EIFuzzCND: UMA ESTRATÉGIA INCREMENTAL PARA CLASSIFICAÇÃO MULTI CLASSE E DETECÇÃO DE NOVIDADES EM FLUXOS DE DADOS

Mestrando : Lucas Ricardo Duarte Bruzzone
Orientadora: Prof. Dra. Heloisa de Arruda Camargo

São Carlos – SP
Outubro/2023

Introdução

Necessidade de um algoritmo que se adapte ao surgimento e mudança de conceitos de forma contínua.

- Cibersegurança
- Detecção de fraudes
- Controle de qualidade

Conceitos básicos

- Tarefa de classificação multi classe
- Técnicas de agrupamento
  - Agrupamento Fuzzy
    - Pertinência: Grau de pertencimento de um ponto de dados a um grupo
    - Tipicidade: Quão representativo um ponto é dentro de um grupo

Aprendizado de máquina em fluxo de dados

Características	Aprendizado de máquina clássico	Aprendizado de máquina em FD
Número de passes	Múltiplos	Único
Tempo	Ilimitado	Tempo real
Memória	Ilimitada	Limitada
Resultado	Preciso	Aproximado
Número de conceitos	Único	Múltiplos

Tabela 2.1: Comparações entre aprendizado de máquina tradicional e em fluxo de dados (NGUYEN; WOON; NG, 2015)

Estrutura de micro-grupo

- Supervised Possibilistic Fuzzy Micro-Cluster (SPFMiCs)

Supervised Possibilistic Fuzzy Micro-Cluster

Vetor de atributos que sumariza os dados

- M^e: Soma linear dos graus de pertinência dos exemplos elevados a alpha(α)
- T^e : Soma linear dos graus de tipicidade t dos exemplos elevados a theta(θ)
- $\overline{\mathrm{CF1}_u^e}$:soma linear dos exemplos ponderados por suas pertinências
- $\overline{\mathrm{CF1}_T^e}$: soma linear dos exemplos ponderados por suas tipicidades
- SSD^e: soma das distâncias dos exemplos para o protótipo do grupo, elevadas a α e ponderadas pelas pertinências de cada exemplo.
- **N** : Número de exemplos pertencentes ao grupos
- **T** : Tempo de chegada do exemplo mais atual
- **class_id** : Rótulo da classe

Detecção de Novidade em Fluxo de Dados

- Detecção de novidades
  - Mudança de conceito
  - Evolução de conceito
- Framework
  - Offline
  - Online
- Latência de rótulos
  - Intermediária
  - Extrema

Revisão Bibliográfica

Título do Estudo	Autores (Ano)	Latência	Abordagem "Fuzzy"
Streaming Autoencoder (SA)	Dong (2018) (DONG; JAPKOWICZ, 2018)	Intermediária	Não
Enhanced Classifier for Data Streams with Novel Class Miner (ECSMiner)	Masud et al. (2010) (MASUD et al., 2010)	Intermediária	Não
Multiclass learning algorithm for DS Active Learning (MINAS-AL)	de Souza et al. (2016) (FARIA et al., 2016b)	Intermediária	Não
Grid-based Clustering for Concept-drifting Data Streams (GC3)	Sethi et al. (2016) (SETHI; KANTARDZIC; HU, 2016)	Intermediária	Não
WiSARD-based Change Detection System (WCDS)	Cardoso et al. (2017) (CARDOSO; FRANÇA; GAMA, 2017)	Extrema	Não
MINAS (Multi-Instance Learning Algorithm from Noisy, Unlabeled Data Streams)	de Souza et al. (2016) (FARIA et al., 2016b)	Extrema	Não
SENNE (Stream-based Evolving Nearest Neighbor)	Cai et al. (2019) (CAI et al., 2019)	Extrema	Não
CluStream	Aggarwal et al. (2003) (AGGARWAL et al., 2003)	Extrema	Não
FuzzND	da Silva et al. (2018) (SILVA et al., 2018)	Extrema	Sim
PFuzzND	da Silva et al. (2018) (SILVA; CAMARGO, 2020)	Extrema	Sim
eClass	Angelov et al. (2008) (ANGELOV; ZHOU, 2008)	Intermediária	Sim
EFuzzCND	Cristiani et al. (2021) (CRISTIANI; CAMARGO, 2021)	Intermediária	Sim

Tabela 3.2: Estudos Selecionados para Detecção de Novidades em Fluxos de Dados

- Estudos selecionados a partir do processo de revisão sistemática:
  - Entendimento do estado da arte
  - Identificar limitações e possíveis melhorias nos trabalhos propostos

Principais limitações

- Capacidade de se adaptar rapidamente a mudança de conceito em cenários de latência extrema
- Dependência de grande quantidade de dados rotulados em cenários de latência intermediária
- EFuzzCND
  - Assume que todos os rótulos estarão disponíveis
  - Modelo construído na fase offline é atualizado somente no cenário do latência intermediária e após um tempo definido.

Enhanced Incremental Fuzzy Clustering for Novelty Detection

- Atualização Incremental do modelo de classes conhecidas (MCC)
  - Permite que o modelo se ajuste mais rapidamente a mudança de conceito
  - Útil principalmente em cenários de latência extrema onde não se tem a presença do rótulo verdadeiro no fluxo de dados
- Percentual de presença de rótulos verdadeiros
  - Considera uma quantidade limitada de rótulos verdadeiros em cenários de latência intermediária
- Implementação da Matriz de Confusão Incremental (MCI)
  - Melhor qualidade de informação em relação ao desempenho do algoritmo

