ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

RENATA LUIZA STANGE

Adaptatividade em Aprendizagem de Máquina: Conceitos e Estudo de Caso

São Paulo

2011

RENATA LUIZA STANGE

Adaptatividade em Aprendizagem de Máquina: Conceitos e Estudo de Caso

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

São Paulo

2011

RENATA LUIZA STANGE

Adaptatividade em Aprendizagem de Máquina: Conceitos e Estudo de Caso

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

Área de concentração: Engenharia da Computação e Sistemas Digitais

Orientador: Prof. Dr. João José Neto

São Paulo

2011

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de dezembro de 2011.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Stange, Renata Luiza
Adaptatividade em aprendizagem de máquina: conceitos e estudo de caso / R.L. Stange. -- ed.rev. -- São Paulo, 2011.
98 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Aprendizado computacional 2. Reconhecimento de padrões 3. Adaptatividade 4. Tecnologia adaptativa 5. Aprendizagem incremental 6. Classificadores I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

DEDICATÓRIA

Ao Henry, marido, pelo incentivo e pela paciência. Aos meus pais e familiares pela compreensão da ausência no convívio familiar.

AGRADECIMENTOS

Deus, pela força espiritual nos momentos de difíceis durante o desenvolvimento desta dissertação.

Ao Prof. Dr. João José Neto, pela oportunidade e paciência durante a orientação, pelo constante incentivo durante todo o trabalho de pesquisa, pela compreensão e conselhos nos momentos difíceis, e principalmente pela sua verdadeira amizade.

Aos professores Dr. Ricardo Luis De Azevedo Da Rocha e a Drª Angela Hum Tchemra pelas observações e sugestões apresentadas no exame de qualificação, que foram de grande valia para o enriquecimento deste trabalho.

A Dr. Fabiana Soares Santana, por compartilhar seu conhecimento em pesquisa e sempre estar disposta a sugerir, revisar e ensinar. Também por sua amizade, hospedagens e cafés ao longo desses anos.

Ao amigo Luciano Ogiboski pelo incentivo para iniciar a pós-graduação na Universidade de São Paulo.

Enfim, a todos os colegas e professores, que influenciaram de alguma forma na realização deste trabalho, em especial aos do Departamento de Engenharia da Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

RESUMO

A aprendizagem incremental requer que o mecanismo de aprendizagem seja baseado no acúmulo dinâmico da informação extraída das experiências realizadas. A aprendizagem de máquina usando adaptatividade considera a integração de técnicas de aprendizagem de máquina simbólicas com técnicas adaptativas para a solução de problemas de aprendizagem. A palavra adaptatividade sugere a capacidade de modificação do conjunto de regras aprendidas em resposta a eventos que podem ocorrer durante o processo de aprendizagem, ou então autoajustes no conjunto de parâmetros. Os dispositivos adaptativos que possuem a capacidade de reter em suas regras informações extraídas de suas entradas podem acumular informações, para que sejam utilizadas quando forem necessárias. As estratégias de interesse para a incorporação da adaptatividade incluem a utilização de métodos e técnicas de aprendizagem de máquina, em particular as que implementam aprendizado supervisionado e tomada de decisão. O objetivo deste trabalho é explorar a utilização de técnicas adaptativas no processo de aprendizado por máquina, tanto de forma exclusiva como em conjunto com outras técnicas de aprendizagem. Para atingir este objetivo, propõe-se aqui a utilização de dispositivos adaptativos para representar o conhecimento adquirido através da aprendizagem incremental. Além disso, é feito um estudo de caso que combina aprendizagem de máquina com técnicas adaptativas para implementar um esquema de aprendizagem autônoma de estratégias, com o objetivo de vencer uma particular instância do jogo que é apresentado. A aprendizagem de um jogo exige a tomada de decisão, que é um processo complexo e dinâmico. Com a finalidade de fornecer um substrato geral para a criação, manipulação e análise de regras em problemas de tomada de decisão, utilizando tabelas de decisão adaptativas, a ferramenta de software *Adapt*-DT foi implementada. Um exemplo ilustrativo utilizando tabelas de decisão adaptativa como meio para a representação de conhecimento é apresentado, para exercitar a utilização da ferramenta. Isto permite concluir que os dispositivos adaptativos podem ser utilizados para representar o conhecimento adequadamente, com vantagens sobre outros métodos tradicionais.

Palavras-chave: Adaptatividade. Tecnologia Adaptativa. Aprendizado de Máquina. Aprendizagem Incremental. Reconhecimento de Padrões. Classificadores. Tomada de Decisão.

ABSTRACT

Incremental learning requires a learning mechanism based on the information extracted from dynamically accumulated experiments. Adaptivity-oriented machine-learning combines adaptive techniques with symbolic ones for solving machine-learning problems. The term “adaptivity” means the ability of a learning process to change its own set of rules in response to events occurred during the learning process, or, equivalently, self-tuning the set of parameters. The adaptive devices with withhold information ability inside their rules, extracted from input from their own set of rules, can accumulate information to be used whenever they are necessary. The strategies of interest to adopt adaptivity include the use of machine learning techniques and methods, particularly the ones that implement supervised learning and decision-making. This work purposes to investigate the application of adaptive techniques in machine learning process, either exclusively and in cooperation with other techniques. In order to achieve this target, the use of adaptive devices to represent the knowledge gathered through incremental learning is proposed. Additionally, a case study that combines both machine learning and adaptive techniques to implement a scheme of autonomous learning strategies is also performed with the goal of winning an instance of the simple game. Decision-making is required to learning how to play a game, which is a complex and dynamic process. So as to provide a general framework for the creation, manipulation and analysis of rules in decision-making problems using adaptive decision tables, the *Adapt*-DT tool was implemented. An illustrative example using adaptive decision tables as a means to represent knowledge is introduced to the tool evaluation. This supports the conclusion that adaptive devices can be used to adequately represent the knowledge, with advantages over other traditional methods.

Keywords: Adaptivity. Adaptive Tecnhology. Machine Learning. Incremental Learning. Pattern Recognition. Classifiers. Decision-Making.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AD	Autômato Adaptativo
ADAPT-DT	*Adaptive Decision Table*
ADAPT-TREE	*Adaptive Tree*
ADAPTOOLS	*Adaptive Tools*
CART	*Classification and Regression Trees*, no original em inglês, ou Classificação e Regressão de árvores
DNA	*Deoxyribonucleic Acid*, no original em inglês, ou Ácido Desoxirribonucléico.
EPUSP	Escola Politécnica da Universidade de São Paulo
LTA	Laboratório de Linguagens e Técnicas Adaptativas
ML	*Machine Learning*, no original em inglês, ou Aprendizagem de Máquina.
PCS	Departamento de Engenharia da Computação e Sistemas Digitais
STAD	*Statecharts* Adaptativos
TTT	Tic-tac-toe
TD	Tabela de Decisão
TDA	Tabela de Decisão Adaptativa
UCI	*University of California - Irvine*

SUMÁRIO

1 INTRODUÇÃO

O termo “Aprendizado por Máquina”, ou aprendizagem de máquina (ML, no original em inglês *Machine Learning*), refere-se ao funcionamento de sistemas computacionais capazes de aprender e modificar o seu comportamento em resposta a estímulos externos, ou através de experiências acumuladas durante sua operação (ALPAYDIN, 2010). Aprendizagem de máquina é uma área de pesquisa que estuda métodos, técnicas e ferramentas computacionais relacionadas à aquisição de novos conhecimentos e, novas habilidades para melhorar o desempenho de algoritmos por meio da experiência (MITCHELL, 1997; ALPAYDIN, 2010).

A adaptatividade é uma característica atribuída ao comportamento automodificável de sistemas computacionais. Este comportamento autônomo ocorre em resposta a estímulos de entrada e ao histórico de operação desses sistemas (NETO, 2001). As pesquisas em adaptatividade investigam soluções para diversos problemas complexos de teoria da computação (NETO, 2000), de aprendizagem de máquina (PISTORI; NETO, 2003a), de tomada de decisão (TCHEMRA, 2009) e de engenharia da computação (PISTORI, 2003), entre outros.

A tecnologia adaptativa corresponde ao conjunto de ferramentas, métodos e técnicas, que permitem solucionar problemas práticos utilizando modelos baseados em dispositivos adaptativos. O modelo geral para um dispositivo adaptativo é definido como um conjunto finito de regras que pode sofrer modificações dinamicamente (NETO, 2001).

O trabalho proposto nesta dissertação tem como objetivo investigar questões relacionadas à utilização da adaptatividade no processo de aprendizado por máquinas.

1.1. MOTIVAÇÃO

A elaboração deste trabalho é motivada no encontro de três assuntos complementares: Tecnologia Adaptativa, Aprendizagem de Máquina e Tomada de Decisão.

A área de aprendizagem de máquina tem-se mostrado uma rica fonte de pesquisa para a exploração prática das aplicações dos fundamentos da tecnologia adaptativa.

Neto e Iwai (1998) apresentam o mecanismo de inferência ativo nos autômatos adaptativos. Como exemplo ilustrativo, o autômato adaptativo é utilizado para o aprendizado supervisionado de linguagens regulares. Um conjunto de amostras positivas[1] e negativas[2] da linguagem é submetido ao autômato, que deve inferir as sentenças aceitas ou rejeitadas.

Pistori e Neto (2002) propõem um algoritmo de indução de árvores de decisão utilizando técnicas adaptativas, que combina estratégias sintáticas e estatísticas, chamado *AdapTree*.

Outras experiências bem-sucedidas em aprendizagem de máquina utilizando dispositivos adaptativos incluem: aprendizagem de modelos para distribuição de espécies (PARIENTE; NETO; SANTANA, 2005; STANGE et al., 2011), classificação de padrões geométricos (HIRAKAWA; SARAIVA; CUGNASCA, 2007), decodificação do alfabeto de LIBRAS (DIAS; SOUZA; PISTORI, 2006), localização de padrões em imagens (PISTORI; NETO, 2004), identificação de diagnósticos médicos (GANZELI et al., 2010) e mineração de dados (TCHEMRA; CAMARGO, 2009), entre outras.

O processo de aprendizagem de máquina traz a tomada de decisão de forma crucial. A tomada de decisão exige um processo de raciocínio em que as informações já adquiridas e as novas informações, quando comparadas entre si, possam levar a novas informações e, com isso, influenciar o processo (TCHEMRA, 2007). Esse processo de raciocínio é muitas vezes complexo e dinâmico, uma vez que as decisões devem ser flexíveis, pois eventualmente dependem de vários fatores e prioridades que nem sempre são fáceis de identificar antes de iniciar o processo de aprendizagem.

De acordo com Neto (2000), a resolução de problemas complexos e de natureza dinâmica utilizando a tecnologia adaptativa pode ser mais expressiva do que a utilização de métodos tradicionais, em alguns casos.

Em aprendizagem de máquina, por exemplo, uma das dificuldades está relacionada à representação do conhecimento humano em uma linguagem simbólica que tenha grande poder de expressividade. Métodos tradicionais para essa representação

[1] Conjunto de sentenças que pertence à linguagem.
[2] Conjunto de sentenças que não pertence à linguagem.

incluem o uso de regras de produção, árvores de decisão e redes Bayesianas, entre outros. O uso de dispositivos adaptativos pode agregar expressividade à representação do conhecimento e contribuir para o crescimento dessa área. Os autômatos adaptativos, por exemplo, além de possuírem o mesmo poder de expressão da Máquina de Turing (ROCHA; NETO, 2000), são eficientes e de fácil visualização, pois são baseados em modelos de autômatos finitos (NETO, 1993).

Contudo, a exploração de questões referentes à aprendizagem de máquina utilizando a tecnologia adaptativa se aplica, de maneira abrangente, ao tratamento de problemas de tomada de decisão.

1.2. OBJETIVOS

O objetivo geral deste trabalho é explorar a utilização de técnicas adaptativas no processo de aprendizado por máquina, tanto de forma exclusiva como em conjunto com outras técnicas de aprendizagem. Para atingir este objetivo, propõe-se aqui a utilização de dispositivos adaptativos para representar o conhecimento adquirido através da aprendizagem incremental.

Os objetivos específicos são:

- Estudar o comportamento dos dispositivos adaptativos como mecanismos de inferência através de um estudo de caso.
- Projetar uma ferramenta computacional de apoio à tomada de decisão em aprendizagem de máquina baseada em dispositivos adaptativos.

1.3. JUSTIFICATIVA

A incorporação da adaptatividade no processo de aprendizagem captura um aspecto fundamental da aprendizagem, que trata da adaptação dinâmica das regras de aprendizagem em função de sua interação com o ambiente (PISTORI, 2003).

Um fator relevante na forma de representar o conhecimento adquirido no processo de aprendizagem é o grau de compreensibilidade proporcionado ao ser humano ou especialista do sistema. Para Michalski (1983), alguns sistemas de aprendizagem podem ser como caixas-pretas, isto é, as regras que estão sendo aprendidas podem não

ser facilmente interpretadas e compreendidas por humanos, o que dificulta a detecção de erros e validação do processo de aprendizagem.

A utilização de técnicas adaptativas na aprendizagem de máquina tem como objetivo simplificar o entendimento das regras de aprendizagem por humanos ou especialistas podendo desta forma facilitar a detecção de erros e validação do processo de aprendizagem.

De fato, enquanto o projetista do sistema de aprendizagem se preocupa com o engenho de algoritmos, seja ajustando parâmetros, escolhendo ou testando métodos – o que de fato é a sua função – os interesses do especialista do sistema são diferentes e estão mais voltados à necessidade de se solucionar um problema de forma efetiva, sem a necessidade de interferência no processo de aprendizagem.

Em relação ao formalismo utilizado na representação de problemas e algoritmos, uma questão recorrente na teoria da computação é a busca do equilíbrio entre expressividade e usabilidade. Por exemplo, as Máquinas de Turing são altamente expressivas, porém sua utilização direta é difícil, enquanto as Máquinas de Estados Finitos são fáceis de usar, mas seu poder de expressão é restrito (PISTORI, 2003). As técnicas adaptativas (NETO, 2001) permitem aumentar a expressividade de formalismos convencionais, tais como autômatos, árvores de decisão e outros, sem prejudicar a sua usabilidade.

Uma questão levantada por Pistori (2003) é a carência de ferramentas computacionais para a disseminação e expansão da tecnologia adaptativa, uma vez que a integração entre teoria e prática facilitaria a compreensão e utilização da tecnologia. As ferramentas disponíveis para uso da tecnologia adaptativa são: 1) ADAPTOOLS (PISTORI; NETO, 2003b), para autômatos adaptativos; 2) STAD e STAD-S (ALMEIDA JUNIOR, 1995), para *statecharts* adaptativos; e 3) LASSUS (BASSETO, 2000), para geração de música por computador usando redes de Markov adaptativas. Essas ferramentas estão disponíveis para *download* no site do laboratório de Linguagens e Técnicas Adaptativas do Departamento de Engenharia da Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo[3].

[3] Disponível em http://www.pcs.usp.br/~lta

1.4. METODOLOGIA

Para alcançar os objetivos propostos neste trabalho, optou-se por uma pesquisa de natureza exploratória. O estudo utiliza-se de revisão bibliográfica e de uma investigação empírica sobre a adaptatividade no contexto de aprendizagem de máquina.

1.5. ORGANIZAÇÃO DO TRABALHO

O trabalho relatado nesta dissertação inclui os capítulos descritos a seguir.

O capítulo 1 apresenta uma visão geral do trabalho, incluindo a motivação e as justificativas para a realização da pesquisa. Os objetivos gerais e específicos do trabalho são apresentados neste capítulo, bem como a metodologia utilizada.

O capítulo 2 contém os principais conceitos de Aprendizado por Máquinas, necessários para o entendimento desta dissertação.

Uma visão geral da Tecnologia Adaptativa e um resumo dos dispositivos adaptativos são apresentados no capítulo 3.

O desenvolvimento do trabalho, que trata da investigação da adaptatividade em aprendizagem de máquina é descrito no capítulo 4.

O capítulo 5 mostra os aspectos de projeto e implementação de uma ferramenta baseada em tabelas de decisão adaptativas para simular problemas de tomada de decisão. Também inclui um exemplo da aplicação da ferramenta na tomada de decisão em aprendizagem de máquina.

E, finalmente, as considerações finais encontram-se no capítulo 6, seguidas das sugestões para a continuidade do trabalho, no capítulo 7.

PARTE I

Conceitos

2 APRENDIZADO POR MÁQUINA

A definição de aprendizado por máquina é:

> Um programa de computador é dito aprender a partir de uma experiência *E* com respeito a uma classe de tarefas *T* e medida de desempenho *P*, se seu desempenho nas tarefas em *T*, segundo a medida *P*, melhora com a experiência *E* (MITCHELL, 1997, p.2).

Há várias situações nas quais o aprendizado por máquina é desejável. De modo geral, os sistemas computacionais capazes de aprender são utilizados na solução de problemas que não podem ser resolvidos por métodos tradicionais de programação, tais como os imperativos, funcionais ou orientados a objetos (PISTORI, 2003; PRATI, 2006). A princípio (PRATI, 2006), ainda não se conhece um algoritmo implementado por métodos tradicionais de programação que seja capaz de reconhecer, por exemplo, caracteres escritos à mão. No entanto, utilizando técnicas de aprendizagem de máquina é possível projetar um sistema computacional que aprenda a reconhecer caracteres escritos a mão, através da observação de uma grande quantidade de manuscritos.

Podemos dizer que nos paradigmas tradicionais de programação, o projetista do sistema ou desenvolvedor é exclusivamente encarregado de encontrar a representação computacional implementável da solução do problema a ser resolvido (MITCHELL, 1997 apud PISTORI, 2003). Por outro lado, os métodos de aprendizagem de máquina oferecem ao projetista do sistema recursos para criar um sistema computacional capaz de obter uma solução automática (ou semi-automática), alcançada a partir de exemplos particulares do problema (MITCHELL, 1997).

O mesmo entendimento pode ser sobreposto em outros casos, nos quais um sistema computacional deve observar um conjunto de fatos e ser capaz de distinguir características de interesse nesses fatos (ex.: observar uma cadeia de caracteres e discriminar as sequências de DNA). Conforme Pistori (2003) o mesmo ambiente de aprendizado pode ser utilizado na solução de problemas que são diferentes do proposto originalmente.

2.1. CARACTERÍSTICAS GERAIS DOS SISTEMAS DE APRENDIZADO

Os sistemas de aprendizado por máquinas possuem características peculiares que possibilitam uma classificação não exclusiva desses sistemas em função da linguagem de descrição, modo de aprendizado, paradigma de aprendizado, formas e tarefa de aprendizado. A Tabela 1 apresenta de forma resumida essa classificação (PRATI, 2006).

Tabela 1 – Características gerais dos sistemas de aprendizado por máquinas

CLASSIFICAÇÃO DOS SISTEMAS DE ML				
MODOS DE APRENDIZADO	PARADIGMAS DE APRENDIZADO	LINGUAGEM DE DESCRIÇÃO	FORMAS DE APRENDIZADO	TAREFAS DE APRENDIZADO
Supervisionado	Simbólico	Exemplos	Incremental	Classificação
Não Supervisionado	Estatístico	Hipóteses	Não Incremental	Regressão
Semissupervisionado	Conexionista	Conhecimento de domínio		
	Genético			

Com a finalidade de obter um sistema de aprendizado capaz de representar computacionalmente um determinado problema, bem como a sua solução, é necessário descrever objetos, processos e situações que fazem parte do seu domínio (MONARD; BARANAUKAS, 2000).

Existem diferentes linguagens de descrição com diferentes complexidades capazes de descrever exemplos (casos observados), hipóteses e conhecimento de domínio (ex.: lógica de atributos, lógica proposicional, lógica relacional, funções matemáticas, etc.).

Neste trabalho é adotado um tipo de linguagem para descrição de exemplos baseada na lógica de atributos, amplamente adotada em algoritmos de aprendizagem. Neste tipo de linguagem, um exemplo é descrito por um conjunto de atributos ou características que assumem diversos valores. Cada exemplo é formado pela conjunção do par atributo e valor que representa um caso observado, ou uma instância do problema (ex.: dor = sim $\wedge$ febre=sim $\wedge$ classe=doente). A classe é um atributo especial definido em alguns exemplos, que representa a saída do algoritmo para aquela instância. É dito que um exemplo é rotulado quando apresenta o valor correspondente ao atributo classe.

Em geral, os algoritmos de aprendizado de máquina têm como entrada um conjunto de exemplos, que podem ser considerados formas de representar os estímulos externos ou experiências que permitem adquirir conhecimento sobre algo. De forma mais abrangente (RUSSEL; NORVIG, 2002), o termo inferência é utilizado para referir-se a aquisição de novos conhecimentos a partir de um conhecimento prévio sobre algo que se deseja aprender. Considerando a inferência realizada sobre um conjunto de exemplos, existem diferentes estratégias de aprendizado, tal como a indução. A indução é uma forma de inferência que permite obter conclusões genéricas a partir de um conjunto particular de exemplos ou fatos observados (RUSSEL; NORVIG, 2002).

No que se refere ao modo como os algoritmos aprendem a partir de exemplos, os sistemas de ML são classificados em supervisionado, não supervisionado e semissupervisionado. Na aprendizagem supervisionada, é fornecido ao sistema de aprendizado um conjunto de exemplos com a saída conhecida, ou seja, cada exemplo observado é descrito por um conjunto de atributos, e o pelo valor da classe à qual o exemplo pertence (RUSSEL; NORVIG, 2002). Na aprendizagem não supervisionada, os algoritmos assumem que não se conhece a classe à qual os exemplos pertencem e procuram encontrar nos valores de atributos similaridades ou diferenças que possam, respectivamente, agrupar os exemplos pertencentes à mesma classe ou dispersar os exemplos de classes distintas (RUSSEL; NORVIG, 2002). O aprendizado semissupervisionado combina o modo de aprendizagem supervisionado e não supervisionado, utilizando um pequeno conjunto de exemplos rotulados e um conjunto de exemplos não rotulados.

A representação do conhecimento extraído a partir dos exemplos pode ser feita de várias formas, tais como regras de produção, árvores de decisão, *naïve Bayes*, rede neural artificial e Máquinas de Vetores de Suporte, entre outras. Essas diferentes representações são derivadas dos paradigmas de aprendizado.

O paradigma simbólico utiliza estruturas gráficas ou lógicas para representar o que foi aprendido e podem ser na forma de expressões lógicas, árvores de decisão e regras de produção, entre outras. As representações estatísticas assumem que os valores de

atributos de cada exemplo estão normalmente distribuídos[4], e então usam os dados fornecidos para determinar média, variância, probabilidades, etc. As estruturas conexionistas são representadas por um conjunto de parâmetros interligados por fórmulas matemáticas não triviais, como é o caso das redes neurais artificiais, que são construções matemáticas inspiradas no funcionamento do neurônio biológico (BISHOP, 1995). O paradigma genético é derivado do modelo evolucionário de aprendizagem (HOLLAND, 1975). Em uma analogia direta com a teoria de Darwin, onde as espécies mais adaptadas ao ambiente sobrevivem, um modelo de aprendizagem genético possui uma população de exemplos que competem entre si para fazer a predição de novos exemplos (PRATI, 2006). Similarmente, os operadores genéticos de reprodução, cruzamento, mutação e inversão são aplicados à população para geração de novos indivíduos ou exemplos.

Conforme a disponibilidade dos exemplos necessários para que a máquina possa aprender algo e a forma de aprendizado, os algoritmos de aprendizagem podem ser classificados em (PRATI, 2006): 1) Não incremental, para iniciar a operação do algoritmo de aprendizagem é necessário que todos os exemplos estejam disponíveis simultaneamente; e 2) Incremental, onde não há necessidade de fornecer todos os exemplos simultaneamente, novos exemplos podem ser disponibilizados e submetidos ao algoritmo de aprendizagem.

Na aprendizagem supervisionada, a tarefa de aprendizado é determinada como classificação ou regressão em função do tipo do atributo classe, que pode ser discreto ou contínuo. Quando o rótulo da classe é um valor discreto, a tarefa de aprendizado é chamada classificação (DUDA; HART; STORK, 2001). Caso o rótulo da classe seja um valor contínuo, a tarefa é denominada regressão (DUDA; HART; STORK, 2001).

