídos que podem ocorrer, valores baixos para o limiar de adaptação θ_{adapt}, ocasionam em um alto número de exemplos classificados como desconhecidos. Dessa maneira, por meio de análise de experimentos empíricos buscando números relativamente baixos de exemplos classificados como desconhecidos, juntamente com um bom desempenho do método em termos de classificação, foram escolhidos os valores para cada limiar de adaptação.

Destaca-se que os valores de parâmetros definidos para o método *MINAS*, estão de acordo com os experimentos em (FARIA et al., 2016b), exceto o experimento do conjunto de dados RBF que apesar de não estar presente nos supramencionados,

optou-se por atribuir os mesmos valores de parâmetros. Ademais, o número mínimo de exemplos em um micro-grupo válido $numMinExCluster$ é definido a partir da razão $tamanho(short\_memory)/k\_short$. A estratégia para o cálculo do parâmetro $threshold$, TV1, é definida como uma medida de coesão, calculada por meio do desvio padrão das distâncias euclidianas entre os exemplos do micro-grupo mais próximo e seu centróide, multiplicado por um fator f. Entretanto, é válido mencionar que os experimentos aqui citados tem como finalidade a validação dos métodos propostos. Portanto, há espaço para diferentes combinações de parâmetros do algoritmo *MINAS* afim de comparar de forma mais detalhada o desempenho de cada um dos métodos.

Tabela 5.5 – Valores de parâmetros escolhidos para o algoritmo *MINAS*

MINAS			
Fase		Parâmetro	Valor
Offline	$algOff$	algoritmo de agrupamento	*Clustream*
	k_class	número de grupos por classe	100
Online	$algOn$	algoritmo de agrupamento utilizado nessa fase	*Clustream*
	$threshold$	limiar referente a identificação de padrões de novidade de extensões de classe	1.1
	$thresholdStrategy$	estratégia utilizada para o cáculo do $threshold$	TV1
	T	Número mínimo de exemplos desconhecidos para execução da DN	2000
	P	limiar temporal de remoção de micro-grupos	4000
	ts	limiar temporal de remoção de exemplos desconhecidos da memória de curto prazo	4000
	$window\_size$	intervalo onde ocorre as remoções citadas acima	4000
DN	k_short	número de grupos	100
	$numMinExCluster$	número mínimo de exemplos em um grupo válido	*

5.4 Métricas de Avaliação

A avaliação das técnicas utilizadas nos experimentos foi baseada na matriz de confusão retangular incremental proposta por Faria et al. (2015) por ter sido a metodologia de avaliação utilizada na proposta do método MINAS (FARIA et al., 2016b). De modo geral, esta matriz adapta-se incrementalmente a medida que novos exemplos são classificados e novos padrões novidades são descobertos ou removidos do modelo, bem como quando classes conhecidas são removidas. Portanto, apresenta-se como uma metodologia adequada para a avaliação e validação dos métodos propostos.

Segundo Faria et al. (2015), levando-se em conta que os dados provenientes do FCD são não rotulados, os PNs detectados pelo modelo não possuem uma ligação direta com as classes do problema. Entretanto, para que seja possível o cálculo de medidas de desempenho que consideram os PNs detectados faz-se necessário a associação dos mesmos às classes do problema. Assim, essa metodologia supõe que todo PN está associado a uma classe, e uma classe pode ser relacionada a mais de um PN.

Para definir a classe que um determinado PN ($micro_{PN}$) será associado, verifica-se a quantidade de exemplos de cada classe do problema que foram classificados com o

rótulo de $micro_{PN}$. A classe majoritária, ou seja, a que obteve o maior número de exemplos classificados com o rótulo de $micro_{PN}$ será a classe associada ao $micro_{PN}$.

Para observar o desempenho de cada algoritmo ao longo do tempo, foram apresentados os resultados de avaliação a cada intervalo de 1000 pontos, denominandos momentos de avaliação. Dessa maneira, foi calculada a medida Macro F-Score (SOKOLOVA; LAPALME, 2009) considerando apenas os exemplos explicados pelo modelo, ou seja, aqueles classificados com rótulo de PNs ou rótulo de classes conhecidas. Além disso, a taxa de exemplos classificados como desconhecidos pelo algoritmo foi determinada pela medida *Unknown Rate* (UnkR) (FARIA et al., 2016b), definida em **??**. Nessa medida, $\#C$ é o número total de classes, $\#UnkR_i$ é o número de exemplos da classe C_i classificados como desconhecidos, e Exc_i é o número total de exemplos da classe C_i. Os resultados são apresentados em Capítulo 6.

$$Unkr = \frac{1}{\#C}(\sum_{i=1}^{\#C} \frac{\#UnkR_i}{\#Exc_i}) \tag{5.1}$$

5.5 Considerações Finais

As características para os experimentos detalhados neste capítulo foram estabelecidas com a finalidade de identificar e avaliar o comportamento das abordagens propostas para a DN multi classe em FCDs em diferentes ambientes, como conjuntos de exemplos com distribuição estacionária, conjuntos com distribuição não-estacionária e conjuntos de exemplos reais utilizados na literatura.

A validação das propostas foi produzida perante técnicas já conhecidas da literatura, que produzem resultados relevantes dentro do AFD.

A análise comentada dos resultados (Capítulo 6) esclarece e comprova a hipótese de que as propostas para *FuzzND* e *PFuzzND* produzem resultados comparáveis em relação ao método os quais foram baseados.

6

Resultados e Análise de Experimentos

O detalhamento apresentado no Capítulo 5 corresponde aos experimentos desenvolvidos para avaliação das propostas de aprendizado *fuzzy* baseadas no *Framework Offline Online*. No total, foram 35 variações parâmetros, aplicados aos 6 conjuntos de dados discutidos para o método *PfuzzND* e 25 para o método *FuzzND*.

Neste capítulo é apresentado apenas um subconjunto das tabelas e gráficos produzidos, em consequência da alta dimensionalidade do conjunto total de resultados [1]. A seleção do conteúdo para exposição neste documento foi feita com o objetivo de esclarecer comportamentos e decorrências da utilização das técnicas e parâmetros descritos no Capítulo 5 e Capítulo 4.

A análise e discussão dos resultados está divida em 6 seções, de acordo com os conjuntos de dados selecionados para validação. Nos gráficos apresentados a seguir, as linhas pontilhadas verticais indicam o surgimento do primeiro exemplo de uma nova classe, já as linhas verticais contínuas representam o momento que houve a detecção de um padrão novidade pelo método.

6.1 MOA

Figuras 6.1, 6.2 e 6.3 apresentam os resultados das medidas Macro F-Score e UnkR para 90 momentos de avaliação no conjunto de dados MOA, em cada algoritmo. Como explicado no Capítulo 5 o conjunto de dados MOA tem característica não estacionária e contém o surgimento de 2 classes ao longo do tempo, a partir dos momentos de avaliação 24 e 59.

Com relação aos resultados, o algoritmo *MINAS* (Figura 6.3) foi capaz de detectar as classes novas que surgiram ao longo do FCD, nos momentos 27 e 60 evidenciado pelos picos de aumento da medida UnkR e redução da mesma após tais momentos, além dos constantes resultados de Macro F-Score iguais a 1, o que significa que o método foi capaz

[1] O conjunto total de resultados está disponível no repositório: `http://github.com/tpinhoda/`.

de classificar corretamente todas os exemplos que pode explicar. Por outro lado, a medida UnkR alcança outros picos, antes e depois dos relacionados ao surgimento real das novas classes, seguidos de detecções de PNs, o que pode indicar uma reação às extensões de classes ou de PNs que são identificadas durante o procedimento de DN. Verifica-se nesses resultados também a demora na redução da medida UnkR mesmo após a detecção de um novo PN.

Referente ao método *FuzzND* (Figura 6.1), os resultados demonstram uma melhor resposta ao surgimento de nova classes, já que suas medidas de UnkR diminuem de maneira mais rápida, e de maneira similar ao método *MINAS*, ter sido capaz de produzir resultados de Macro F-score constantes e iguais a 1. De modo geral, o método também pode detectar o surgimento de classes novas, comprovado pelo aumento da medida UnkR e sua redução imediata após detecção de um novo PN nos momentos 26 e 61. Além disso, como o método pode detectar extensões de classe e de PNs durante a etapa de classificação, a adaptação ocorre de forma mais rápida em relação ao método MINAS, o que torna o método menos suscetível a produção de valores altos de UnkR quando não relacionados ao surgimento de novas classes. Entretanto, ainda podem ocorrer detecções de PNs não referentes ao surgimento de novas classes, porém nesse conjunto de dados essa característica ocorre somente uma vez.

Concernente ao algoritmo *PFuzzND*, obteve um comportamento semelhante ao método *FuzzND* (Figura 6.2). Entretanto, foi capaz de detectar apenas PNs relacionados ao surgimento de novas classes, pois assim como o método *FuzzND*, *PfuzzND* também é apto a detectar extensões de classe ou de PNs durante o processo de classificação, o que explica os baixos valores de UnkR durante quase todos os momentos de avaliação, exceto no surgimento de novas classes. A detecção de PNs relacionados apenas ao surgimento de novas classes demonstra o melhor desempenho do método *PFuzzND* em relação aos outros dois. Entretanto, é valido destacar que seus parâmetros foram ajustados de maneira a propiciar os resultados obtidos.

