Aprendizagem de Regras de Decisão Utilizando Técnicas Adaptativas: Experimentos Preliminares

F. K. O. Takatuzi e R. L. Stange

***Resumo*—Uma das formas de aprendizagem de máquina é a indução de regras a partir de exemplos, particularmente a extração de regras de classificação. A maioria dos trabalhos nesta área, se concentram na geração de regras na forma intermediária de árvores de decisão, para posteriormente converter cada caminho da árvore em uma regra. Este artigo propõe um algoritmo de geração de regras que utiliza as árvores de decisão adaptativas como estrutura intermediária. O objetivo é que tal algoritmo, seja um candidato credível para uso em problemas de aprendizagem incremental de regras de classificação. O algoritmo foi testado e comparado com algoritmos que utilizam diferentes abordagens, tais como aprendizagem estatística, aprendizagem de regras e indução de árvores de decisão. Os experimentos apresentados, sugerem que o algoritmo adaptativo para geração regras possui um resultado próximo a alguns algoritmos clássicos. Para um primeiro experimento, a eficiência do algoritmo foi satisfatória, visto que o método de generalização do algoritmo ainda precisa de ajustes, já que este utiliza apenas a frequência relativa dos atributos para escolha de atributos que compõem cada regra.**

***Index Terms*—Tecnologia adaptativa, aprendizagem incremental, indução de regras, classificação.**

I. Introdução

A Aprendizagem de Máquina é uma área da Inteligência Artificial que estuda a capacidade que os sistemas computacionais possuem de aprender e modificar seu comportamento, através da experiência adquirida a fim de melhorar seu desempenho em execuções posteriores [1]. Existem diversos métodos que são utilizados para a busca de soluções de problemas em aprendizagem de máquina, tais como o Aprendizado Bayesiano, as Redes Neurais Artificiais (RNA), os métodos de indução de árvores de decisão e a aprendizagem de regras de decisão. Os métodos de aprendizagem baseados em regras são partes integrantes de sistemas em inteligência artificial, porém, a adoção desses métodos em problemas de classificação ainda é modesta [2]. Uma visão geral dos métodos baseados em regras se concentra em classes de regras do tipo "*Se..então*". Para [3] o conjunto de regras do tipo "*Se..então*"é uma das formas mais expressivas e legíveis para representar hipóteses em problemas de aprendizagem. As regras de decisão são utilizadas para representar o conhecimento obtido a partir dos dados em aplicações diversas [4] [5] [6].

De forma geral, a maioria dos algoritmos são não incrementais, ou seja, não possuem a capacidade de incluir novos exemplos de como instâncias particulares do problema são resolvidas ao longo de sua execução. A necessidade de buscar soluções alternativas de aprendizagem incremental é de grande relevância [7] [8] [9], pois existem diversas aplicações em que novas informações vão surgindo ou se alterando ao longo do tempo, formando assim um cenário em que a utilização de algoritmos incrementais é mais conveniente [10]. Em soluções não incrementais, sempre que uma nova informação é disponibilizada, o processo de aprendizagem é repetido e a estrutura que representa o conhecimento é reconstruída. Em árvores de decisão por exemplo, a reconstrução da árvore implica em um custo computacional elevado [11]. Dessa forma, a utilização de métodos não incrementais se torna ineficiente, uma vez que as informações repassadas ao sistema de aprendizagem estão em constante modificação. Neste caso, torna-se necessário implementar processos dinâmicos baseados na aprendizagem incremental, de maneira que a base de conhecimento possa ser atualizada sem que todo o processo de aprendizagem necessite ser repetido.

Um conceito que tem sido adotado na solução de problemas com características dinâmicas é a adaptatividade. De acordo com Neto (2011) a adaptatividade refere-se a capacidade que os sistemas guiados por regras possuem de alterar em seu próprio comportamento, em função de seu comportamento atual e de estímulos de entrada.

Neste trabalho, é proposto um algoritmo para aprendizagem incremental de regras de decisão a partir de exemplos. O algoritmo foi capaz de gerar, a partir de um conjunto de exemplos de treinamento, uma árvore de decisão como resultado, onde cada caminho da árvore é representado por um cojunto de regras. Tal árvore, possui características adaptativas que permitem que sua estrutura interna seja alterada durante o processo de aprendizagem e classificação, possibilitando uma nova forma de aprendizagem incremental.