Algoritmo EIFuzzCND

Fase Offline do EIFuzzCND

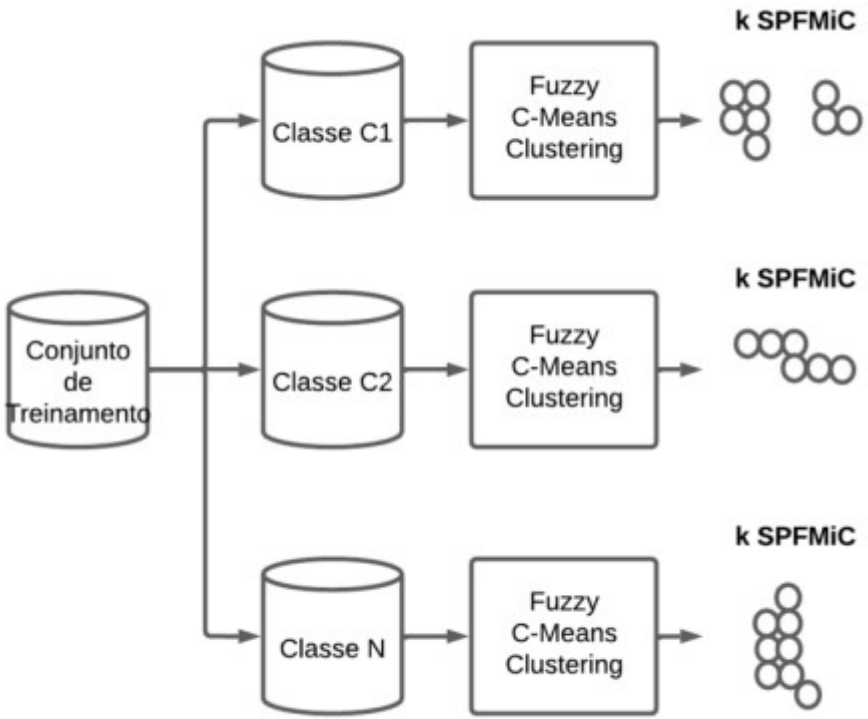


Figura 4.1: Fase Offline - Algoritmo do modelo formado por EIFuzzCND

Algorithm 1 Algoritmo da Fase Offline

Require: *trainSet*, fuzzification, K, minWeight = 0, α, θ

Ensure: MCC - Initial State

1: $examplesByClass \leftarrow$ separateByClasses($trainSet$)
2: **for** each class **do**
3: $clusters \leftarrow$ FuzzyC-Means($examplesByClass$, K, fuzzification)
4: $SPFMiCs \leftarrow$ SummaryClusters($examplesByClass$, $clusters$, class, α, θ, $minWeight$)
5: $MCC \leftarrow MCC \cup SPFMiCs$
6: **end for**
7: **return** MCC

Fase Online do EIFuzzCND

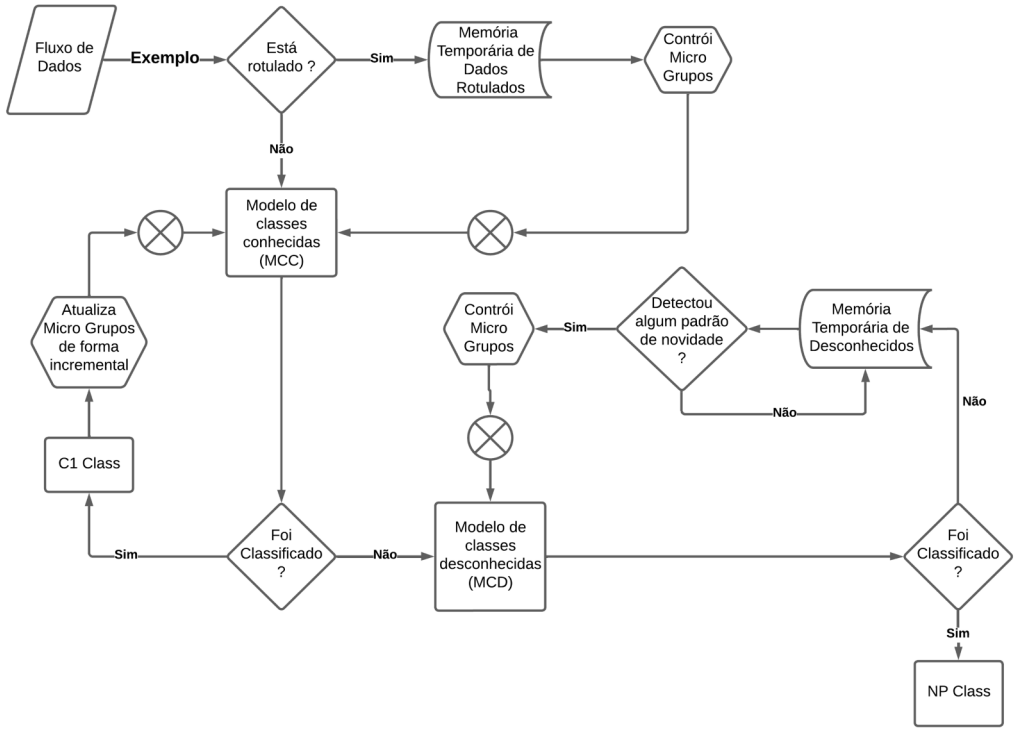


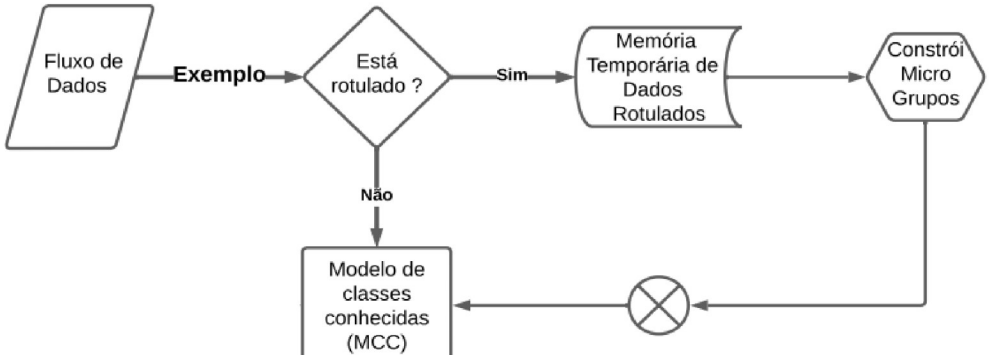
Figura 4.2: Fase Online - Algoritmo EIFuzzND

Algorithm 2 Algoritmo da Fase Online

Require: *dataStream*, *MCC*, *L*, *tChunk*, *T*, *phi*, *percentLabeled*
Ensure: *LabeledData*