6.2 RBF

Referente ao conjunto de dados RBF as Figura 6.4, Figura 6.5, Figura 6.6 apresentam os resultados das medidas Macro F-Score e UnkR para 48 momentos de avaliação. O conjunto de dados RBF tem característica estacionária, portanto não apresenta mudanças de conceito ao longo do tempo. Entretanto, contém o surgimento de classes a partir dos momentos de avaliação 8 e 42.

Nesse conjunto de dados o algoritmo *MINAS* (Figura 6.6) apresentou altos valores de UnkR seguidos por uma alta quantidade de detecções de PNs, embora tenha produzido constantes resultados de Macro F-Score iguais a 1, o que significa que o método foi capaz de

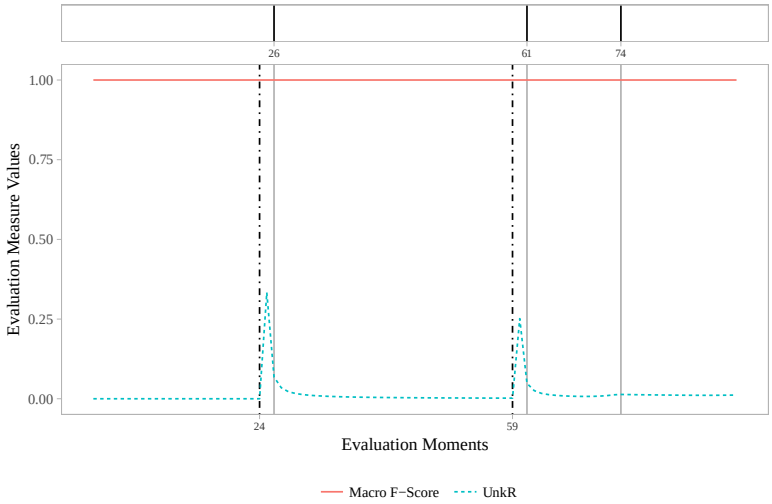


Figura 6.1 – Macro F-Score e UnkR do algoritmo *FuzzND* no conjunto de dados MOA

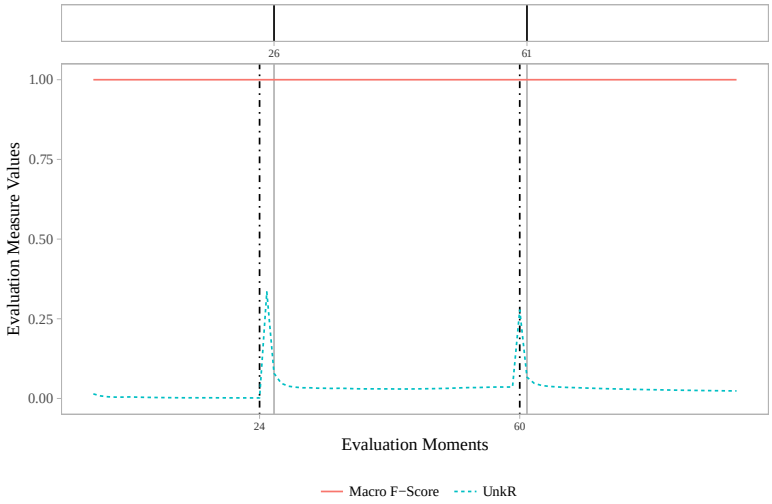


Figura 6.2 – Macro F-Score e UnkR do algoritmo *PFuzzND* no conjunto de dados MOA

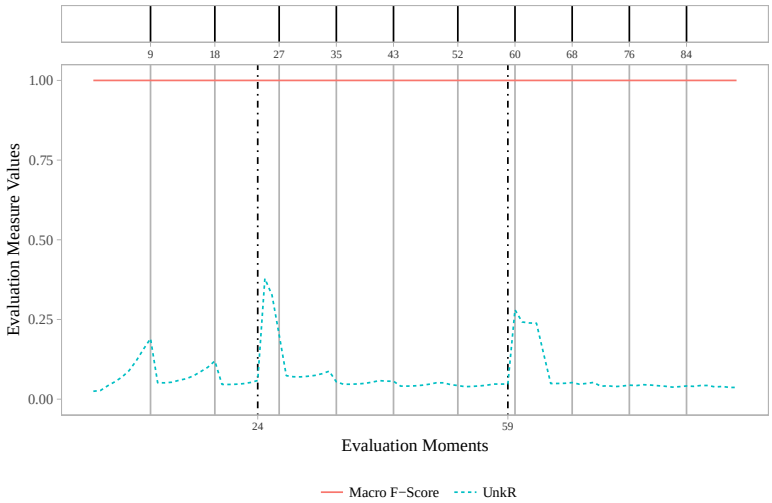


Figura 6.3 – Macro F-Score e UnkR do algoritmo *MINAS* no conjunto de dados MOA

classificar corretamente todas os exemplos que pode explicar. Entretanto, a medida UnkR alcança valores próximos de 0.5 até o momento de avaliação 4 inferindo-se que o método classificou como desconhecido um número próximo da metade dos exemplos processados. Constata-se então que o método obteve um modelo inicial inadequado, que por sua vez interferiu nos resultados durante todo o FCD. Apesar do método conseguir se adaptar e sofrer uma relativa redução na sua medida UnkR, essa ainda manteve-se próxima de 0.25, ou seja, pelo menos 1/4 dos exemplos processados foram classificados como desconhecidos. Além disso, o método apresentou uma considerável demora para detectar PNs referentes ao surgimento de uma nova classe durante os momentos de avaliação 42 e 47. De modo geral, apesar de uma inicialização inadequada o método conseguiu se adaptar e produzir resultados consideravelmente bons.

Referente ao resultados do algoritmo *FuzzND* (Figura 6.4), destaca-se os baixos valores da medida UnkR até o momento de avaliação 8, inferindo-se dessa maneira que o método obteve um bom modelo inicial. Além disso, assim como nos resultados de Macro F-Score do método *MINAS*, *FuzzND* também foi capaz de classificar de forma correta todos os exemplos que pode explicar, fato evidenciado pelos valores constantes de Macro F-Score igual a 1. Entretanto, este método apresentou uma considerável demora para a detecção de PNs relacionados ao surgimento da primeira classe nova, o que causou o aumento na medida UnkR que foi moderadamente reduzida após a detecção de PNs e incorporação dos mesmos no modelo nos momentos de avaliação 14 e 22. Apesar do aumento na medida UnkR, *FuzzND* conseguiu adaptar-se de forma correta ao surgimento de novas classes e detectar uma quantidade de PNs próxima ao número de novas classes que sugiram no FCD.

Em respeito ao método *PFuzzND* (Figura 6.5), foram obtidos os melhores resultados em relação aos outro algoritmos, por conseguir adaptar-se de maneira rápida ao surgimento de novas classes e manter a medida UnkR com valores próximos de 0. Além disso pode classificar de modo correto todos os exemplos não classificados como desconhecidos. Infere-se então que o algoritmo *PFuzzND* foi capaz de melhor representar a distribuição dos dados nesse experimento.

6.3 SynEDC

O resultados relacionados ao conjunto de dados SynEDC são apresentados nas Figura 6.7, Figura 6.8, Figura 6.9 por meio das medidas Macro F-Score e UnkR para 370 momentos de avaliação. O conjunto de dados SynEDC tem característica não estacionária, logo apresenta mudanças de conceito ao longo do tempo. Ademais, contém o surgimento de novas classes a partir dos momentos de avaliação (158, 146, 132, 121, 107, 95, 83, 71, 58, 46, 33, 21, 8), sendo que mais tarde nos momentos (171, 183,196,208,221,233,246,258,