II. Trabalhos Relacionados

Desde [12], a tecnologia adaptativa têm sido utilizada como uma alternativa para resolução de problemas complexos, principalmente pela capacidade de expressão dos dipositivos adaptativos que, por meio da adição da adaptatividade, permite tornar dispositivos convencionais, tais como autômatos finitos e árvores de decisão, mais expressivos [13]. Outra característica atribuída aos dispositivos adaptativos é que, por serem baseados em dispositivos guiados por regras, podem proporcionar uma melhor interpretação pelos especialistas do domínio. Essa é uma das desvantagens de muitas técnicas de modelagem estatística, comumente usadas em aprendizado de máquina, o modelo resultante é difícil de interpretar [14]. Estudos preliminaries, apresentados em [15], mostram um caso particular da utilização da adaptatividade em aprendizagem, onde as tabelas de decisão adaptativas são utilizadas para representar o modelo de aprendizagem. Este trabalho mostra que

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: kenjitakatuzi@gmail.com, rlgomes@utfpr.edu.br.

este formalismo permite representar de maneira satisfatória um problema de aprendizagem de máquina, bem como possibilitar a identificação de melhorias para o processo de aprendizagem, pois o formato das regras facilita a interpretação do modelo de aprendizagem pelo projetista ou especialista.

[10], propuseram um algoritmo de indução de árvores de decisão utilizando técnicas adaptativas, que combina estratégias sintáticas e estatísticas, chamado *AdapTree*. Neste caso, não parece ser vantajoso considerar que os dispositivos adaptativos apenas melhoram a expressividade e compreensibilidade da solução, comparados a dispositivos não adaptativos. Assim, objetivo é melhorar o entendimento das regras, mas também cumprir exigências de eficiência.

Em [16] foi apresentado um método híbrido para aprender regras de classificação baseado na estratégia de cobertura sequencial, cujos métodos aplicam uma estratégia de decomposição do problema, na qual a tarefa de encontrar uma base de regras completa é reduzida a uma sequência de subproblemas, onde para cada subproblema a solução é uma única regra. Além disso, foi incorporado ao método a capacidade de automodificação do conjunto de regras, que permite inspecionar iterativamente o conhecimento extraído, podendo substituir uma ou mais regras para simplificar o conhecimento ou melhorar a capacidade de previsão do conjunto.

III. Referencial Teórico

A. Aprendizagem de regras de decisão

Uma visão geral dos métodos baseados em regras para representação do conhecimento e aprendizagem se concentra em classes de regras do tipo "*Se..então*". Para [3] o conjunto de regras deste tipo é um das formas mais expressivas e legíveis para representar hipóteses em aprendizagem de máquina. A forma geral de uma regra "*Se..então*"é:

`P` $\rightarrow$ `Q` ou `Se P então Q`, onde:

- `P` é uma proposição que pode conter um conjunto arbitrário de n pares de atributos - valor, $P = \text{condition}_1 \wedge \ldots \wedge \text{condition}_n$ (n é o tamanho da regra).
- `Q` é o valor do atributo categórico da classe.

Segundo [3], uma forma de aprender um conjunto de regras é construir uma árvore de decisão e converter o resultado para um conjunto de regras. A árvore de decisão pode ser mapeada para um conjunto de regras, transformando cada caminho da raiz até cada nó folha em uma regra. Outra maneira é extrair as regras diretamente do conjunto de treinamento, sem utilizar uma estrutura intermediária. Em resumo, as formas de aprender um conjunto de regras são:

- **Método 1:** Construir uma árvore de decisão e fazer a conversão para um conjunto de regras. Dado um conjunto de treinamento P, os passos para a contrução de uma árvore de decisão são [17]
  1) Se P contém um ou mais exemplos, todos pertencentes à mesma classe c_i então a árvore de decisão para P é um nó folha rotulado pela classe c_i;
  2) Se P não contém exemplos então a árvore é uma folha e a classe associada deve ser determinada a partir de informações além de P (ex: utilizar a classe mais frequente para o nó-pai desse nó);
  3) Se P contém exemplos que pertencem a várias classes então dividir P em subconjuntos mais "puros".
  4) Os passos 1, 2 e 3 são aplicados recursivamente para cada subconjunto de exemplos de treinamento de maneira que, em cada nó, as arestas levam para as subárvores construídas a partir do subconjunto de exemplos P.
- **Método 2:** Utilizar um algoritmo de cobertura sequencial (Sequential Covering):
  1) Invocar APRENDA-UMA-REGRA sobre todos os exemplos;
  2) Remover os exemplos cobertos pela regra aprendida;
  3) Repetir o processo até atingir a fração desejada de exemplos cobertos pelas regras.

