Confidence in Decision-Making Through Adaptive Devices

R. S. Okada and J. J. Neto

Abstract— **This paper presents a confidence-oriented model for decision-making devices, capable of acquiring new information at runtime. Typically, machine-learning strategies consist of building an abstraction – such as decision trees - out of a training data before putting it into use, assuming that the data is accurate. Learning at runtime, however, not only requires handling uncertain data, but it also needs a flexible device to accept new information dynamically. In this paper, we introduce an environment for decision-making, which, along with adaptive methods, allows the handling of dynamic learning under uncertainty. It also introduces some of the state-of-art methods for dynamic learning.**

Keywords— **Decision-Making, Adaptive Device, Case-Based Reasoning, Confidence Interval, k-Nearest Neighbor, Instance-Based Learning.**

I. INTRODUÇÃO

Tomada de decisão pode ser compreendido como um processo cognitivo em que uma ação deve ser escolhida e tomada como resposta a um estímulo externo – este, formado por um conjunto de pares atributo-valor que representam um evento [1]. Escolher uma ação, então, torna-se uma questão de comparar as alternativas possíveis e escolher aquela que melhor se adequa ao problema.

Tipicamente, modelos computacionais de tomada de decisão tentam imitar a forma com que humanos realizam esta mesma tarefa. Contudo, sabe-se que este processo não é perfeitamente racional [2], sofrendo influência de fatores emocionais [3] e experiências passadas [4], comumente ignoradas pelos modelos de decisão ou de aprendizado de máquina.

Fatores temporais também se mostram influentes, uma vez que as pessoas, de modo geral, direcionam sua atenção para eventos mais recentes, frequentemente ignorando ou esquecendo aprendizados obtidos em um tempo distante. O foco em eventos recentes nos permite uma rápida adaptação às mudanças comportamentais no processo a ser prognosticado [5] – a este comportamento, dá-se o nome *concept drift*.

Dispositivos de estrutura rígida, contudo, só conseguem alterar sua estrutura reconstruindo-a novamente, ação de difícil realização em tempo de execução. Por não se adaptarem a mudanças externas, suas taxas de acertos se degradam ao longo do tempo, fazendo-se necessário reconstruí-los para evitar perda de desempenho.

Este trabalho visa à criação de um ambiente para tomada de decisão adaptativa, capaz de adquirir novos conhecimentos ao longo de sua execução, sendo fundamentado no raciocínio baseado em casos e em um sistema de realimentações de resultados. Este ambiente deve ser reutilizável, permitindo o uso de diferentes algoritmos de tomada de decisão – desde que tenham comportamento adaptativo.

R. S. Okada, Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, rsuzuki.okada@gmail.com

J. J. Neto, Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, jjneto@gmail.com

II. CONCEITOS

A. Tecnologia Adaptativa:

O termo *adaptatividade* é empregado para descrever dispositivos capazes de modificar seu próprio comportamento como resposta aos dados de entrada, sem a interferência de agentes externos [6]. De forma geral, um dispositivo adaptativo é composto por um dispositivo convencional – como exemplo, autômatos de estados finitos, gramáticas e tabelas de decisão – e uma camada adaptativa, que contém a lógica necessária para alterar a estrutura do dispositivo subjacente [7], conforme ilustra a Figura 1.

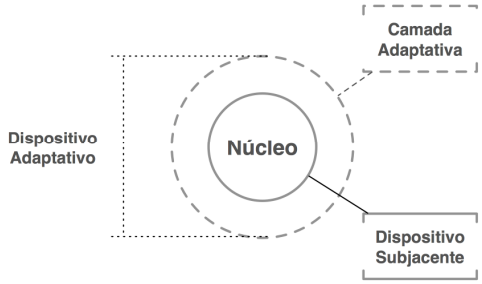

Figura 1. Estrutura de um dispositivo adaptativo.