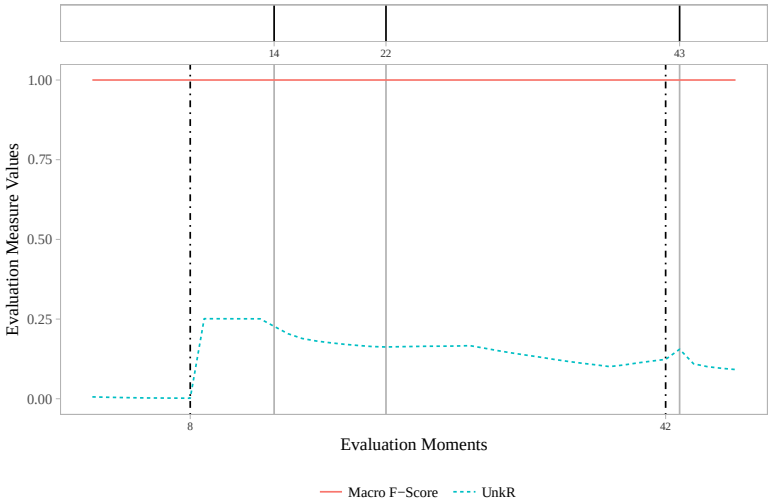


Figura 6.4 – Macro F-Score e UnkR do algoritmo *FuzzND* no conjunto de dados RBF

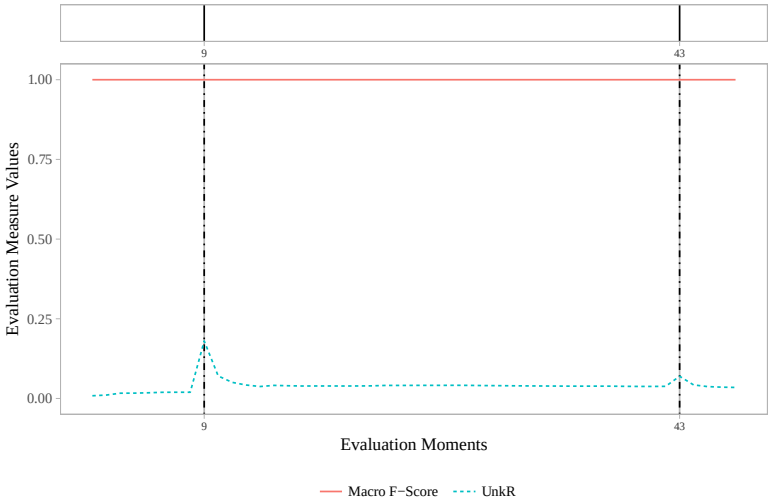


Figura 6.5 – Macro F-Score e UnkR do algoritmo *PFuzzND* no conjunto de dados RBF

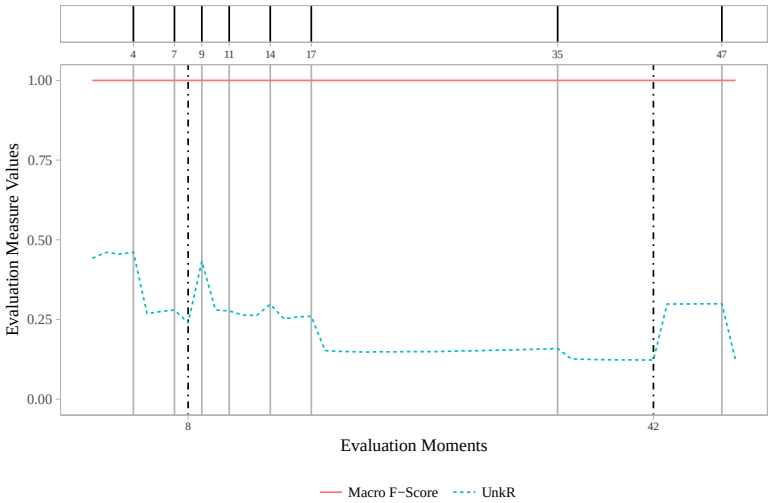


Figura 6.6 – Macro F-Score e UnkR do algoritmo *MINAS* no conjunto de dados RBF

271,283,296,308,321,333,346,358) ocorre o ressurgimento de classes, que podem ser caracterizadas como classes recorrentes, ou seja, classes que surgem ao longo do FCD entre intervalos de tempo grandes.

Com relação aos resultados, o algoritmo *MINAS* (Figura 6.9), não detectou todas a novas classes que surgiram ao longo do FCD acarretando a redução da medida Macro F-Score. Dentre as 14 novas classes que surgiram até o momento de avaliação 171 o método foi capaz de detectar um total de 8. Ademais, após momento 171 a presença de picos na medida UnkR indicam que o método pode ter relacionado grupos válidos de exemplos desconhecidos a classes recorrentes. Entretanto, como a detecção de novas classes nos momentos anteriores não foram totalmente eficazes, o método não conseguiu uma melhora no seu desempenho quanto ao processo de classificação, evidenciado pela redução moderada da medida Macro F-Score ao longo do tempo.

Os métodos *FuzzND* e *PFuzzND* obtiveram resultados muito semelhantes (Figura 6.7,Figura 6.8). Em comparação com o algoritmo *MINAS*, ambos os métodos conseguiram detectar todas as novas classes que surgiram ao longo do FCD. Importante ressaltar que os métodos propostos não tratam classes recorrentes como o algoritmo *MINAS*, portanto assumem que classes recorrentes são novas classes. De modo geral, ambos os métodos conseguiram reagir rapidamente ao surgimento de novas classes. Entretanto, o método *PFuzzND* apresentou picos menores da medida UnkR, principalmente no início do FCD. Além de manter-se com a medida Macro F-Score igual a 1 durante todo o fluxo, diferentemente do método *FuzzND* que apresenta uma redução dessa medida ao fim do FCD. Esses resultados, no entanto, precisam ser melhor avaliados a fim de identificar se existe uma diferença estatística entre eles.

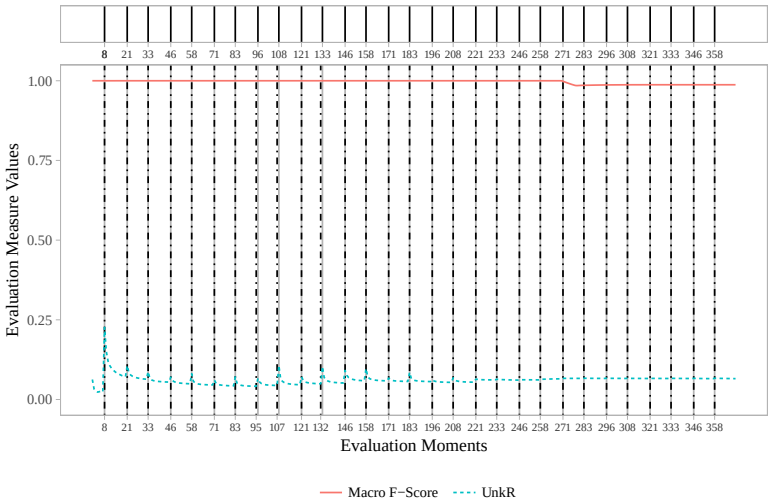


Figura 6.7 – Macro F-Score e UnkR do algoritmo *FuzzND* no conjunto de dados SynEDC

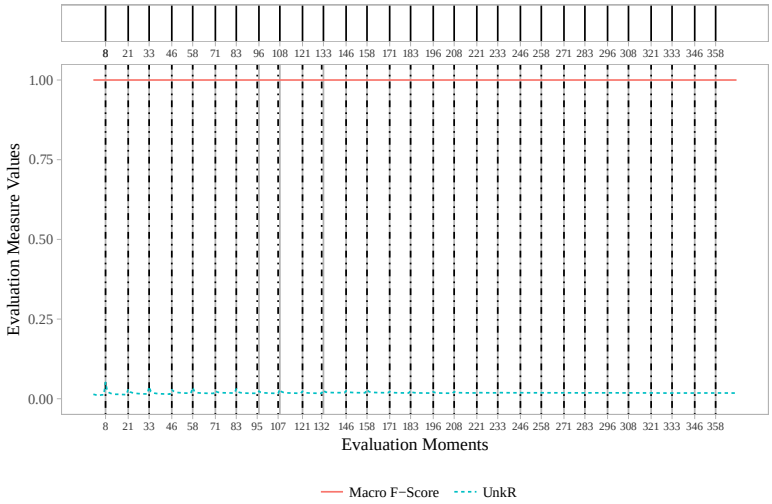


Figura 6.8 – Macro F-Score e UnkR do algoritmo *PFuzzND* no conjunto de dados SynEDC

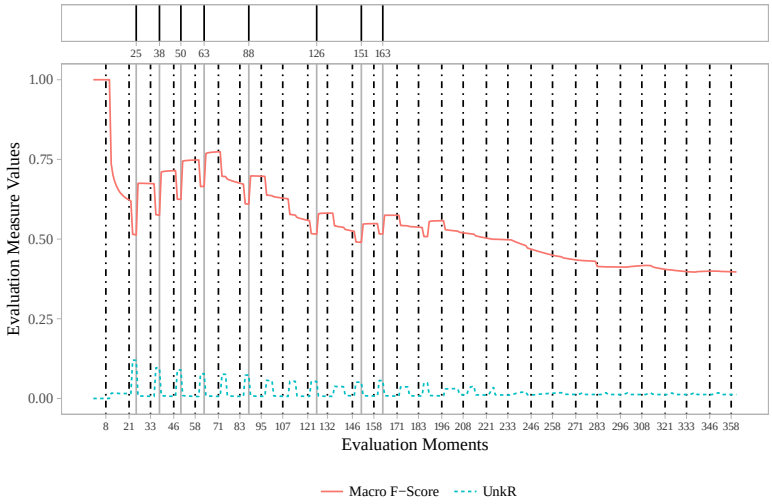


Figura 6.9 – Macro F-Score e UnkR do algoritmo *MINAS* no conjunto de dados SynEDC

6.4 KDD99

Figura 6.10, Figura 6.11 apresentam os resultados das medidas Macro F-Score e UnkR para 442 momentos de avaliação no conjunto de dados KDDD99 para os algoritmos *MINAS* e *PFuzzND*. O método *FuzzND* não foi avaliado nesse conjunto devido ao seu processo de classificação que não consegue lidar com conjuntos de dados que apresentam somente uma classe por um longo período de tempo, característica essa predominante no conjunto de dados KDD99. Dessa maneira, como explicado no Capítulo 5 o conjunto de dados KDD99 é um conjunto real que possui alta dimensionalidade e complexidade na distribuição de seus dados. Destaca-se que para este e os demais conjunto de dados reais, não serão apresentadas linhas verticais pontilhadas indicando o surgimento real de novas classes, pois demandariam um alto custo de tempo para identificá-los além de que poderiam dificultar a leitura dos gráficos de resultados.