Tendo em vista os objetivos deste trabalho e a abrangência dos conceitos relacionados ao aprendizado por máquinas, optou-se por discutir os assuntos relacionados à aprendizagem através de indução lógica, modo de aprendizado supervisionado, problemas de classificação, aprendizagem incremental e técnicas híbridas envolvendo aprendizado simbólico e estatístico, em destaque na Figura 1.

[4] A Distribuição Gaussiana é uma das mais importantes distribuições da estatística, inteiramente descrita por seus parâmetros de média e desvio padrão.

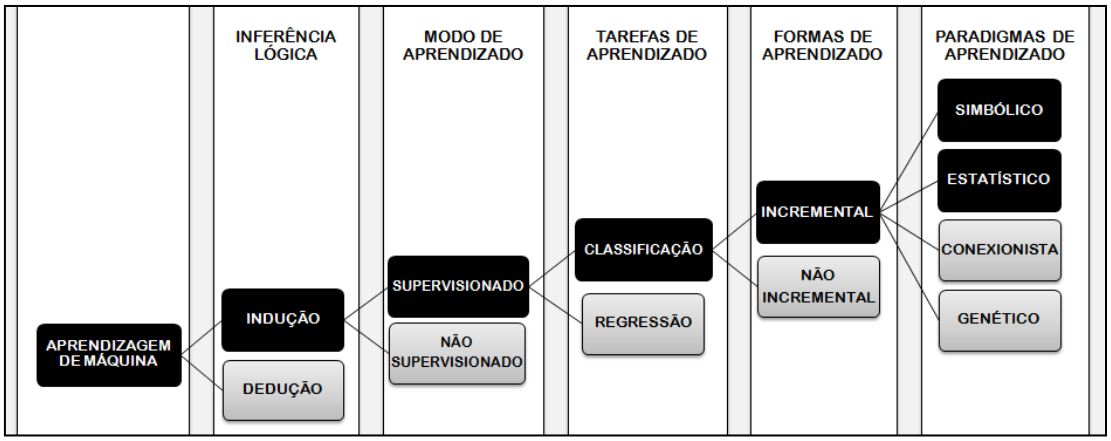


Figura 1 – Conceitos relacionados à aprendizagem de máquina.

Este trabalho procura seguir um padrão de notação e terminologia para tratar questões relacionadas a aprendizado de máquina, que são descritas a seguir.

Um algoritmo de aprendizado supervisionado recebe como entrada um conjunto de exemplos de treinamento $P = \{\rho_1,..., \rho_n\}$, onde n é referente ao número de exemplos do conjunto P. Um exemplo ρ_i, $i \in \{1, 2,..., n\}$, é uma tupla (A, ω_j) que descreve um conjunto finito de d atributos $A = (\alpha_1,...,\alpha_d)$ e uma classe $\omega_j \in \Omega = \{\omega_1, \omega_2,..., \omega_c\}$, $j \in \{1, 2,..., c\}$, onde c refere-se ao número de classes distintas no conjunto P. Seja $V = \{\nu_1,... \nu_m\}$ um conjunto de m valores possíveis para cada atributo α_k, $k \in \{1, 2,..., d\}$, $\nu_{i,k}$ indica o valor do atributo α_k no exemplo ρ_i e $\omega_{i,k}$ indica o valor da classe ω_j no exemplo ρ_i. A Tabela 2 mostra uma representação para os dados de treinamento no formato atributo-valor.

Tabela 2 – Representação em tabela do conjunto de treinamento no formato atributo-valor.

	α_1	...	α_d	Ω
ρ_1	$\nu_{1,1}$	$\nu_{i,k}$	$\nu_{1,d}$	$\omega_{i,k}$
...	...	...	...	...
ρ_n	$\nu_{n,1}$	$\nu_{i,k}$	$\nu_{i,k}$	$\omega_{i,k}$

Para ilustrar um conjunto de treinamento, considere o problema de decidir se as condições do tempo estão favoráveis para jogar tênis. Seja P o conjunto de treinamento da Tabela 3, extraído de (MITCHELL, 1997), o conjunto de atributos A = {Tempo, Temperatura, Umidade, Vento} e as classes Ω = {sim, não}.

Tabela 3 – Conjunto de exemplos de treinamento.

CONJUNTO DE TREINAMENTO					
Exemplos de Treinamento	Atributos				
	α_1 Tempo	α_2 Temperatura	α_3 Umidade	α_4 Vento	Ω Jogar Tênis
ρ_1	ensolarado	quente	alta	fraco	não
ρ_2	ensolarado	quente	alta	forte	não
ρ_3	nublado	quente	alta	fraco	sim
ρ_4	chuvoso	regular	alta	fraco	sim
ρ_5	chuvoso	fria	normal	fraco	sim
ρ_6	chuvoso	fria	normal	forte	não
ρ_7	nublado	fria	normal	fraco	sim
ρ_8	ensolarado	fria	normal	fraco	sim
ρ_9	Ensolarado	Fria	Normal	Fraco	Sim
ρ_{10}	chuvoso	regular	normal	forte	sim
ρ_{11}	ensolarado	regular	normal	forte	sim
ρ_{12}	nublado	regular	alta	forte	sim
ρ_{13}	nublado	quente	normal	fraco	sim
ρ_{14}	chuvoso	regular	alta	forte	não

Seja $\mu_j \in M = \{\mu_1,..., \mu_j\}$ um algoritmo aplicável ao conjunto *P*, onde M é um conjunto de algoritmos de aprendizagem supervisionada. Em uma tarefa de classificação, um algoritmo de aprendizagem μ_j é aplicado a um conjunto de exemplos de treinamento *P* e uma hipótese *h* é gerada. Seja χ um novo padrão a ser classificado, a hipótese *h* deve predizer o valor correspondente à ω_j.

Para avaliar o desempenho da hipótese gerada um conjunto de testes é submetido ao classificador. Seja $T = \{\tau_1,..., \tau_t\}$ um conjunto de testes, τ_l é uma tupla (A, ω_j) tal que $l \in \{1, 2,..., t\}$, onde *t* é o número de exemplos testes no conjunto *T*. Idealmente, o conjunto de teste *T* não deve conter exemplos em comum com o conjunto de treinamento *P*. O classificador pode ser avaliado calculando o erro de classificação. O erro de classificação ou simplesmente taxa erro, ξ, é a medida final do desempenho de um classificador *h*.

A Figura 2 representa o diagrama geral do processo de aprendizado supervisionado para classificação.

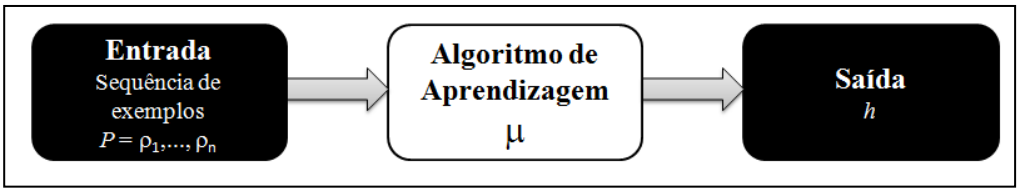


Figura 2 – Processo de aprendizado por máquina simplificado.

2.2. APRENDIZAGEM EM RECONHECIMENTO DE PADRÕES

A aprendizagem de máquina teve sua origem na computação, ao passo que o reconhecimento de padrões tem suas origens na engenharia. No entanto, essas atividades podem ser vistas como duas facetas de mesmo campo (BISHOP, 2006).

O campo da aprendizagem de máquina incorpora o reconhecimento de padrões. O reconhecimento de padrão, de acordo com Theodoridis e Koutroumbas (2006), é a descoberta de regularidades em dados através de algoritmos computacionais e uso dessas informações para classificar objetos em categorias ou classes. O termo genérico “padrão” é utilizado para referir-se a essas regularidades.

Gonzalez e Thomason (1978) definem reconhecimento de padrões como a classificação de um dado de entrada através da seleção de atributos importantes a partir de uma grande quantidade de exemplos de treinamento.

Em linhas gerais, o problema de reconhecimento de padrões é um problema de classificação (DUDA; HART; STORK, 2001). O objetivo do reconhecimento de padrão é a classificação de objetos em categorias ou classes (THEODORIDIS; KOUTROUMBAS, 2006).

A Figura 3 mostra um diagrama geral para o processo de reconhecimento de padrões para aprendizagem supervisionada e tarefa de classificação (DUDA; HART; STORK, 2001). A seguir são descritas as etapas identificadas na Figura 3 de acordo com Duda, Hart e Stork (2001).

O processo de reconhecimento de padrões é dividido basicamente em aprendizagem e classificação: na componente de aprendizagem a finalidade é obter, a partir de um conjunto de exemplos de treinamento *P*, um conjunto de regras ou classificador *h*

capaz de reconhecer um padrão χ; na componente de classificação, o conjunto de regras *h* é utilizado para atribuir uma classe ω_j para novos indivíduos χ.

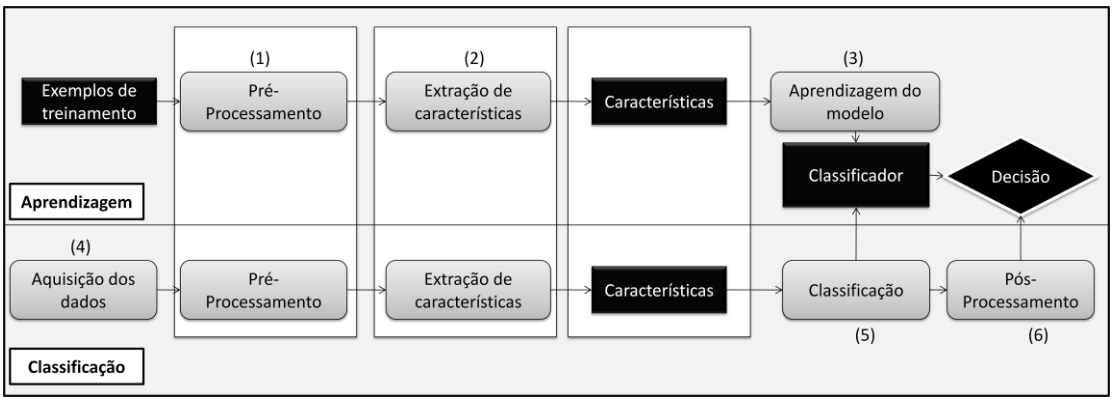


Figura 3 – Diagrama geral para o processo de reconhecimento de padrão.

Na aprendizagem e na classificação são fornecidos, respectivamente, os exemplos de treinamento e os dados a serem classificados, que podem sofrer um pré-processamento. A finalidade do pré-processamento, tanto na fase de aprendizagem quanto de classificação, é ajustar eventuais ruídos nos dados (ex.: substituir valores de atributos inconsistentes, inserir valores de atributos ausentes, etc.) ou realizar qualquer tratamento sobre os dados capturados para simplificar operações subsequentes (ex.: ajustar distorção de imagem).

A extração de características é a tarefa de extrair um subconjunto de atributos de um dado capaz de identificá-lo. Por exemplo, no problema de classificação da flor Íris (FISHER, 1936), o comprimento das pétalas e a largura das sépalas são características relevantes para discriminar uma Íris Virgínica de uma Íris Versicolor. Para classificar essa flor é necessário observar essas características, que são consideradas discriminantes.

Na aprendizagem do modelo, as características discriminantes extraídas na fase anterior são mapeadas entre grupos de padrões por algoritmos de aprendizagem. A forma de mapear as características depende da técnica implementada pelo algoritmo de aprendizagem (ex.: indução de árvore, *naïve* Bayes, etc.). Em linhas gerais, na aprendizagem indutiva supervisionada o modelo é a hipótese gerada a partir do conjunto de exemplos.

Na fase de classificação, a aquisição de dados realiza as medições das variáveis físicas dos dados (ex.: câmera para capturar imagens, microfone para capturar som).

Na componente classificação, as características extraídas dos indivíduos na aquisição dos dados são analisadas em conformidade com o modelo aprendido, e o classificador atribui classes aos novos indivíduos.

Após a classificação, uma avaliação de confiança nas decisões tomadas pelo classificador pode ser realizada no pós-processamento. Nesta etapa também podem ocorrer combinações de classificadores (*Ensemble Classification*).

Para construir um ambiente de aprendizagem para reconhecimento de padrões, conforme descrito acima, o projetista deve seguir um ciclo de projeto, mostrado na Figura 4.

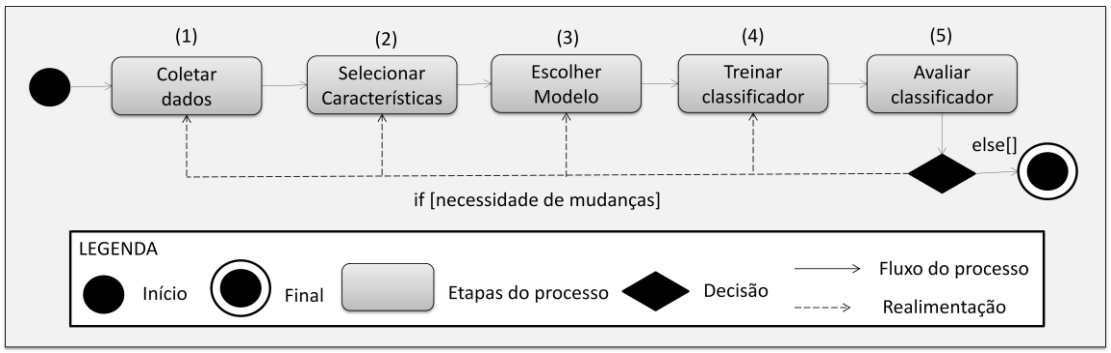


Figura 4 – Ciclo do projeto de reconhecimento de padrões.

O primeiro passo no processo de construção de um sistema para reconhecimento de padrões é definir como será a coleta de dados (ex.: utilizar sensores para capturar som ou medir temperatura). A quantidade de ruídos nos dados coletados depende, entre outros, de aspectos de iluminação e largura de banda. Para realizar experimento no ambiente de aprendizado, também se pode coletar dados a partir de bases de dados (ex.: *Linguistic Data Consortium*[5], UCI *Machine Learning Repository*[6]). Uma questão difícil para o projetista é decidir quando o conjunto de dados coletado é suficientemente grande e representativo para o aprendizado do modelo.

Na seleção de características, o projetista deve decidir quais são as características apropriadas para discriminar os dados coletados. Essa decisão depende do domínio do

[5] Disponível em http://www.ldc.upenn.edu/.
[6] Disponível em http://archive.ics.uci.edu/ml/

problema, como reconhecimento de imagens, reconhecimento da fala e reconhecimento de formas. Por exemplo, na discriminação entre cromossomos (TSAI; FU, 1980), o comprimento do braço de um cromossomo permite distinguir um cromossomo metacêntrico de um cromossomo submetacêntrico, portanto é uma característica capaz de identificar diferentes cromossomos. As características selecionadas nesta fase são as mesmas que devem ser extraídas na fase de aprendizagem e classificação, mostradas anteriormente na Figura 3.

Na escolha do modelo, o projetista deve escolher o método de aprendizagem que será aplicado para o problema de reconhecimento de padrão, tais como: estatísticos, conexionistas, etc. O método de aprendizagem escolhido deriva a forma de representação dos indivíduos, por exemplo, uma matriz, uma árvore de decisão, etc. Em um problema de classificação o modelo é chamado de classificador.

Após definir os métodos e a forma de representar o modelo, o objetivo é projetar a fase de aprendizagem. Nesta fase, o projetista deve decidir como as regras de classificação serão obtidas a partir dos dados de treinamento (ex.: aprendizagem supervisionada, aprendizagem não supervisionada).

Por fim, na avaliação do classificador, o projetista deve testar o classificador para saber o quão bem o modelo escolhido está classificando novos indivíduos, e se necessário refazer as fases anteriores. Em linhas gerais, o projetista pode considerar diferentes aspectos para a avaliação de um classificador, tais como taxa de acerto, facilidade de interpretação do modelo, tempo de treinamento, e outros (DUDA; HART; STORK, 2001).

2.3. MÉTODOS DE APRENDIZADO ESTATÍSTICOS *vs.* DETERMINÍSTICOS.

Na prática, a escolha de um método para a construção de um sistema computacional com habilidades para aprender é um problema difícil. A decisão muitas vezes é baseada em quais métodos estão disponíveis, ou convenientemente, em quais métodos são mais conhecidos pelo projetista (JAIN; DUIN; MAO, 2000).

Contudo, cada método tem o seu domínio de aplicação, sendo um método mais apropriado que os outros, conforme o caso (DUDA; HART; STORK, 2001).

De acordo com Jain, Duin e Mao (2000), os principais métodos de aprendizagem aplicados no reconhecimento de padrões são os métodos baseados em estatística (HASTIE; TIBSHIRANI; FRIEDMAN, 2001) e os métodos sintáticos ou estruturais (PAVLIDIS, 1980). Além desses, existem diversos outros métodos, tais como os métodos difusos (BEZDEK, 1981), baseados na lógica *fuzzy*, e os conexionistas (BISHOP, 1995), porém esses métodos não serão abordados neste trabalho, pois não serão utilizados no desenvolvimento do estudo de caso.

Neste trabalho, os métodos sintáticos ou estruturais também são tratados como métodos determinísticos. O termo "determinístico", neste contexto, se refere não ao fato de não haver mais de uma possível situação seguinte em cada passo da utilização do dispositivo, mas apenas ao fato de que tais passos sejam conduzidos aplicando-se os métodos utilizados a exemplos concretos, e não apenas com base em informações estatísticas.

Os métodos estatísticos são baseados em modelos probabilísticos para geração de padrões. A classificação é ancorada em estimativas e na teoria da decisão (DUDA; HART; STORK, 2001; HASTIE; TIBSHIRANI; FRIEDMAN, 2001). No modelo de aprendizagem baseado em estatística, cada padrão é representado em termos de d características ou medições, onde d é o número de características ou atributos. A meta é escolher atributos que separem vetores de padrão, pertencentes a diferentes categorias, em regiões compactas. A eficácia da representação do espaço (conjunto de atributos) é determinada por quão bem os padrões de classes distintas podem ser separados. Dado um conjunto de padrões de treinamento, o objetivo é estabelecer fronteiras de decisão no espaço de característica. As fronteiras de decisão separam padrões pertencentes a diferentes categorias (JAIN; DUIN; MAO, 2000).

Os métodos estatísticos pressupõem uma distribuição de probabilidade associada ao espaço de características. A aprendizagem é formulada em termos estatísticos e probabilísticos. Os métodos estatísticos utilizam variáveis aleatórias sobre um espaço amostral. Quanto menor a variância das variáveis aleatórias, maior a precisão. Assim, para soluções de alta precisão a variância deve ser muito pequena (DUDA; HART; STORK, 2001).

Métodos estatísticos buscam encontrar padrões em um espaço amostral sobre uma massa de dados. Os métodos estatísticos são apropriados quando se mostram irrelevantes as informações específicas acerca de cada indivíduo, pois esses métodos massificam as amostras de treinamento e eliminam a importância do indivíduo. Um indivíduo pode ser representado por um exemplo particular de treinamento ou um novo padrão a ser reconhecido. A meta dos sistemas de aprendizado baseado em métodos estatísticos é reconhecer novos indivíduos com uma taxa de erro aceitável. Em geral, os métodos estatísticos são adequados para evidenciar tendências no espaço amostral, tais como as tendências de mercado na bolsa de valores.

Bunke e Kandel (1990) consideram que a utilização de métodos estatísticos para a análise de padrões complexos exige o uso de recursos adicionais, tais como a utilização de métodos determinísticos auxiliares.

Os métodos determinísticos são apropriados quando o processo de aprendizagem envolve padrões complexos (JAIN; DUIN; MAO, 2000). Métodos determinísticos pressupõem que um padrão a ser aprendido seja decomposto em subpadrões ou primitivas (FU, 1982). Cada padrão é representado por um conjunto de primitivas e suas relações. No reconhecimento de novos indivíduos, os métodos determinísticos buscam mapear a estrutura relativa aos indivíduos. Cada indivíduo torna-se, desta maneira, relevante e particular, podendo proporcionar alta precisão na identificação de novos indivíduos.

Para Jain, Duin e Mao (2000), os métodos determinísticos no reconhecimento de padrões são intuitivamente atraentes por fornecerem a descrição do padrão a partir das primitivas. Fu (1982) considera os métodos sintáticos e estruturais apropriados para problemas de reconhecimento de padrões, quando os padrões apresentam uma estrutura bem definida. Em particular, uma das dificuldades na utilização dos métodos determinísticos está na extração de primitivas a partir de um indivíduo.

Entretanto, além da aplicação individual de cada método em diferentes domínios, existe a possibilidade de combinar métodos. A combinação de métodos pode agregar as vantagens e capacidades de diferentes métodos (JAIN; DUIN; MAO, 2000).

Em particular, métodos estatísticos utilizados de maneira independente não são adequados para a solução de problemas de aprendizagem que necessitem

fundamentalmente levar em conta cada indivíduo e seus relacionamentos (JAIN; DUIN; MAO, 2000). Por outro lado, utilizar métodos determinísticos para investigar uma grande quantidade de exemplos, quando cada indivíduo e suas relações são relevantes, pode exigir um alto custo computacional (JAIN; DUIN; MAO, 2000). Nestes casos, uma solução promissora é a utilização de métodos híbridos.

Um método híbrido costuma ser apropriado para a busca de uma solução quando se tem um problema de natureza estruturada e dinâmica. Também é adequado nas situações em que as relações entre os indivíduos, bem como seu comportamento individual e coletivo, são de grande importância. Para exemplificar, suponha a aplicação de métodos estatísticos e determinísticos sobre uma grande quantidade de exemplos de treinamento. Os métodos estatísticos têm a função de localizar a região provável da solução procurada, convergindo rapidamente para o espaço de possíveis soluções. Na sequência, métodos determinísticos têm a função de refinar a solução do processo, considerando os aspectos estruturais relacionados aos indivíduos.

Neste trabalho, não se propõe um estudo exaustivo dos métodos de aprendizagem, por isso são apresentados apenas alguns métodos, usualmente adotados na implementação de algoritmos de aprendizado.

Um estudo comparativo sobre as técnicas utilizadas para a construção de classificadores e suas aplicações é mostrado em (STANGE; NETO, 2010). Neste trabalho é apresentada uma proposta para a construção de classificadores, baseada em métodos híbridos incluindo métodos adaptativos.

2.3.1. Método Bayesiano e naïve Bayes

O método bayesiano é baseado na teoria da decisão estatística e utiliza o Teorema de *Bayes* para determinar a distribuição da probabilidade de padrões pertencentes a cada classe (DUDA; HART; STORK, 2001; MITCHELL, 1997).

A seguir uma breve descrição dos conceitos de probabilidade e estatística necessários para o entendimento do método Bayesiano.

Um experimento (*E*) é um processo em que o resultado não é conhecido antes de ser observado, porém no qual os possíveis resultados são conhecidos (ex.: resultado do

lançamento de um dado). O espaço amostral (L_E) é conjunto de todos os possíveis resultados do experimento E (ex.: as seis faces do dado {1, 2, 3, 4, 5, 6}). Um evento I ($I \subseteq L_E$) é um subconjunto qualquer do espaço amostral (ex.: As faces pares {2, 4, 6}).

Probabilidade é uma medida da chance de ocorrência de um fenômeno de interesse. A probabilidade associada a um evento I é proporcional ao tamanho do conjunto associado ao evento em relação ao tamanho do espaço amostral (ex.: probabilidade de ocorrer uma face par de um dado: $P(I) = \frac{3}{6}$, *P(I)* denota a probabilidade do evento *I*).

A probabilidade condicional de um evento é probabilidade de ocorrência de um evento *x* quando se dispõe da informação que outro evento ocorreu.