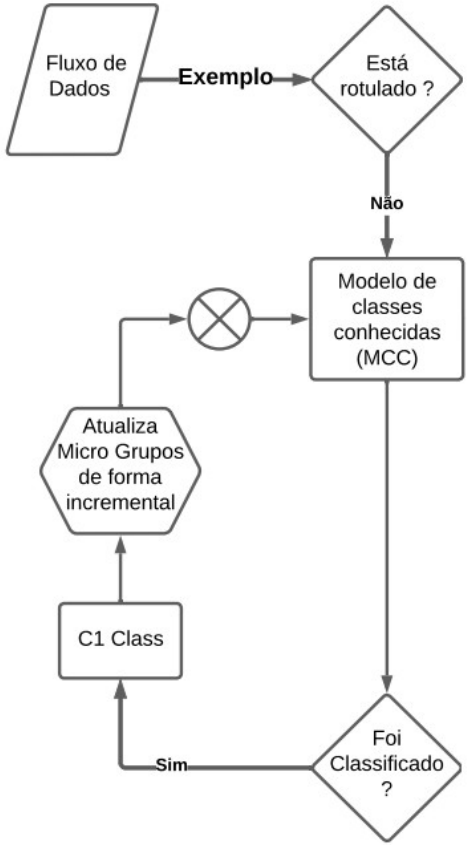
```
1:  unknownInstances ← empty list
2:  labeledInstances ← empty list
3:  for each instance inst in dataStream do
4:      label ← MCC.predict(inst)
5:      if label = −1 then
6:          label ← MCD.predict(inst)
7:          if label = −1 then
8:              add inst to unknownInstances
9:              if size of unknownInstances ≥ T then
10:                 unknownInstances ← multiClassNoveltyDetection(unknownInstances)
11:             end if
12:         end if
13:     end if
14:     if currentTime ≥ latencyDelay then
15:         if Math.random() < percentLabeled OR labeledMem is empty then
16:             labeledMem ← addToList(labeledExample)
17:         end if
18:         if size(labeledMem) ≥ tChunk then
19:             MCC ← trainModel(labeledMem, tempo)
20:             clear(labeledMem)
21:         end if
22:     end if
23: end for
24: return testSetLabeled
```

Presença dos rótulos verdadeiros




```
14:  if currentTime ≥ latencyDelay then
15:    if Math.random() < percentLabeled OR labeledMem is empty then
16:      labeledMem ← addToList(labeledExample)
17:    end if
18:    if size(labeledMem) ≥ tChunk then
19:      MCC ← trainModel(labeledMem, tempo)
20:      clear(labeledMem)
21:    end if
```

Atualização Incremental

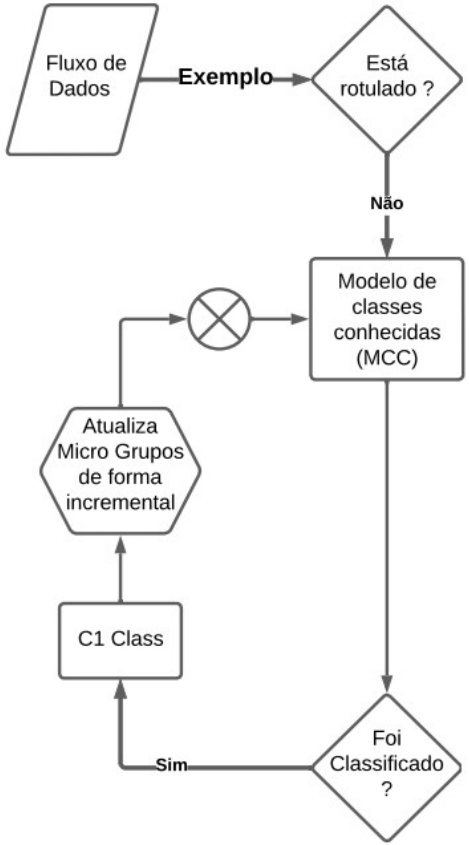


3: **for** each instance *inst* in *dataStream* **do**
4: $label \leftarrow MCC.predict(inst)$

Algorithm 3 Algoritmo da Atualização Incremental

Require: MCC, exemplo, classes
1: **for** each SPFMiC in MCC **do**
2: *distancia* ← calculaDistanciaEuclidiana(SPFMiC.centroide, exemplo)
3: **if** distancia <= raio com peso do objeto SPFMiC **then**
4: *tipicidade* ← calculaTipicidade(objetoSPFMiC.centroide, exemplo)
5: *pertinencia* ← calculaPertinencia(objetoSPFMiC.centroide, exemplo)
6: *listaObjetosSelecionados*.append(objetoSPFMiC)
7: **end if**
8: **end for**
9: **if** *listaObjetosSelecionados* is empty **then**
10: **return** -1
11: **else**
12: *ObjetoSelecionado* ← seleciona(listaObjetosSelecionados, tipicidade)
13: *SPFMiCAtualizado* ← atualizaSPFMiC(objetoSelecionado, exemplo, Pertinencia, Tipicidade)
14: **return** *MCC*
15: **end if**

Atualização SPFMiC



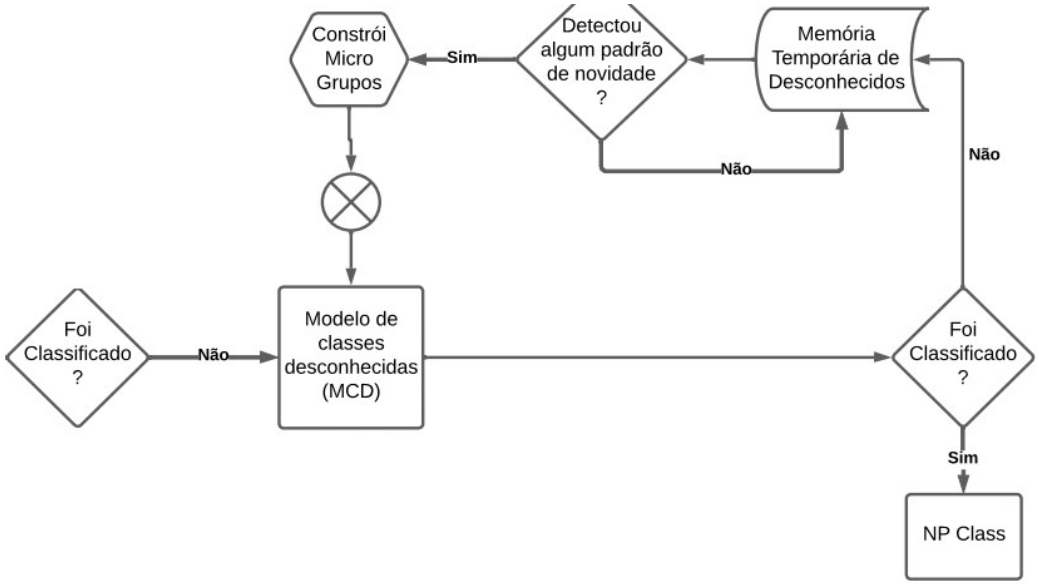
Algorithm 4 Método atualizaSPFMiC

1: **Input:** $exemplo$, $pertinencia$, $tipicidade$
2: $dist \leftarrow$ calculaDistanciaEuclidiana(exemplo, centroide)
3: $N \leftarrow N + 1$
4: $Me \leftarrow Me + pertinencia^{\alpha}$
5: $Te \leftarrow Te + tipicidade^{\theta}$
6: $SSDe \leftarrow SSDe + dist^2 \times pertinencia$
7: $atualizaCF1pertinencia(exemplo(i), pertinencia)$
8: $atualizaCF1tipicidades(exemplo(i), tipicidades)$
9: atualizaCentroide()