Com relação ao método *MINAS*, percebe-se que inicialmente o método apresenta altos índices de UnkR juntamente com baixos valores de Macro F-Score. Infere-se por meio disso que o classificador inicial gerado na fase *offline* não provou-se adequado para os dados iniciais do FCD. Entretanto, após uma série de adaptações e detecções de PNs por parte do método, produz-se resultados das duas medidas de maneira constante, porém como este é um conjunto de dados desbalanceado, é válido ressaltar que os índices Macro F-Score podem indicar a correta classificação apenas de classes predominantes.

O método *PFuzzND*, entretanto, apresentou resultados similares ao método *MINAS*. De modo geral, o algoritmo apresentou altos valores da medida UnkR inicialmente, constatando-se assim a geração de um classificador inicial inadequado para o FCD. Entretanto, o método apresenta uma constante nos resultados de ambas as medidas de avaliação após determinado momento. Em comparação com o método *MINAS*, apesar dos valores de Macro F-Score do método *PFuzzND* apresentarem resultados melhores, ainda que influenciados negativamente pelo desbalanceamento de classes, os índices de UnkR são moderadamente mais elevados e há uma maior detecção de PNs, o que caracteriza uma maior inflexibilidade no processo de classificação. Nesse caso, é importante destacar que apesar da flexibilização do processo de classificação por meio de teoria de conjuntos *fuzzy*, a proposta apresentou-se mais inflexível do que o método não *fuzzy*. Resguardadas as devidas considerações a respeito dos parâmetros escolhidos, esse comportamento pode ter ocorrido devido a ocorrência de exemplos idênticos nesse conjunto de dados, que desconfigura o cálculo de tipicidade obtido por meio da média das distâncias do exemplos para o centróide do micro-grupo. Nessas circunstâncias, micro-grupos criados a partir de um conjunto de dados cuja maioria dos exemplos são idênticos, apresentarão as médias das distâncias muito próximas de 0, o que faz com que a maioria dos novos exemplos sejam distantes da média e possuam valores de tipicidade muito baixos, logo sendo classificados como desconhecidos.

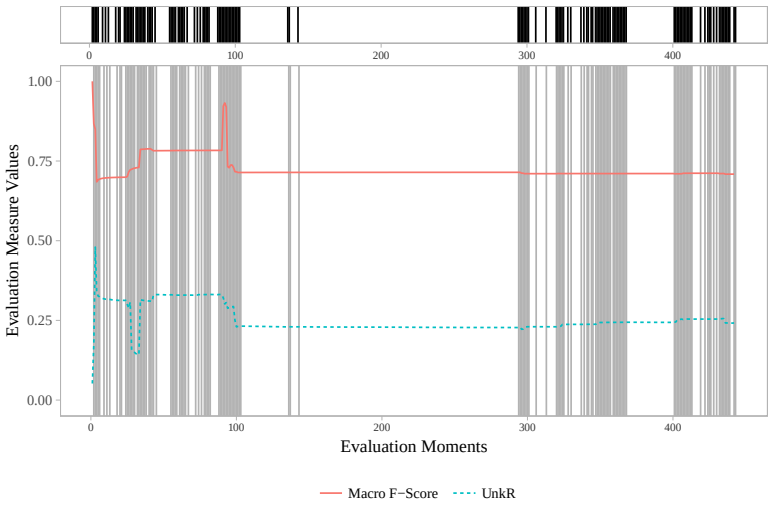


Figura 6.10 – Macro F-Score e UnkR do algoritmo *PFuzzND* no conjunto de dados KDD99

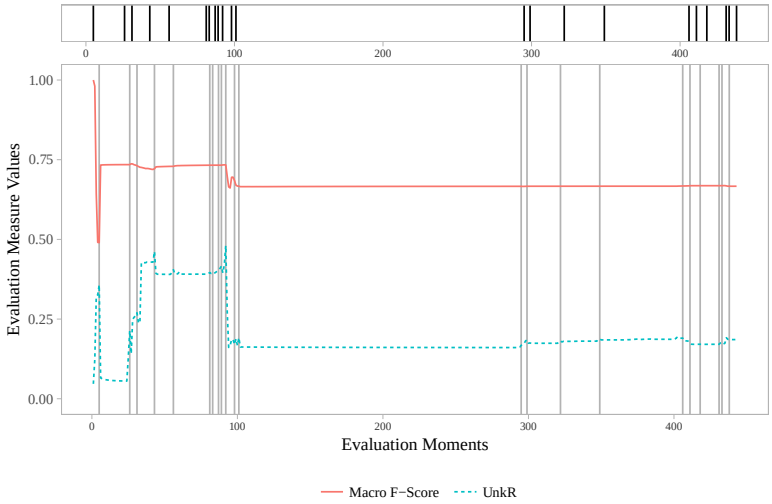


Figura 6.11 – Macro F-Score e UnkR do algoritmo *MINAS* no conjunto de dados KDD99

6.5 FcTe

Os resultados relacionados ao conjunto de dados FcTe são apresentados nas Figura 6.13 e Figura 6.12 por meio das medidas Macro F-Score e UnkR para 450 momentos de avaliação. Assim como o conjunto de dados KDD99 este também é um conjunto de dados real que apresenta uma alta dimensionalidade, distribuição de dados complexa e desbalanceamento de classes. Além disso, por potencialmente apresentar sobreposição de classe (Figura 5.4b) não foram feitos experimentos com o método *FuzzND*, pelo fato do mesmo não conseguir tratar este tipo de problema. Ademais, assim como na avaliação do conjunto de dados KDD99 não houve a identificação manual do surgimento de novas classes, logo, os gráficos não apresentam linhas verticais pontilhadas.

O método *Minas* apresentou valores de Macro F-Score muito baixos apesar de possuir uma taxa de UnkR próxima de 0 ao longo do FCD. Infere-se por meio disto que o modelo de decisão não foi adequado durante todo o FCD e os PNs detectados não foram o suficiente para uma melhora considerável do modelo. A potencial existência de classes sobrepostas visualizadas na Figura 5.4b pode ter causado os resultados obtidos pelo método, já que micro-grupos de classes diferentes e sobrepostos podem causar classificações errôneas pelo modelo.

Por outro lado, o método *PFuzzND* obteve bons resultados de ambas as medidas ao longo do FCD. Entretanto, precisou detectar muito mais PNs que o método MINAS. Acredita-se portanto, que a grande quantidade de PNs detectados influenciou de modo positivo nos resultados, ajudando o método a melhor classificar novos exemplos. Porém, torna-se impossível detectar por meio das medidas de avaliação utilizadas nesse trabalho o surgimento de novas classes. Constata-se por meio destes resultados que o modelo de decisão baseado em micro-grupos não é o ideal pra este conjunto de dados.

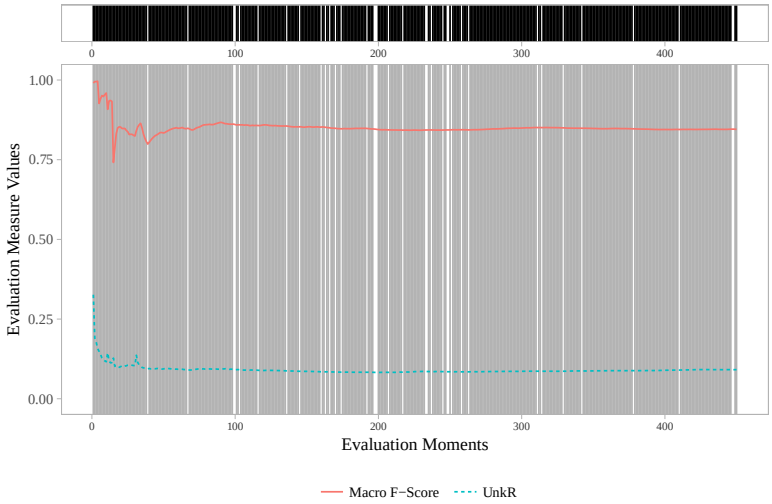


Figura 6.12 – Macro F-Score e UnkR do algoritmo *PFuzzND* no conjunto de dados FcTe