O método bayesiano supõe que a probabilidade $P(\omega_i)$ de cada classe ω_i e as densidades de probabilidade condicionais $\rho(\alpha|\ \omega_i)$ de α com respeito a cada uma das classes ω_i, i = 1, 2,...,c, são fornecidas. Com as condicionais, o Teorema de Bayes é aplicado e a probabilidade $P(\omega_i|x)$ calculada (DUDA; HART; STORK, 2001; THEODORIDIS; KOUTROUMBAS, 2006), assim:

$$P(\omega_i|x) = \frac{P(\omega_i)\rho(x|\omega_i)}{\rho(x)}$$

(1) Equação

$$\rho(x) = \sum_{i\neq j}^{c} P(\omega_j)\ \rho(x|\ \omega_j)$$

(2) Equação

Onde:

– $P(\omega_i)$ é a probabilidade a *priori*, que expressa o conhecimento sobre os parâmetros antes de examinar os dados;

– $\rho(x|\omega_i)$ é a densidade condicional ou verossimilhança. A função densidade de probabilidade associa cada possível valor da variável aleatória x à sua probabilidade de ocorrência;

– ρ (x) é a evidência, onde x são valores observados; e

– $P(\omega_i|x)$ é probabilidade a *posteriori*.

Como exemplo é apresentado uma das formas de representar o conhecimento adquirido através de métodos Bayesianos, trata-se do classificador *naïve Bayes* (DUDA; HART; STORK, 2001; MITCHELL, 1997).

O *naïve Bayes* é utiliza um modelo simples de classificação, porém proporciona bons resultados mesmo quando comparados aos obtidos com classificadores mais complexos (DUDA; HART; STORK, 2001).

Seja o conjunto de treinamento para o problema de decidir se as condições do tempo estão favoráveis para jogar tênis. apresentado na Tabela 3, o processo de construção do *naïve Bayes* é a seguir:

1: Calcular a probabilidade de cada classe ω_i ou a priori utilizando a fórmula:

$$P(\omega_i) = \frac{\text{nº de instancias } \omega_i}{\text{nº de instancias de treinamento}}$$

$$\text{Exemplo} = \begin{cases} P(sim) = \frac{9}{14} \\ P(não) = \frac{5}{14} \end{cases}$$

2: Calcular a probabilidade de cada atributo do conjunto $A = \{\alpha_1,..., \alpha_d\}$ em relação à possível classe do conjunto $\Omega = (\omega_1,...\omega_c)$, ou densidade condicional, aplicando a fórmula:

$$\rho(\alpha_j = v_k|\omega_i) = \frac{\text{nº de inst. com } \alpha_j = v_k \text{ para } \omega_i}{\text{nº de inst. com } \omega_i}$$

$$\text{Exemplo} = \rho(\text{ensolarado}|\text{sim}) = \frac{2}{9}$$

3: Calcular cada probabilidade da classe de χ usando o Teorema de Bayes, a *posteriori* com a equação:

$$P(\omega_i|\mathrm{A}) = \frac{\mathrm{P}(\mathrm{A}|\omega_i)\mathrm{P}(\omega_i)}{\mathrm{P}(\mathrm{A})}$$

Como o objetivo é maximizar $P(\omega_i|\mathrm{A})$, escrevemos:

$$\arg\max \mathrm{P}(\omega_\mathrm{i}|\mathrm{A}) = \arg\max \mathrm{P}(\mathrm{A}|\omega_\mathrm{i})\mathrm{P}(\omega_\mathrm{i})$$

Considerando que os atributos são independentes, o cálculo de $P(A|\omega_i)$ é transformado em:

$$P(A|\omega_i) = P(\alpha_1|\omega_i) \ldots P(\alpha_d|\omega_i)$$

O resultado final da fórmula do classificador *naïve* Bayes:

$$\arg\max P(\omega_i|A) = \arg\max \prod_d P(\alpha_d|\omega_i)\, P(\omega_i)$$

A fórmula obtém o resultado de maior probabilidade considerando cada ω_i, é a possível classe da instância.

4: Para finalizar, classificar o novo padrão de entrada χ = {ensolarado, frio, normal, forte}.

Calcular $P(\omega_1|A)$ e $P(\omega_2|A)$:

$$P(\omega_1|A) = P(\alpha_1|\omega_1)P(\alpha_2|\omega_1)P(\alpha_3|\omega_1)P(\alpha_4|\omega_1)P(\omega_1)$$
$$= 0.33\text{x}0.33\text{x}0.66\text{x}0.22\text{x}0.64 = 0{,}0101$$

$$P(\omega_2|A) = P(\alpha_1|\omega_2)P(\alpha_2|\omega_2)P(\alpha_3|\omega_2)P(\alpha_4|\omega_2)P(\omega_2)$$
$$= 0.6\text{x}0.2\text{x}0.2\text{x}0.6\text{x}0.36 = 0{,}0051$$

5: Conclusão: $P(\omega_1|A) > P(\omega_2|A)$ então a provável classe de χ é "sim". Assim, se $P(\omega_1|A) > P(\omega_2|A)$ então classificar χ com o valor de ω_1 senão classificar χ com o valor de ω_2

2.3.2. Indução de Árvores de Decisão

A indução de árvores de decisão é considerada uma das formas mais simples de escrever programas de aprendizagem, apesar disso, em geral, obtém bons resultados (MITCHELL, 1997; RUSSEL; NORVIG, 2002).

Na árvore de decisão, o primeiro nó é chamado raiz e por convenção fica no topo da árvore. A partir do nó raiz, sucessivas ramificações (também chamadas de links ou ramos) são ligadas a outros nós, chamados de nós internos. As ligações ocorrem até atingir um nó terminal, ou folha.

Para tomar uma decisão utilizando a árvore, o dispositivo é percorrido da raiz para as folhas, testando os valores dos atributos em nós sucessivos e, quando uma folha é alcançada a decisão é tomada (WITTEN; EIBE, 2005).

Diversos algoritmos de indução de árvores foram propostos, tais como ID3 (QUINLAN, 1986), C4.5 (QUINLAN, 1993), ID5 (UTGOFF, 1989), ITI (UTGOFF; BERKMAN; CLOUSE, 1997), etc. Uma família de algoritmos bastante popular para a construção de uma árvore decisão a partir de um conjunto de treinamento é chamada *Top-Down Induction of Decision Tree* (TDIDT) (QUINLAN, 1986). O arcabouço do algoritmo TDIDT é descrito a seguir na Tabela 4, e se baseia em três possibilidades sobre um conjunto de P contendo as classes $\Omega = \{\omega_1, \omega_2, ..., \omega_c\}$.

Tabela 4 – Método TDIDT para a construção de árvores de decisão.

CONSTRUÇÃO DA ÁRVORE DE DECISÃO	
1:	Se P contém um ou mais exemplos, todos pertencentes à mesma classe ω_i, então a árvore de decisão para P é um nó folha que identifica a classe ω_i;
2:	Se P não contém exemplos então a árvore é uma folha e a classe deve ser determinada a partir de informações externas (ex. escolher a classe baseada no conhecimento do domínio do problema);
3:	Se P contém exemplos que pertencem a diferentes classes então dividir P em subconjuntos $S_1,..., S_n$ que são, ou tendem a representar classes únicas;
4:	Os passos 1, 2 e 3 são aplicados recursivamente para cada um dos subconjuntos S_i, com i variando de 1 até n.

Essas considerações sugerem um processo conhecido como particionamento recursivo para construção de árvores: dado um conjunto de exemplos de treinamento em um nó, declarar o nó como folha, ou encontrar outra maneira de dividir o conjunto em um novo subconjunto. Essa maneira de dividir o conjunto é uma das características que

diferem entre si os algoritmos de indução de árvores.

TDIDT é base para vários algoritmos de indução de árvores de decisão, dentre os mais populares podemos citar o ID3, C4.5 e CART (BREIMAN et al., 1984).

Em particular, o ID3 utiliza uma medida estatística para escolher o atributo que melhor divide os exemplos de treinamento, chamada de ganho de informação. O objetivo é produzir subconjuntos mais "puros", ou seja, com menos exemplos pertencentes a classes distintas. Para entender a medida utilizada pelo ID3 será apresentado o conceito de entropia e sua relação com ganho de informação.

A entropia é uma medida comumente usada em teoria de informação que caracteriza a homogeneidade de uma coleção arbitrária de exemplos (MITCHELL, 1997). Seja *L* um subconjunto contendo exemplos retirados de um conjunto $P = \{\rho_1,..., \rho_j\}$. Se todos os elementos de *L* são iguais entre si, a entropia é mínima. O valor máximo de entropia é obtido quando todos os exemplos de *P* aparecem em igual quantidade em *L*.

A entropia é inversamente proporcional à quantidade de informação (PISTORI, 2003). Quando todos os elementos do conjunto *L* possuem o mesmo valor do atributo classe, ou ω_j (entropia mínima) pode-se concluir com 100% de possibilidade de acerto que um exemplo retirado aleatoriamente de *L* terá valor ω_j (quantidade máxima de informação). Por outro lado, se a entropia de *L* é máxima, a possibilidade de acerto do valor ω_j de um exemplo retirado aleatoriamente de P é de $1/ |P|$.

Para ilustrar, seja *P* o conjunto de treinamento da Tabela 3, extraído de (MITCHELL, 1997), S_+ um subconjunto de *P* com exemplos de treinamento pertencentes à classe ω_1 = "sim" e S_- um subconjunto de *P* com exemplos de treinamento pertencentes à classe ω_2 = "não". A proporção de exemplos pertencentes à classe ω_1 é P_+ e proporção de exemplos pertencentes à classe ω_2 é P_-. É calculada a entropia de $S = \{S_+, S_-\} \in P$ com relação aos exemplos classificados como $\Omega = \{\omega_1, \omega_2\}$ através da fórmula (MITCHELL, 1997):

$$\text{Entropia (S)} = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

Aplicando a fórmula ao subconjunto S temos:

$$\text{Entropia (S)} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

A entropia possui o valor numérico 0 (zero), se todos os membros de S pertencem à mesma classe e valor 1 (um), quando S contém um número igual de exemplos S_+ e S_-. Quando S contém números desiguais de exemplos S_+ e S_-, a entropia está entre 0 e 1.

O ganho de informação mede a redução da entropia causada pela divisão dos exemplos de *P* de acordo com os valores do atributo de $A = (\alpha_1,...,\alpha_d)$ (MITCHELL, 1997). O melhor atributo é aquele com o maior ganho de informação. Mais precisamente, o ganho de informação, *Ganho(S, A)* de um atributo *A*, relativo a um conjunto de exemplos S, é definido como (MITCHELL, 1997):

$$Ganho(S, A) = Entropia(S) - \left(\sum_{v \in V} \frac{|S_v|}{|S|}. Entropia(S_v)\right)$$

Considere o ganho de informação do atributo "Vento" e V = {forte, fraco} temos:

$$Ganho(S, \alpha) = Entropia(S) - \left(\frac{S_{forte}}{S} Entropia(S_{forte}) + \frac{S_{fraco}}{S} Entropia(S_{fraco})\right)$$

Calculando a entropia do atributo "Vento" em relação aos valores de atributo "fraco" e "forte", temos:

$$\text{Entropia}\left(S_{fraco}\right) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.811$$

$$\text{Entropia}\left(S_{forte}\right) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$$

Calculando o ganho de informação para o atributo "Vento", temos:

$$Ganho(S, "Vento") = 0.94 - \left(\frac{8}{14} * 0.811 + \frac{6}{14} * 1\right) = 0.048$$

O mesmo procedimento é aplicado aos atributos "Tempo", "Umidade" e "Temperatura", que possuem respectivamente, $Ganho(S, "Tempo") = 0.24$, $Ganho(S, "Umidade") = 0.151$, $Ganho(S, "Temperatura") = 0.029$. Na primeira iteração do algoritmo, o atributo "Tempo" é escolhido como o atributo que melhor particiona o conjunto S, portanto é definido como nó raiz da árvore. Um ramo para cada possível valor do atributo "Tempo" é anexado ao nó (ex.: ensolarado, nublado e

chuvoso).

O processo de selecionar um atributo e dividir os exemplos de treinamento repete-se a cada nó descendente não folha. Os atributos que já estiverem incorporados na árvore (ex.: “Tempo”) são excluídos do cálculo, de modo que aparecem apenas uma vez na árvore. O processo continua para cada novo nó descendente até que uma das duas condições seja atingida: todos os atributos foram incluídos na árvore ou os exemplos de treinamento associados a este nó possuem o mesmo valor de atributo (ex.: entropia é zero)

O resultado final do algoritmo constrói a árvore de decisão da Figura 5.

Após a geração da árvore, a primeira pergunta a ser feita ao jogador poderia ser se o tempo está ensolarado, chuvoso ou nublado. A seguir, dependendo da resposta outras perguntas podem ser feitas até que se chegue a uma conclusão, obtida em um nó terminal.

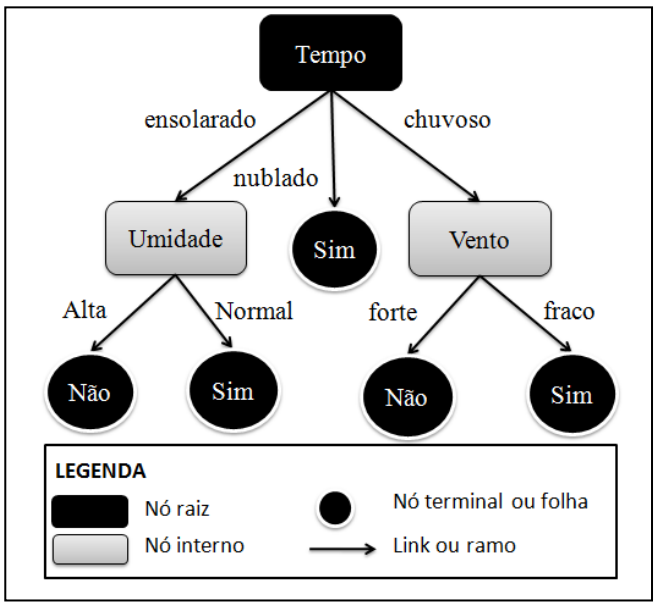


Figura 5 – Árvore de Decisão construída a partir do ID3.

3 TECNOLOGIA ADAPTATIVA

Este capítulo tem a finalidade de descrever sucintamente algumas técnicas adaptativas que têm sido desenvolvidas por pesquisadores em tecnologia adaptativa.

Acredita-se que esta descrição é suficiente para o entendimento desta dissertação, porém, se necessário um estudo mais detalhado destas técnicas, recomenda-se a leitura das referências indicadas no texto.

A pesquisa de técnicas adaptativas tem a finalidade de propor soluções alternativas para diversos problemas utilizando dispositivos adaptativos. Um dispositivo adaptativo é composto por um dispositivo guiado por regras e um mecanismo subjacente que permite a modificação do conjunto de regras.

Um dispositivo guiado por regras é qualquer abstração formal que tem seu comportamento descrito por um conjunto finito de regras (NETO, 2009). Uma formulação para um dispositivo guiado por regras encontra-se resumida na Tabela 5 e sua formulação completa, em (NETO, 2001).

Tabela 5 – Formulação de um dispositivo guiado por regras.

FORMULAÇÃO - DISPOSITIVO GUIADO POR REGRAS	
$ND = (C, R, S, c_0, A)$	
C	Conjunto de todas as possíveis configurações do dispositivo;
R	Relação de mudança de configuração, $R \subseteq C \times (S \cup \{ \varepsilon \}) \times C$;
S	Conjunto de estímulos de entrada do dispositivo;
c_0	Configuração inicial única do dispositivo, pertence ao conjunto C;
A	Conjunto das configurações de aceitação do dispositivo.

Os dispositivos adaptativos surgiram da busca por formalismos simples de usar, como os autômatos de estados finitos, porém capazes de representar problemas mais complexos de teoria da computação (NETO, 2000).

No formalismo adaptativo, dispositivos mais poderosos, em relação à capacidade de expressão, podem ser obtidos a partir da incorporação de um mecanismo adaptativo atrelado aos recursos oferecidos por um dispositivo mais simples (PISTORI, 2003). O mecanismo adaptativo é representado por um conjunto de funções adaptativas, as quais são conectadas às regras do dispositivo convencional. As funções adaptativas são capazes de alterar convenientemente o conjunto de regras do dispositivo

convencional, também chamado de subjacente, provocando uma mudança de comportamento no dispositivo adaptativo.

Uma formulação para os dispositivos adaptativos é mostrada na Tabela 6 (NETO, 2009).

Tabela 6 – Formulação de um dispositivo adaptativo básico.

FORMULAÇÃO - DISPOSITIVO ADAPTATIVO BÁSICO	
AD = (C, RA, S, AA, c_0, RA_0, A)	
C	Conjunto de todas as possíveis configurações do dispositivo adaptativo;
RA	Conjunto de todas as regras adaptativas do dispositivo adaptativo;
S	Conjunto de estímulos de entrada do dispositivo adaptativo;
AA	Conjunto de funções adaptativas φ, incluindo a função nula λ;
c_0	Configuração inicial $c_0 \in C$ do dispositivo adaptativo, que deve ser única;
RA_0	Conjunto inicial de regras adaptativas do dispositivo adaptativo;
A	Conjunto de todas as configurações de aceitação do dispositivo adaptativo;

As funções adaptativas são descritas por um cabeçalho e um corpo, similar às funções em linguagens de programação, conforme Tabela 7.

Tabela 7 – Declaração das funções adaptativas.

DECLARAÇÃO - FUNÇÃO ADAPTATIVA φ	
CABEÇALHO	
Nome φ	Uma função adaptativa é referenciada por um nome;
Parâmetros	Ênupla ordenada (ρ_1, ρ_2,..., ρ_p), de p≥0 parâmetros formais;
Variáveis	Um conjunto {υ_1, υ_2,..., υ_v}, de $v \geq 0$ variáveis. Os valores das variáveis são preenchidos uma única vez pelas ações adaptativas de consulta;
Geradores	Um conjunto {χ_1, χ_2,..., χ_g}, de $g \geq 0$ geradores. Os valores dos geradores são preenchidos com novos valores a cada chamada da função adaptativa.
CORPO (βμ, Δ, αν)	
Função adaptativa anterior	Chamada βμ de uma função adaptativa anterior (μ) executada antes da aplicação das ações adaptativas elementares definidas no conjunto Δ.
Função adaptativa posterior	Chamada αν de uma função adaptativa posterior (ν) executada após a aplicação das ações adaptativas elementares definidas no conjunto Δ.
Núcleo da função adaptativa	Núcleo da função adaptativa φ consiste do conjunto $\Delta = \{\delta_1, \delta_2,..., \delta_e\}$, contendo $e \geq 0$ ações adaptativas elementares δ_i, para $0 \leq i < e$.

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 consulta, remoção e inserção de regra, apresentada na Tabela 8.

Tabela 8 – Ações adaptativas elementares.

AÇÕES ADAPTATIVAS ELEMENTARES δ_i	
	$\delta_i = \otimes [r]$
$\otimes$	Representa um operador aplicado à regra r e ao conjunto RA_t de regras que definem o comportamento do dispositivo adaptativo AD.
RA_t	Conjunto RA no passo t de operação do dispositivo AD.
r	Padrão para a regra sobre a qual deve incidir a ação adaptativa elementar.
	OPERADORES $\otimes$
Inserção +	Inclui no conjunto RA_t uma nova regra r.
Remoção -	Elimina do conjunto RA_t a regra r.
Consulta ?	Pesquisa em RA_t o padrão r e preenche variáveis com os resultados da busca.

Têm sido alvos frequentes de pesquisa os seguintes dispositivos adaptativos: os autômatos adaptativos (NETO, 1993), as árvores de decisão adaptativas (PISTORI, 2003) e as tabelas de decisão adaptativa (NETO, 2001; TCHEMRA, 2009).

3.1. AUTÔMATOS ADAPTATIVOS

Autômato Adaptativo (AA) é uma classe de autômatos com uma característica peculiar, a capacidade de poder modificar a sua própria estrutura durante sua execução (NETO, 2001). Essas alterações são realizadas através de ações adaptativas associadas a algumas das transições do autômato que permitem a consulta, inserção e remoção de transições.

O modelo formal dos autômatos adaptativos pode ser representado com base nos dispositivos adaptativos guiados por regras, cujo dispositivo subjacente é um autômato (NETO, 2001).

A fim de simplificar a definição de autômato, considere uma máquina de estados finitos que partindo de um estado inicial e consumindo uma cadeia de entrada, transita entre seus estados de acordo com um conjunto de regras de transição. Se após consumir uma cadeia de entrada, um autômato parar sobre um de seus estados finais, diz-se que a cadeia é reconhecida pelo autômato. Ao conjunto de cadeias que a máquina reconhece dá-se o nome de linguagem, de modo que cada tipo de máquina (autômato finito, Máquina de Turing, etc.) reconhece apenas uma classe de linguagens específica. Essas classes de linguagens são definidas com base na Hierarquia de Chomsky (LEWIS; PAPADIMITRIOU, 2000) que determina, para cada classe de linguagem, um conjunto de regras gramaticais que, quando aplicadas,

geram apenas cadeias daquela classe.

A representação gráfica de um Autômato Adaptativo é mostrada na Figura 6. Na figura, as funções adaptativas associadas a algumas das transições do autômato permitem a edição das transições entre seus estados.

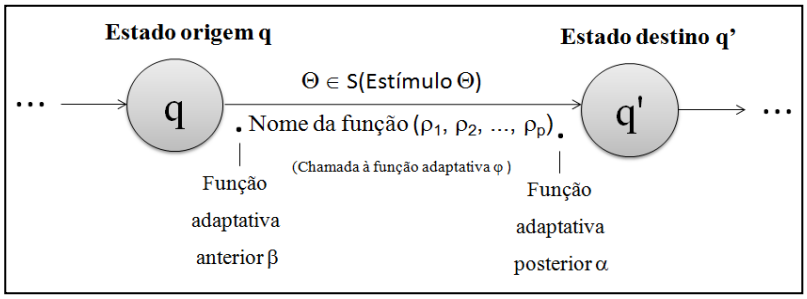


Figura 6 – Representação gráfica de um Autômato Adaptativo.

Autômatos Adaptativos possuem as mesmas características das máquinas de estados finitos, porém são capazes de computar a mesma classe de linguagens que as Máquinas de Turing. É provado em (ROCHA; NETO, 2001) que os Autômatos Finitos Adaptativos são equivalentes, em poder computacional, a Máquinas de Turing.

A seguir, na Tabela 9 a descrição de uma técnica proposta em (NETO; IWAI 1998) para a construção do autômato.

Tabela 9 – Passos para a construção de um autômato adaptativo

CONSTRUÇÃO DO AUTÔMATO ADAPTATIVO	
1:	Iniciar com um autômato conhecido;
2:	Modificar o autômato incrementalmente para aceitar sucessivos casos de amostras positivas;
3:	Modificar o autômato incrementalmente para rejeitar sucessivos casos de amostras negativas;
4:	Repetir as ações até que o treinamento seja considerado aceitável.

3.2. ÁRVORES DE DECISÃO ADAPTATIVAS

Em (PISTORI, 2003; PISTORI, H.; NETO, 2003a), 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 comum 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 anexadas a alguns ramos da árvore que permitem a inserção e/ou remoção de subárvores.

Um caso especial de árvore de decisão adaptativa obtida pelo algoritmo *AdapTree* (PISTORI, 2003; PISTORI; NETO, 2002). O *AdapTree* é um algoritmo incremental de aprendizagem de máquina supervisionado, que permite intercalar as fases de treinamento e teste da árvore. Outra característica relevante do *AdapTree* é o tratamento de valores contínuos, ausentes e inconsistentes[7] do conjunto de exemplos de treinamento.

O tratamento de valores contínuos é feito através da discretização, baseado no método de discretização de Fayyad e Irani (FAYYAD; IRANI, 1993). Os valores ausentes são substituídos, através de um pré-processamento do conjunto de treinamento, aplica-se neste caso o mesmo método usado no algoritmo CN2 (CLARK; NIBLETT, 1989). Valores inconsistentes são tratados por contadores, associados às folhas da árvore de decisão adaptativa. O contador permite que o valor da classe associado a cada folha seja sempre aquele de maior frequência entre os exemplos inconsistentes. Citando Pistori (2003) [...] "no caso de valores inconsistentes, a árvore "decidirá heuristicamente" pela classe mais comum, entre os exemplos inconsistentes". A formulação da árvore de Decisão Adaptativa é apresentada na Tabela 10 (PISTORI, 2003).