B. Dispositivos Adaptativos

Um dispositivo guiado por regras é qualquer abstração formal que tem seu comportamento descrito por um conjunto finito de regras. No formalismo adaptativo, dispositivos adaptativos poder ser obtidos a partir da incorporação de um mecanismo adaptativo atrelado aos recursos oferecidos por um dispositivo guiado por regras. O mecanismo adaptativo é representado por um conjunto de funções adaptativas, as quais são conectadas às regras do dispositivo guiado por regras. As funções adaptativas são capazes de alterar o conjunto de regras do dispositivo, ocasionando a mudança de comportamento no dispositivo adaptativo.

Ações adaptativas podem ser definidas em termos de abstrações que são as funções adaptativas, de modo similar às chamadas de funções em linguagens de programação. As funções adaptativas são descritas por um cabeçalho e um corpo. O cabeçalho da função é composto por um nome, variáveis, parâmetros e geradores. Uma função adaptativa é referenciada por um nome ϕ, os parâmetros p são uma ênupla ordenada (ρ_1, ρ_2,..., ρ_n), de $n \geq 0$ parâmetros formais, as variáveis v são formadas por um conjunto (ν_1, ν_2,..., ν_m), de $m \geq 0$ variáveis ν, os geradores g formam um conjunto (γ_1, γ_2,..., γ_k), de $k \geq 0$ geradores. Os valores das variáveis são preenchidos uma única vez pelas ações elementares de inspeção da função adaptativa. Já os valores dos geradores são atualizados a cada chamada da função adaptativa. O corpo da função é composto por uma função adaptativa anterior e uma função adaptativa posterior e o núcleo da função.

As funções adaptativas, em seu núcleo, referem-se a operações básicas de edição do conjunto de regras que definem o dispositivo adaptativo, representadas pelas ações adaptativas elementares de inspeção, remoção e inserção de regra, onde:

1) *ação adaptativa elementar inspeção*: permite a consulta ao conjunto de regras em busca de um dado padrão de regra.
2) *ação adaptativa elementar remoção*: remove regras do conjunto corrente correspondentes a um dado padrão de regra.

3) *ação adaptativa elementar inserção*: insere uma regra no conjunto corrente de acordo com um dado padrão de regra.

A formulação para dispositivos guiados por regras e dispositivos adaptativos pode ser encontrada em [18].

C. Árvores de Decisão Adaptativas

Em [19], foi apresentado um dispositivo adaptativo cujo mecanismo subjacente é uma árvore de decisão. Esse dispositivo, chamado de árvore de decisão adaptativa, permite que a estrutura hierárquica de uma árvore de decisão possa ser dinamicamente alterada durante o processo de decisão, quando a árvore é percorrida da raiz para as folhas. A capacidade de automodificação é obtida através das funções adaptativas incorporadas a alguns ramos da árvore que permitem a inspeção, a inserção e remoção de subárvores. O mecanismo subjacente das árvores de decisão adaptativas é denominado árvores de decisão não-determinísticas e generaliza o conceito de árvore de decisão. A formulação da árvore de decisão adaptativa é apresentada a seguir, de acordo com [19].