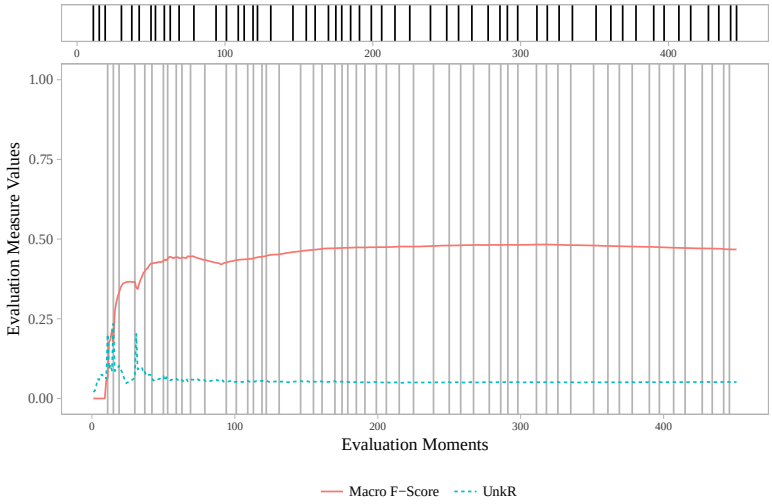


Figura 6.13 – Macro F-Score e UnkR do algoritmo *MINAS* no conjunto de dados FcTe

6.6 CoverType

Referente ao conjunto de dados CoverType as Figura 6.14 e Figura 6.15 apresentam os resultados das medidas Macro F-Score e UnkR para 580 momentos de avaliação. Esse é um conjunto de dados real que, de maneira similar ao conjunto de dados FcTe, apresenta potencialmente sobreposição de classes, como visualizado na Figura 5.4a além de possuir desbalanceamento de classe. Portanto, não houve avaliação do método *FuzzND*. Ademais, também não houve a identificação manual de surgimento de novas classes.

De maneira geral, o método *MINAS* não apresentou um bom desempenho nesse conjunto de dados, obtendo valores de Macro F-Score abaixo de 0.5 ao longo da maior parte do FCD, apesar de ter produzido baixos índices de UnkR. Por outro lado, o método *PFuzzND* obteve valores de Macro F-Score melhores, entretanto, ainda próximos de 0.5. Ademais, *PFuzzND* produziu valores de UnkR relativamente maiores além de ter

detectado mais PNs. O desbalanceamento desse conjunto de dados juntamente com sobreposição de classe, influenciaram negativamente nos resultados gerados por ambos os algoritmos. Em especial, classes sobrepostas influenciam no cálculo de tipicidade adotado por *PFuzzND*, dessa maneira classes sobrepostas produzem valores similares de tipicidade o que pode causar classificações errôneas por parte do método.

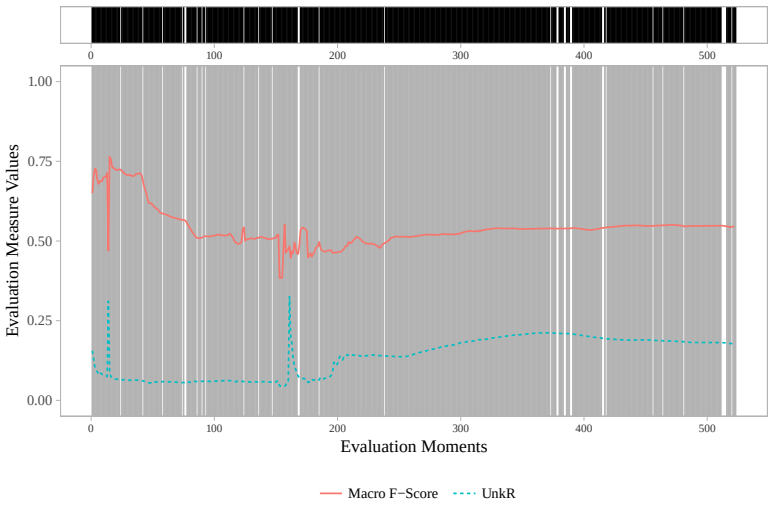


Figura 6.14 – Macro F-Score e UnkR do algoritmo *PFuzzND* no conjunto de dados CoverType

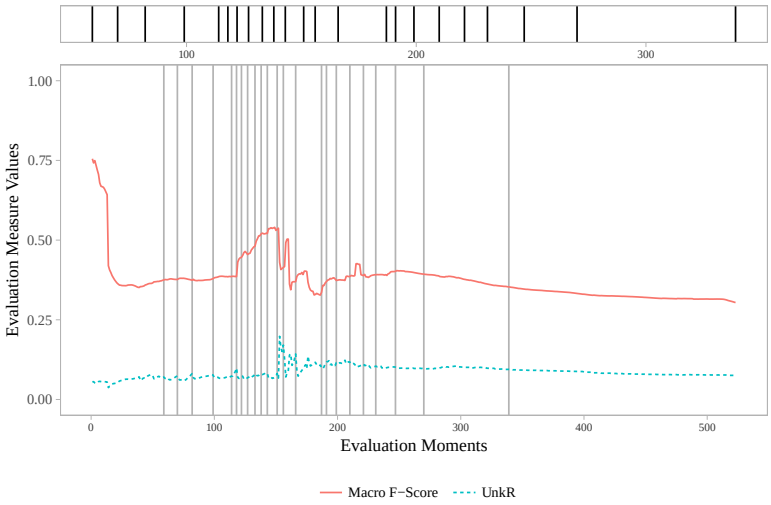


Figura 6.15 – Macro F-Score e UnkR do algoritmo *MINAS* no conjunto de dados CoverType

6.7 Considerações Finais

O corpo de resultados comentados neste capítulo configura um subconjunto do total obtido pela execução dos 60 experimentos aos conjuntos de exemplos selecionados. Apesar do conjunto reduzido de resultados apresentados neste capítulo, destaques importantes são expostos.

Esses destaques dizem respeito a um levantamento de características de comportamento das distintas combinações de parâmetros definidas para experimentos, analisando os resultados produzidos em variados aspectos, como a compatibilidade entre os micro-grupos e conjunto de exemplos, avaliação das propostas levando-se em consideração a influência dos micro-grupos *fuzzy* no aprendizado baseado no *Framework Offline-Online*.

Quanto aos experimentos realizados com as abordagens propostas no Capítulo 4, *FuzzND* e *PFuzzND*, os resultados produzidos foram interessantes por apontarem valores comparáveis a abordagem tida como base, pela consistência observada entre as duas métricas e, também, por apresentar valores melhores ou iguais de Macro F-Score em todos os experimentos, indicando um melhor desempenho em termos de classificação adquiridos por meio da flexibilização desse processo.

7

Conclusões

Fluxos de Dados representam um domínio de problemas que tende a aumentar, acompanhando a evolução tecnológica. Estratégias para extração de conhecimento a partir de Fluxo de Dados são uma tendência dentro das pesquisas na área de Aprendizado de Máquina.

As características imprevisíveis dos domínios de Fluxo de Dados geram dificuldades no processo de aprendizagem, encorajando a busca por aprendizado flexível, por exemplo, pela integração de conceitos da teoria de conjuntos *fuzzy*. Propostas baseadas em conceitos *fuzzy* têm o objetivo de colaborar para flexibilidade e adaptabilidade do conhecimento aprendido.

Dentro da tarefa de DN em FCD por meio de técnicas *fuzzy*, a maioria das propostas limitam-se a domínios específicos e consideram a existência de apenas duas classes (Normal e Anormal). Por outro lado, a maioria das abordagens para a DN multiclasse em FCD não representam as imprecisões que podem ocorrer ao longo do tempo, o que pode ser prejudicial em um contexto no qual a distribuição dos dados é mutável com o tempo, ocasionando representações inadequadas.

Contribuições

Este trabalho apresenta duas novas abordagem para DN multiclasse em FCD, que introduz maior flexibilidade no processo de aprendizagem por meio da aplicação de conceitos da teoria de conjuntos *fuzzy* para construção e gerenciamento de uma estrutura de sumarização para a fase *online* do *Framework Offline-Online*.

As principais contribuições do trabalho são descritas a seguir:

- **Definição de estruturas de sumarização *fuzzy* supervisionada:** Em particular, sobre as estruturas de sumarização propostas, pode-se ressaltar que a estrutura

SFMiC (SILVA et al., 2017), sua extensão (SILVA et al., 2018) e a estrutura SPFMiC oferecem representações de qualidade para FCDs.

- **Extensões *fuzzy* do método MINAS:** Os métodos propostos *FuzzND* (SILVA et al., 2018) e *PFuzzND* foram capazes de melhor reagir as mudanças ocorridas nos conjuntos de dados sintéticos e produzir resultados comparáveis em conjutos reais

Os experimentos realizados comprovam que as propostas são comparáveis, ou melhores ao método base não-*fuzzy*, mostrando-se vantajosas para representação geral e identificação de novidades em conjuntos de exemplos.