O primeiro formalismo geral empregando o conceito de adaptatividade para um dispositivo guiado a regras foi introduzido por Neto [8], apesar de que o conceito de estrutura automodificável já houvesse sido explorado em formalismos individuais [9], [10], [11], [12]. Um dos primeiros exemplos notáveis deste tipo de mecanismo pode ser encontrado nos autômatos de estados finitos, em que o uso de funções adaptativas permite a criação e remoção de transições, bem como utilizar novos estados, conforme necessário – fato que faz com que este autômato seja Turing-equivalente [12]. Deve-se ressaltar que esta estrutura flexível permite reutilizar formalismos consolidados e torná-los adaptativos, ao custo de um pequeno acréscimo em sua complexidade [7].

Como exemplo de reuso, Tchemra fez uso desta tecnologia para permitir alterações no conjunto de regras de uma tabela de decisão, permitindo que novas regras fossem criadas tanto ao executar uma regra quanto ao verificar que nenhuma regra pode ser executada – neste caso, invoca um segundo método de decisão para que tome a decisão, cuja solução será

incorporada à tabela através de funções adaptativas [13]. A esta tabela, foi dado o nome de Tabela de Decisão Adaptativa Estendida (*TDAE*). A Figura 2 ilustra seu funcionamento, em que uma tabela de decisão convencional utiliza uma camada adaptativa para chamar funções adaptativas em situações em que não possui uma regra que se adeque ao problema, ou quando executa uma regra adaptativa. O método auxiliar utilizado por Tchemra, no caso, foi o AHP (*Analytic Hierarchy Process*) [13].

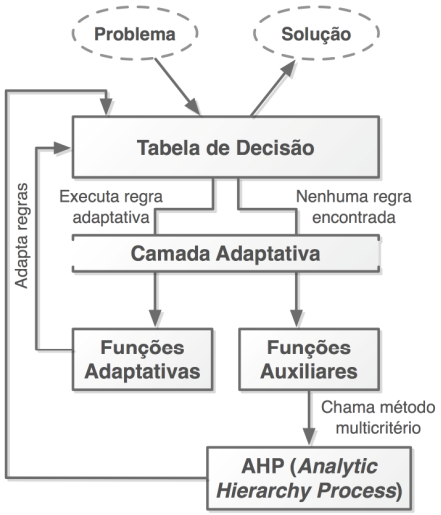

Figura 2. Fluxo de mensagens de uma Tabela de Decisão Adaptativa Estendida (*TDAE*).

Uma importante contribuição da TDAE foi o uso de tecnologia híbrida através da adaptatividade, permitindo que o dispositivo incorpore um novo conhecimento de forma incrementando, utilizando, para tanto, outro dispositivo, mais capaz em termos de potencial de decisão. Esta propriedade é base constituinte do ambiente aqui proposto, conforme está detalhado na seção III.

Outros exemplos de empregos deste tipo de técnica, além da tabela de decisão, podem ser encontrados em casos como processamento de linguagens naturais [14] e composição musical [15]. Ferramentais didáticos também foram desenvolvidos como forma de criar, aplicar e testar técnicas adaptativas, a exemplo do *AdapTools* [16].

B. Sistema de Apoio à Decisão:

Sistema de apoio à decisão (do inglês "*decision support system*") é um artifício computacional utilizado para dar apoio a um tomador de decisão, ajudando-o na tarefa de solucionar problemas [17]. Em geral, é formado por uma base de conhecimento, um modelo de decisão e uma interface gráfica [18], e deve ser treinado antes de ser efetivamente utilizado.

O treinamento de um sistema de suporte à decisão, classicamente, se dá através de um conjunto de amostras de casos conhecidos, fornecendo uma base de conhecimento inicial. O termo "*eager learning*" é dado aos métodos de aprendizados que tentam construir uma abstração durante o treinamento (ex: construção de uma árvore de decisão via ID3), enquanto o termo "*lazy learning*" é atribuído para os métodos que não fazem uma generalização a partir destes dados até que seja necessário durante sua execução [19] (ex: classificador *k-NN*). Em geral, o treinamento deve ocorrer quando se deseja atualizar a base de conhecimento, reconstruindo as abstrações criadas pelos métodos de aprendizado.