Definição 1. Uma árvore de decisão não-determinística, ou árvore-DND, é uma 7-upla T =(I, Σ, Γ, f, c, A, R), onde:

- I: é o conjunto de exemplos;
- Σ: é o conjunto de atributos, incluindo o atributo classe;
- Γ: são os valores de atributos, incluindo o símbolo "?", denominado valor ausente que deve ser usado para representar valores ausentes, desconhecidos ou inexistentes;
- f: $\Sigma \rightarrow 2^{\Gamma}$: é uma função que determina o domínio de cada atributo;
- $c \in \Sigma$: define o atributo classe;
- A: $I \times \Sigma \rightarrow \Gamma$ é uma função binária usada para descrever os elementos do conjunto de exemplos, associando exemplos, atributos e valores de atributos.
- R: é uma estrutura hierárquica finita denominada subárvore e definida recursivamente como:
  - **Folha**: contendo um identificador único e um valor para o atributo classe;
  - **Não Folha**: uma (n+2-upla) (id, a, (v_1, R_1),..., (v_n, R_n)), onde id é um identificador, $a \in \Sigma$ -{c} é um atributo, $v_i \in f(a)$ são valores para $1 \leq i \leq n$ e todos os elementos R_i, $1 \leq i \leq n$ são subárvores.

A árvore de decisão não-determinística adaptativa, ou árvore-DND adaptativa, é uma árvore DND acrescida de uma camada adaptativa, que permite a modificação do conjunto R. Estas alterções podem ocorrer antes ou depois de cada execução de uma ação elementar, que podem inspecionar, remover ou inserir subárvores.

Definição 2. Uma árvore-DND adaptativa é uma dupla AT = (CS_0, CA), onde:

- CS_0: é o mecanismo subjacente, com uma leve modificação que permite o acoplamento de ações adaptativas a cada uma das subárvores de R;
- CA: mecanismo adaptativo, formado pelo conjunto das funções adaptativas.

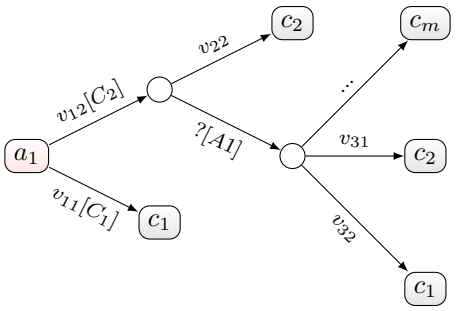

Figura 1. Exemplo de árvore de decisão adaptativa.

A operação de uma árvore- DND adaptativa opera de acordo com o mecanismo subjacente, até que uma função adaptativa seja disparada.

IV. Algoritmo de aprendizagem

O algoritmo de aprendizagem está baseado no funcionamento do *Adaptree*, proposto por [20], no qual a ideia geral é transformar cada exemplo de treinamento em um caminho partindo da raiz até a folha. Porém é necessário evidenciar algumas diferenças: 1) As funções adaptativas estão acopladas nos ramos e não aos nós como no *Adaptree*, isso permite a remoção e inserção de ramos e não de subárvores. 2) Cada caminho da árvore é transformado em uma regra do tipo "*Se..então*".

A árvore de decisão adaptativa gerada é composta pelo conjunto de atributos Σ, que possui uma ordenação arbitrária podendo ser representada por uma sequência $a_1, ..., a_n$, na qual n = número total de atributos, $c = a_n$ representa o atributo classe. A estrutura inicial S da árvore contém um valor inicial desconhecido ? na raiz e a camada adaptativa Φ é composta por um conjunto de funções adaptativas $A_1, ..., A_k$, podendo estas conter ações elementares de inserção ($+$), remoção ($-$) ou consulta (?). As funções adaptativas estão acopladas a alguns ramos específicos e quando acionadas, conferem à árvore a capacidade de automodificação.

O processo de aprendizagem pode ser divido em três etapas distintas, sendo elas *pré-processamento*, *aprendizagem* e *classificação*, que serão descritas a seguir:

A. Pré-processamento

O algoritmo realiza a leitura e pré-processamento do conjunto de dados, executando uma função inicial que impõe uma ordem aos atributos, de acordo com a forma que estão dispostos no conjunto de treinamento. Esta ordem é mantida ao longo de toda a execução do algoritmo. Por convenção, o primeiro atributo a ser lido é alocado na raiz da árvore, os demais atributos são os nós internos e o último atributo é a classe.

B. Aprendizagem

O algoritmo recebe uma nova instância de treinamento e busca na estrutura da árvore um caminho que corresponda

a instância que está sendo processada. Esta correspondência está baseada na comparação dos valores do atributo raiz e do atributo folha, ou seja, o algoritmo verifica se existe um caminho na estrutura S, cujo valor do atributo raiz e o valor do atributo folha sejam iguais aos valores da instância de treinamento. Na aprendizagem o algoritmo executa em dois modos: modo de aprendizagem não adaptativo e modo de aprendizagem adaptativo.