Algorithm 5 Método atualizaCentroide

1: $nAtributos \leftarrow$ length of CF1pertinencias
2: $centroide \leftarrow$ array of size $nAtributos$
3: **for** $i \leftarrow 0$ **to** $nAtributos - 1$ **do**
4: $\quad centroide[i] \leftarrow \frac{\alpha \times CF1pertinencias[i] + \theta \times CF1tipicidades[i]}{\alpha \times Te + \theta \times Me}$
5: **end for**

Detecção de novidades




```
5:   if label = −1 then
6:     label ← MCD.predict(inst)
7:     if label = −1 then
8:       add inst to unknownInstances
9:       if size of unknownInstances ≥ T then
10:        unknownInstances ← multiClassNoveltyDetection(unknownInstances)
11:      end if
12:    end if
13:  end if
```

Experimentos e métricas de avaliação

Experimentos e métricas de avaliação

Dataset	Instâncias	Instâncias(Offline)	Atributos	Classes	Classes(Offline)	Sintético
MOA3	100,000	10,000	4	4	2	Sim
RBF	48,588	2,000	2	5	3	Sim
SynEDC	400,000	30,000	54	20	6	Sim
KDD	490,000	48,791	34	5	4	Não
CoverType	581,000	47,045	54	7	7	Não

Tabela 5.1: Características dos datasets utilizados na avaliação dos algoritmos.

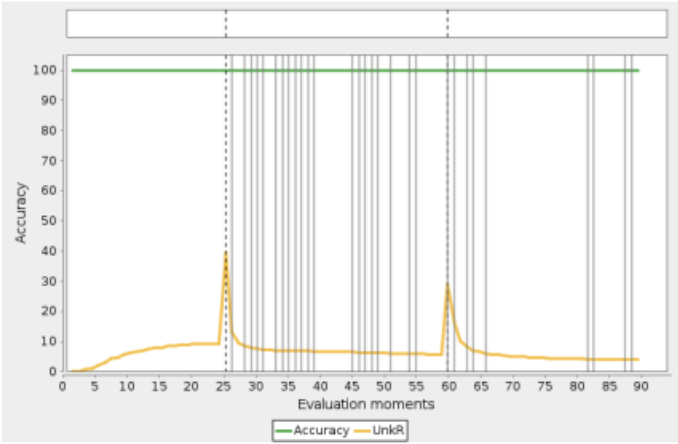
- Algoritmos selecionados
  - EFuzzCND
  - ECSMiner
  - MINAS
- Métricas de avaliação

Resultados

Latência Extrema

MOA3

(a) EIFuzzCND



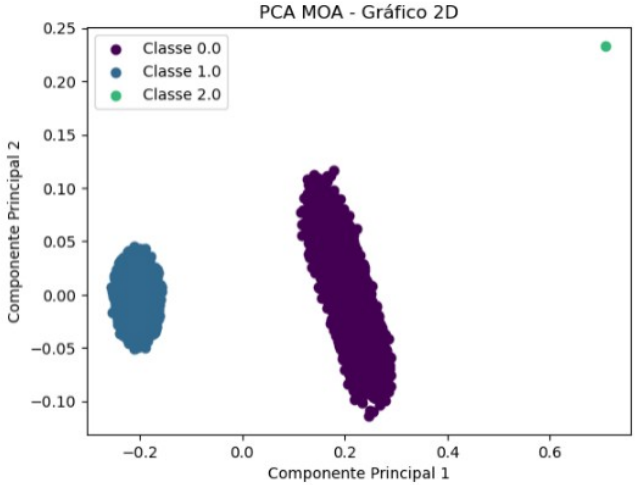
(b) EFuzzCND

Classes	0.0	1.0	UnkMem	2.0	3.0	NP1	NP2	NP3	NP4
0.0	26228	96	14	0	0	0	0	2286	15
1.0	110	28414	11	0	0	0	0	0	0
UnkMem	0	0	0	0	0	0	0	0	0
2.0	0	0	0	0	0	15914	0	0	0
3.0	0	0	0	0	0	0	6558	0	0

Tabela 6.2: MCI representando o último momento de avaliação no dataset MOA3

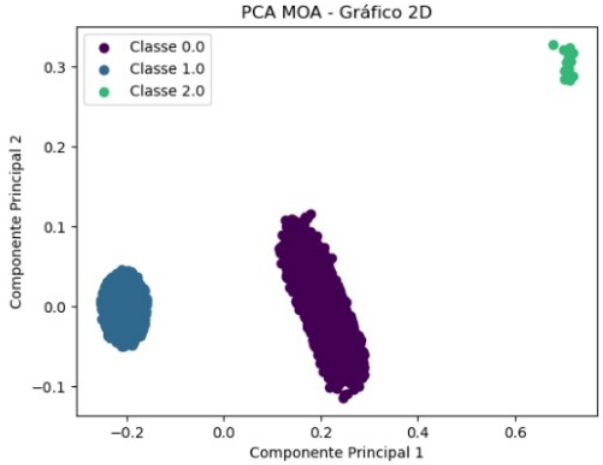
MOA3

Classes	0.0	1.0	UnkMem	2.0
0.0	12565	0	13	0
1.0	0	12410	11	0
UnkMem	0	0	0	0
2.0	0	0	1	0



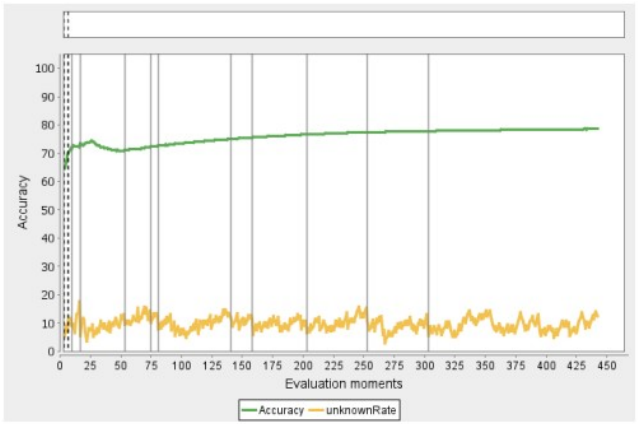
(a) Momento de avaliação 25.000

Classes	0.0	1.0	UnkMem	2.0	NP1
0.0	12595	0	13	0	0
1.0	0	12431	11	0	0
UnkMem	0	0	0	0	0
2.0	0	0	0	0	16