O sistema também deve levar em consideração os tipos de critérios que podem ser medidos e recebidos como entrada. Tipicamente, os problemas de classificação e de tomada de decisão utilizam critérios de dois tipos básicos: *nominais* (também chamados de *categóricos* ou *discretos*) e *contínuos*. Critérios nominais são aqueles que possuem um conjunto finitos de valores que podem ser atribuídos (ex: o critério "*tempo*" pode assumir valores tais como "*ensolarado*", "*chuvoso*", "*nublado*", etc.). Já critérios contínuos podem assumir valores numéricos quaisquer (ex: "*temperatura*" pode assumir valores reais acima de 0K – limite inferior conhecido).

Deve-se notar que os nem todos os métodos de aprendizado, classificação e de aprendizado têm capacidade de tratar ambos os tipos de critérios. Como exemplo, árvores de decisão construídas usando-se o algoritmo ID3 [20] tratam apenas atributos nominais, enquanto SVM (*Support-Vector Machines* [21]) opera apenas com valores numéricos. Métodos mais robustos, como C4.5 [22] e K* [23], lidam com ambos nativamente.

Em determinados casos, pode ser desejável converter os critérios de um tipo para outro. Um caso estudado foi o classificador de Bayes [24] o qual, ainda que consiga tratar os dois tipos por construção, obtém resultados melhores quando os atributos contínuos são convertidos para um conjunto finito de valores – passo conhecido como discretização. Uma das técnicas mais simples de discretização é o EWD (*Equal Width Discretization*), que divide o espaço de valores numéricos em múltiplas bandas de igual largura, cada banda correspondendo a um valor discreto [24]. Técnicas mais robustas incluem o RMEP (*Recursive Minimal Entropy Partitioning*) e uso de clusters [25].

O processo contrário, em transformar um critério nominal em contínuo, pode ser feito substituindo os valores nominais por um conjunto de critérios com valores binários mutuamente exclusivos. Cada valor binário representa uma categoria possível do critério nominal a ser convertido [26].

Inicialmente, os modelos de decisão eram imutáveis, sendo construídos uma única vez durante treinamento e mantidos até que um novo treinamento fosse realizado, com base em dados mais recentes. O uso de agentes inteligentes tem ganhado força neste ramo como maneira de aumentar o potencial do sistema [27], mas, em geral, não trata a questão da imutabilidade da abstração. Os conceitos *adaptativo* e *evolucionário* têm sido considerados como sendo os próximos passos do desenvolvimento dos sistemas de tomada de decisão [28].

C. Aprendizado Incremental:

O termo *aprendizado incremental* se refere à capacidade que apresenta um sistema de manter-se atualizado a partir de informações recebidas durante sua execução [29]. Esta

propriedade permite que o sistema consiga não só agregar conhecimentos antes desconhecidos, mas também se adequar a problemas dinâmicos, em que soluções anteriores podem não ser mais válidas para a situação atual [30].

Para métodos de aprendizado do tipo *lazy learning*, a capacidade incremental pode ser obtida com certa facilidade, uma vez que estes métodos não mantêm uma abstração e/ou generalização extraída do treinamento. Em especial, métodos baseados no aprendizado através de instâncias (IBL – *Instance-Based Learning*) são naturalmente incrementais por armazenarem apenas as instâncias treinadas previamente e utilizando-as como parâmetro de comparação (o termo instância se aplica a um conjunto de critérios-valores utilizados como entradas do sistema). Exemplos de métodos com *lazy learning* incluem o classificador de Bayes, *k-Nearest Neighbor* [31] e K* [23].

Contudo, adicionar capacidade incremental em métodos de aprendizado do tipo *eager learning* mostra-se mais complexa, uma vez que isso implica em alterar uma abstração de forma dinâmica – uma tarefa pouco trivial, considerando-se que as alterações dependem da estrutura criada pelo método de aprendizado. Um dos exemplos conhecidos neste tipo de aprendizado é o ITI (*Incremental Tree Inducer*), capaz de manter uma árvore de decisão atualizada de forma incremental, reestruturando-a a cada novo exemplo [32].

Nota-se também que, em conjunto com a capacidade de aprendizado incremental, é desejável obter o efeito contrário, de "esquecer" conhecimento de forma incremental, já que, em determinados problemas, soluções podem ser válidas apenas por um período de tempo, fazendo-se necessário um mecanismo para descartá-las após seu prazo de validade [5].