Os resultados obtidos são um indicativo de que as propostas são significativas e contribuem para o futuro desenvolvimento de extensões e novas técnicas fundamentadas nas propostas apresentadas neste trabalho.

Finalmente, as abordagens *fuzzy* baseadas no método *MINAS* contribuem para o avanço das comunidades de pesquisa em aprendizado flexível e aprendizado em fluxo de dados.

Limitações e Trabalhos Futuros

Os resultados produzidos pelas abordagens propostas sugerem que o conhecimento gerado utilizando os métodos *FuzzND* ou *PFuzzND* é efetivo quanto a avaliações métricas. Essas abordagens são comparáveis e obtém melhores resultados que sua versão não-*fuzzy*.

Todavia, são identificados alguns pontos de melhoria e de extensão para as propostas apresentadas neste documento.

- *Parametrização do algoritmo:* Apresenta-se como uma das principais limitações dos métodos propostos, bem como pelo algoritmo base *MINAS*. Uma quantidade relativamente grande de parâmetros deve ser configurada, por exemplo, número de grupos por classe, limiares para a execução do processo de DN, critérios de validação de um novo grupo, e limiares relacionados a identificação de extensões e *outliers*. De maneira geral, cada conjunto de dados pode apresentar uma combinação diferente de valores de parâmetros, e encontrar a melhor combinação não é uma tarefa fácil. Por meio dos experimentos desenvolvidos, notou-se uma piora no desempenho preditivo quando os parâmetros não são configurados adequadamente. Ademais, os valores escolhidos para os limiares de adaptação θ_{class} e θ_{adapt}, apresentam-se como críticos para o desempenho do algoritmo.

  Portanto, novas estratégias devem ser estudas de maneira a torna estes parâmetros adaptativos e diminuir a sensibilidade dos métodos quanto a eles.

- **Modelo baseado em hiperesferas:** Apesar dos métodos propostos apresentarem uma representação mais flexível por meio de micro-grupos *fuzzy*. Nos experimentos com conjunto de dados reais notou-se desempenhos preditivos muito baixos o que pode ser causado pelo viés imposto pelo método de superfícies de decisão em formatos hiperesféricos, quando os conjuntos de dados não apresentam esta característica.

  Dessa maneira, novos classificadores devem ser estudados com o intuito de melhor definir superfícies de decisão em conjuntos de dados complexos, por exemplo, que podem apresentar sobreposição de classes.

- **Tratamento de mudanças de conceito abruptas**: De modo geral uma das limitações das propostas desse trabalho bem como do método base, é a dificuldade na identificação de mudanças de conceitos abruptas e por conseguinte a adaptação dos métodos, quando estas existem.

  Portanto, faz-se necessário o estudo de técnicas capazes de identificar de forma correta este tipo de mudança e adaptar o método de forma rápida

- **Comparação com outras abordagens:** Os experimentos definidos para avaliar a proposta envolvem apenas o método que serviu como base. Como descrito neste documento, existem outras abordagens de DN em FCD, inclusive que utilizam técnicas *fuzzy*.

  É fundamental que seja realizada a comparação das novas abordagens com as técnicas de DN em FCD já existentes. Também deve haver a preocupação quanto à utilização de métricas de avaliação, pois as diferentes abordagens foram avaliadas com diferentes métricas.

Referências

ABDALLAH, Z. S. et al. Anynovel: detection of novel concepts in evolving data streams. *Evolving Systems*, Springer, v. 7, n. 2, p. 73–93, 2016.

AGGARWAL, C. C. On classification of graph streams. In: SIAM. *Proceedings of the 2011 SIAM International Conference on Data Mining*. [S.l.], 2011. p. 652–663.

AGGARWAL, C. C. An introduction to outlier analysis. In: *Outlier Analysis*. [S.l.]: Springer, 2013. p. 1–40.

AGGARWAL, C. C. *A Survey of Stream Classification Algorithms*. 2014.

AGGARWAL, C. C. et al. A Framework for Clustering Evolving Data Streams. In: *Proceedings of the 29th International Conference on Very Large Data bases*. [S.l.]: Morgan Kaufmann, 2003. v. 29, p. 81–92.

AGGARWAL, C. C. et al. A framework for clustering evolving data streams. In: VLDB ENDOWMENT. *Proceedings of the 29th international conference on Very large data bases-Volume 29*. [S.l.], 2003. p. 81–92.

AGGARWAL, C. C. et al. On demand classification of data streams. In: ACM. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2004. p. 503–508.

AGGARWAL, C. C.; YU, P. S. A Framework for Clustering Uncertain Data Streams. In: *2008 IEEE 24th International Conference on Data Engineering*. [S.l.]: IEEE, 2008. v. 00, p. 150–159.

AL-KHATEEB, T. et al. Stream classification with recurring and novel class detection using class-based ensemble. In: IEEE. *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. [S.l.], 2012. p. 31–40.

AL-KHATEEB, T. M. et al. Cloud guided stream classification using class-based ensemble. In: IEEE. *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. [S.l.], 2012. p. 694–701.

ALBERTINI, M. K.; MELLO, R. F. de. A self-organizing neural network for detecting novelties. In: ACM. *Proceedings of the 2007 ACM symposium on Applied computing*. [S.l.], 2007. p. 462–466.

ANGELOV, P.; RAMEZANI, R.; ZHOU, X. Autonomous novelty detection and object tracking in video streams using evolving clustering and takagi-sugeno type neuro-fuzzy system. In: IEEE. *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. [S.l.], 2008. p. 1456–1463.

ANGELOV, P. P.; ZHOU, X. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 16, n. 6, p. 1462–1475, 2008.

BABCOCK, B. et al. Models and issues in data stream systems. In: ACM. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. [S.l.], 2002. p. 1–16.

BACHE, K.; LICHMAN, M. *UCI machine learning repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

BEZDEK, J. C.; EHRLICH, R.; FULL, W. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, Elsevier, v. 10, n. 2-3, p. 191–203, 1984.

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

BIFET, A. et al. MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. In: *JMLR: Workshop and Conference Proceedings*. [S.l.]: Microtome Publishing, 2010. v. 11, p. 44–50.

BOUCHACHIA, A.; VANARET, C. GT2FC: An Online Growing Interval Type-2 Self-Learning Fuzzy Classifier. *IEEE Transactions on Fuzzy Systems*, v. 22, n. 4, p. 999–1018, 2014.

BREVE, F.; ZHAO, L. Semi-supervised Learning with Concept Drift Using Particle Dynamics Applied to Network Intrusion Detection Data. In: *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*. [S.l.]: IEEE, 2013. p. 335–340.

CAMPELLO, R. J.; HRUSCHKA, E. R. A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems*, Elsevier, v. 157, n. 21, p. 2858–2875, 2006.

CAO, F. et al. Density-Based Clustering over an Evolving Data Stream with Noise. In: *Proceedings of the 2006 SIAM International Conference on Data Mining*. [S.l.]: Society for Industrial and Applied Mathematics, 2006. p. 328–339.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Outlier detection: A survey. *ACM Computing Surveys*, 2007.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM, v. 41, n. 3, p. 15, 2009.

CHANDOLA, V.; BORIAH, S.; KUMAR, V. Understanding categorical similarity measures for outlier detection. *Technical report 08–008, University of Minnesota*, 2008.

CHAWLA, S.; SUN, P. Slom: a new measure for local spatial outliers. *Knowledge and Information Systems*, Springer, v. 9, n. 4, p. 412–429, 2006.

CHEN, Y. et al. Multi-dimensional regression analysis of time-series data streams. In: VLDB ENDOWMENT. *Proceedings of the 28th international conference on Very Large Data Bases*. [S.l.], 2002. p. 323–334.

CHEN, Y.; TU, L. Density-based clustering for real-time stream data. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*. [S.l.]: ACM Press, 2007. p. 133–142.

CHENG, Y. et al. Learning to Group Web Text Incorporating Prior Information. In: *2011 IEEE 11th International Conference on Data Mining Workshops*. [S.l.]: IEEE, 2011. p. 212–219.

CINTRA, M. E.; MONARD, M. C.; CAMARGO, H. A. A fuzzy decision tree algorithm based on c4. 5. *Mathware & Soft Computing*, v. 20, p. 56–62, 2013.

Computational Intelligence Group. *Data Stream Repository*. 2017. Department of Computing, Federal University of São Carlos, São Carlos, Brazil. Disponível em: <http://github.com/CIG-UFSCar/DS_Datasets>.

COULL, S. et al. Intrusion detection: A bioinformatics approach. In: IEEE. *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. [S.l.], 2003. p. 24–33.

DECKER, K. M.; FOCARDI, S. Technology overview: A report on data mining. Citeseer, 1995.

DESHPANDE, A. et al. Model-driven data acquisition in sensor networks. In: VLDB ENDOWMENT. *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. [S.l.], 2004. p. 588–599.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: ACM. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2000. p. 71–80.

ESKIN, E. et al. A geometric framework for unsupervised anomaly detection. In: *Applications of data mining in computer security*. [S.l.]: Springer, 2002. p. 77–101.