Tabela 10 – Formulação da Árvore de Decisão Adaptativa.

FORMULAÇÃO – ADAPTREE	
AD = ((P, A, V, Ω, ω_j, R), AA)	
P	Conjunto de exemplos de treinamento, $P = \{\rho_1,..., \rho_n\}$.
A	Conjunto de atributos, $A = (\alpha_1,...,\alpha_d)$.
V	Conjunto dos valores possíveis de cada atributo, $? \subseteq V$.
Ω	Conjunto das possíveis classes do conjunto P, $\Omega = \{\omega_1, \omega_2,..., \omega_c\}$.
ω_j	A classe de um exemplo $\rho_i \in P$.
R	Estrutura hierárquica finita, denominada subárvore.
AA	Conjunto de funções adaptativas $AA = \{\varphi_1, \varphi_2, ...,\varphi_d\}$, uma para cada atributo A.

O mecanismo de construção da árvore é apresentado, sucintamente, na Tabela 11, de acordo com (PISTORI, 2003).

[7] Pistori (2003) chama de exemplos de treinamento inconsistentes, dois exemplos que apresentam o mesmo valor para todos os atributos, exceto para o atributo classe.

Tabela 11 – Passos para a construção de um autômato adaptativo.

CONSTRUÇÃO DA ÁRVORE DE DECISÃO	
1:	Iniciar a construção com uma árvore com um nó folha contendo um valor ausente "?", uma função adaptativa φ_i e conjunto de treinamento $P = \{\rho_1,..., \rho_n\}$.
	Função Adaptativa φ_i { ?[(ρ_i; *nome_do_atributo_classe*); ? α_k)] - Obtém o valor do atributo classe. +[(*r; ?α_k)] - Adiciona folha (substituindo a subárvore que gerou a chamada desta função adaptativa), *r é um gerador de nós}
2:	À medida que os exemplos α_k vão sendo lidos, a árvore vai crescendo.
3:	Repetir as ações até que árvore atinja a altura |A|.

A Figura 7 mostra a representação gráfica de uma árvore de decisão adaptativa criada a partir dos exemplos de treinamento da Tabela 3.

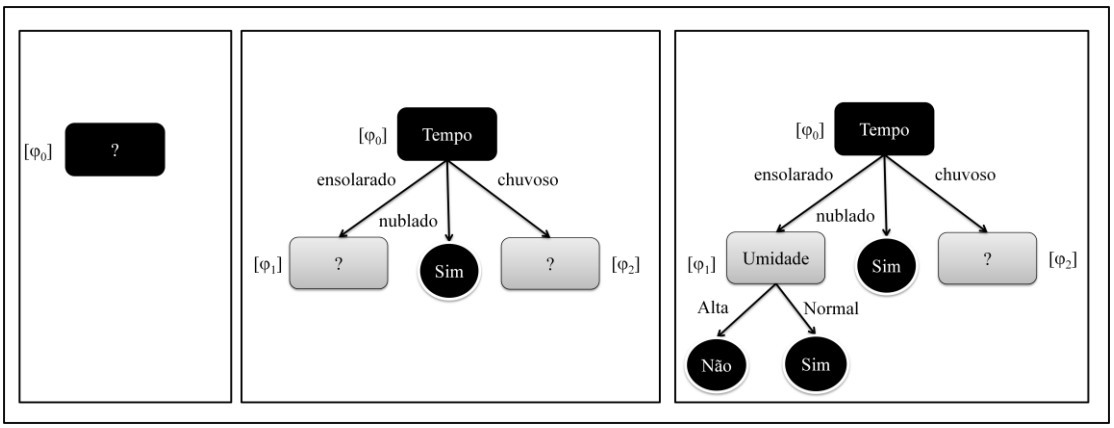


Figura 7 – Evolução do algoritmo *AdapTree* durante o aprendizado.

3.3. TABELAS DE DECISÃO ADAPTATIVAS.

Nas tabelas de decisão adaptativas o dispositivo subjacente é uma tabela de decisão convencional. Na representação gráfica, conforme a Tabela 12, uma tabela de decisão convencional é composta por colunas que representam conjuntos de regras associadas a condições e ações (HUGHES; SHANK; STEIN, 1968). A primeira coluna, partindo da primeira linha da tabela, representa um conjunto de condições e, a seguir, encontra-se um conjunto de ações. A tabela de decisão opera verificando as condições de acordo com os valores definidos nas colunas de regras. Quando a condição é satisfeita de acordo com uma determinada regra, essa regra é considerada válida e todas as ações a ela associadas são executadas.

Uma versão adaptativa de uma tabela de decisão convencional pode ser obtida adicionando-se a ela linhas, onde são incluídas as funções adaptativas. Além disso, a

cada coluna que representa uma regra simples, deve ser adicionada uma chamada para uma função adaptativa associada à execução de uma regra em particular. Com isso, sempre que uma regra adaptativa é aplicada, uma ação adaptativa é invocada, permitindo que sejam feitas mudanças no conjunto de regras.

Tabela 12 – Tabela de decisão convencional.

		$\mathbf{r_1}$	$\mathbf{r_2}$	**...**	$\mathbf{r_k}$
Condições	condição$_1$	c_{11}	c_{21}	...	c_{k1}
	...	...	...	...	...
	condição$_n$	c_{1n}	c_{2n}	...	c_{kn}
Ações	acão$_1$	a_{11}	a_{21}	...	a_{k1}
	...	...	...	...	...
	acão$_m$	a_{1m}	a_{2m}	...	a_{kn}

A formulação da tabela de decisão é apresentada na Tabela 13.

Tabela 13 – Formulação da tabela de decisão convencional.

FORMULAÇÃO – TABELA DE DECISÃO CONVENCIONAL	
	TD = (CT, R, CV, t_0, AT, A)
CT	Conjunto de todas as configurações possíveis da tabela de decisão;
R	Conjunto finito de regras de decisão: R = $\{r_i, 1 \leq i \leq n\}$. Cada regra $r_i = (c_{ij}, a_{ik}) \in R$, onde: c_{ij} representa um valor para a condição c_j na regra r_i, a_{ik} representa um valor para a ação a_k da regra r_i;
CV	Conjunto finito dos valores c_{ij} válidos para as condições C_j
t_0	Configuração inicial da tabela de decisão;
AT	Conjunto de configurações aceitas da tabela de decisão;
A	Conjunto finito de alternativas do problema;

Uma tabela de decisão adaptativa, cujo formalismo está apresentado em (NETO, 2001; TCHEMRA, 2009), é um dispositivo guiado por regras que permite modificações dinâmicas no conjunto de regras que compõe a tabela de decisões, através de ações adaptativas. As tabelas de decisão adaptativas são formadas por um conjunto de condições, ações, regras e funções adaptativas. A estrutura geral gráfica de uma tabela de decisão adaptativa é apresentada na Tabela 14, baseada no formato descrito em (NETO, 2001) e na tabela de decisão convencional descrita em (HUGHES; SHANK; STEIN, 1968).

Na Tabela 14, as linhas que compõem a tabela de decisão TD correspondem ao dispositivo guiado por regras. O mecanismo adaptativo é obtido acrescentando um

conjunto de linhas na tabela para a definição das funções adaptativas.

Tabela 14 – Representação gráfica de uma possível Tabela de Decisão Adaptativa.

			Ações adaptativas	**r_1**	**r_2**	**...**	**r_k**
	Cabeçalhos (Tags)→		(H,+, -,?)	S	R	R	E
Tabela de Decisão TD	Condições	$condição_1$		c_{11}	c_{21}	...	c_{k1}
		...		...	...	...	...
		$condição_n$		c_{1n}	c_{2n}	...	c_{kn}
	Ações	$acão_1$		a_{11}	a_{21}	...	a_{k1}
		...		...	...	...	...
		$acão_m$		a_{1m}	a_{2m}	...	a_{kn}
Funções Adaptativas	Anterior	Nome φ		Chamadas às ações adaptativas			
	Posterior	Nome φ					
	Parâmetros	ρ_1					
		...					
		ρ_p					
	Variáveis	υ_1					
		...					
		υ_v					
	Geradores	g_1					
		...					
		g_l					

Na execução de uma regra em uma TDA, primeiramente são verificadas as regras não adaptativas e, se uma única delas se aplica, as ações correspondentes são executadas. Caso mais de uma regra não adaptativa satisfaça a condição, as ações correspondentes às mesmas devem ser aplicadas em paralelo, como previamente definido para tabelas de decisão convencionais. Porém, se nenhuma regra não adaptativa satisfizer a condição, trata-se de uma condição não prevista e, no caso de uma tabela de decisão convencional, não haveria como prosseguir.

Os elementos da Tabela de Decisão Adaptativa são descritos na Tabela 15.

Tabela 15 – Elementos da Tabela de Decisão Adaptativa.

ELEMENTOS DA TDA
1. Linha Cabeçalhos → (*Tags*): especifica o cabeçalho das colunas. O título de cada coluna pode ser: a. "H" que representa o cabeçalho da função adaptativa; b. "?", "-" ou "+" que indicam respectivamente uma ação elementar de consulta, remoção e inserção; c. "S", "E" ou "R" que indicam respectivamente uma regra delimitadora inicial, final ou uma regra da tabela de decisão convencional.
2. Linhas das Condições: cada linha de condição (condição$_1$,...,condição$_n$) corresponde a um rótulo de condição;
3. Valores das condições: em suas células, são indicados os valores das condições (c_{11},...,c_{kn}). Cada valor pode corresponder a entradas limitadas ("S", "N" ou branco) ou a entradas estendidas;
4. Linhas das Ações: cada linha representa uma ação (ação$_1$,..., ação$_m$), que pode ser executado em resposta ao conjunto de condições;
5. Valores das Ações: devem ser marcadas as células das ações (a_{11},...,a_{kn}) que serão executadas quando as regras são avaliadas;
6 Funções adaptativas: as linhas nome da função adaptativa (Nome φ), parâmetros (ρ_1,...ρ_p), variáveis (υ_1,...,υ_v) e geradores (g_1,...,g_l) são definidas, convenientemente.
7. Chamadas às ações adaptativas: em suas células assinalam-se aquelas que serão chamadas antes ou depois da aplicação da regra.

PARTE II

Aplicação e Ferramenta

4 ADAPTATIVIDADE EM APRENDIZAGEM DE MÁQUINA

A utilização de dispositivos adaptativos na mecanização da aprendizagem permite a inserção de um componente de adaptação nas regras que são aprendidas. Esse componente deve proporcionar ao algoritmo de aprendizagem a capacidade de aprender regras sem a necessidade de estímulos externos.

A aplicação de técnicas adaptativas proporciona um bom entendimento dos resultados obtidos com a aprendizagem, por isso pode ser útil se comparado com outras técnicas de aprendizagem que fornecem bons resultados, mas são incapazes de explicar como chegaram ao resultado fornecido (ex.: redes neurais).

Neste trabalho, o processo de aprendizagem foi divido nas seguintes etapas: *Interface*, *Inferência* e *Memória*. O termo mecanismo de aprendizado se refere a essas três etapas.

A *Interface* estabelece a comunicação entre as informações externas e o mecanismo de aprendizado. Entende-se por informações externas qualquer estímulo de entrada provocado por um agente inteligente, seja ele humano (externo) ou a própria máquina (realimentação). De modo geral, a função da *Interface* é receber as informações e alimentar o mecanismo de aprendizado.

A *Inferência* está em constante ciclo de aprendizagem e é responsável por analisar as informações capturadas e agregar conhecimento sobre elas, para melhorar o desempenho na tomada de decisão. O conhecimento adquirido é representado por um conjunto de regras, que pode ser alterado a cada ciclo de aprendizagem. A situação corrente do conjunto de regras é representada por dispositivos adaptativos.

A *Memória* representa o conjunto de regras. Tudo que o dispositivo aprende deve ser armazenado em uma base de regras. Posteriormente, a base de regras é utilizada na tomada de decisão e, depois é possível inferir novas regras a partir dos dados da base. Nos dispositivos adaptativos a memória está acoplada ao próprio dispositivo, que é uma das vantagens de sua utilização.

Para exercitar a utilização da adaptatividade em aprendizagem de máquina foi escolhido um tipo particular de jogo, classificado como jogos dinâmicos. Esse é um tipo de jogo em que o processo de interação estratégica se desenvolve em etapas sucessivas. Nestes jogos, a decisão tomada por um jogador considera as decisões tomadas pelos demais jogadores. Assim, os jogadores fazem escolhas a partir do que os outros jogadores decidiram no passado. Além disso, nesse tipo de interação, as escolhas presentes exigem considerar as consequências futuras, uma vez que o oponente pode retaliar em etapas posteriores do jogo (WIDYANTORO; VEMBRINA, 2009).

O matemático e filósofo alemão Ernst Friedrich Ferdinand Zermelo (1817-1953) provou um teorema que conduz à ideia de que jogos dinâmicos de duas pessoas, com ações sequenciais e informação completa, tais como os jogos da velha ou de xadrez, são perfeitamente determinados, ou seja, possuem solução definida. Neste caso, se um jogador souber aplicar a estratégia correta, dificilmente perderá o jogo.

Em linhas gerais, o processo de aprendizagem de uma estratégia para vencer uma instância de um jogo provoca a adaptação do comportamento dos jogadores para melhorar o desempenho na tomada de decisão. Cada tomador de decisão em um jogo é denominado jogador. Esse adota estratégias, que conduz às decisões para atingir seus objetivos, para cada informação que um jogador possa ter e em cada momento que ele é chamado a realizar uma ação.

A seguir é apresentado um estudo de caso que mostra o processo de aprendizagem de uma estratégia para jogar o *Tic-Tac-Toe*, popularmente conhecido como jogo da velha, é apresentado.

4.1. ESTUDO DE CASO: APRENDENDO A JOGAR O TIC-TAC-TOE

O objetivo do estudo é investigar a utilização de dispositivos adaptativos em problemas reais e aplicar técnicas adaptativas para extração de regras desses dispositivos. A aplicação escolhida, apesar de simples, tem sido utilizada na aplicação de diferentes técnicas, tais como redes neurais artificiais (YAU; TEO, 2007) e distância de Hamilton (RAJANI; DAR; BISWAS; RAMESHA, 2011). Isso permite a fácil associação entre a utilização de técnicas adaptativas e as diferentes técnicas de aprendizado de máquina.

Tic-Tac-Toe (TTT), ou jogo da velha, é um jogo de tabuleiro de dois jogadores, onde cada jogador tem a sua vez para marcar um tabuleiro de tamanho 3×3 com o seu próprio símbolo (“X” para o primeiro jogador e “O” para o segundo jogador).

Os jogos de tabuleiros são casos de estudos interessantes em aprendizagem de máquina, se consideramos que o resultado dessa experiência representa uma habilidade humana (WIDYANTORO; VEMBRINA, 2009). A capacidade de jogar representa muitas vezes um desafio aos jogadores que devem ser criativos para criar planos de ação, combinar estratégias de jogos e aprender com a experiência.

O nível de dificuldade de um jogo de tabuleiro varia de um simples jogo como o TTT ou um quebra-cabeça até os que necessitam de uma estratégia complicada, como as aplicadas nos jogos de dama e xadrez.

As posições do tabuleiro são representadas por b_{ij}, onde i é linha da matriz 3×3 e j é a coluna, representadas na Figura 9.

b_{11}	b_{12}	b_{13}
b_{21}	b_{22}	b_{23}
b_{31}	b_{32}	b_{33}

Figura 8 – Posições do Tabuleiro do TTT.

As regras do jogo são as seguintes (BECK, 2008):

a) O jogo começa com o tabuleiro vazio;

b) Um jogador assume a vez marcando o seu símbolo em uma das posições vazias do tabuleiro;

c) O jogador que forma uma linha reta completa (de ponta a ponta), ou seja, uma linha vertical, horizontal ou diagonal primeiro é o vencedor;

d) O jogo empata se nenhuma linha reta é formada e não existem posições vazias no tabuleiro.

O estado inicial da partida é o tabuleiro vazio. Em cada etapa da partida, que será chamada de jogada, um jogador faz um movimento que consiste em colocar um “X” em uma das posições vazias do tabuleiro e o outro jogador faz um movimento que consiste em colocar um “O” em uma das posições vazias.

A Figura 9 mostra uma possível configuração de três tabuleiros de jogos.

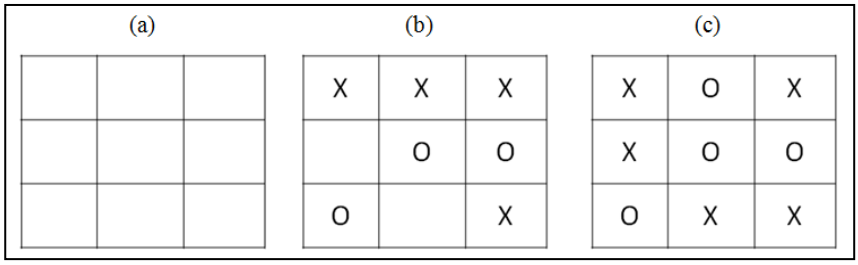


Figura 9 – Três diferentes configurações para o tabuleiro do TTT: (a) Configuração inicial, (b) Jogador “X” vence o jogo, (c) Jogo empatado.

Neste exemplo ilustrativo, o objetivo final da aprendizagem é descobrir uma estratégia para vencer o TTT. A Figura 10 mostra o mecanismo de aprendizado refinado para o aprendizado de estratégias para um jogo dinâmico com informação completa.

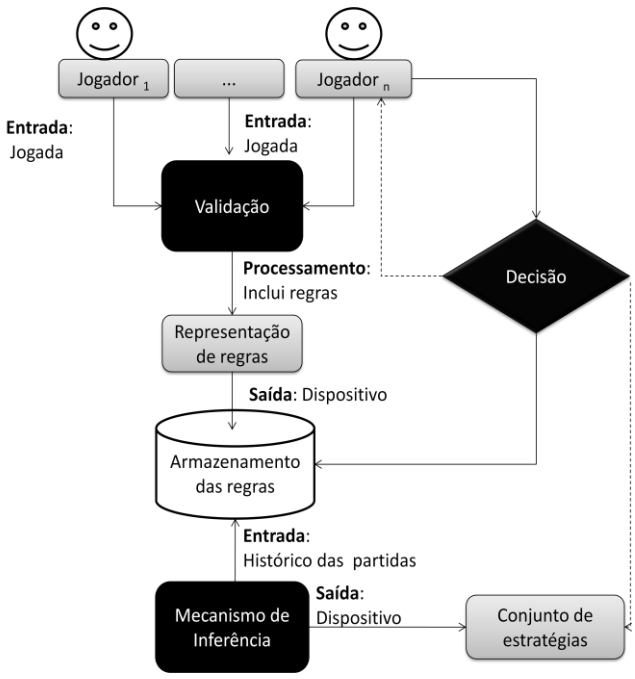


Figura 10 – Mecanismo de aprendizado de um jogo

A Figura 10 mostra as relações entre os componentes que compõem o mecanismo de aprendizado, a saber: *Jogadores* (Jogador$_1$, Jogador$_2$,..., Jogador$_n$), *Validação*,

Representação de Regras, *Armazenamento de Regras*, *Mecanismo de Inferência*, *Conjunto de Estratégias* e *Decisão*.

O TTT consiste de um jogo com dois jogadores, neste exemplo, sendo um deles a própria aplicação, a partir deste momento chamada de "Jogador", enquanto o outro jogador pode ser um ser humano, que será chamado de "Oponente". A jogada de "Jogador" é representada pelo símbolo "X" e a de "Oponente" pelo símbolo "O". A componente *Jogadores* representa o canal de comunicação entre o mecanismo de aprendizado e o ambiente externo (*Interface*). Cada jogador deve fornecer uma entrada de dados para o funcionamento do mecanismo de aprendizado. Aqui, cada entrada é uma jogada, representada por $\delta(b_{ij})$, onde $\delta = \{X,O\}$. Para escolher uma jogada (*Decisão*, Figura 10), "Jogador" pode consultar a base de regras (*Armazenamento de regras*, Figura 10).

A *Validação* é a componente que recebe os estímulos de entrada. Neste exemplo, a validação é representada por uma jogada que indica a posição do tabuleiro escolhida por um jogador, e determina se esta entrada (ex.: $X(b_{23})$, que significa "Jogador" escolhe a posição b_{23} do tabuleiro) é válida (ex.: verifica se posição b_{23} do tabuleiro existe e está vazia). No aprendizado de uma partida, cada jogada é uma regra a ser validada e deve ser capturada e posteriormente representada por um dispositivo apropriado.

Na *Representação de Regras*, um dispositivo adaptativo pode ser uma solução alternativa se considerarmos que, idealmente, o mecanismo de aprendizagem deve ser capaz de incorporar novas regras de forma autônoma.

Segundo Pressman (2006), sistemas com características autônomas favorecem a utilização de modelos formais em sua especificação. As Máquinas de Estados Finitos, por exemplo, são modelos formais que favorecem a Engenharia de Software na especificação do comportamento de sistemas reativos, que se torna mais precisa e menos sujeita a ambiguidades. Esses modelos facilitam o entendimento do sistema.

Particularmente, os autômatos adaptativos podem evoluir gradualmente, modificando sua topologia inicial e, sucessivamente, inserir ou excluir transições de seu próprio conjunto de transições, como resultado da execução de ações adaptativas (NETO; IWAI, 1998).

A proposta é utilizar um autômato adaptativo para representar o conjunto de regras aprendidas em cada partida disputada. Porém, sua representação possui aplicações práticas restritas, se for considerado como dispositivo subjacente o autômato finito, que possui a informação de saída limitada à lógica binária aceita/rejeita (MENEZES, 2011). Assim, com a finalidade de extrair cada movimento do autômato no decorrer de seu aprendizado, aplica-se uma extensão de autômatos conhecida como máquina de *Mealy*, que possui saídas associadas às transições. A máquina de *Mealy* é um autômato finito modificado capaz de gerar uma palavra de saída para cada transição da máquina, a qual inclusive pode ser vazia (MENEZES, 2011).

Na definição da máquina de *Mealy* é incorporado um alfabeto de símbolos de saída Δ, que pode ser o mesmo alfabeto de entrada. A função de transição pode ser representada como um diagrama, assim como nos autômatos finitos, adicionando a cada transição a saída associada, quando diferente da palavra vazia.

A representação gráfica deste autômato é mostrada na Figura 11.

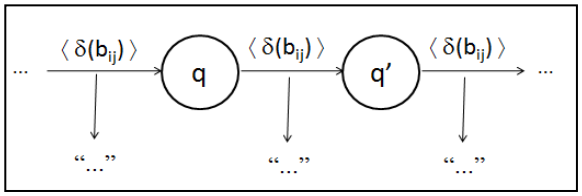


Figura 11 – Máquina de *Mealy*.

A seguir a simbologia utilizada no diagrama de transição da Máquina de *Mealy*.

- ⟨...⟩ entrada fornecida pelo usuário (ex.: por "Jogador").
- "..." saída gerada pela transição (ex.: jogada realizada).

A entrada representa o conjunto (finito) de todos os possíveis símbolos que são estímulos de entrada válidos do autômato. Os estados q e q' representam os estados de origem e destino de uma transição.

Para fins de apresentação, as saídas da máquina serão ocultadas e será definido como representação equivalente o autômato adaptativo, apresentado na Figura 12. Assim a transição $(q, \langle ... \rangle) \rightarrow q'$ da máquina de *Mealy* é equivalente $(q, \Theta) \rangle\rangle q$ do autômato adaptativo.

As saídas do autômato marcam as escolhas efetuadas por “Jogador” e “Oponente”, a cada jogada e a cada partida. O caminho que determina o sucesso ou o insucesso na partida é mapeado através das saídas geradas pelo autômato.