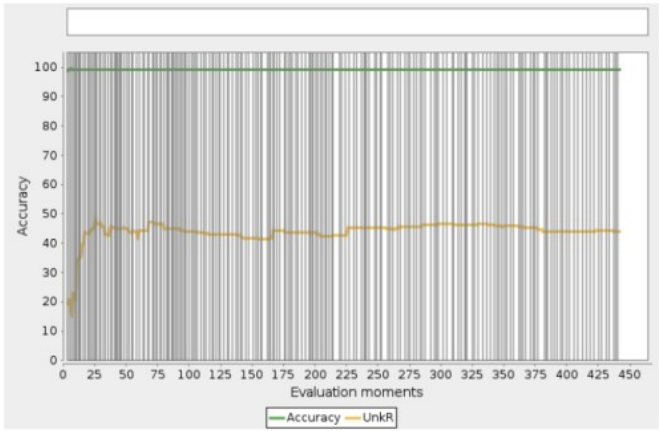
(b) Momento de avaliação 25.066

KDD99



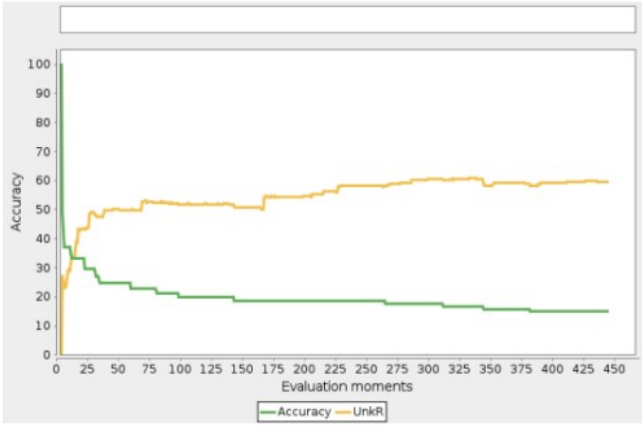
(a) EIFuzzCND

- Maior valor de acurácia
- Taxa de desconhecidos menor e estável



(b) EFuzzCND

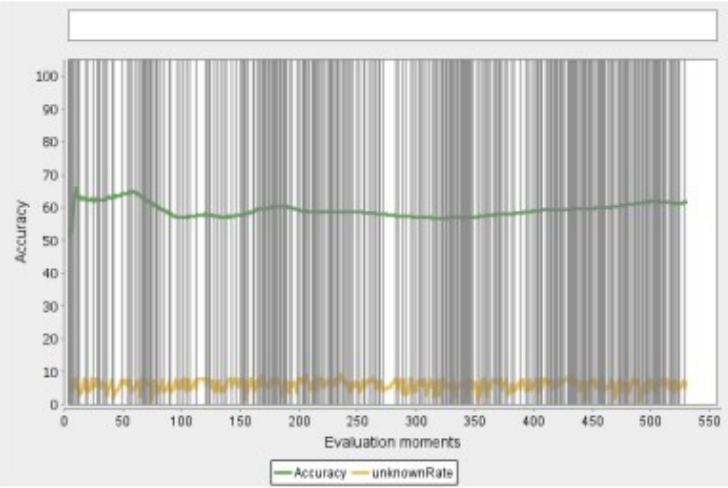
- Alta taxa de desconhecidos
- Maior quantidade de detecção de padrões de novidade sem necessidade



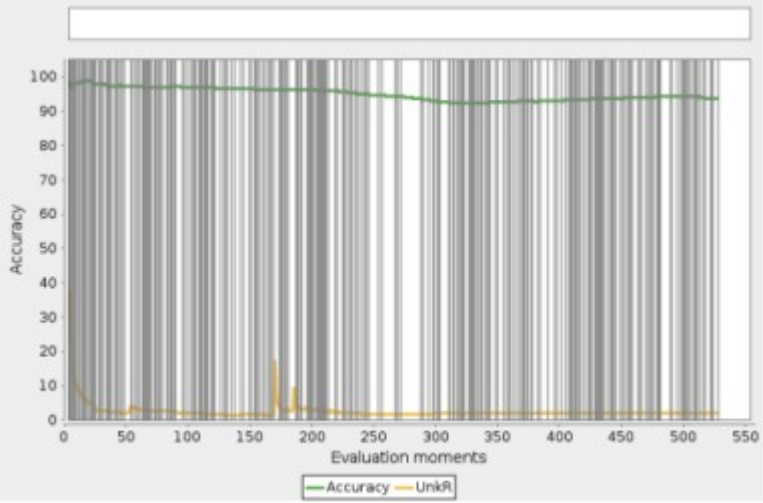
(c) MINAS

- Alta taxa de desconhecidos
- Baixo valor de acurácia

CoverType



(a) EIFuzzCND



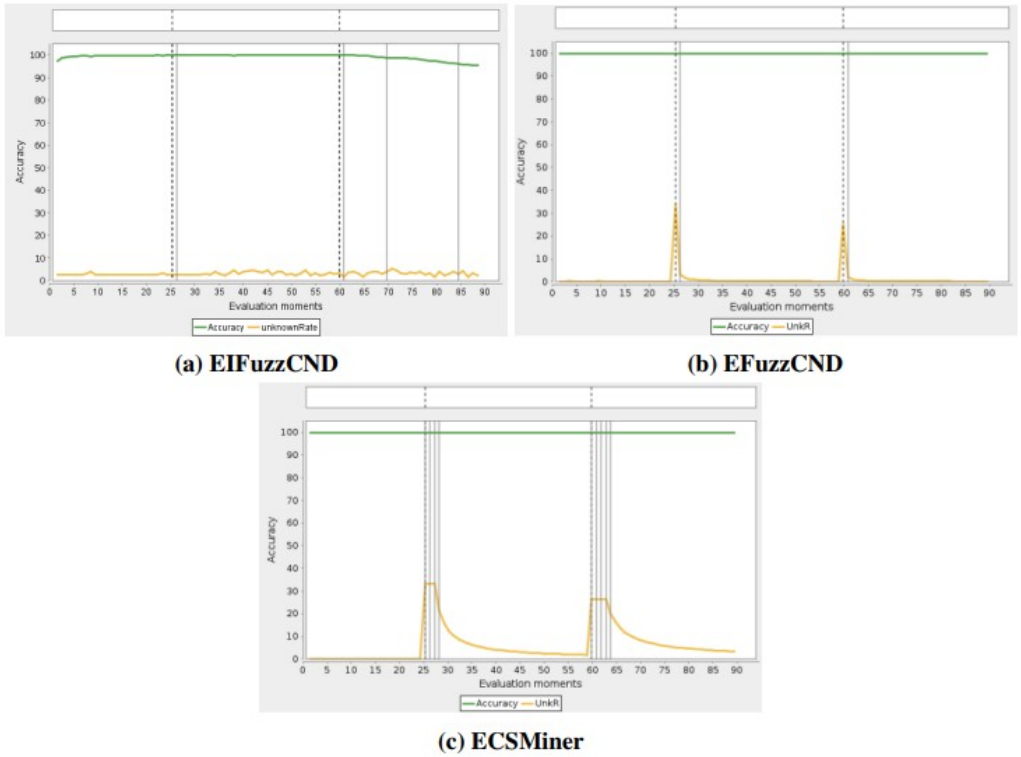
(b) EFuzzCND

Em conjuntos de dados complexos onde existe a sobreposição de classes, a inclusão de novas instâncias ao longo do tempo pode resultar em uma dificuldade crescente em separar claramente essas classes. Uma abordagem fuzzy comum pode não ser tão afetada por essa sobreposição