1) Modo de aprendizagem não adaptativo. Se não existir um caminho correspondente, o algoritmo constrói um novo caminho, de maneira que o valor do nó raiz e do nó folha assumem os valores contidos na instância de treinamento. Os valores dos nós internos são denotados por um valor "?", o qual representa um valor de atributo ainda desconhecido. Esses valores poderão ser preenchidos com base nos valores de atributos das futuras instâncias de treinamento. Além disso, aos ramos que possuem este símbolo, também está acoplada uma função adaptativa de classificação. Essas funções adaptativas poderão ser utilizadas posteriormente para a classificação de uma nova instância. Quando existe um caminho correspondente à instância de treinamento, o algoritmo executa o mecanismo estatístico que busca o atributo de maior relevancia, para modificar a árvore e melhorar a classificação. A medida estatística utilizada é baseada na frequência relativa de cada valor de atributo lido no caminho encontrado, em relação ao total de instâncias recebidas até o momento.

2) Modo de aprendizagem adaptativo. Se existir um caminho, o algoritmo por meio das funções adaptativas de aprendizagem (Algoritmo 1) acopladas aos ramos do nó raiz, modifica convenientemente este caminho de forma a melhor generalizar a árvore. A função adaptativa recebe como parâmetro o atributo-valor mais relevante, encontrado pelo mecanismo estatístico. Assim, por meio do mecanismo adaptativo é realizada a atualização do ramo correspondente ao atributo relevante.

Algoritmo 1: FUNÇÃO ADAPTATIVA DE APRENDIZAGEM.

1 **Função Adaptativa** A_k (a_j, a_{j+1}, $valor$)
2 **início**
3 | Consulta: $?[(a_j, ?) \rightarrow a_{j+1}]$
4 | Remoção: $-[(a_j, ?) \rightarrow a_{j+1}]$;
5 | Inserção: $+[(a_j, valor) \rightarrow a_{j+1}]$
6 **fim**

C. Classificação

O algoritmo recebe uma nova instância de teste a fim de classificá-la. Para isso, realiza uma busca em profundidade, percorrendo cada um dos caminhos da árvore até que se encontre um caminho capaz de classificar a nova instância. Nesta etapa, a utilização da tecnologia adaptativa torna-se possível conferir à árvore a capacidade de automodificação durante a classificação. Neste caso, se existir algum caminho da árvore com valores de atributo *default*, ou seja, contendo um valor "?", uma função adaptativa é disparada. Essa função executa uma ação elementar de remoção e outra de inserção (Algoritmo 2), que permite modificar o valor do atributo *default*, para um valor que permita a classificação da instância de teste. Os parâmetros a_j e *valor* são referentes à instância corrente. Após a classificação da instância, essa alteração é descartada e a árvore original é preservada para uma nova classificação.

Algoritmo 2: FUNÇÃO ADAPTATIVA DE CLASSIFICAÇÃO.

1 **Função Adaptativa** C_k (a_j, a_{j+1}, valor)
2 **início**
3 | Remoção: $-[(a_j, ?) \rightarrow a_{j+1}]$;
4 | Inserção: $+[(a_j, valor) \rightarrow a_{j+1}]$
5 **fim**

Por exemplo, supondo a execução da regra R_5, mostrada a seguir:

- Regra R5 antes da modificação:
  `Se` $((a_1 = v_{12}) \wedge (a_2 = v_{21}) \wedge (a_3 = ?)[C_1] \wedge (a_4 = v_{43}))$ `então` c_1

Para a classificação da instância $\alpha = v_{12}v_{21}v_{33}v_{43}$ a função adaptativa C_1 é acionada, tendo como parâmetro $a_j = a_3$ e $valor = v_{33}$. Neste caso, é executada a ação elementar de remoção $-[(a_3, ?) \rightarrow a_4]$ e a ação elementar de inserção $+[(a_3, a_3) \rightarrow a_4]$. A regra resultante da modificação é apresentada na sequência e classifica a instância como pertencente a classe c_1.