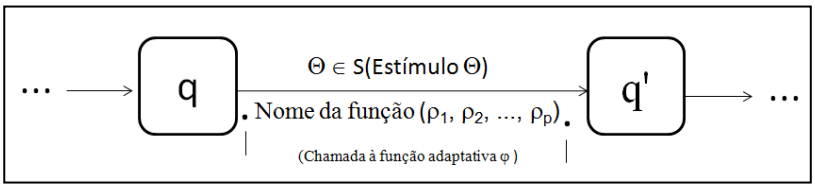


Figura 12 – Autômato Adaptativo com saída oculta.

Na Tabela 16 descreve-se a técnica utilizada para a construção do autômato baseada em (NETO; IWAI, 1998), que representa o comportamento dos jogadores.

Tabela 16 – Passos para a construção de um autômato adaptativo para um jogador de TTT.

CONSTRUÇÃO DO AUTÔMATO ADAPTATIVO	
1:	Iniciar com um autômato conhecido;
2:	Modificar o autômato incrementalmente para aceitar jogadas sucessivas na qual o jogador vença a partida;
3:	Modificar o autômato incrementalmente para rejeitar jogadas sucessivas na qual o jogador perca a partida;
4:	Modificar o autômato incrementalmente para aceitar jogadas sucessivas na qual jogador empate a partida;
5:	Repetir as ações até que o treinamento seja considerado aceitável[8].

Ao se pensar no jogo como uma máquina de estados que recebe uma entrada, sendo a entrada uma jogada ($\delta(b_{ij})$), o tabuleiro começa no estado vazio e os estados mudam a cada vez que a máquina recebe uma entrada. Alguns estados são especiais quando são atingidos, por exemplo, quando a partida acaba, esses estados são os estados finais. Alguns estados finais determinam uma vitória para “Jogador” e outros, uma vitória para “Oponente”, enquanto os demais estados significam que o jogo terminou com um empate.

As funções adaptativas do autômato impõem uma ordenação, pré-determinada, ao conjunto de atributos. É essa ordenação que determina em que jogada da partida irão ocorrer os diversos atributos. Na medida em que exemplos vão sendo submetidos ao dispositivo, a estrutura vai se modificando, na forma de um autômato.

[8] Neste ponto é necessário determinar um limiar para não ocorrer um ajuste excessivo ao conjunto de treinamento.

Seja o conjunto de atributos A = {α_0, α_1, α_2, α_3, α_4, α_5, α_6, α_7, α_8, α_9}, onde: α_0 = "1ª jogada", α_1 = "2ª jogada"; α_2 = "3ª jogada"; α_3 = "4ª jogada"; α_4 = "5ª jogada"; α_5 = "6ª jogada"; α_6= "7ª jogada"; α_7 = "8ª jogada"; α_8 = "9ª jogada"; α_9= "Resultado da partida".

A técnica descrita na Tabela 16 é implementada pelo algoritmo da Tabela 17.

Tabela 17 – Algoritmo para capturar as jogadas dos jogadores em cada partida.

ALGORITMO 1 – Aprendizado utilizando um Autômato Adaptativo.	
1:	Para i:= 0 até n [passo 1] faça
2:	Pesquisar no conjunto de transições do autômato se existe alguma transição ?(q_i, $\sigma_?$) $\rangle\rangle$ $q_?$, onde q_i é conhecido. Para todas as transições consumindo σ com estado inicial q_i e estado destino $q_?$, retornar os valores $\sigma_?$ e $q_?$.
2.1:	Se encontrar apenas uma transição: retorna os valores de $\sigma_?$ e $q_?$, aplique a regra.
2.2:	Se não encontrar nenhuma transição: Criar uma nova transição consumindo um símbolo A≠ε com estado inicial q_i e estado destino $q_{\#j}$, para j:= i+1. O símbolo # é utilizado para indicar que um novo estado foi gerado.
	2.2.1: Remova a transição (q_i ,$\sigma_?$) $\rangle\rangle$ $q_?$
	2.2.2: Insira a transição (q_i ,A) $\rangle\rangle$ $q_{\#j}$ onde A é uma posição vazia do tabuleiro escolhida e j:= i+1.
	2.2.3: Insira a transição: (q_i , ε[.φ_i]) $\rangle\rangle$ $q_{\#j}$
	2.2.4: Inserir a transição: ($q_{\#j,}$ ε[.φ_j]) $\rangle\rangle$ q_f
3:	Escolher a posição σ do tabuleiro.

Cada atributo de α_0 a α_8 pode conter os seguintes valores de atributos {X(b_{11}), X(b_{12}), X(b_{13}), X(b_{21}), X(b_{22}), X(b_{23}), X(b_{31}), X(b_{32}), X(b_{33}), O(b_{11}), O(b_{12}), O(b_{13}), O(b_{21}), O(b_{22}), O(b_{23}), O(b_{31}), O(b_{32}), O(b_{33}), θ }, onde θ significa posição do tabuleiro vazia.

Para os possíveis valores de α_9, considere-se o conjunto de classes distintas Ω = {ω_1, ω_2, ω_3}, onde: ω_1 = "P"; ω_2 = "N"; e ω_3 = "E", representando respectivamente os resultados: "Jogador" venceu a partida (Positivo), "Jogador" perdeu a partida (Negativo) e "Jogador" empatou a partida.

Este exemplo ilustrativo considera o conjunto de dados de treinamento *P*, apresentados na Tabela 18.

Tabela 18 – Dados de treinamento referentes a seis partidas de TTT.

CONJUNTO DE TREINAMENTO						
Atributos	Instancias de Treinamento					
	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
α_0	$X(b_{22})$	$X(b_{11})$	$X(b_{12})$	$X(b_{31})$	$X(b_{22})$	$X(b_{22})$
α_1	$O(b_{11})$	$O(b_{33})$	$O(b_{31})$	$O(b_{22})$	$O(b_{13})$	$O(b_{11})$
α_2	$X(b_{33})$	$X(b_{31})$	$X(b_{33})$	$X(b_{11})$	$X(b_{33})$	$X(b_{13})$
α_3	$O(b_{12})$	$O(b_{21})$	$O(b_{11})$	$O(b_{21})$	$O(b_{11})$	$O(b_{31})$
α_4	$X(b_{13})$	$X(b_{13})$	$X(b_{13})$	$X(b_{23})$	$X(b_{21})$	$X(b_{21})$
α_5	$O(b_{31})$	$O(b_{22})$	$O(b_{31})$	$O(b_{12})$	$O(b_{12})$	$O(b_{23})$
α_6	$X(b_{23})$	$X(b_{12})$	θ	$X(b_{31})$	θ	$X(b_{12})$
α_7	θ	θ	θ	$O(b_{33})$	θ	$O(b_{31})$
α_8	θ	θ	θ	$X(b_{13})$	θ	$X(b_{33})$
α_9	P	P	N	E	N	E

Nesta fase de *Representação de regras*, cada regra ou jogada deve ser capturada por autômatos distintos, seja de "Jogador" ou de "Oponente". As jogadas de "Oponente" são importantes para que "Jogador" aprenda com a derrota. Assim, "Jogador" pode fazer escolhas a partir do que "Oponente" decidiu no passado.

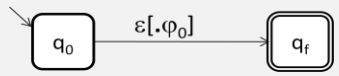
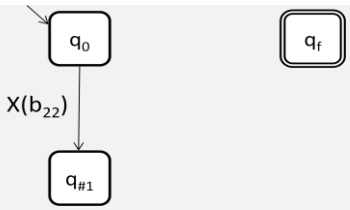
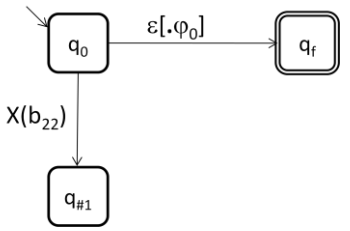
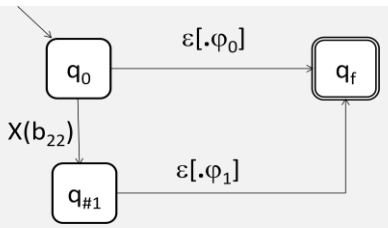
A primeira partida tem a função de calibrar o mecanismo de aprendizado, "Jogador" realiza suas jogadas sem auxílio do tomador de decisão (*Decisão*, Figura 10), pois ainda não existe um histórico de operações (partidas) que possa ser consultado (*Armazenamento de Regras*, Figura 10).

No *Armazenamento de Regras*, as jogadas capturadas pelos autômatos são armazenadas na memória do mecanismo de aprendizado e representam o histórico das partidas. A memória também é chamada de base de regras e as informações contidas nesta base podem ser utilizadas futuramente para melhorar o modelo de aprendizagem através da experiência. Em dispositivos adaptativos, as regras ficam armazenadas na estrutura do próprio dispositivo. Nos autômatos adaptativos, as regras ficam armazenadas através dos estados e transições, nas tabelas de decisão adaptativas, são representadas pelos valores de condições e ações e, nas árvores de decisão adaptativas, são representadas através dos caminhos que guiam uma decisão da raiz até as folhas.

A Tabela 19 mostra a evolução do autômato que representa as jogadas de "Jogador" para a primeira jogada.

Na Partida$_1$, uma possível configuração final do autômato adaptativo inferido, após quatro jogadas de “Jogador”, é representada pela Figura 13.

Tabela 19 – Evolução do autômato que representa o comportamento de "Jogador".

1ª JOGADA – Evolução do Autômato	
- Ação elementar de inspeção:?(q_i, $\sigma_?$) $\rangle\rangle$ $q_?$ - Retorna $\sigma = \varepsilon[.\varphi_0]$ e $q = q_f$ - Saída $\Theta = q_f$	
- Ação elementar de remoção: -(q_i, $\varepsilon[.\varphi_i]$) $\rangle\rangle$ q_f - Remove a transição (q_0, ε $[.\varphi_0]$) $\rangle\rangle$ q_f	
- Ação elementar de inserção: +(q_i ,A) $\rangle\rangle$ q_i - Insere a transição (q_0, $X(b_{22})$ $\rangle\rangle$ $q_{\#1}$	
- Ação elementar de inserção: +(q_i ,$\varepsilon[.\varphi_i]$) $\rangle\rangle$ q_f - Insere a transição +(q_0 , $\varepsilon[.\varphi_0]$) $\rangle\rangle$ q_f	
- Ação elementar de inserção:+(q_j ,$\varepsilon[.\varphi_i]$) $\rangle\rangle$ q_f - Insere a transição ($q_{\#1}$, $\varepsilon[.\varphi_1]$) $\rangle\rangle$ q_f	

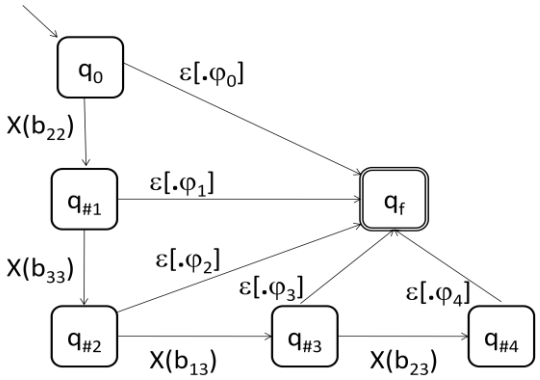
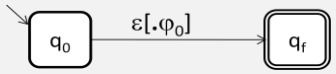
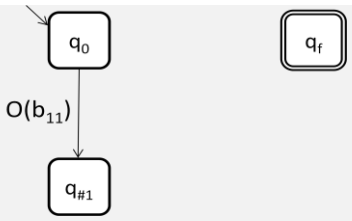
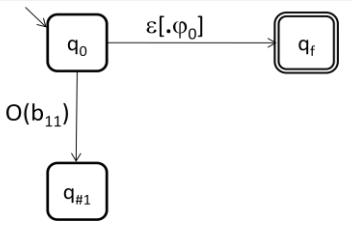
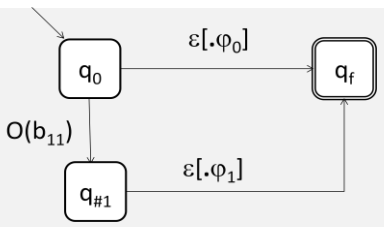


Figura 13 – Autômato Coletor de jogadas de "Jogador"

A Tabela 20 mostra a evolução do autômato que captura as jogadas de “Oponente” para a primeira jogada.

Tabela 20 – Evolução do autômato que representa o comportamento de "Oponente".

1ª JOGADA – Evolução do Autômato	
- Ação elementar de inspeção: $?(q_i, \sigma_?) \rangle\rangle\ q_?$ - Retorna $\sigma = \varepsilon[.\varphi_0]$ e $q = q_f$ - Saída $\Theta = q_f$	
- Ação elementar de remoção: $-(q_i, \varepsilon[.\varphi_i]) \rangle\rangle\ q_f$ - Remove a transição $(q_0, \varepsilon\ [.\varphi_0]) \rangle\rangle\ q_f$	
- Ação elementar de inserção: $+(q_i, A) \rangle\rangle\ q_i$ - Insere a transição $(q_0, O(b_{11}) \rangle\rangle\ q_{\#1}$	
- Ação elementar de inserção: $+(q_i, \varepsilon[.\varphi_i]) \rangle\rangle\ q_f$ - Insere a transição $+(q_0, \varepsilon[.\varphi_0]) \rangle\rangle\ q_f$	
- Ação elementar de inserção: $+(q_i, \varepsilon[.\varphi_i]) \rangle\rangle\ q_f$ - Insere a transição $(q_{\#1}, \varepsilon[.\varphi_1]) \rangle\rangle\ q_f$	

Na Partida$_1$, uma possível configuração final do autômato adaptativo inferido, após três jogadas de “Oponente”, é mostrada pela Figura 14.

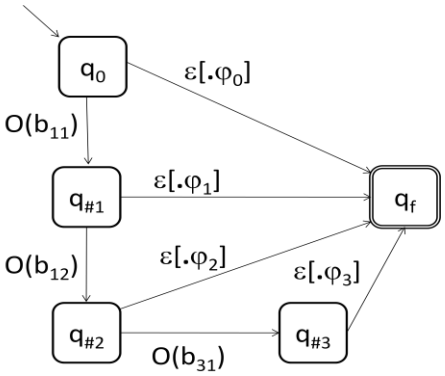


Figura 14 – Autômato Coletor de jogadas de "Oponente"

O mecanismo de aprendizado captura tanto as jogadas de "Jogador" quanto de "Oponente" e, dependendo do resultado da partida (venceu, perdeu ou empatou), repete as jogadas nas partidas posteriores.

Uma vantagem comparada a outros mecanismos de representação, tais como *naïve* Bayes, é que o autômato armazena a sequência das jogadas, assim é possível saber a ordem em que as jogadas foram efetuadas no tabuleiro. Essa relação de ordem das jogadas fica incorporada na estrutura do autômato.

Neste trabalho, o modelo de aprendizagem incremental inclui uma memória para armazenar as operações realizadas na base de regras adaptativas. O conjunto de transições (jogadas) resultante de cada partida é inserido em uma Tabela de Decisão Adaptativa, porém é relevante destacar que o próprio autômato adaptativo pode representar o conjunto de regras adquiridas. Apesar disso, diferentes dispositivos adaptativos podem apresentar uma solução mais aderente a um problema em particular.

Neste exemplo, se considerarmos que o objetivo final é obter um conjunto de regras para auxílio à tomada de decisão em um jogo de tabuleiro, é conveniente à utilização da TDA. Neto (2009b) considera que sistemas que envolvem tomadas de decisão podem utilizar Tabelas de Decisão Adaptativas para a construção de sistemas inteligentes que recebem dados de treinamento, aprendem a classificá-los e, futuramente, classificam novos dados.

Outra motivação para a representação da utilizando uma TDA está fundamentada em Neto (2001, apud TCHEMRA, 2009), que articula "[...] os dispositivos adaptativos dirigidos por regras podem dar maior flexibilidade às tabelas de decisão, permitindo, não somente a consulta às regras, como também a inclusão e a exclusão de regras durante a operação do dispositivo, transformando, assim, a tabela de decisão numa ferramenta mais poderosa".

A Tabela 21 mostra a construção da TDA durante o processo de aprendizagem de regras. As jogadas capturadas pelo autômato são representadas por um conjunto de regras do tipo "Se c*ondições então Ação"*, onde as condições são as jogadas válidas que levam o "Jogador" a um resultado. A ação é o resultado obtido na partida. O conjunto de regras é utilizado para jogar novas partidas.

Tabela 21 – Construção da Tabela de Decisão Adaptativa.

CONSTRUÇÃO DA TABELA DE DECISÃO ADAPTATIVA	
1:	Iniciar uma TDA conhecida;
2:	Buscar no conjunto de regras R uma ou mais regras que satisfaçam as condições de entrada P_k:
	2.1: Caso encontre uma ou mais regras aplicáveis, inclua todas elas em um conjunto de regras aplicáveis;
	2.2: Caso contrário, prepara o dispositivo para uma nova entrada, aplicando o passo 4;
3:	Extrair as regras aplicáveis do conjunto de regras:
	3.1: Caso apenas uma regra seja aplicável, o dispositivo executa a ação determinada pela regra; essa situação define uma operação determinística;
	3.2: Caso mais de uma regra seja aplicável, uma situação de não determinismo é detectada. As regras são executadas em paralelo.
4:	Decodificar as ações
	4.1: No caso de uma regra não adaptativa, extrair as ações associadas às regras;
	4.2: No caso de uma regra adaptativa, extrair as ações adaptativas associadas às regras.
5:	Executar as ações:
	5.1: No caso de uma regra não adaptativa, executar as rotinas semânticas associadas a cada ação.
	5.2: No caso de uma regra adaptativa:
	5.2.1: Para uma função adaptativa posterior, executar o passo 5.1 e na sequência o passo 5.3.
	5.2.2: Para uma função adaptativa anterior, executar o passo 5.3 e na sequência executar o passo 5.1.
	5.3: Executar as ações adaptativas.

A partir da $Partida_2$, há regras na base de regras e o "Jogador" pode decidir as jogadas de acordo com as jogadas passadas. Porém, alguns movimentos podem ser aleatórios, sem qualquer auxílio do tomador de decisões, considerando que a posição do tabuleiro está ocupada pelo Oponente. A *Decisão* é baseada nas regras armazenadas na Tabela de Decisão Adaptativa, caso nenhuma regra seja aplicável, deve-se decidir aleatoriamente e aprender uma nova regra.

Considere o algoritmo para construção da TDA descrito na Tabela 21.

A saída do algoritmo é um reconhecedor de partidas τ, neste caso representado por um dispositivo do tipo tabela de decisão adaptativa. Dado uma nova partida χ, o reconhecedor deve verificar se a partida já foi realizada anteriormente, caso já tenha sido realizada aplica a regra aprendida na partida. Caso contrário, aprende uma nova regra.

A configuração inicial da TDA está representada na Tabela 22. Na configuração inicial apenas uma regra adaptativa, P_a, é inserida na Tabela. No decorrer de sua execução, novas regras vão sendo inseridas.

Tabela 22 – Configuração Inicial da Tabela de Decisão Adaptativa.

Cabeçalhos (Tags)			**H**	**?**	**+**	**P_a**
Condições	$Condição_1$			p_0	v_0	-
	$Condição_2$			p_1	v_1	-
	$Condição_3$			p_2	v_2	-
	$Condição_4$			p_3	v_3	-
	$Condição_5$			p_4	v_4	-
	$Condição_6$			p_5	v_5	-
	$Condição_7$			p_6	v_6	-
	$Condição_7$			p_7	v_7	-
	$Condição_8$			p_8	v_8	-
Ações	$Ação_1$			p_9	v_9	?
Funções Adaptativas	Anterior	f_1	B	√	√	√
	Posterior					
	Parâmetros	p_0	P			p_0
		...	...			...
		p_9	P			p_9
	Variáveis	v_0	V			
		...	...			
		v_9	V			
	Geradores					

O resultado da execução da TDA para o conjunto de jogadas capturadas pelos autômatos de Tabela 19 e Tabela 20 através do conjunto de treinamento da Tabela 18 é o conjunto de partidas $P = \{P_1, P_2,...,P_k\}$, onde k é o número de partidas, conforme Tabela 23. As condições representam a ordem das jogadas capturadas pelo autômato, tanto de "Jogador" quanto de "Oponente".

Tabela 23 – Tabela de Decisão Adaptativa após 6 modificações.

Cabeçalhos (Tags)			H	?	+	P_1	P_2	P_3	P_4	P_5	P_6	P_a
Condições	Jogada_1_x			p_0	v_0	$X(b_{22})$	$X(b_{11})$	$X(b_{12})$	$X(b_{31})$	$X(b_{22})$	$X(b_{22})$	θ
	Jogada_1_o			p_1	v_1	$O(b_{11})$	$O(b_{33})$	$O(b_{31})$	$O(b_{22})$	$O(b_{13})$	$O(b_{11})$	θ
	Jogada_2_x			p_2	v_2	$X(b_{33})$	$X(b_{31})$	$X(b_{33})$	$X(b_{11})$	$X(b_{33})$	$X(b_{13})$	θ
	Jogada_2_o			p_3	v_3	$O(b_{12})$	$O(b_{21})$	$O(b_{11})$	$O(b_{21})$	$O(b_{11})$	$O(b_{31})$	θ
	Jogada_3_x			p_4	v_4	$X(b_{13})$	$X(b_{13})$	$X(b_{13})$	$X(b_{23})$	$X(b_{21})$	$X(b_{21})$	θ
	Jogada_3_o			p_5	v_5	$O(b_{31})$	$O(b_{22})$	$O(b_{31})$	$O(b_{12})$	$O(b_{12})$	$O(b_{23})$	θ
	Jogada_4_x			p_6	v_6	$X(b_{23})$	$X(b_{12})$	θ	$X(b_{31})$	θ	$X(b_{12})$	θ
	Jogada_4_o			p_7	v_7	θ	θ	θ	$O(b_{33})$	θ	$O(b_{31})$	θ
	Jogada_5_x			p_8	v_8	θ	θ	θ	$X(b_{13})$	θ	$X(b_{33})$	θ
Ações	Resultado			p_9	v_9	P	P	N	E	N	E	?
Funções Adaptativas	Anterior	f_1	B	√	√							√
	Posterior											
	Parâmetros	p_0	P									p_0
		...	...									...
		p_9	P									p_9
	Variáveis	v_0	V									
		...	...									
		v_9	V									

O exemplo a seguir ilustra a operação da TDA para representar o conhecimento adquirido pelo autômato.

Considere o exemplo de uma partida χ = {$X(b_{22})$, $O(b_{11})$, $X(b_{13})$, $O(b_{31})$, $X(b_{21})$, $O(b_{23})$, $X(b_{12})$, $O(b_{31})$, $X(b_{33})$, Empate}.

Neste caso, não existe uma regra aplicável, ou seja, uma partida já realizada com a mesma configuração. Porém, é sabido que χ é um exemplo de partida que "Jogador" venceu e deve ser adicionado à TDA. Na TDA a regra adaptativa P_a é satisfeita toda vez que nenhuma regra não adaptativa for aplicável. A regra P_a possui a seguinte função adaptativa associada:

Exemplo do pseudocódigo para declaração de uma função adaptativa.

```
Função Adaptativa f1(Parâmetros: p1, p2, p3, p4, p5, p6, p7, p8, p9){
  Variáveis: v1, v2, v3, v4, v5, v6, v7, v8, v9
  Geradores: necessários para gerar novas regras
  Ações elementares Δ{
     δ1:?[v1, v2, v3, v4, v5, v6, v7, v8, v9] (Existe uma regra que satisfaça as condições χ).
     δ2:+[v1,v2,v3,v4,v5,v6,v7, v8, v9](Inserir uma regra Pk).
  }
}
```

No caso de aplicação da regra P_a em χ, os valores dos atributos são atribuídos aos parâmetros da função f_1. A função recebe os valores como parâmetro e atribui os valores dos parâmetros para as variáveis. Na sequência, a ação elementar de consulta pesquisa se existe uma regra em R que satisfaça χ e a ação elementar de inserção inclui uma regra P_k, para n= 1, 2,... k-1, onde n é o número de regras atuais da TDA.