Resultados

Latência Intermediária

MOA3



(a) EIFuzzCND

(b) EFuzzCND

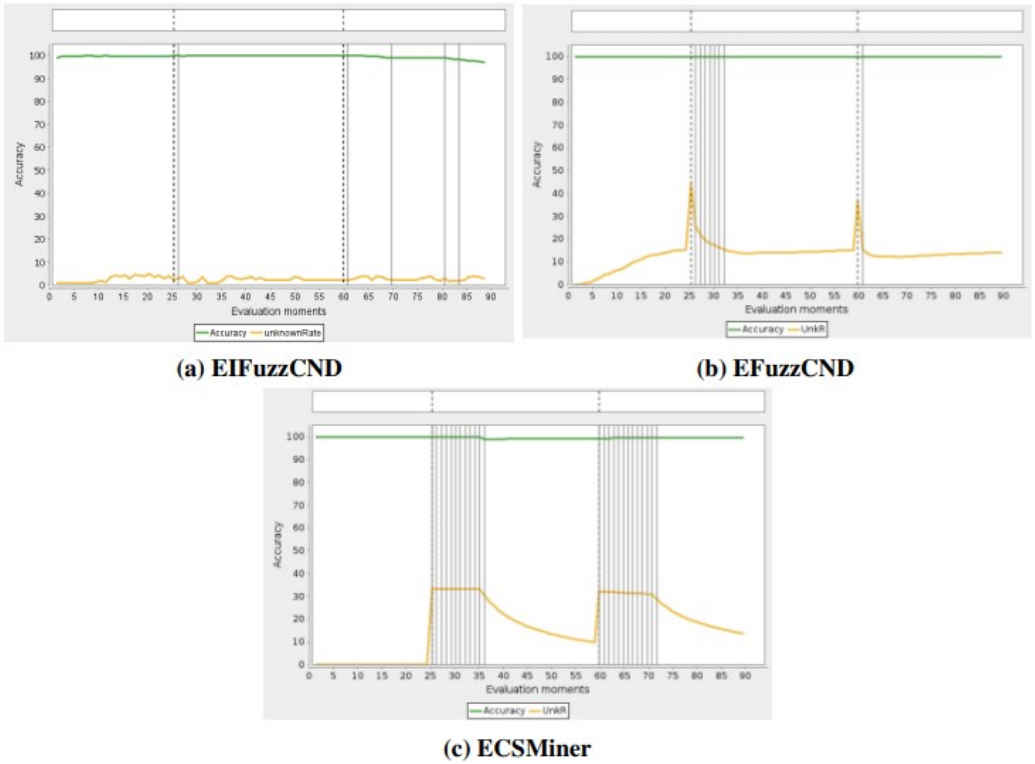
(c) ECSMiner

Figura 6.13: Acurácia e taxa de desconhecidos dos algoritmos, considerando latência intermediária de 2000, para o conjunto de dados MOA3.

Em cenários de menor valor de latência o algoritmo EFuzzCND apresentou melhor desempenho.

Por não ser incremental e ser depende dos rótulos verdadeiros para a classificação, o algoritmo EfuzzCND pode manter uma acurácia mais alta quando confrontado com novidades.

MOA3



(a) EIFuzzCND

(b) EFuzzCND

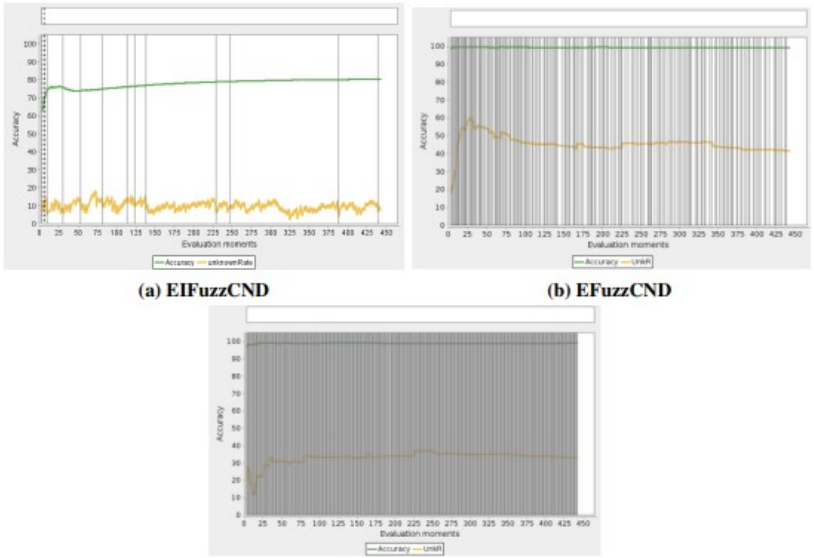
(c) ECSMiner

Figura 6.15: Acurácia e taxa de desconhecidos dos algoritmos, considerando latência intermediária de 10000, para o conjunto de dados MOA3.

Porém quando o valor de latência intermediária aumenta o desempenho do algoritmo EFuzzCND cai enquando o algoritmo EIFuzzCND mantém um comportamento estável e robusto.

Logo resultados obtidos indicam que, à medida que o valor da latência aumenta, o EIFuzzCND se destaca como uma solução mais eficiente e confiável em comparação com o EFuzzCND.