FARIA, E. R. et al. Novelty detection in data streams. *Artificial Intelligence Review*, Springer, v. 45, n. 2, p. 235–269, 2016.

FARIA, E. R. de et al. Minas: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, Springer, v. 30, n. 3, p. 640–680, 2016.

FARIA, E. R. de et al. Evaluation of multiclass novelty detection algorithms for data streams. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 27, n. 11, p. 2961–2973, 2015.

FARID, D. M.; RAHMAN, C. M. Novel class detection in concept-drifting data stream mining employing decision tree. In: IEEE. *Electrical & Computer Engineering (ICECE), 2012 7th International Conference on*. [S.l.], 2012. p. 630–633.

FARID, D. M. et al. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications*, Elsevier, v. 40, n. 15, p. 5895–5906, 2013.

FDEZ-RIVEROLA, F. et al. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications*, v. 33, n. 1, p. 36–48, 2007.

FILEV, D. P.; TSENG, F. Novelty detection based machine health prognostics. In: IEEE. *Evolving Fuzzy Systems, 2006 International Symposium on*. [S.l.], 2006. p. 193–199.

FILEV, D. P.; TSENG, F. Real time novelty detection modeling for machine health prognostics. In: IEEE. *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American*. [S.l.], 2006. p. 529–534.

FILIPPONE, M.; SANGUINETTI, G. Novelty detection in autoregressive models using information theoretic measures. *Techn. Ber. Department of Computer Science, University of Sheffield*, 2009.

GAMA, J. *Knowledge discovery from data streams*. [S.l.]: CRC Press, 2010.

GAMA, J.; GABER, M. M. (Ed.). *Learning from Data Streams: Processing Techniques in Sensor Networks*. [S.l.]: Springer, 2007. 244 p.

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

GOGOI, P. et al. A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, Br Computer Soc, p. bxr026, 2011.

GOH, K.-S.; CHANG, E. Y.; LI, B. Using one-class and two-class svms for multiclass image annotation. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 17, n. 10, p. 1333–1346, 2005.

GÓMEZ, J.; GONZÁLEZ, F.; DASGUPTA, D. An immuno-fuzzy approach to anomaly detection. In: IEEE. *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*. [S.l.], 2003. v. 2, p. 1219–1224.

GUHA, S. et al. Clustering data streams. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. [S.l.]: IEEE Comput. Soc, 2000. p. 359–366.

GUSTAFSON, D. E. G. D. E.; KESSEL, W. C. K. W. C. Fuzzy clustering with a fuzzy covariance matrix. *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, v. 17, n. 2, p. 761–766, 1978.

HAGAN, M. T.; MENHAJ, M. B. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, IEEE, v. 5, n. 6, p. 989–993, 1994.

HAHSLER, M.; BOLANOS, M.; FORREST, J. *streamMOA: Interface for MOA Stream Clustering Algorithms*. [S.l.], 2015. R package version 1.1-2. Disponível em: <https://CRAN.R-project.org/package=streamMOA>.

HAHSLER, M.; BOLANOS, M.; FORREST, J. *stream: Infrastructure for Data Stream Mining*. [S.l.], 2016. R package version 1.2-3. Disponível em: <https://CRAN.R-project-.org/package=stream>.

HANLEY, J. A.; MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, v. 143, n. 1, p. 29–36, 1982.

HAQUE, A.; KHAN, L.; BARON, M. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In: *AAAI*. [S.l.: s.n.], 2016. p. 1652–1658.

HARTIGAN, J. A.; WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, JSTOR, v. 28, n. 1, p. 100–108, 1979.

HASHEMI, S.; YANG, Y. Flexible decision tree for data stream classification in the presence of concept change, noise and missing values. *Data Mining and Knowledge Discovery*, Springer, v. 19, n. 1, p. 95–131, 2009.

HAYAT, M. Z. et al. Content-based concept drift detection for email spam filtering. In: IEEE. *Telecommunications (IST), 2010 5th International Symposium on*. [S.l.], 2010. p. 531–536.

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

HELLER, K. A. et al. One class support vector machines for detecting anomalous windows registry accesses. In: *Proc. of the workshop on Data Mining for Computer Security*. [S.l.: s.n.], 2003. v. 9.

HORE, P. et al. Online fuzzy c means. In: *NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society*. [S.l.]: IEEE, 2008. p. 1–5.

HORE, P.; HALL, L. O.; GOLDGOF, D. B. Creating Streaming Iterative Soft Clustering Algorithms. In: *NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society*. [S.l.]: IEEE, 2007. p. 484–488.

HORE, P.; HALL, L. O.; GOLDGOF, D. B. Single Pass Fuzzy C Means. In: *2007 IEEE International Fuzzy Systems Conference*. [S.l.]: IEEE, 2007. p. 1–7.

HULTEN, G.; DOMINGOS, P. *VFML - A toolkit for mining high-speed time-changing data streams*. 2003. Disponível em: <http://www.cs.washington.edu/dm/vfml/>.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: ACM. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2001. p. 97–106.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.]: ACM Press, 2001. p. 97–106.

ISAZADEH, A.; MAHAN, F.; PEDRYCZ, W. Mflexdt: multi flexible fuzzy decision tree for data stream classification. *Soft Computing*, Springer, v. 20, n. 9, p. 3719–3733, 2016.

JOLLIFFE, I. *Principal component analysis*. [S.l.]: Wiley Online Library, 2002.

KANAGAL, B.; DESHPANDE, A. Online filtering, smoothing and probabilistic modeling of streaming data. In: IEEE. *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. [S.l.], 2008. p. 1160–1169.

KAPOOR, A. et al. Gaussian processes for object categorization. *International journal of computer vision*, Springer, v. 88, n. 2, p. 169–188, 2010.

KEMMLER, M. et al. One-class classification with gaussian processes. *Pattern Recognition*, Elsevier, v. 46, n. 12, p. 3507–3518, 2013.

KEOGH, E. et al. Finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*, Springer, v. 11, n. 1, p. 1–27, 2007.

KIM, H.-C.; LEE, J. Pseudo-density estimation for clustering with gaussian processes. *Advances in Neural Networks-ISNN 2006*, Springer, p. 1238–1243, 2006.

KMIECIAK, M. R.; STEFANOWSKI, J. Semi-supervised approach to handle sudden concept drift. *Control and Cybernetics*, v. 40, n. 3, p. 667–695, 2011.

KRANEN, P. et al. The ClusTree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, Springer-Verlag, v. 29, n. 2, p. 249–272, 2011.

KRAWCZYK, B. Diversity in ensembles for one-class classification. In: *New Trends in Databases and Information Systems*. [S.l.]: Springer, 2013. p. 119–129.

KRISHNAPURAM, R.; KELLER, J. M. A possibilistic approach to clustering. *IEEE transactions on fuzzy systems*, IEEE, v. 1, n. 2, p. 98–110, 1993.

KRISHNAPURAM, R.; KELLER, J. M. The possibilistic C-means algorithm: Insights and recommendations. *IEEE Transactions on Fuzzy Systems*, v. 4, n. 3, p. 385–393, 1996.

KUROSE, J. et al. An end-user-responsive sensor network architecture for hazardous weather detection, prediction and response. In: SPRINGER. *Asian Internet Engineering Conference*. [S.l.], 2006. p. 1–15.

LABROCHE, N. Online fuzzy medoid based clustering algorithms. *Neurocomputing*, v. 126, p. 141–150, 2014.

LE, T. et al. Multiple distribution data description learning algorithm for novelty detection. *Advances in Knowledge Discovery and Data Mining*, Springer, p. 246–257, 2011.

LEITE, D.; COSTA, P.; GOMIDE, F. Evolving granular neural network for semi-supervised data stream classification. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2010. p. 1–8.

LEITE, D.; COSTA, P.; GOMIDE, F. Evolving granular neural network for semi-supervised data stream classification. In: IEEE. *Neural Networks (IJCNN), The 2010 International Joint Conference on*. [S.l.], 2010. p. 1–8.

LEMOS, A.; CAMINHAS, W.; GOMIDE, F. Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Information Sciences*, v. 220, p. 64–85, 2013.

LEMOS, A.; CAMINHAS, W.; GOMIDE, F. Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Information Sciences*, Elsevier, v. 220, p. 64–85, 2013.

LI, X.; CROFT, W. B. Improving novelty detection for general topics using sentence level information patterns. In: ACM. *Proceedings of the 15th ACM international conference on Information and knowledge management*. [S.l.], 2006. p. 238–247.

LI, Y. et al. A study of large-scale data clustering based on fuzzy clustering. *Soft Computing*, Springer Berlin Heidelberg, v. 20, n. 8, p. 3231–3242, 2016.

LING, C.; LING-JUN, Z.; LI, T. Stream data classification using improved fisher discriminate analysis. *J. Computers*, v. 4, n. 3, p. 208–214, 2009.

LIU, Y.; ZHOU, Y. Online Detection of Concept Drift in Visual Tracking. In: *Neural Information Processing*. [S.l.]: Springer International Publishing, 2014. p. 159–166.