No domínio do TTT, cada jogador enfrenta um desafio básico: como criar uma estratégia flexível o suficiente para lidar com certas situações de mudança a cada jogada. A estratégia é formada por um conjunto de regras que permite aumentar as chances de vitória de um jogador. O mecanismo de aprendizagem deve aprender não somente a aplicar jogadas válidas, mas também como aplicar uma jogada que garanta uma vitória ou, no pior caso, leve a partida a um empate.

Após várias partidas, a base de regras se torna uma rica fonte de dados que representa o comportamento dos jogadores e estimula a execução de uma próxima etapa do aprendizado, que é a descoberta de estratégias de jogo.

O objetivo da descoberta de uma estratégia para jogar o TTT considera que os jogadores devem usar os meios mais adequados para atingir seus objetivos, seja

vencer ou não perder. Assim, após diversas partidas o jogador acumula experiências e essa experiência pode conduzir os jogadores a um comportamento estratégico. Cada jogador toma decisões considerando que elas terão efeitos sobre os outros jogadores, bem como as decisões dos outros jogadores terão efeitos sobre suas decisões. A estratégia tenta minimizar o efeito negativo de cada decisão.

Esta etapa de aprendizagem ou descoberta de uma estratégia é representada pelo *Mecanismo de Inferência*.

O *Mecanismo de Inferência* atua sobre o conjunto de regras aprendidas, armazenado na base de regras, e infere um conjunto de estratégias que poderão ser consultadas na tomada de decisão dos jogadores.

Neste trabalho, a aprendizagem de estratégias é baseada nas medidas de ganho de informação e entropia, similar ao ID3.

A entropia, no contexto da teoria de informação, pode ser considerada como uma medida da quantidade de informação que uma pessoa necessita para organizar seus conhecimentos e descobrir uma regra. Analogamente, será adotada como a medida para organizar o conhecimento adquirido por um jogador e descobrir um conjunto de regras para realizar as jogadas de forma estratégica.

Quanto mais alternativas uma sistema de tomada de decisão possui (ex.: mais jogadas possíveis), mais informações são necessárias para aprender a tomá-las (maior entropia). Se um sistema de tomada de decisão não tem alternativas, não é necessária nenhuma informação (a entropia é 0).

Tabela 24 – Algoritmo para inferência de regras utilizando ganho de informação.

DESCOBERTA DE UMA ESTRATÉGIA PARA JOGAR TTT	
1:	Para cada jogada K de "Jogador" faça:
	a. Crie um estado que representa a jogada K.
	b. Calcule o ganho de informação de cada jogada (valor de atributo). $$Ganho(S,A) = Entropia(S) - \sum_{v \in V} \frac{\lvert S_v \rvert}{\lvert S \rvert} . Entropia(S_v)$$ Onde: $$\text{Entropia (S)} = -\text{P}_+\log_2\text{P}_+ - \text{P\_}\log_2\text{P\_}$$
	c. Estender a árvore adicionando um ramo para cada valor do atributo.
	d. Dividir o conjunto de exemplos P (tendo em conta o valor do atributo escolhido) e passe os subconjuntos para as folhas da árvore.
	e. Repetir os passos para cada novo nó gerado.

Seja *P* o conjunto de exemplos de treinamento armazenado na base de regras do mecanismo de aprendizagem. A base de regras é composta por 958 exemplos (626 exemplos de vitória e 332 exemplos de derrota) extraídos do repositório de dados UCI Machine Learning Repository. Estes dados foram tratados de forma que a ordem das jogadas fossem representados no conjunto. Seja 9 o total de atributos, cada atributo corresponde a um uma posição do tabuleiro de TTT.

Aplicando a fórmula ao subconjunto S temos:

$$\text{Entropia (S)} = -\frac{626}{958}\log_2\frac{626}{958} - \frac{332}{958}\log_2\frac{332}{958} = 0.93$$

Considere os resultados obtidos com o calculo do ganho de informação: $\text{Ganho}(S, b_{11}) = 0.93 - 0.91 = 0.02$, $\text{Ganho}(S, b_{12}) = 0.93 - 0.92 = 0.01$, $\text{Ganho}(S, b_{13}) = 0.93 - 0.91 = 0.02$, $\text{Ganho}(S, b_{21}) = 0.93 - 0.92 = 0.01$, $\text{Ganho}(S, b_{22}) = 0.93 - 0.84 = 0.09$, $\text{Ganho}(S, b_{23}) = 0.93 - 0.96 = -0.03$, $\text{Ganho}(S, b_{31}) = 0.93 - 0.91 = 0.02$, $\text{Ganho}(S, b_{32}) = 0.93 - 0.92 = 0.01$, $\text{Ganho}(S, b_{33}) = 0.93 - 0.91 = 0.01$.

Na primeira iteração do algoritmo, é possível concluir que a jogada com maior ganho de informação é escolher a posição central do tabuleiro, ou seja, $\delta(b_{22})$. A partir da segunda jogada, o ganho de informação é calculado baseado na jogada de "Oponente"

A Figura 15 mostra a possibilidade de jogadas e seus valores correspondentes.

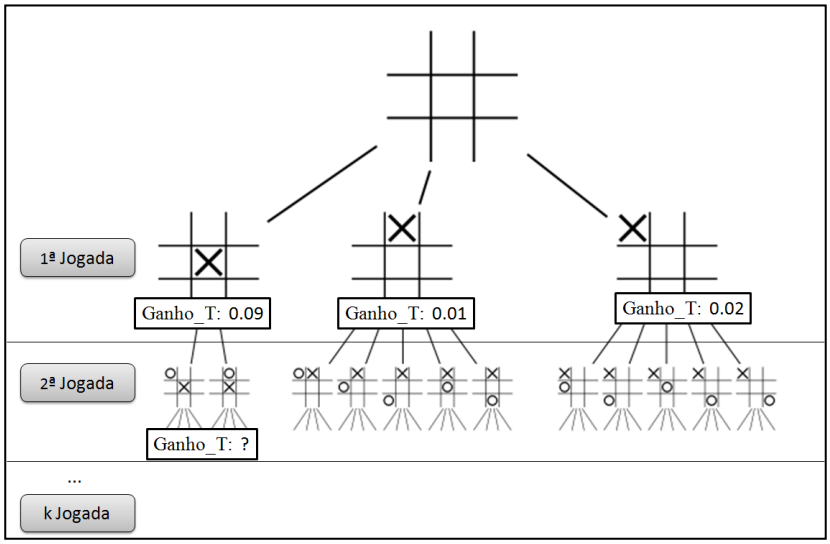


Figura 15 – Representação das possibilidades de jogadas com ganho de informação.

Na sequência, os cálculos de ganho de informação são refeitos, considerando cada possível jogada de "Oponente", e assim sucessivamente até obter uma árvore que seja capaz de auxiliar o jogador na tomada de decisão aplicando uma estratégia.

4.1.1. Análise das estratégias

Esta seção mostra como analisar o comportamento dos jogadores a partir das regras geradas do conjunto de estratégias. As estratégias de "Jogador" consideram que este deve iniciar a partida, porém é perfeitamente possível extrair uma estratégia que permita a "Oponente" iniciar a partida.

A *Decisão* das jogadas utilizando uma estratégia é baseada no *Conjunto de estratégias*.

Existem diversos algoritmos disponíveis para decidir as jogadas de "Jogador" em resposta aos movimentos do "Oponente" durante o jogo. Na Tabela 25 é apresentada uma abordagem heurística simples para automatizar os movimentos do computador, para o nível iniciante.

Tabela 25 – Roteiro para a tomada de decisão baseada no conjunto de estratégias.

TOMADA DE DECISÃO	
1:	Verifique se existe um movimento que o computador pode fazer de modo que obterá maior ganho de informação.
	a. Em caso afirmativo, preencher o quadrado relevante.
	b. Senão, verifique se há um movimento que irá bloquear uma vitória para o outro jogador e preencha o quadrado correspondente.
	c. Senão, preencha o quadrado que fica na linha/coluna com o número máximo de células livres de marcas.

O TTT com estratégias deve permitir que, para uma particular instância do jogo, "Jogador" vença ou pelo menos empate a partida. As estratégias inferidas a partir da base de regras são as seguintes:

1. *Se* Jodada_1_x *então* escolha X(b_{22}): "Jogador" deve fazer a jogada de abertura, neste caso a melhor jogada é escolher a posição central do tabuleiro com maior ganho de informação (Ganho_T: 0.09).

Na sequência, o adversário pode marcar uma lateral ou um canto. Por exemplo, caso o "Oponente" marque um dos cantos, é observado que "Jogador" deve escolher o canto oposto, formando uma linha diagonal, conforme a Figura 16.

2. Se Jodada_2_o = O(b_{33}) *então* escolha X(b_{11}): Dependendo da opção escolhida pelo "Oponente", "Jogador" deve escolher aquela com maior ganho de informação. Para a jogada O(b_{33}), O(b_{11}), O(b_{13}) e O(b_{31}) a melhor opção é o canto oposto, caso ele esteja vazio. Caso contrário, escolhe-se a próxima opção com melhor ganho de informação, e assim sucessivamente.

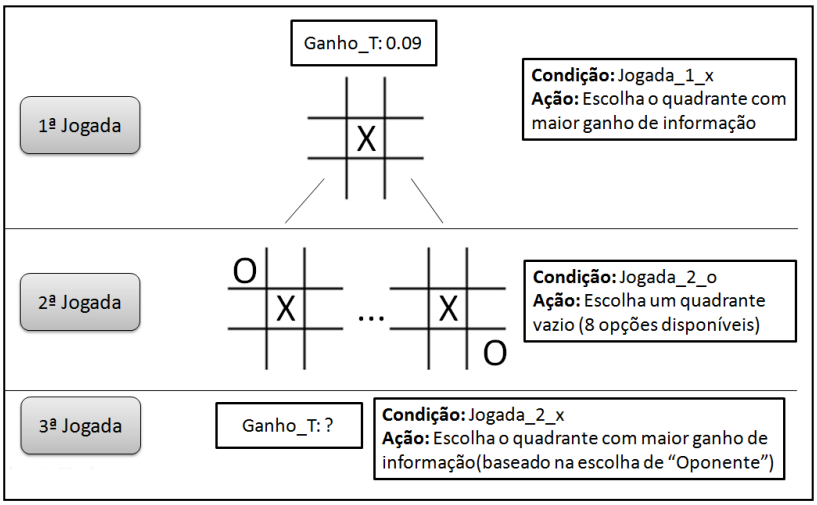


Figura 16 – "Jogador" escolhe a 3ª jogada baseada na escolha do oponente.

Se o próximo movimento de "Oponente" for adjacente da marca anterior, "Jogador" vai ter boas chances de vencer, como mostrado Figura 17.

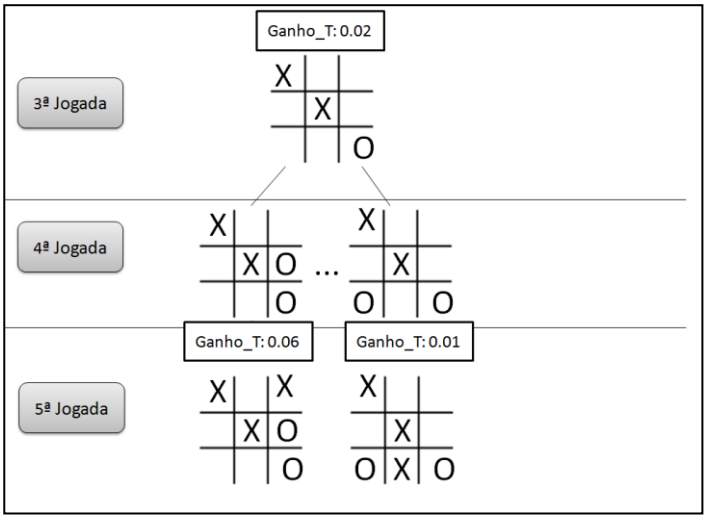


Figura 17 – "Jogador" escolhe a 5ª jogada baseada na escolha do oponente.

3. Se Jodada_2_o = O(b_{23}) *então* escolha X(b_{13}): Neste caso, o "Jogador" deve retaliar o "Oponente". O termo retaliar significa que "Jogador" deve bloquear todas as tentativas de vencer do "Oponente".

Considerando que "Jogador" tome as decisões baseadas nas regras que definem uma estratégia, o clássico TTT pode ser jogado de modo que "Jogador" é direcionado para uma vitória ou um empate.

5 IMPLEMENTAÇÃO DE UMA FERRAMENTA PARA TOMADA DE DECISÃO

Com a finalidade de apoiar a resolução de problemas de tomada de decisão, em particular de aprendizagem de máquina, uma ferramenta computacional *Adapt*-DT foi implementada, baseada em técnicas adaptativas. Esta ferramenta tem a função de fornecer um substrato geral para a criação, manipulação e análise de regras em problemas de tomada de decisão utilizando tabelas de decisão adaptativas.

Este capítulo apresenta a ferramenta *Adapt*-DT, sua arquitetura básica, aplicação e resultados obtidos. Posteriormente, é apresentado um exemplo ilustrativo para testar a utilização da ferramenta na tomada de decisão em um problema de aprendizagem de máquina.

A ferramenta foi implementada utilizando a linguagem de programação Java. A escolha da linguagem pretende facilitar a integração com outras ferramentas já desenvolvidas no laboratório de Linguagens e Técnicas Adaptativas (http://www.pcs.usp.br/~lta), caso seja conveniente.

O projeto da ferramenta permite a integração com outros pacotes de softwares, por exemplo, com pacote de software Weka[9]. O Weka reúne vários algoritmos de aprendizado de máquina em um único ambiente.

A Figura 18 apresenta o diagrama de classes da ferramenta para utilização das tabelas de decisão adaptativas.

As classes *Field, Condition, Action, Rule, Engine* e *Label* correspondem ao substrato de uso geral do sistema. Essas classes permitem definir as condições, ações e regras da tabela de decisão adaptativa.

Field é a superclasse responsável por modificar e acessar os valores dos campos condições e ações.

[9] Disponível em http://www.cs.waikato.ac.nz/ml/weka

Condition e *Action* estendem a classe *Field*. Para exemplificar, em uma tabela de decisão, os objetos do tipo *Condition* e *Action* correspondem aos campos da tabela de decisão. No caso de uma árvore de decisão, os objetos do tipo *Condition* correspondem aos nós da árvore. Os valores atribuídos a cada condição correspondem aos ramos dos nós. Os objetos do tipo *Action* são as folhas da árvore.

A classe *Label* é utilizada para rotular as condições e as ações. O rótulo é apenas o nome dado a cada condição e ação (ex.: temperatura e febre são rótulos para condições, classe é um rótulo para ação). A classe *Rule* é utilizada para criar uma regra. Uma regra é composta por um conjunto de valores de condições e ações. As classes *AdaptiveRules, AdaptiveFunction, ElementaryActions, Parameters, Variables* e *Generators* permitem a incorporação da camada adaptativa no dispositivo guiado por regras. A classe *AdaptiveRules* estende as funções da classe *Rule* e permite a definição das regras adaptativas. A classe *Engine* é o motor do sistema guiado por regras e implementa os seguintes métodos: *searchcondition( )* - busca no conjunto de regras uma ou mais regras que satisfaçam as condições; *extractrule( )* - extrai a regra que satisfaz as condições de entrada, se ela existir; *decodeaction( )* - interpreta a regra e determina as ações associadas a ela; *applyaction( )*: aplica as ações correspondentes à regra.

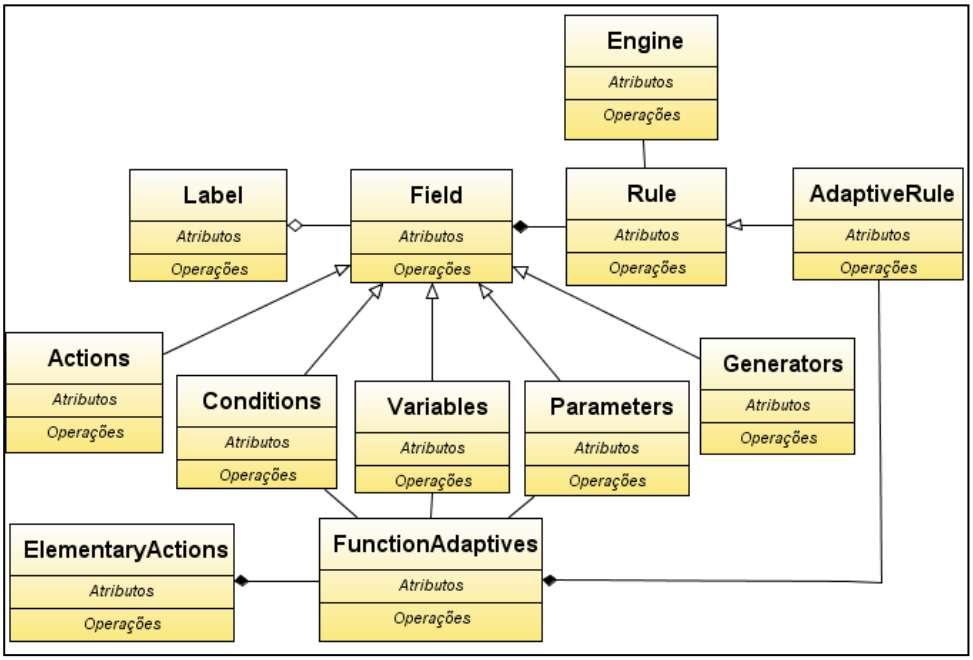


Figura 18 – Diagrama de classes para implementação de uma TDA.

Na Tabela 26, é feita a descrição do algoritmo adotado para a tomada de decisão da *Adapt*-DT:

Tabela 26 – Operação da Tabela de Decisão Adaptativa.

ALGORITMO – OPERAÇÃO DA TDA
1. Buscar em um conjunto de regras R uma ou mais regras que satisfaçam as condições de entrada: a. Caso encontre uma ou mais regras aplicáveis, armazena todas elas em um conjunto de regras aplicáveis; b. Caso contrário, retorna uma mensagem informando a inexistência de regras aplicáveis e prepara o dispositivo para uma nova entrada, posicionado a execução no passo 1.
2. Extrair as regras aplicáveis do conjunto de regras: a. Caso apenas uma regra seja aplicável, executar o passo 3; b. Caso mais de uma regra seja aplicável, uma situação de não determinismo é detectada. A seguinte sequência de operações é executada neste caso: i. Um objeto da classe *NonDeterministic* é instanciado; ii. Esse objeto cria um *thread* para cada regra aplicável. Cada *thread* cria uma nova instância da tabela de decisão adaptativa em operação; iii. O objeto *NonDeterministic* fornece os parâmetros necessários para continuar o passo. Cada nova instância da tabela de decisão continua o processo executando os passos 3 e 4, para decodificar a regra e executar as ações, respectivamente; iv. Após executar os passos 3 e 4, os resultados são retornados para a instância *NonDeterministic*; v. A instância *NonDeterministic* apresenta os resultados obtidos com a aplicação de cada uma das regras.
3. Decodificar as ações: a. No caso de uma regra não adaptativa: i. Extrair as ações associadas às regras. b. No caso de uma regra adaptativa: i. Extrair as ações adaptativas associadas às regras.
4. Executar as ações: a. No caso de uma regra não adaptativa: i. Executar as rotinas semânticas associadas a cada ação. b. No caso de uma regra adaptativa: i. Executar as rotinas semânticas associadas a cada ação. c. No caso de uma regra adaptativa: i. Para uma função adaptativa posterior, executar o passo 4.a.i e na sequência executar o passo 4.b.iii; ii. Para uma função adaptativa anterior, executar o passo 4.b.iii e na sequência executar o passo 4.a.i. iii. Executar as ações adaptativas.

5.1. FUNCIONALIDADES E INTERFACE GRÁFICA

Nesta seção são destacadas as principais funções da ferramenta. É apresentada a interface gráfica e o modo de utilização da ferramenta.

A Figura 19 mostra a interface principal do programa. A interface do *Adapt*-DT possibilita a visualização das regras de decisão bem como os resultados do algoritmo de tomada de decisão. Essa característica permite ao projetista do sistema acompanhar a aplicação das regras, servindo como suporte ao processo decisório.

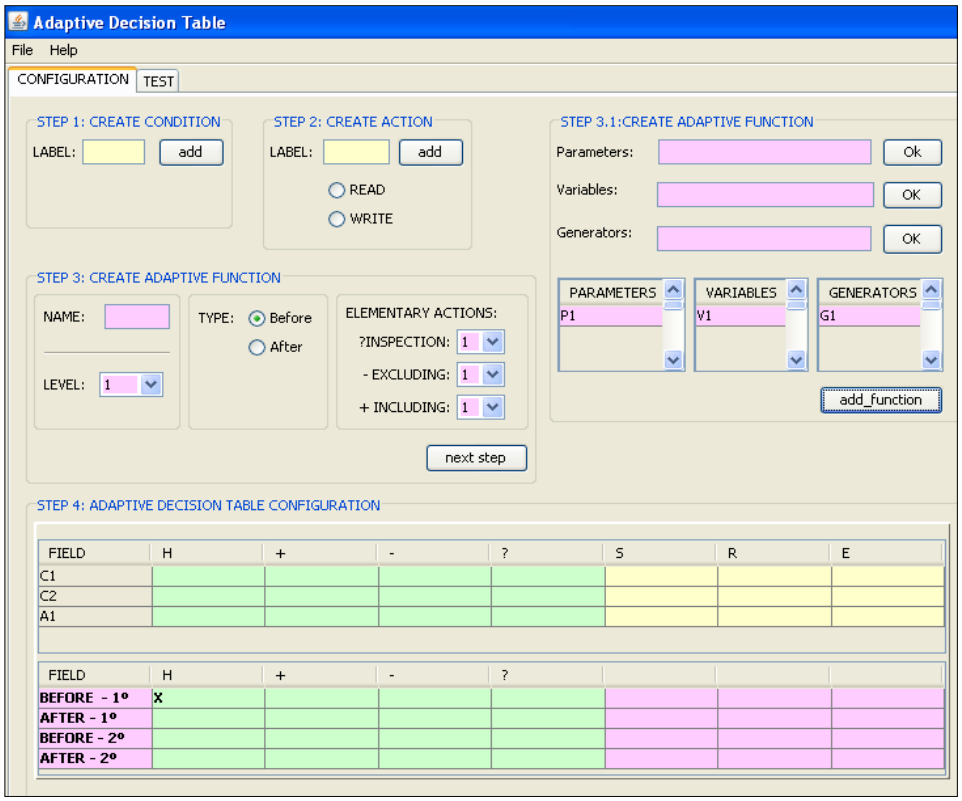
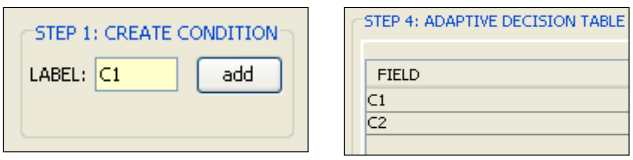


Figura 19 – Tela inicial da ferramenta.

Na sequência, é apresentada passo a passo a especificação e utilização da ferramenta, e optou-se pela expansão de cada passo da tela principal, para que as características da interface e da operação do software pudessem ser descritas de forma mais detalhada.

O primeiro passo para o usuário configurar a tabela é criar os rótulos para as condições, conforme a Figura 20. As *n* condições definidas pelo projetista são apresentadas nas linhas de *1* a *n* da tabela e coluna *Field* da tabela de decisão.

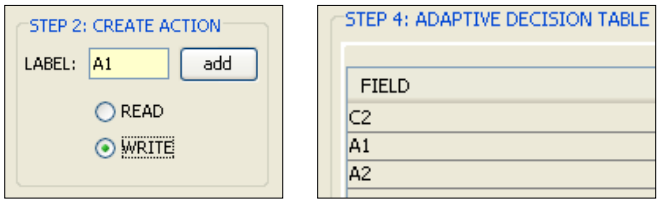


(a) (b)

Figura 20 – (a) Criar rótulo para as condições e (b) Visualizar as condições criadas.