- Regra R5 depois da modificação:
  `Se` $((a_1 = v_{12}) \wedge (a_2 = v_{21}) \wedge (a_3 = v_{33}) \wedge (a_4 = v_{43}))$ `então` c_1

Quando a função de busca retorna mais de uma regra ou caminho, a decisão de classificação é tomada estatísticamente.

V. EXPERIMENTOS PRELIMINARES

O algoritmo foi implementado utilizando a linguagem de programação Java e reutiliza algumas classes do pacote WEKA (Waikato Environment for Knowledge Analysis) [21].

A. Recursos Utilizados

Dentre as limitações do algoritmo, está o fato dele trabalhar apenas com valores de atributos nominais e também ainda não realizar tratamento de valores ausentes (*missing value*), por este motivo foi utilizado apenas 1 conjunto de dados. O conjunto de dados utilizado foi o *lenses* contendo 24 instâncias de treinamento e 4 valores de atributos. Este conjunto de dados descreve quais tipos de lentes de contato se ajustam melhor a cada paciente. Os atributos observados são idade, astigmatismo e produção de lágrima. As classes indicam se o paciente deve usar lentes macias, lentes duras ou se não devem usar lentes.

Quanto aos algoritmos escolhidos, optou-se por utilizar algoritmos de diferentes abordagens de aprendizagem de máquina, por se tratar ainda de estudos preliminares. Os algoritmos utilizados foram:

- NaiveBayesUpdateable: versão incremental do Naive-Bayes;
- JRip ou Ripper: aprendizagem incremental de regras;
- HoeffdingTree: aprendizagem incremental de árvores de decisão;

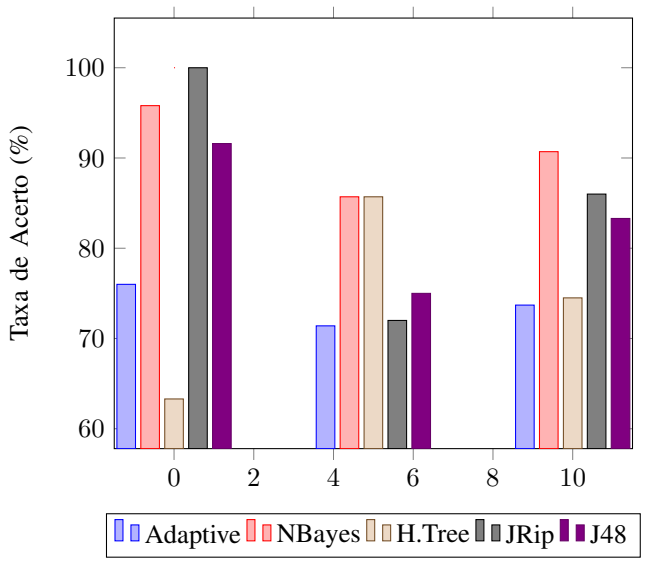

Figura 2. Gráfico comparativo das taxas de acerto

- J48: algoritmo não incremental da família TDIDT.

Os conjuntos de dados utilizados foram retirados do repositório do UCI (UC Irvine Machine Learning Repository) [1]. Todos os algoritmos utilizados possuem o código-fonte disponível na ferramenta WEKA [2].

A medida de desempenho utilizada foi a precisão, baseada nas quantidades de exemplos classificados corretamente, e na porcentagem de acertos obtidos pelo algoritmo.

Os testes foram realizados da seguinte forma:

- **Experimento 1**: foi utilizado o conjunto de dados completo tanto para treino quanto para teste.
- **Experimento 2**: o conjunto de dados completo foi dividido aleatoriamente em dois peças, um conjunto de treinamento e um conjunto de teste, na proporção aproximada de 70% a 30%.

B. Análise dos Resultados

A Tabela I apresenta as taxas de acerto para cada algoritmo, executados sobre cada um dos conjuntos de treinamento e testes. A primeira linha refere-se à Experimento 1, a linha dois refere-se à Experimento 2 e a última linha apresenta a média dos dois experimentos. A Figura 2 apresenta um gráfico dos resultados apresentados na Tabela I.

Tabela I
COMPARAÇÃO DAS TAXAS DE ACERTO.