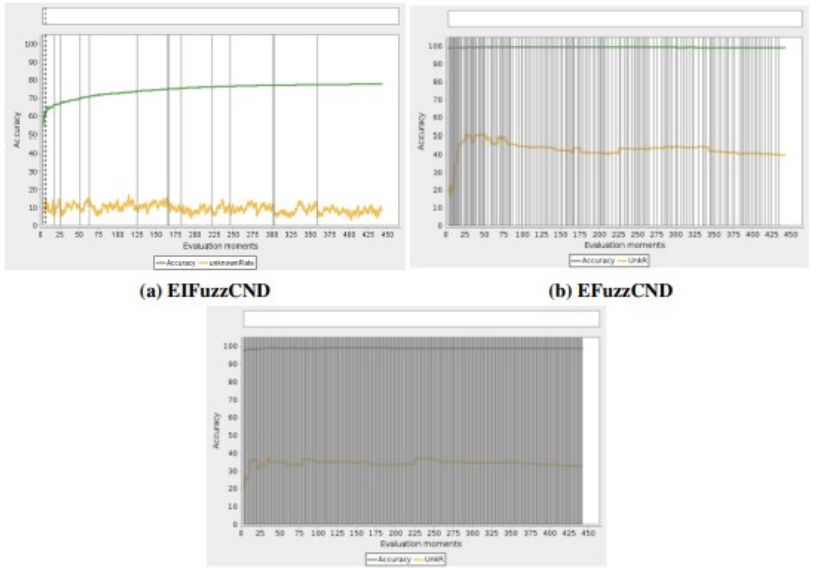
KDD99



(a) EIFuzzCND (b) EFuzzCND

(c) ECSMiner

Figura 6.19: Acurácia e taxa de desconhecidos dos algoritmos, considerando latência intermediária de 2000, para o conjunto de dados KDD.

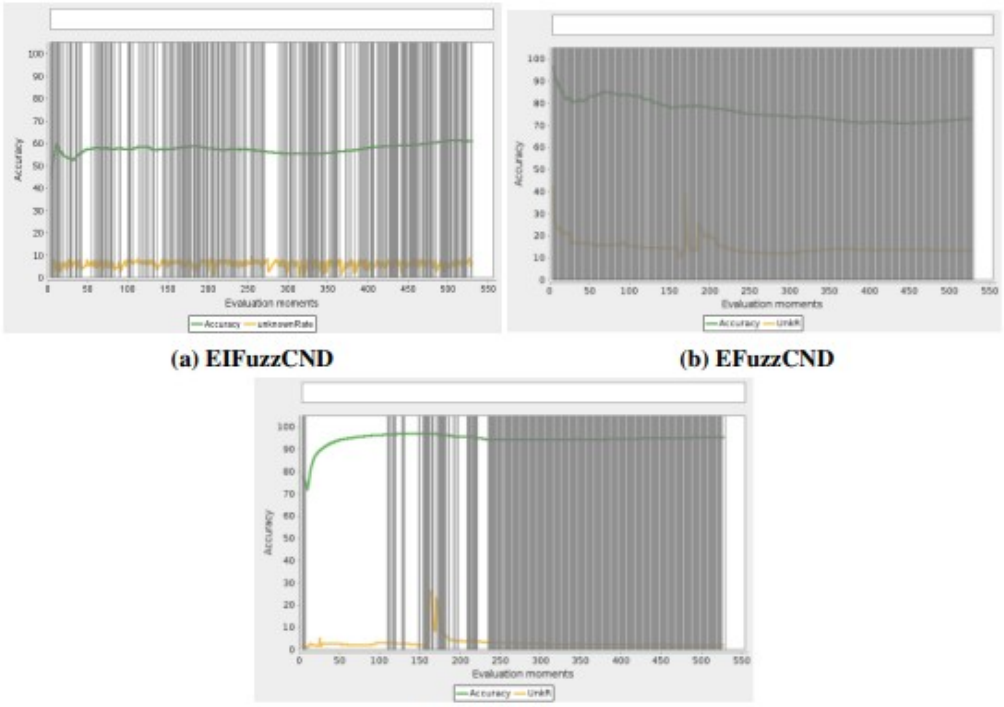


(a) EIFuzzCND (b) EFuzzCND

(c) ECSMiner

Figura 6.21: Acurácia e taxa de desconhecidos dos algoritmos, considerando latência intermediária de 10000, para o conjunto de dados KDD.

CoverType



(a) EIFuzzCND

(b) EFuzzCND

(c) ECSMiner

Figura 6.22: Acurácia e taxa de desconhecidos dos algoritmos, considerando latência intermediária de 2000, para o conjunto de dados CoverType.

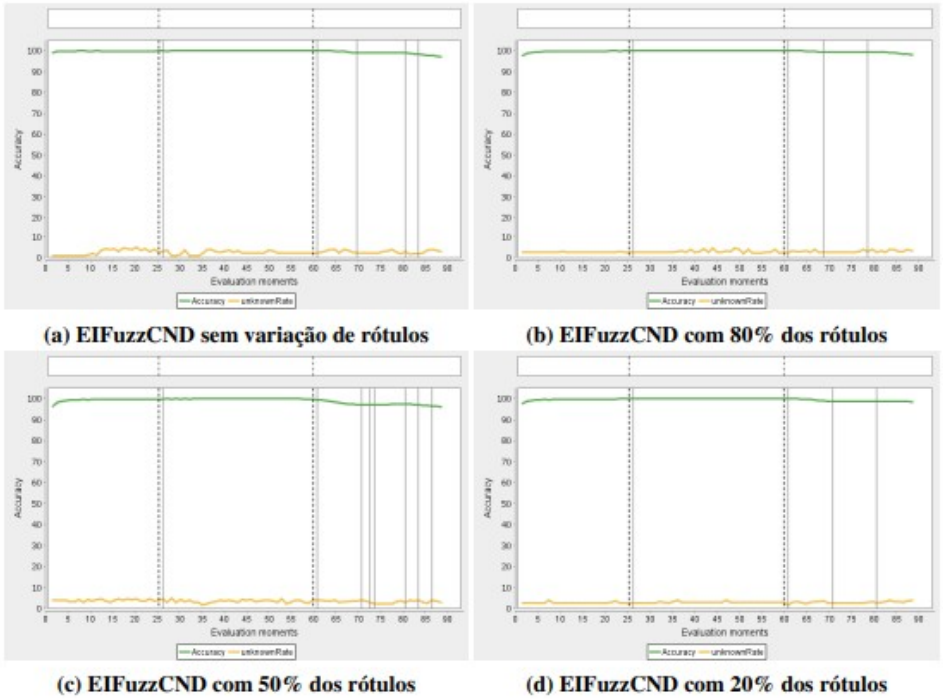
O algoritmo ECSMiner obteve resultados consistentes e satisfatórios. Essa performance pode ser atribuída à utilização de uma abordagem de clusterização normal pelo ECSMiner, em contraste com os algoritmos EIFuzzCND e EFuzzCND, que utilizam clusterização fuzzy.

A clusterização normal adotada pelo ECSMiner pode ter contribuído para uma melhor adaptação aos padrões específicos presentes no dataset CoverType.