O segundo passo é criar os rótulos para as ações, conforme a Figura 21. Além de definir um rótulo, é necessário escolher a operação que deve ser realizada pela ação.

Inicialmente são permitidas apenas ações de leitura e escrita, conforme as opções *Read* e *Write*, mostradas na mesma figura. As *m* ações definidas pelo projetista são apresentadas nas linhas *n+1* a *m* da tabela e coluna *Field* da tabela de decisão.



(a) (b)

Figura 21 – (a) Criar rótulo para as ações e (b) Visualizar as ações criadas.

Após definir rótulos para as condições e ações, o terceiro passo é declarar as funções adaptativas.

A criação das funções adaptativas ocorre como descrito a seguir. Primeiramente, é definido o cabeçalho da função. No cabeçalho, é escolhido o nome da função adaptativa e o tipo da função que pode ser "*Before*" para função adaptativa anterior ou "*After*" para função adaptativa posterior. Na sequência, o usuário deve definir a quantidade de ações elementares para cada um dos tipos possíveis de ações adaptativas, a saber: "?" para consulta, "-" para exclusão e "+" para inclusão.

A Figura 22 apresenta o primeiro passo para a declaração da função adaptativa e o botão *next_step*, que habilita o próximo passo para a declaração da função adaptativa.

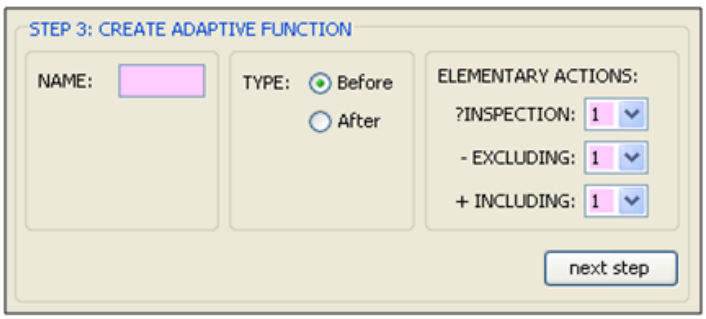
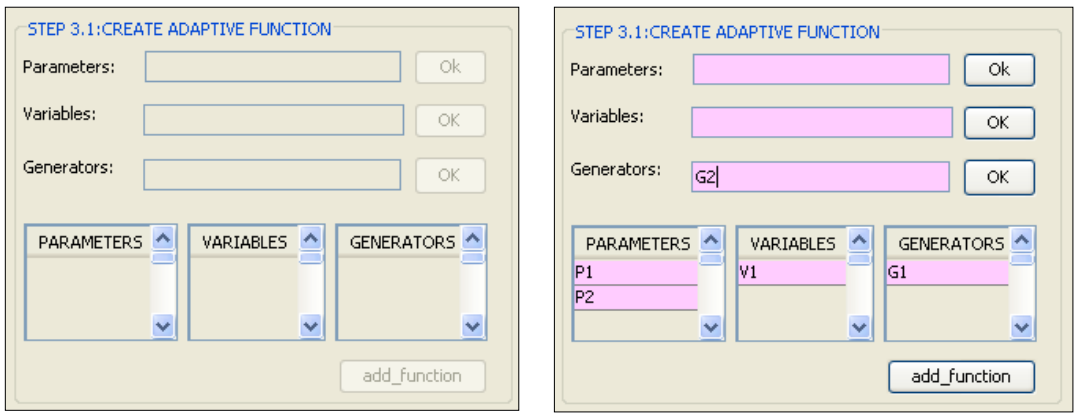


Figura 22 – Declaração das funções adaptativas.

No passo seguinte, são definidos parâmetros e, se necessário, variáveis e geradores. Os parâmetros são utilizados para referenciar os valores passados como argumento nas chamadas às funções adaptativas. As variáveis são utilizadas para armazenar os valores resultantes de uma ação elementar de consulta. Os geradores são utilizados

para referenciar e gerar novos valores durante a execução de uma ação elementar. A Figura 23 mostra, respectivamente, os campos desabilitados e habilitados para preenchimento dos dados.



(a) (b)

Figura 23 – (a) Preenchimento desabilitado e (b) habilitado para definição de parâmetros, variáveis e geradores.

Na Figura 23, o botão *add_function* adiciona uma função a uma lista de funções. A lista de funções é mostrada na Figura 24. Para visualizar o conteúdo das funções adaptativas é necessário selecionar a função desejada na lista, conforme mostrado na mesma figura.

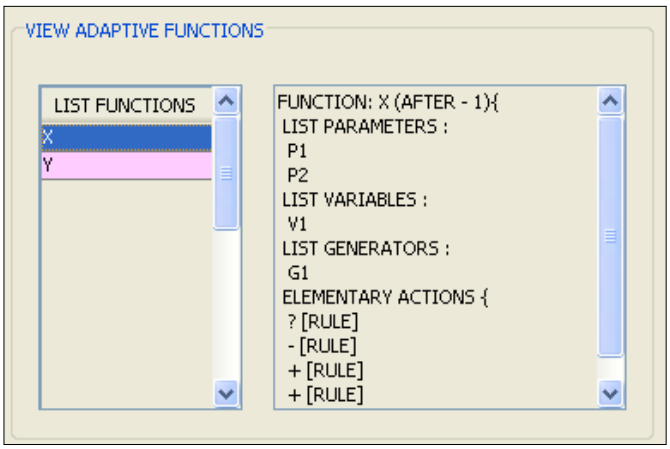


Figura 24 – Lista de funções adaptativas e visualização da função adaptativa.

Após definir as funções adaptativas, o próximo passo é configurar a tabela de decisão adaptativa. A Figura 25 mostra a tabela de decisão adaptativa com as regras S e E pré-definidas. “S” significa à regra inicial, caso seja necessário definir uma ordem para a busca de regras. “E” corresponde à regra final, ou seja, não existem regras a serem buscadas após essa regra.

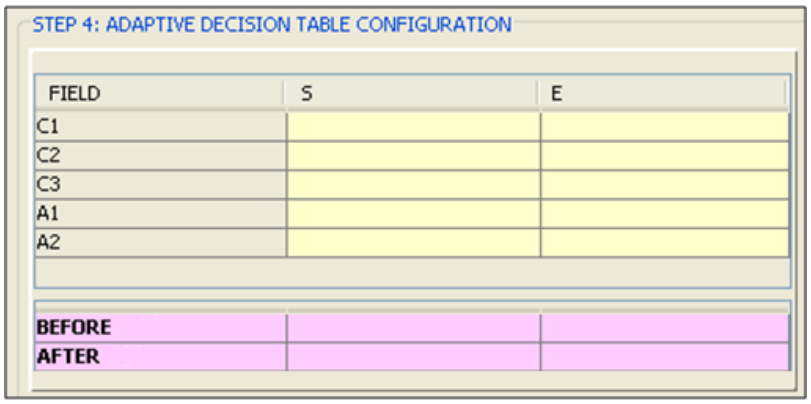


Figura 25 – Regra inicial e final pré-definidas.

A Figura 26 apresenta a tabela com a inclusão da regra R, em destaque. O botão *add_rule* permite adicionar regras na tabela. O botão *del_rule* remove a regra selecionada na tabela. Apenas as regras S e E não podem ser removidas. O *remove_rows* remove as linhas selecionadas. Observe que a tabela é divida em duas partes principais: as linhas amarelas correspondem às condições e ações da regra; as linhas rosa correspondem às chamadas de funções adaptativas.

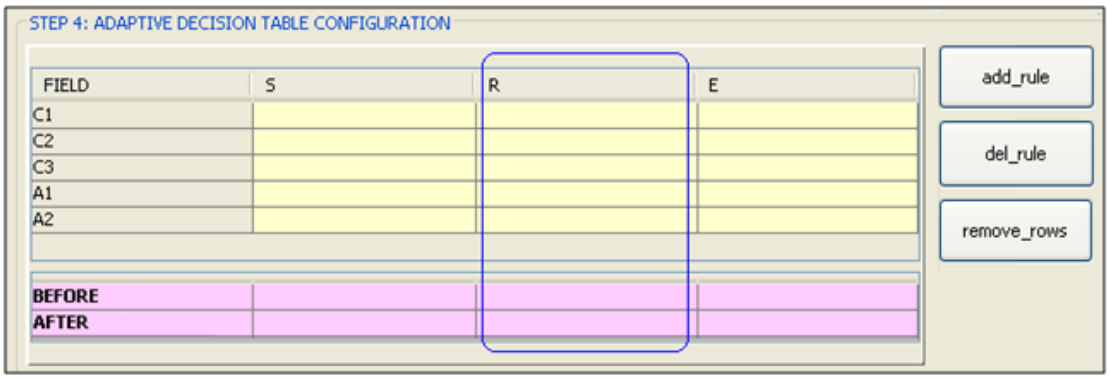


Figura 26 – Inclusão e exclusão de regras R.

Com as condições, ações e funções adaptativas definidas, o próximo passo é configurar as regras do dispositivo.

5.2. APLICAÇÃO DA FERRAMENTA

A seguir, é apresentado um exemplo ilustrativo para explorar a utilização da *Adapt*-DT na simulação de problemas de classificação.

Considere o conjunto de atributos A = {α_1, α_2, α_3, α_4, α_5, α_6, α_7}, onde:

- α_1 = “temperatura corporal”, com temperatura corporal = {quente, frio}; α_2 = “cobertura da pele”, com cobertura da pele = {cabelos, escamas, pelos, penas,

espinhos, não possui}; α_3 = "ovíparo", com ovíparo = {sim,não}; α_4 = "criatura aquática", com criatura aquática = {sim,não,semi}; α_5 = "criatura aérea", com criatura aérea = {sim,não}; α_6= "possui pernas", com possui pernas = {sim,não}; e α_7 = "hiberna", com hiberna = {sim,não}.

Seja Ω o conjunto de classes distintas, onde: ω_1 = "Mamífero"; ω_2 = "Pássaro"; ω_3 = "Peixe"; ω_4 = "Réptil"; e ω_5 = "Anfíbio".

O conjunto de dados de treinamento *P*, extraído de (TAN; STEINBACH; KUMAR, 2005) é apresentado na Tabela 27.

Tabela 27 – Conjunto de dados de treinamento.

CONJUNTO DE TREINAMENTO									
Exemplos de Treinamento		Atributos							
		α_1	α_2	α_3	α_4	α_5	α_6	α_7	Ω Classe
ρ_1	Humano	quente	cabelos	não	não	não	sim	não	Mamífero
ρ_2	Cobra	fria	escamas	sim	não	não	não	sim	Réptil
ρ_3	Salmão	fria	escamas	sim	sim	não	não	não	Peixe
ρ_4	Baleia	quente	cabelos		sim	não	não	não	Mamífero
ρ_5	Sapo	fria	não possui	sim	semi	não	sim	sim	Anfíbio
ρ_6	Morcego	quente	cabelo	não	não	sim	sim	sim	Mamífero
ρ_7	Pomba	quente	penas	sim	não	sim	sim	não	Pássaro
ρ_8	Gato	quente	pelos	não	não	não	sim	não	Mamífero
ρ_9	Tartaruga	fria	escamas	sim	semi	não	sim	não	Réptil
ρ_{10}	Pinguim	quente	penas	sim	semi	não	sim	não	Pássaro
ρ_{11}	Porco Espinho	quente	espinhos	não	não	não	sim	sim	Mamífero
ρ_{12}	Enguia	fria	escamas	sim	sim	não	não	não	Peixe
ρ_{13}	Salamandra	fria	não possui	sim	semi	não	sim	sim	Anfíbio

Considere um algoritmo de aprendizagem, a saída do algoritmo de aprendizagem é um classificador τ. Neste caso, representado por um conjunto de regras do tipo *"Se...então"* e apresentado em uma tabela de decisão convencional. Dado um novo padrão χ, o classificador deve predizer o valor correspondente da classe Ω associada ao padrão. O conjunto *R* inferido do conjunto *P* é mostrado na Figura 27.

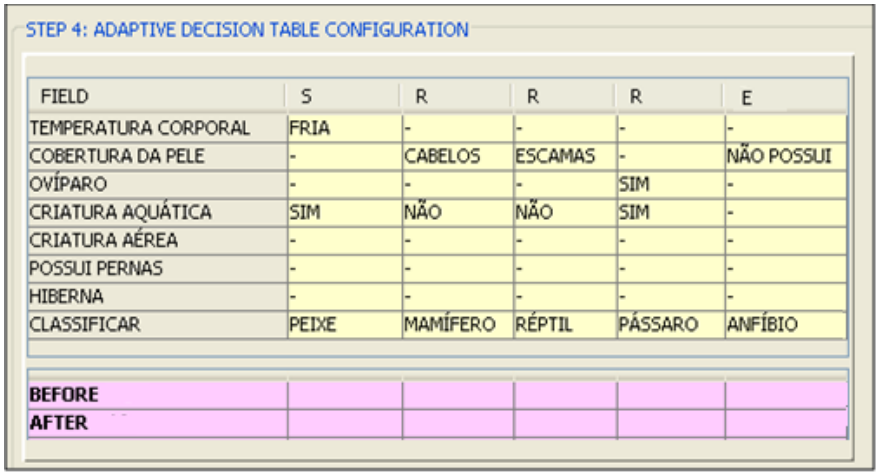


Figura 27 – Regras de classificação na tabela de decisão.

As tabelas de decisão, assim como as árvores de decisões ou redes neurais (WITTEN; EIBE, 2005), são modelos de classificação usados para o reconhecimento de padrões.

Na fase de avaliação do processo de aprendizagem, os dados de testes são submetidos ao classificador. Como resultado, quatro casos resultantes são esperados.

Caso 1: Apenas uma regra do conjunto *R* é aplicável ao dado de entrada e o classificador rotula o padrão corretamente. Por exemplo, para χ = {fria, escamas, sim, não, não, não, sim} a regra aplicável é terceira regra. Neste caso, o padrão é classificado como réptil. O exemplo χ é uma cobra, portanto, a classificação está correta.

Para simular a classificação no *Caso 1*, utilizando a ferramenta, os dados de entrada devem ser carregados de um arquivo com extensão *.txt*, conforme Figura 28.

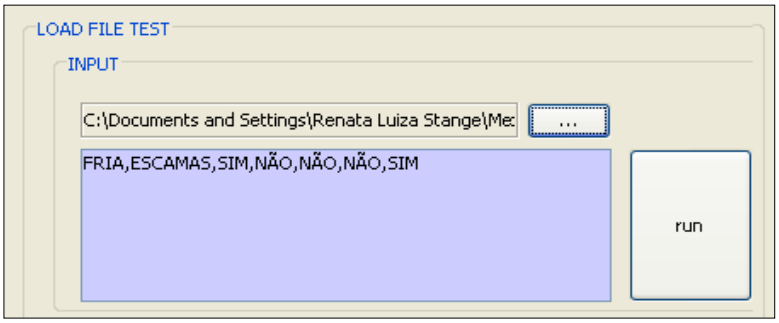


Figura 28 – Carregar o arquivo de entrada.

A Figura 29 mostra o formato do arquivo de entrada. Cada condição a ser testada é separada por uma vírgula (,).

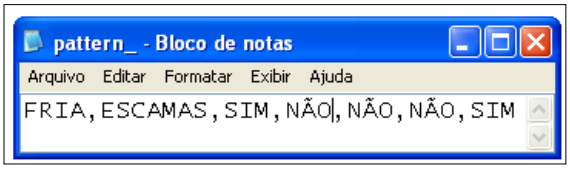


Figura 29 – Formato dos dados de entrada.

Para a entrada do arquivo da Figura 29, com o conjunto de regras da Figura 27, o resultado da execução é mostrado na Figura 30.

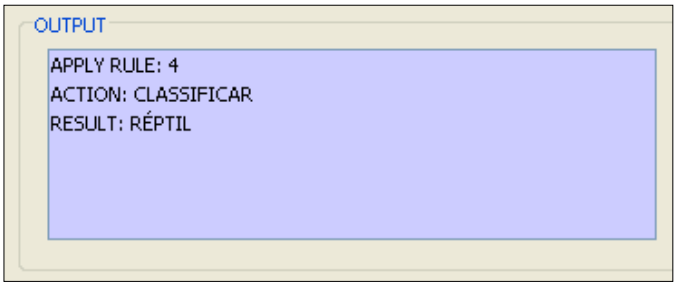


Figura 30 – Saída do dispositivo.

Caso 2: Nenhuma regra do conjunto *R* é aplicável. Isso significa que o classificador não é capaz de classificar χ. Os motivos podem ser diversos, tais como: 1) χ não é um padrão, neste caso, χ deve ser descartado; 2) χ é um padrão, porém o classificador não aprendeu as regras necessárias para classificá-lo, neste caso, este problema ocorreu na fase de aprendizagem e os motivos podem ser: o método de aprendizagem não é adequado, por exemplo, os atributos selecionados pelo método para discriminar as classes são insuficientes, ou não são bons discriminantes; 3) Ocorrência de problemas nas amostras de treinamento (dados ruidosos, dados ausentes, quantidade de dados insuficiente); χ é um novo padrão que deverá ser aprendido, a *priori*, não pertencia ao conjunto de treinamento *P*.

Para ilustrar o *Caso 2*, considere χ = {quente, pelos, não, não, não, sim, não}. Neste caso, não existe uma regra aplicável. Porém, é sabido que χ é um exemplo de gato e deveria ser classificado como mamífero. O algoritmo de aprendizagem não foi capaz de aprender a regra que classifica um gato como mamífero a partir do conjunto P. Para isso, uma regra adaptativa E é inserida ao conjunto *R*, conforme Figura 27. Essa regra é satisfeita toda vez que nenhuma regra convencional for aplicável.

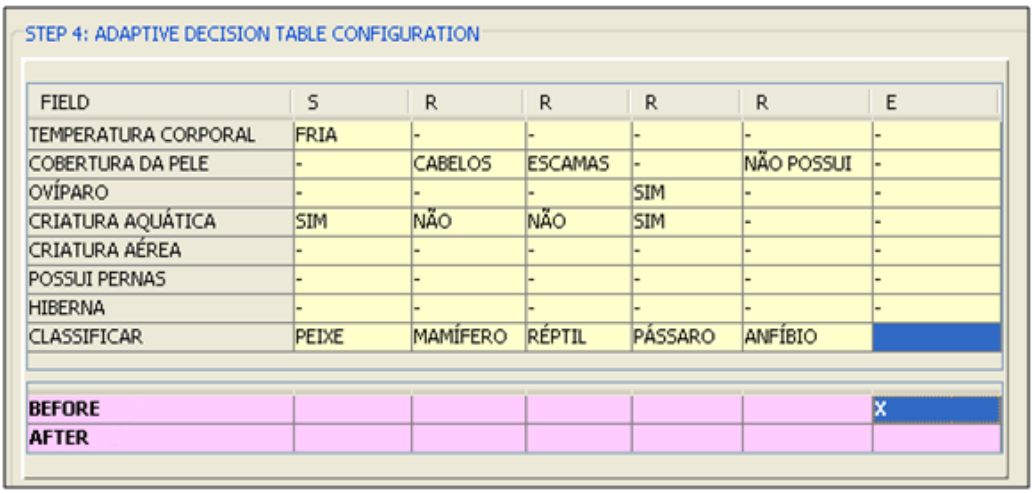


Figura 31 – Um exemplo de configuração inicial para a tabela de decisão adaptativa.

A regra *E* é uma regra adaptativa e possui a função adaptativa X associada, especificada a seguir:

Exemplo do pseudocódigo para declaração de uma função adaptativa.

```
Função Adaptativa X(Parâmetros: p1, p2, p3, p4, p5, p6, p7){
  Variáveis: v1, v2, v3, v4, v5, v6, v7
  Geradores: Geradores de rótulos *gt
  Ações elementares Δ{
     δ1:?[v1, v2, v3, v4, v5, v6, v7] (Existe uma regra que satisfaça as condições χ).
     δ2:-[v1,v2,v3,v4,v5,v6,v7](Se existir, remover).
     δ3:+[p1,p2,p3,p4,p5,p6,p7,*g1](Inserir uma regra rt).
  }
}
```

A função adaptativa X previamente declarada, fica disponível para edição do corpo da função, conforme Figura 32. O corpo da função adaptativa deve especificar as modificações que podem ocorrer no conjunto de regras da tabela de decisão adaptativa.

No caso de aplicação da regra E em χ, a ação elementar de consulta irá pesquisar se existe uma regra em *R* que satisfaça χ. Se existir, a ação elementar de remoção (δ_2) irá removê-la. Em seguida, a ação elementar de inserção (δ_3) inclui uma nova regra no conjunto R. Por exemplo, ao receber as entradas χ_1 = {quente, pelos, não, não, não,

sim, não}, χ_2 = {quente, pelos, não, não, não, sim, não} e χ_3 = {quente, penas, sim, semi, não, sim, não}, a regra E será aplicada e o conjunto de regras R será modificado.

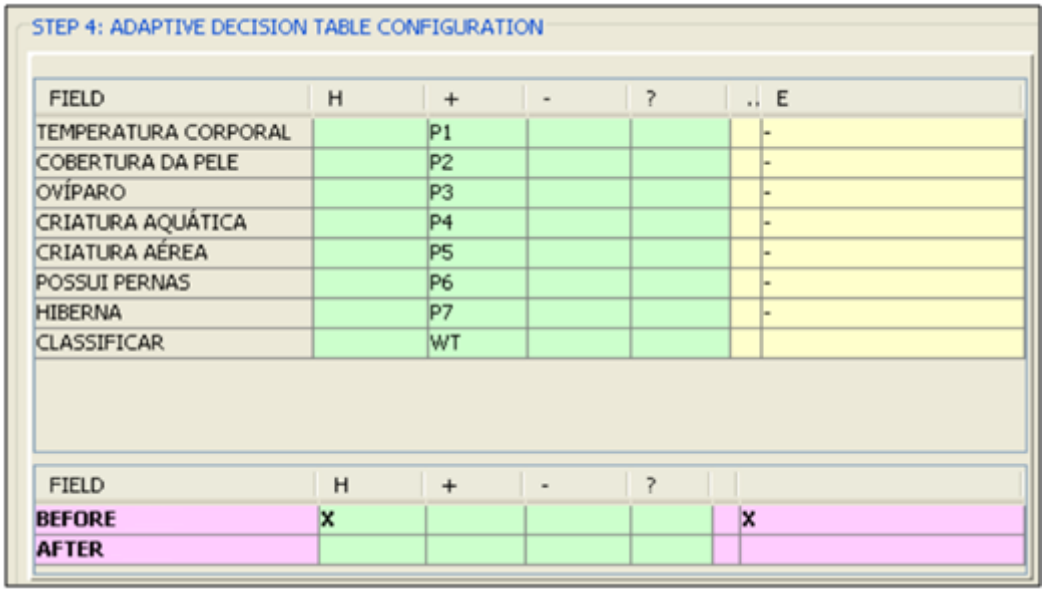


Figura 32 – Associando funções adaptativas às regras.

A Figura 32, mostra no cabeçalho da função, representado pela coluna "H", uma função adaptativa chamada "X". A linha "BEFORE" indica que "X" é uma função adaptativa anterior. Os cabeçalhos das colunas "+", "-" e "?" indicam, respectivamente, as ações elementares de inserção, remoção e consulta da função adaptativa "X". Para exemplificar, apenas o corpo da ação elementar de inserção está preenchido.

A Tabela 28 mostra a tabela de decisão adaptativa após sucessivas transformações. As transformações do dispositivo são representadas por TDA_j para j = 0, 1, 2, ... *m* onde m é o número de modificações do dispositivo. Optou-se pela apresentação em tabela e não diretamente na ferramenta, por questões de legibilidade da figura.

Posteriormente, as novas regras deverão ser observadas para inferir novas regras de classificação e assim aprender a classificar corretamente novos padrões não cobertos pelas regras iniciais.

Contudo, é importante destacar que, no exemplo da Tabela 28, apenas uma regra adaptativa foi inserida, porém é possível incluir outras regras adaptativas com funções adaptativas capazes de realizar diferentes modificações na tabela de decisão adaptativa.