Dados	***Adaptive***	**NBayes**	**H. Tree**	**JRip**	**J48**
Exp. 1	76%	95.8%	63.3	100%	91.6
Exp. 2	71.4%	85.7%	85.7%	72%	75%
Média	73.7%	90.7%	74.5%	86%	83.3%

De acordo com o gráfico, o resultado do novo algoritmo não oscilou muito entre os dois tipos de experimento, obtendo uma diferença de apenas 1%, o que pode ser considerado pouco significativo em termos estatísticos. Os algoritmos NaiveBayesUpdateable e HoeffdingTree obtiveram o maior desempenho no Experimento 2, porém este último obteve o pior desempenho no Experimento 1. No Experimento 1 o algoritmo JRip foi o algoritmo que obteve a melhor performance, porém no Experimento 2 o seu resultado foi um dos piores, com uma diferença inferior a 1% do Adaptive. Esse comportamento também foi observado no J48, que obteve um bom resultado no Experimento 1, e uma pequena queda no Experimento 2. Na média, os algoritmos que menos oscilaram os resultados foram primeiramente o Adaptive, em segundo o NaiveBayesUpdateable e na sequência o J48. É importante destacar que o JRip e o J48 são algortimos não incrementais, desta forma eles possuem a vantagem de possuir todo o conjunto de treinamento de antemão para a tomada de decisão e construção do classificador. Já em relação ao NaiveBayesUpdateable, apesar de ser incremental, este possui vantagem em relação ao Adaptative quanto ao método estatístico utilizado.

[1]Disponível em: `http://archive.ics.uci.edu/ml/index.php`
[2]Disponível em: `http://www.cs.waikato.ac.nz/ml/weka/`

VI. CONSIDERAÇÕES FINAIS

Para um estudo preliminar, os resultados foram considerados satisfatórios, visto que mesmo o algoritmo tendo obtido um desempenho inferior a outros algoritmos como o NaiveBayesUpdateable (versão incremental do NaiveBayes) e o J48 (implementação do C4.5), o resultado obtido em Experimento 2 foi similar ao JRip, um algoritmo incremental que também utiliza uma abordagem de aprendizagem de regras. A busca por melhores resultados pode ser conduzida de diferentes modos em trabalhos futuros. Primeiro, foi utilizado apenas um conjunto de dados, além disso nota-se que o conjunto de dados é relativamente pequeno, sendo uma extensão natural deste trabalho a coleta de novos dados. Porém, para isso será necessário implementação o tratamento de valores ausentes e um método de discretização para lidar com atributos numéricos. Em segundo, notou-se que a regras geradas pelo algoritmo ainda possuem muitos valores de atributos *default*, e isso dificulta a classificação, pois faz com que o algoritmo retorne vários caminhos possíveis para classificar uma dada instância e a decisão seja tomada pelo mecanismo estatítico. Neste ponto, considera-se a introdução de uma medida estatística mais sofisticada, como o ganho de informação por exemplo, já que a frequência relativa é uma medida bastante ingênua para a escolha do melhor atributo.

Além disso, em trabalhos futuros pretende-se realizar novos experimentos em função da complexidade das regras geradas pelo algoritmo. A complexidade de um conjunto de regras pode ser medida em termos de, pelo menos, três parâmetros [22]

Intuitivamente, o algoritmo parece ter bom desempenho em conjunto de dados onde pelo menos uns dos atributos é claramente mais relevante para uma determinada classe. Para conjunto de dados que em que todos os atributos contribuem em igual peso na resposta, a taxa de acerto do algoritmo é reduzida em função da medida estatística adotada. Porém para isso, serão necessários novos experimentos e comparação apenas com algoritmos baseados em regras.

REFERÊNCIAS

[1] E. Alpaydin, *Introduction to Machine Learning*, ser. Adaptive computation and machine learning. MIT Press, 2014. [Online]. Available: https://books.google.com.br/books?id=NP5bBAAAQBAJ

[2] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2012. [Online]. Available: https://books.google.com.br/books?id=Br33IRC3PkQC

[3] T. Mitchell, *Machine Learning*, ser. McGraw-Hill International Editions. McGraw-Hill, 1997. [Online]. Available: https://books.google.com.br/books?id=EoYBngEACAAJ

[4] D. Garcia, J. C. Gamez, A. Gonzalez, and R. Perez, "Using a sequential covering strategy for discovering fuzzy rules incrementally," in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Aug 2015, pp. 1–8.