Resultados

Latência Intermediária com variação de rótulos

MOA3



(a) EIFuzzCND sem variação de rótulos

(b) EIFuzzCND com 80% dos rótulos

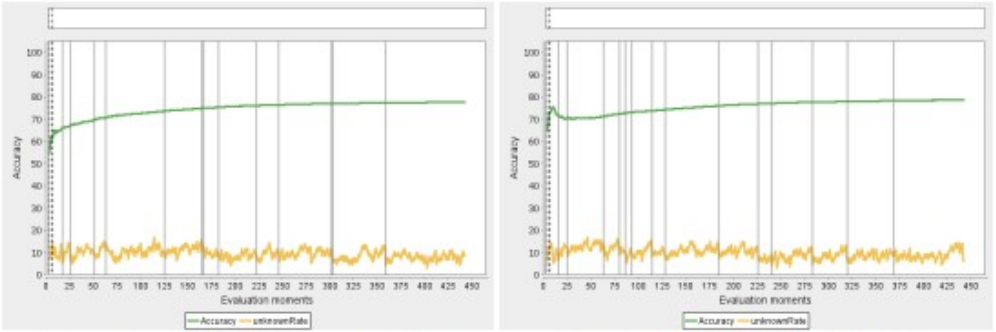
(c) EIFuzzCND com 50% dos rótulos

(d) EIFuzzCND com 20% dos rótulos

Figura 6.28: Acurácia e taxa de desconhecidos do EIFuzzCND, considerando latência intermediária de 10000 com variação de rótulos, para o conjunto de dados MOA.

Pela sua característica incremental o algoritmo conseguiu lidar de maneira eficiente com as variações nos valores percentuais, mantendo um bom desempenho na detecção de novidades.

KDD99



(a) EIFuzzCND sem variação de rótulos

(b) EIFuzzCND com 80% dos rótulos

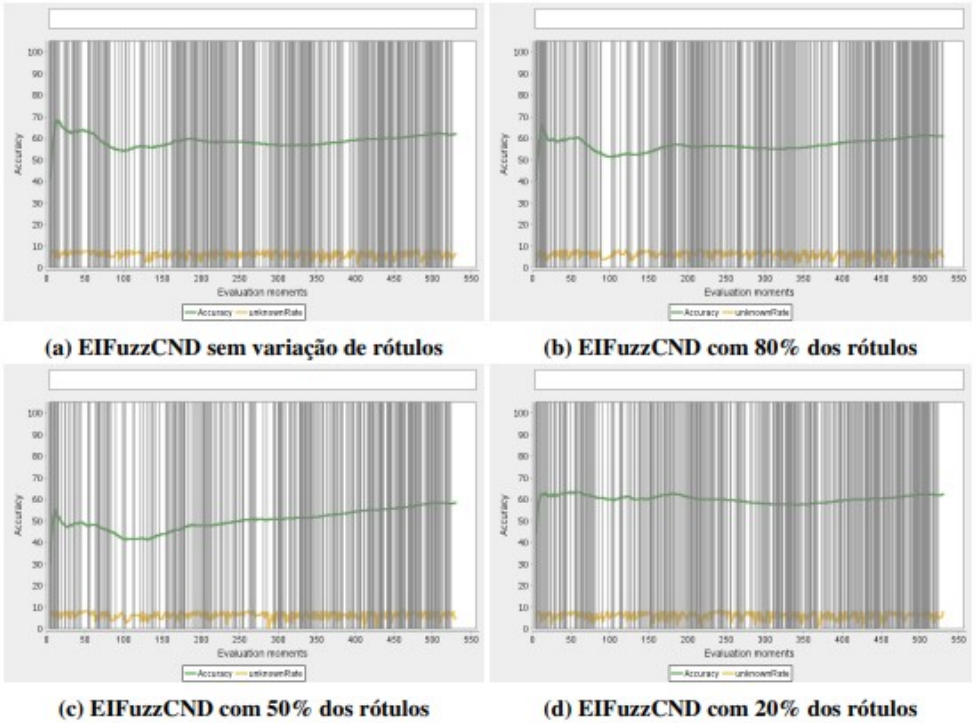
(c) EIFuzzCND com 50% dos rótulos

(d) EIFuzzCND com 20% dos rótulos

Figura 6.30: Acurácia e taxa de desconhecidos do EIFuzzCND, considerando latência intermediária de 10000 com variação de rótulos, para o conjunto de dados KDD.

Conseguiu lidar de forma mais eficiente com as variações nos rótulos, mantendo um desempenho consistente na classificação correta das instâncias.

CoverType



(a) EIFuzzCND sem variação de rótulos

(b) EIFuzzCND com 80% dos rótulos

(c) EIFuzzCND com 50% dos rótulos

(d) EIFuzzCND com 20% dos rótulos

Figura 6.31: Acurácia e taxa de desconhecidos do EIFuzzCND, considerando latência intermediária de 10000 com variação de rótulos, para o conjunto de dados CoverType.

Uma explicação possível é que as características intrínsecas desse conjunto de dados não se adéquem bem à abordagem de clusterização fuzzy utilizada pelo algoritmo. Pode haver uma falta de representatividade dos padrões de novidade no espaço de atributos considerado ou uma dificuldade em definir limites claros entre as diferentes classes e as instâncias desconhecidas.

Resultados

- Latência Extrema
  - Taxa de acurácia: Superou o MINAS e o EFuzzCND mantendo uma taxa de classificação correta superior
  - Taxa de desconhecidos: Manteve uma baixa e constante taxa de desconhecidos
- Latência Intermediária
  - Detecção de novidades: Eficaz em identificar padrões novos no fluxo de dados mesmo com um atraso moderado.
  - Estabilidade: Mostrou estabilidade no desempenho, indicando sua capacidade de lidar com diferentes níveis de disponibilidade de rótulos

Conclusão

- Implementação da atualização incremental permitiu que o modelo se adaptasse de forma mais rápida a mudança de conceito, principalmente em cenários de latência extrema
- Variação da presença de rótulos no cenário de latência intermediária permitiu avaliar a robustez e estabilidade do algoritmo
- Implementação da Matriz de Confusão Incremental permitiu melhor visualização e análise do comportamento do algoritmo

Trabalhos Futuros

- Melhoria no tratamento de classes desbalanceadas
- Criação de métricas específicas para detecção de novidades como sensibilidade a taxa de falsos positivos e adaptação a mudança de conceito
- Comparação e avaliação com outros conjuntos de dados e outros algoritmos
- Avaliação em cenários de latência nula
- Exploração de estratégias de adaptação de parâmetros como o índice de similaridade (φ).