Tabela 28 – Tabela de decisão adaptativa após 3 modificações TDA_3.

Cabeçalhos (Tags)			H	?	-	+	r_1	r_2	r_3	r_4	r_5	r_6	r_{t1}	r_{t2}	r_{t3}
Condições	Temperatura corporal			p_1	v_1	v_1	fria	-	-	-	-	-	quente	quente	quente
	Cobertura da pele			p_2	v_2	v_2	-	cabelo	escamas	-	Não possui	-	pelos	espinhos	penas
	Ovíparo			p_3	v_3	v_3	-	-	-	sim	-	-	não	não	sim
	Criatura aérea			p_4	v_4	v_4	-	-	-	-	-	-	não	não	semi
	Criatura aquática			p_5	v_5	v_5	sim	não	não	não	-	-	não	não	não
	Possui pernas			p_6	v_6	v_6	-	-	-	-	-	-	sim	sim	sim
	Hiberna			p_7	v_7	v_7	-	-	-	-	-	-	não	sim	não
Ações	Classe					$*g_1$	peixe	mamífero	réptil	pássaro	anfíbio	?	$*g_1$	$*g_2$	$*g_3$
Funções adaptativas	Anterior	f_1	B	√	√	√						√			
	Posterior														
	Parâmetros	p_1	P									p_1			
		...	...									...			
		p_7	P									p_7			
	Variáveis	v_1	V												
		...	...												
		v_7	V												
	Geradores	g_1	G												

Caso 3: Existe mais de uma regra aplicável no conjunto *R*. Trata-se, portanto, de uma situação não determinística. Isso pode ocorrer quando os dados de treinamento são ambíguos ou quando o conjunto de treinamento é pequeno, entre outros motivos. Na implementação da ferramenta para simulação de tabelas de decisão adaptativas, é proposta uma solução utilizando o mecanismo de *threads*. Um *thread* é um fluxo de controle sequencial em um programa. A programação *multi-threaded* é uma forma de programação paralela, onde vários *threads* são executados concorrentemente em um programa. Os *threads* executam em um mesmo espaço de memória, podendo trabalhar concorrentemente sobre dados compartilhados.

Caso 4: Existe uma regra aplicável no conjunto *R*, porém o classificador está classificando os padrões de maneira incorreta. Isso pode ocorrer por várias razões, por exemplo, o método de aprendizagem pode ser inadequado para as características dos dados de treinamento ou os dados de treinamento podem ser insuficientes para uma boa taxa de aprendizagem. (DUDA; HART; STORK, 2001). Neste caso, é necessário rever a construção do classificador.

PARTE III

Considerações Finais

6 CONSIDERAÇÕES FINAIS

O estudo comparativo dos métodos estatísticos e determinísticos permitiu a identificação de problemas de aprendizagem que podem ser solucionados utilizando adaptatividade, tais como os problemas de natureza estruturada e dinâmica. O levantamento das principais características dos métodos estatísticos e determinísticos indicou a aplicação de métodos híbridos, incluindo técnicas adaptativas.

Os métodos disponíveis para os problemas de aprendizagem propõem diversas soluções na tentativa de construir sistemas de aprendizado eficientes e com um grau de precisão razoável na tomada de decisão. Na prática, em problemas de processamento da língua natural, um método híbrido pode ser uma alternativa de busca de solução no processo de aprendizagem. Por exemplo, aplicando uma técnica estatística (ex.: *naïve Bayes*), a busca por uma possível solução em amostras grandes é reduzida, o seja, o espaço amostral é reduzido. Em seguida, mecanismos de inferência gramatical e gramática adaptativa podem ser utilizados para tratamento de características estruturais e dinâmicas do problema. A decisão final não seria baseada apenas nas regras baseadas em probabilidades, mas também na relevância de cada indivíduo da amostra.

Para exemplificar, podemos citar a ferramenta Google *Translate*[10], que utiliza métodos estatísticos para aprender a traduzir textos em diferentes idiomas. O poder da ferramenta de tradução está no uso de supercomputadores para processar um imenso banco dados. O banco de dados foi formado inicialmente com textos oficiais da ONU vertidos para seis idiomas (exemplos de treinamento) e continua crescendo gradualmente com a inclusão de traduções sugeridas por usuários. Atualmente (desde 2010), o grupo de pesquisa da Google tem pesquisado sobre a inserção de regras gramaticais nos algoritmos de tradução do Google Translate (PAVÃO JUNIOR, 2010). Segundo Miles Osborne, pesquisador da Universidade de Edimburgo, na Escócia, que trabalhou no projeto do Google, [...] “os métodos estatísticos são limitados para esse tipo de problema e a redução das altas taxas de erro, inevitáveis nas traduções realizadas usando uma técnica puramente estatística, acabam

[10] Disponível em http://translate.google.com.

necessitando de uma gramática". A utilização de um método híbrido tende a compor textos mais fluentes.

A aprendizagem de máquina usando adaptatividade considera a integração de técnicas de aprendizagem de máquina, determinísticas ou estatísticas, e técnicas adaptativas para a construção de sistemas de aprendizado.

Na aprendizagem tradicional, ou seja, sem a utilização de técnicas adaptativas, os ajustes para a melhoria no processo de aprendizagem, em geral, ocorrem na fase de treinamento. Porém, muitas informações relevantes podem ser capturadas na fase de utilização do sistema (ex.: na fase de classificação). Em geral, isso ocorre quando o problema é de natureza dinâmica. Com a utilização de técnicas adaptativas, através dos dispositivos adaptativos, é permitido descrever de forma natural os aspectos dinâmicos dos problemas de aprendizagem.

A aprendizagem adaptativa é uma forma de adaptar gradualmente o modelo aprendido. A adaptatividade pode ser útil para detectar ajustes no conjunto de regras e modificar o modelo, de maneira incremental. Embora existam maneiras de refazer a estrutura de dispositivos convencionais, tais como árvores e tabelas de decisão, após a obtenção de novos dados, refazer o conjunto de regras pode ser ineficiente, podendo levar à perda parcial ou total de informações que haviam sido anteriormente aprendidas.

A grande vantagem da adaptatividade sobre outras técnicas correntemente utilizadas para a formulação de modelos de representação e de manipulação do conhecimento reside no fato de que:

(a) a informação total, encerrada no dispositivo adaptativo, está representada integralmente no conjunto de regras. A memória é representada pelo próprio dispositivo;

(b) a aprendizagem angariada em cada passo adaptativo do dispositivo encontra-se integralmente confinada à variação sofrida pelo conjunto de regras. É possível determinar o que foi aprendido pelo dispositivo adaptativo analisando as alterações de configuração durante o processo de aprendizagem;

(c) a observação, a identificação, a qualificação e a quantificação da evolução da aprendizagem, bem como a relação de causa e efeito entre a sequência de entradas processadas, a das regras aplicadas e a das variações sofridas pelo conjunto de regras, pode ser efetuado única e exclusivamente pela análise do conjunto de regras do dispositivo adaptativo e das suas variações ao longo da operação do dispositivo.

Em segundo plano, a adaptatividade permite que as diversas características inerentes aos problemas reais de aprendizagem, tais como comportamento dinâmico e estocástico, possam ser tratadas de forma transparente. A interação do especialista no mecanismo de aprendizagem é simplificada e não apresenta complexidades técnicas, pois estas devem ficar a cargo do mecanismo de aprendizagem. De fato, o especialista deve manter o controle da definição do problema, incluindo o conhecimento sobre o domínio, restrições ou preferências através da definição das funções adaptativas. Esta é a parte que, dependendo do nível da adaptatividade, não pode ser "automatizada".

Contudo, os métodos que utilizam dispositivos simbólicos para representar o que foi aprendido são adequados quando é desejável obter um conjunto de regras mais compreensível por especialistas humanos. Considerando que esses dispositivos podem ser representados por um conjunto de regras do tipo "*Se...então*", os dispositivos adaptativos se enquadram na representação simbólica das regras aprendidas. Tal formato utilizado em dispositivos adaptativos fornece uma perspectiva unificada sob a qual as regras de um mecanismo de aprendizado podem ser convertidas e analisadas.

Outro aspecto percebido, similar ao observado em dispositivos baseados em regras, tais como árvores de decisão, expressões lógicas, regras de produção e tabelas de decisão, é a naturalidade que se tem em modelar os problemas de aprendizagem de máquina utilizando diferentes dispositivos adaptativos. Os autômatos adaptativos, as tabelas de decisão adaptativas e as árvores de decisão são equivalentes e facilmente convertidos.

Em particular, os dispositivos adaptativos, tais como autômatos adaptativos, árvores ou tabelas de decisão adaptativas, permitem a representação estrutural das regras. Segundo Michalski (1983), a criação de estruturas simbólicas que sejam compreensíveis e utilizadas por modelos mentais na aprendizagem é mais interessante do que os modelos estatísticos.

A combinação equilibrada entre a obtenção de modelos de aprendizagem compreensíveis e expressivos pode ser adquirida com a utilização de dispositivos adaptativos. Tanto os autômatos adaptativos quanto as tabelas de decisão adaptativas, e intuitivamente, outros dispositivos adaptativos utilizados para representar o conhecimento adquirido em sistemas de aprendizagem, facilitam o entendimento das regras. A estrutura simples e compreensível dos dispositivos adaptativos são ferramentas úteis para a descoberta do conhecimento.

7 CONTRIBUIÇÕES

Com o estudo de caso foi possível concluir que os dispositivos adaptativos permitem representar de forma satisfatória um problema de aprendizagem de máquina, bem como identificar melhorias para o processo de aprendizagem, pois o formato das regras facilita a interpretação do modelo de aprendizagem pelo projetista ou especialista.

Outra constatação é que é possível prever o comportamento de um processo durante a sua análise. Ainda, é possível definir como os dados podem ser utilizados para construir um sistema de aprendizado a partir do processo que os gerou e entender melhor os dados e utilizá-los para ganhar algum tipo de vantagem para melhorar o processo. Muitos sistemas de aprendizado são avaliados apenas em função da precisão na tomada de decisão. Neste trabalho, os sistemas de aprendizado também são avaliados sob a perspectiva de compreensibilidade das regras aprendidas.

A aprendizagem de máquina utilizando tecnologia adaptativa pode ser considerada uma técnica inspirada na aprendizagem indutiva supervisionada com o objetivo de melhorar o entendimento das regras, mas principalmente cumprir exigências de eficiência. Na técnica, uma base de regras é utilizada para representar o comportamento inteligente de um jogador em função de experiências passadas.

As regras (ex.: uma jogada é uma regra) são extraídas da memória e aplicadas durante as partidas podendo gerar novas regras que alteram o comportamento dos jogadores ou a base de regras (inclusão ou exclusão de regras). A probabilidade de aplicação de uma regra que leva o jogador a uma vitória é proporcional ao ganho de informação associada a esta regra (ex.: jogadas com maior ganho de informação têm maior probabilidade de vencer a partida).

A aplicação da medida estatística baseada no ganho de informação pode ser considerada uma forma de avaliar os dispositivos adaptativos. A avaliação é interessante para testar a eficácia e eficiência dos dispositivos adaptativos. Neste caso, não parece ser vantajoso considerar que os dispositivos adaptativos apenas melhoram a expressividade e a compreensibilidade da solução, comparados a dispositivos não adaptativos. Porém, ainda é necessário avançar em formas de avaliar os dispositivos

para analisar o funcionamento das funções adaptativas e determinar se as ações correspondentes estão atingindo os resultados esperados.

Por fim, a implementação de uma ferramenta de software didática e flexível, que permita aos projetistas de software construir sistemas de informação utilizando dispositivos adaptativos é outra contribuição tecnológica do trabalho. A ferramenta computacional apresentada comporta os conceitos relacionados às tabelas de decisão adaptativas de maneira clara, permitindo que usuários ainda não familiarizados com os conceitos de adaptatividade entendam e consigam projetar softwares adaptativos através da ferramenta.

A interface gráfica da *Adapt*-DT conduz o usuário no processo de especificação de sistemas baseado em tabelas de decisão adaptativas. A facilidade de utilização e os aspectos didáticos proporcionados pela interface gráfica tornam a ferramenta apropriada para o ensino de tecnologia adaptativa.

Contudo, a disponibilidade de ferramentas de software pode estimular a pesquisa de fundamentos das técnicas adaptativas. Os fundamentos que sustentam a tecnologia adaptativa são considerados sólidos por estarem baseados em teoria da computação. Porém, nem todos esses fundamentos foram investigados de maneira exaustiva e a disponibilidade de novas ferramentas que integram teoria e prática podem ser úteis para essa investigação.

8 SUGESTÕES PARA TRABALHOS FUTUROS

No desenvolvimento deste trabalho, foram identificadas algumas necessidades de explorar novas soluções em tecnologia adaptativa. Na engenharia de software, a necessidade de uma metodologia para o desenvolvimento de software adaptativo é importante para a especificação de sistemas de aprendizagem utilizando adaptatividade. O desenvolvimento de métodos, técnicas e ferramentas para sistematizar o desenvolvimento de softwares adaptativos seria de grande valia. Um mapeamento dos desafios nesta área e levantamento do estado da arte também representaria uma importante contribuição para o desenvolvimento da pesquisa em tecnologia adaptativa.

Um passo importante para o crescimento da pesquisa em adaptatividade é a determinação de medidas capazes de avaliar o conjunto de regras do dispositivo adaptativo. Através de medidas que permitissem uma análise mais onerosa do conjunto de regras, seria possível identificar se as ações adaptativas aplicadas a este conjunto atingem uma configuração de convergência, a ponto de que a aplicação de novas ações adaptativas não faça mais sentido, de forma que elas possam ser removidas ou alteradas.

9 REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA JUNIOR, J. R.: *Stad: Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos*. 202p. Tese (Doutorado em Engenharia), EPUSP, São Paulo, 1995.

ALPAYDIN, E.: *Introduction to Machine Learning*. MIT Press, 2ª Edição, 2010. ISBN-10: 0-262-01243-X

BASSETO, B. A.: *Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos*. Dissertação (Mestrado em Engenharia), EPUSP, São Paulo, 2000.

BECK, J.: *Combinatorial games: tic-tac-toe theory*. Cambridge University Press, 2008. ISBN 9780521461009.

BEZDEK, J.C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

BISHOP, C. M.: *Pattern Recognition and Machine Learning*. Berlin: Springer, 2006. ISBN 0-387-31073-8.

_____. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. ISBN 0-19-853864-2 (Paperback)

BREIMAN, L.; FRIEDMAN, J; OHLSEN, R.; STONE, C.: *Classification and regression trees*. Wadsworth, Belmont, CA, 1984.

BUNKE, H.; KANDEL, A.: *Hybrid Methods in Pattern Recognition*, Series in Machine Perception and Artificial Intelligence - Vol. 47, 1990.

CLARK, P; NIBLETT, T.: *The CN2 Induction Algorithm*. Machine Learning, 3(4): 261-283, 1989.

DIAS, J. B.; SOUZA, K. P. de; PISTORI, H.: *Conjunto de Treinamento para Algoritmos de Reconhecimento de LIBRAS*. II Workshop de Visão Computacional, São Carlos, Outubro 16-18, 2006.

DUDA, R. O.; HART, P. E.; STORK, D. G.: *Pattern Classification*. 2ª Edição. Wiley, 2001. ISBN: 0471056693.

HOLLAND, J. H., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

FAYYAD, U. M; IRANI, K. B.: *Multi-interval discretization of continuous-valued attributes for classification learning*, in Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, Inc., pp. 1022 - 1027, 1993.

FISHER, R.: *The Use of Multiple Measurements in Taxonomic Problems* In: Annals of Eugenics, 7, p. 179—188, 1936.

FU, K. S.: *Syntactic Pattern Recognition and Applications*. Prentice-Hall, 1982. ISBN: 0138801207

GANZELI, H. S.; BOTTESINI, J. G.; PAZ, L. O.; RIBEIRO, M. F. S.: *Skan-Skin Scanner: software para o reconhecimento de câncer de pele utilizando técnicas adaptativas*. Memórias do WTA 2010 – IV Workshop de Tecnologia Adaptativa, São Paulo, 2010.

GONZALEZ, R.C.; THOMASON, M.G.: *Syntactic pattern recognition: an introduction*. Addison-Wesley Publishing Company, Reading, MA, 1978.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J.H.: *The Elements of Statistical Learning*, 1ª Edição. Springer, 2001.

HIRAKAWA, A. R.; SARAIVA, A. M.; CUGNASCA, C. E.: *Autômatos Adaptativos Aplicados em Automação e Robótica*. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 539-543)

HUGHES, M. L., SHANK, R. M., STEIN, E. S.: *Decision Tables*. Midi Publications, Management Development Institute, Divisions of Information, Industries, Inc., Wayne, Pennsylvania, 1968.

JAIN, A.K.; DUIN, R. P.W.; MAO, J.: *Statistical pattern recognition: A review*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22 (nº1):4–37, 2000.

LEWIS, H. R.; PAPADIMITRIOU, C. H.: *Elementos da Teoria da Computação*. Bookman, 2000. ISBN: 0-13-262478-8

MENEZES, P. B.: *Linguagens Formais e Autômatos*. 6ª Edição. Bookman, 2011. ISBN: 9788577807659

MICHALSKI, R. S.: *A theory and methodology of inductive learning*. In Machine Learning L R. S. Michalski, J. Carbonelli, and T. Mitchell, eds. Palo Alto, CA: Tioga Publishing, 1983.

MITCHELL, T. M.: *Machine Learning*. 1ª Edição. McGraw-Hill, 1997. ISBN: 0070428077.

MONARD, M. C.; BARANAUKAS, J. A.: *Aplicações de Inteligência Artificial: Uma Visão Geral*. São Carlos: Instituto de Ciências Matemáticas e de Computação de São Carlos, 2000.

NETO, J. J.: *Adaptatividade: Generalização Conceitual*. Memórias do WTA 2009, EPUSP, São Paulo, 2009. ISBN 978-85-86686-51-1

_____. *Adaptive Technology and Its Applications*. Encyclopedia of Artificial Intelligence. New York, v.1, p.: 37-44, 2009b.

_____. *Adaptive Rule-Driven Devices: General Formulation and Case Study*. Revista de Engenharia de Computação e Sistemas Digitais, São Paulo, v.1, n.1, p. 45-57, 2001.

_____. *Solving complex problems with Adaptive Automata*. Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): Implementation and Application of Automata 5th International Conference, CIAA 2000, Vol.2088, London, Canada, Springer-Verlag, pp.340, 2000.

_____. *Contribuição à metodologia de construção de compiladores*. 272p. Tese (Livre-Docência), EPUSP, São Paulo, 1993.

NETO, J. J., IWAI, M. K.: *Adaptive Automata for Syntax Learning*. In Anais da XXIV Conferência Latino-americana de Informática - CLEI 98, pg. 135–149, Quito, Equador, 1998.

PARIENTE, C. B.; NETO, J. J.; SANTANA, F. S.: *Towards an Adaptive Implementation of Genetic Algorithms*. I Taller Latinoamericano de Informática para la Biodiversidad (INBI) - CLEI 2007, San José, Costa Rica, 9-12 Octubre, 2007.

PAVÃO JUNIOR, J.: *A língua do Google*. Revista Veja, São Paulo, Editora Abril, 2163 edição, nº 18, p 122-131, 05 de maio de 2010.

PAVLIDIS, T.: *Structural Pattern Recognition*; Springer Series in Electrophysics 1; Springer – Verlag; 1980; ISBN: 3540084630.

PRATI, R. C.: *Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos*. Tese (Doutorado), ICMC-USP, São Carlos, 2006.

PISTORI, H. *Tecnologia Adaptativa em Engenharia de Computação: estado da arte e aplicações*. Tese (Doutorado em Engenharia), Escola Politécnica da USP, 2003.

PISTORI, H.; NETO, J. J.: *Decision Tree Induction using Adaptive FSA*. CLEI Eletronic Journal. Volume 6, Number 1, 2003a.

_____. *AdapTools: Aspectos de Implementação e Utilização*. Boletim Técnico PCS, EPUSP, São Paulo, 2003b.

_____. *AdapTree: Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas*. Anais Conferência Latino Americana de Informática - CLEI 2002. Montevidéu, Uruguai, Novembro, 2002.

_____. *An Experiment on Handshape Sign Recognition using Adaptive Technology: Preliminary Results*. XVII Brazilian Symposium on Artificial Intelligence - SBIA 04. São Luis, September 29 - October 1, 2004

PRESSMAN, R. S.: *Engenharia de Software*. 6ª Edição. Rio de Janeiro: McGraw-Hill, 2006. ISBN: 8586804576.

QUINLAN, J. R.*: Induction of decision trees*. Machine Learning, 1, 81-106, 1986

_____. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann, San Francisco, 1993.

RAJANI, N.F. DAR, G. BISWAS, R. RAMESHA, C.K.: *Solution to the Tic-Tac-Toe Problem Using Hamming Distance Approach in a Neural Network*. In: Second International Conference on Intelligent Systems, Modelling and Simulation - ISMS 2011, Cambodia, 2011.

ROCHA, R.L.; NETO, J.J.: *Autômato adaptativo, limites e complexidade em comparação com máquina de Turing*. In: Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, p. 33-48, 2000.

RUSSEL, S. J.; NORVIG, P.: *Artificial Intelligence: A Modern Approach*. 2º Edição. Prentice-Hall, 2002. ISBN - 10: 0137903952

STANGE, R. L.; NETO, J. J.: *Reconhecimento de Padrões em Classificadores - Comparação de Técnicas e Aplicações*. Memórias do WTA 2010 – Workshop De Tecnologia Adaptativa, EPUSP, São Paulo, 2010.

_____. *Aprendizagem Incremental Usando Tabelas De Decisão Adaptativas*. Memórias do WTA 2011 – Workshop De Tecnologia Adaptativa, EPUSP, São Paulo, 2011.

STANGE, R. L.; GIANNINI, T. C.; SANTANA, F. S.; JOSE, J.; MAURO SARAIVA, A.: *Evaluation of Adaptive Genetic Algorithm to Environmental Modeling of Peponapis and Cucurbita*. Revista IEEE América Latina, v. 9, p. 171-177, 2011

TAN, P.; STEINBACH, M.; KUMAR, V.: *Introduction to Data Mining*, Boston, MA, USA Addison-Wesley Longman Publishing Co., Inc., 2005.

TCHEMRA, A. H.: *Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério***.** Tese (Doutorado em Engenharia), EPUSP, São Paulo, 2009.

_____. *Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão*. I WTA – Workshop sobre Tecnologia Adaptativa, São Paulo, 2007.

TCHEMRA, A. H.; CAMARGO, R.: Descoberta de padrões em bases de dados utilizando Técnicas Adaptativas. III WTA – Workshop sobre Tecnologia Adaptativa, São Paulo, 2009.

THEODORIDIS, S.; KOUTROUMBAS, K.: *Pattern Recognition*. 3º Edição. Academic Press, 2006.

TSAI, W.H.; FU, K.S.: *Attributed grammar - A tool for combining syntactic and statistical approaches to pattern recognition*. IEEE Trans. Syst., Man, and Cybern., SMC-10, pp. 873-885, 1980.

WITTEN, I. H.; EIBE, F.: *Data Mining: Practical Machine Learning Tools and Techniques*. 2ª Edição. Morgan Kaufmann, 2005. ISBN 0-12-088407-0

WIDYANTORO, D. H.; VEMBRINA, Y. G: *Learning to play tic-tac-toe*. In: International Conference on Electrical Engineering and Informatics, 2009.

UTGOFF, P. E.: *Incremental Induction of Decision Trees*. Machine Learning, Machine Learning, 4, 161-186, 1989.

UTGOFF, P.E.; BERKMAN, N.C.; CLOUSE, J.A.: *Decision tree induction based on efficient tree restructuring*. Machine Learning, 1997.

YAU Y.J., TEO J. AND ANTHONY P.: *Evolution and Co-Evolution in Cognitive Neural Agents Synthesis for Tic-Tac-Toe*. IEEE Symposium on Computational Intelligence and Games (CIG 2007), pages 304-311, Hawaii, USA, 2007.