[5] M. Michalak, M. Sikora, and L. Wrobel, "Rule quality measures settings in a sequential covering rule induction algorithm - an empirical approach," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2015, pp. 109–118.

[6] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "A new sequential covering strategy for inducing classification rules with ant colony algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 64–76, Feb 2013.

[7] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 125–141, May 2008. [Online]. Available: https://doi.org/10.1007/s11263-007-0075-7

[8] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, Oct 2011.

[9] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2007, pp. 255–262.

[10] H. Pistori and J. J. Neto, "Adaptree - proposta de um algoritmo para indução de árvores de decisão baseado em técnicas adaptativas." in *Anais Conferência Latino Americana de Informática - CLEI 2002.*, Montevideo, Uruguai, Novembro 2002.

[11] M. L. Yoshida, "Aprendizado supervisionado incremental de redes bayesianas para mineração de dados." Master's thesis, Universidade Federal de São Carlos - UFSCar, 2007.

[12] J. J. Neto, *Solving Complex Problems Efficiently with Adaptive Automata*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 340–342.

[13] J. a. J. Rocha, Ricardo. L. A.; Neto, "Automato adaptativo, limites e complexidade em comparação com máquina de turing." in *Proceedings of the second Congress of Logic Applied to Technology - LAPTEC*. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, 2001, pp. 33–48.

[14] T. A. Plate, "Accuracy versus interpretability in flexible modeling: implementing a tradeoff using gaussian process models," *Behaviour Metrika Special Issue On Interpreting Neural Network Models*, vol. 26, pp. 29–50, 1999.

[15] R. L. Stange and J. J. Neto, "Aprendizagem incremental usando tabelas de decisão adaptativas." *Quinto Workshop de Tecnologia Adaptativa - WTA 2011*, 2011.

[16] ——, "Learning decision rules using adaptive technologies: a hybrid approach based on sequential covering," *Procedia Computer Science*, vol. 109, no. Supplement C, pp. 1188 – 1193, 2017, 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917310712

[17] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986. [Online]. Available: http://dx.doi.org/10.1023/A:1022643204877

[18] J. J. Neto, *Adaptive Rule-Driven Devices - General Formulation and Case Study*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 234–250.

[19] H. Pistori, J. J. Neto, and M. C. Pereira, "Adaptive non-deterministic decision trees: General formulation and case study," *INFOCOMP Journal of Computer Science*, 2006.

[20] H. Pistori, "Tecnologia em engenharia de computação: estado da arte e aplicações," PhD thesis, Escola Politécnica, Universidade de São Paulo, 2003.

[21] I. Witten, E. Frank, and M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, ser. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011. [Online]. Available: https://books.google.com.br/books?id=bDtLM8CODsQC

[22] M. Bramer, *Automatic Induction of Classification Rules from Examples Using N-Prism*. London: Springer London, 2000, pp. 99–121.

Fabio Kenji Oshiro Takatuzi é graduado em Tecnologia em Sistemas para Internet pela Universidade Tecnológica Federal do Paraná (UTFPR) (2017). Atualmente, atua como instrutor de ensino na empresa VitalNet Consultoria em Informática, possuindo experiência no ensino de linguagens de programação, desenvolvimento web e design gráfico. Possui interesse na área da aprendizagem de máquina, em especial, na utilização da tecnologia adaptativa em aplicações para mineração de dados, tomada de decisão e inteligência artificial.

Renata Luiza Stange Atualmente é doutoranda do Programa de Pós-Graduação em Engenharia de Computação do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo (PCS/EPUSP), atuando como pesquisadora no Laboratório de Linguagens e Técnicas Adaptativas (LTA). Mestre em Ciências, também pelo Programa de Pós-Graduação em Engenharia de Computação do PCS/EPUSP (2011). Bacharel em Análise de Sistemas pela Universidade Estadual do Centro-Oeste (2002). Atua como docente na Universidade Tecnológica Federal do Paraná (UTFPR). Tem experiência na área de Ciência da Computação, particularmente em dispositivos adaptativos, tecnologia adaptativa e suas aplicações à inteligência artificial, aprendizagem de máquina e reconhecimento de padrões.