LIU, Y.-H.; LIU, Y.-C.; CHEN, Y.-J. Fast support vector data descriptions for novelty detection. *IEEE Transactions on Neural Networks*, IEEE, v. 21, n. 8, p. 1296–1313, 2010.

LOPES, P. A.; CAMARGO, H. A. Fuzzstream: Fuzzy data stream clustering based on the online-offline framework. In: *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.]: IEEE, 2017.

MAATEN, L. v. d.; HINTON, G. Visualizing data using t-sne. *Journal of machine learning research*, v. 9, n. Nov, p. 2579–2605, 2008.

MAGDY, A.; BASSIOUNY, M. K. SIC-Means: A Semi-fuzzy Approach for Clustering Data Streams Using C-Means. In: *Artificial Neural Networks in Pattern Recognition*. [S.l.]: Springer Berlin Heidelberg, 2010. p. 96–107.

MARIN, L. et al. On-line dynamic adaptation of fuzzy preferences. *Information Sciences*, v. 220, p. 5–21, 2013.

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

MASUD, M. et al. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 23, n. 6, p. 859–874, 2011.

MASUD, M. M. et al. Addressing concept-evolution in concept-drifting data streams. In: IEEE. *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. [S.l.], 2010. p. 929–934.

MCBAIN, J.; TIMUSK, M. Feature extraction for novelty detection as applied to fault detection in machinery. *Pattern Recognition Letters*, Elsevier, v. 32, n. 7, p. 1054–1061, 2011.

MCCALLUM, A.; NIGAM, K.; UNGAR, L. H. Efficient clustering of high-dimensional data sets with application to reference matching. In: ACM. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2000. p. 169–178.

MEYER, D. et al. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. [S.l.], 2015. R package version 1.6-7. Disponível em: <https://CRAN.R-project.org/package=e1071>.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. *Machine learning: An artificial intelligence approach*. [S.l.]: Springer Science & Business Media, 2013.

MILJKOVIĆ, D. Review of novelty detection methods. In: IEEE. *Mipro, 2010 proceedings of the 33rd international convention*. [S.l.], 2010. p. 593–598.

MITCHELL, T. *Machine Learning*. [S.l.]: McGraw-Hill Education, 1997. 414 p.

MOSTAFAVI, S.; AMIRI, A. Extending fuzzy c-means to clustering data streams. In: *20th Iranian Conference on Electrical Engineering (ICEE2012)*. [S.l.]: IEEE, 2012. p. 726–729.

MUÑOZ-MARÍ, J. et al. Semisupervised one-class support vector machines for classification of remote sensing data. *IEEE transactions on geoscience and remote sensing*, IEEE, v. 48, n. 8, p. 3188–3197, 2010.

NAHAR, V. et al. Semi-supervised Learning for Cyberbullying Detection in Social Networks. In: *Databases Theory and Applications*. [S.l.]: Springer International Publishing, 2014. p. 160–171.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. *Knowledge and information systems*, Springer, v. 45, n. 3, p. 535–569, 2015.

O'CALLAGHAN, L. et al. Streaming-data algorithms for high-quality clustering. In: *Proceedings 18th International Conference on Data Engineering*. [S.l.]: IEEE Comput. Soc, 2002. p. 685–694.

PAL, N. R. et al. A possibilistic fuzzy c-means clustering algorithm. *IEEE transactions on fuzzy systems*, IEEE, v. 13, n. 4, p. 517–530, 2005.

PAN, J.; YANG, Q.; PAN, S. Online co-localization in indoor wireless networks by dimension reduction. In: *Proceedings of the National Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2007. p. 1102–1107.

PATCHA, A.; PARK, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, Elsevier, v. 51, n. 12, p. 3448–3470, 2007.

PEDRYCZ, W.; GOMIDE, F. *An Introduction to Fuzzy Sets: Analysis and Design*. [S.l.]: MIT Press, 1998. 465 p.

PERNER, P. Concepts for novelty detection and handling based on a case-based reasoning process scheme. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 22, n. 1, p. 86–91, 2009.

PIATETSKY, G. *R, Python Duel As Top Analytics, Data Science software*. 2016. Disponível em: <http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html>.

PIMENTEL, M. A. et al. A review of novelty detection. *Signal Processing*, Elsevier, v. 99, p. 215–249, 2014.

QUINLAN, J. R. Induction of decision trees. *Machine learning*, Springer, v. 1, n. 1, p. 81–106, 1986.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2016. Disponível em: <https://www.R-project.org/>.

RUSIECKI, A. Robust neural network for novelty detection on data streams. In: SPRINGER. *International Conference on Artificial Intelligence and Soft Computing*. [S.l.], 2012. p. 178–186.

SILVA, D. et al. Semi-supervised classification of characterized patterns for demand forecasting using smart electricity meters. In: *2011 International Conference on Electrical Machines and Systems*. [S.l.]: IEEE, 2011. p. 1–6.

SILVA, T. P. D. et al. A fuzzy multiclass novelty detector for data streams. In: *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.]: IEEE, 2018. (in press).

SILVA, T. P. da et al. A fuzzy variant for on demand data stream classification. In: *BRACIS 2017 ()*. [S.l.: s.n.], 2017.

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

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, Elsevier, v. 45, n. 4, p. 427–437, 2009.

SONG, X. et al. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 19, n. 5, 2007.

SPINOSA, E. J. et al. Novelty detection with application to data streams. *Intelligent Data Analysis*, IOS Press, v. 13, n. 3, p. 405–422, 2009.

SUYKENS, J. A.; VANDEWALLE, J. Least squares support vector machine classifiers. *Neural processing letters*, Springer, v. 9, n. 3, p. 293–300, 1999.

TAN, S. C.; TING, K. M.; LIU, T. F. Fast anomaly detection for streaming data. In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2011. v. 22, n. 1, p. 1511.

TARASSENKO, L. et al. Novelty detection for the identification of masses in mammograms. IET, 1995.

TAX, D. M.; DUIN, R. P. Support vector data description. *Machine learning*, Springer, v. 54, n. 1, p. 45–66, 2004.

TAX, D. M.; JUSZCZAK, P. Kernel whitening for one-class classification. *International Journal of Pattern Recognition and Artificial Intelligence*, World Scientific, v. 17, n. 03, p. 333–347, 2003.

TAYLOR, D. W.; CORNE, D. W. An investigation of the negative selection algorithm for fault detection in refrigeration systems. In: SPRINGER. *International Conference on Artificial Immune Systems*. [S.l.], 2003. p. 34–45.

TRAN, T. et al. Probabilistic inference over rfid streams in mobile environments. In: IEEE. *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. [S.l.], 2009. p. 1096–1107.

WANG, C.-H. Outlier identification and market segmentation using kernel-based clustering techniques. *Expert Systems with Applications*, Elsevier, v. 36, n. 2, p. 3744–3750, 2009.

WANG, H. et al. Mining concept-drifting data streams using ensemble classifiers. In: ACM. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2003. p. 226–235.

WANG, J.; NESKOVIC, P.; COOPER, L. N. Pattern classification via single spheres. In: SPRINGER. *International Conference on Discovery Science*. [S.l.], 2005. p. 241–252.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine learning*, Springer, v. 23, n. 1, p. 69–101, 1996.

WU, F.; WANG, T.; LEE, J. An online adaptive condition-based maintenance method for mechanical systems. *Mechanical Systems and Signal Processing*, Elsevier, v. 24, n. 8, p. 2985–2995, 2010.

WU, X.; LI, P.; HU, X. Learning from Concept Drifting Data Streams with Unlabeled Data. *Neurocomputing*, v. 92, p. 145–155, 2012.

XIONG, X.; CHAN, K. L.; TAN, K. L. Similarity-driven cluster merging method for unsupervised fuzzy clustering. In: AUAI PRESS. *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. [S.l.], 2004. p. 611–618.

YOGITA, Y.; TOSHNIWAL, D. Clustering techniques for streaming data - a survey. In: *2013 3rd IEEE International Advance Computing Conference (IACC)*. [S.l.]: IEEE, 2013. p. 951–956.

YU, Y. et al. Anomaly Intrusion Detection for Evolving Data Stream Based on Semi-supervised Learning. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.]: Springer Berlin Heidelberg, 2009. v. 5506 LNCS, p. 571–578.

ZHANG, J. et al. Novel fault class detection based on novelty detection methods. *Intelligent Computing in Signal Processing and Pattern Recognition*, Springer, p. 982–987, 2006.

ZHANG, P. et al. A framework for application-driven classification of data streams. *Neurocomputing*, v. 92, p. 170–182, 2012.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*. [S.l.]: ACM Press, 1996. p. 103–114.

ZHOU, A. et al. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, v. 15, n. 2, p. 181–214, 2008.

ZHU, L. et al. Study of Fuzzy Weighting Subspace Clustering for Streaming Data. *Journal of Computational Information Systems*, v. 10, n. 14, p. 6305–6314, 2014.