A capacidade de aprendizado incremental é utilizada como base para o modelo dos dispositivos a serem utilizados pelo sistema proposto. A adaptatividade ocorre na forma de inserção e remoção de conhecimento, cuja descrição é feita na seção III.

D. Raciocínio Baseado em Casos:

Quando confrontados com um novo problema, uma das técnicas frequentemente adotadas por humanos para solucioná-lo é reutilizar soluções de problemas passados que tinham semelhanças com o problema atual [33]. A este método de raciocínio, em que uma solução é adotada através de casos previamente enfrentados ao invés de derivar uma nova abstração, é dado o nome de Raciocínio Baseado em Casos (do inglês "*Case-Based Reasoning*", ou então, CBR).

O CBR é um método que se executa, tipicamente, em quatro etapas [34], conforme ilustra a Figura 3:

- ***Retrieve***: busca por casos cujo problema é similar ao caso atual;
- ***Reuse***: adaptação das soluções dos casos encontrados, tornando-os compatíveis com o novo problema;
- ***Revise***: aplicação da solução, verificando se os resultados foram aceitáveis;
- ***Retain***: memorização da nova solução na base de conhecimento, disponibilizando-a como um caso conhecido para futuros problemas.

Note que este método de raciocínio possui um modelo de aprendizado similar ao IBL, pois não mantém uma generalização a partir dos dados conhecidos. Isso permite utilizar um aprendizado incremental neste tipo de raciocínio, cuja aquisição de conhecimento ocorre após confirmar que a solução adotada estava correta.

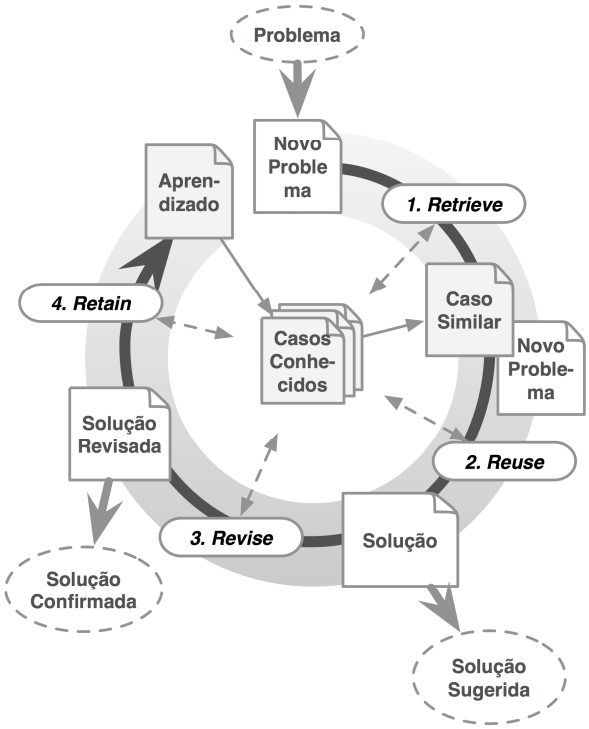

Figura 3. Metodologia adotada pelo raciocínio baseado em casos, conforme descrito por Aamodt [34].

Dois problemas surgem a partir deste raciocínio. Primeiramente, deve-se dispor de uma métrica para calcular a semelhança entre dois casos, que depende do problema a ser resolvido. Em seguida, deve-se notar que, em grande parte dos casos, não é possível assegurar que uma solução adotada está, de fato, correta, devido à possível latência temporal e espacial dos efeitos causados pela solução [35], ou pela falta de realimentação do usuário. Assim, o aprendizado incremental, diferentemente de um treinamento realizado por especialistas, deve lidar com um nível de incerteza muito maior. Este aspecto será tratado na seção seguinte.

E. Tratamento de Incertezas e Confiança de uma Decisão:

Um dos aspectos explorados neste trabalho é a capacidade do sistema decisório de estimar um nível de confiança para suas respostas. Confiar em uma solução significa acreditar que ela é, de fato, correta.

De maneira geral, quanto mais uma ação se repete, mais facilmente as pessoas acreditam que a mesma seja uma verdade. Crença também é fortemente influenciada por realimentações externas: opiniões que reforçam a ideia original aumenta a confiança, enquanto realimentações negativas

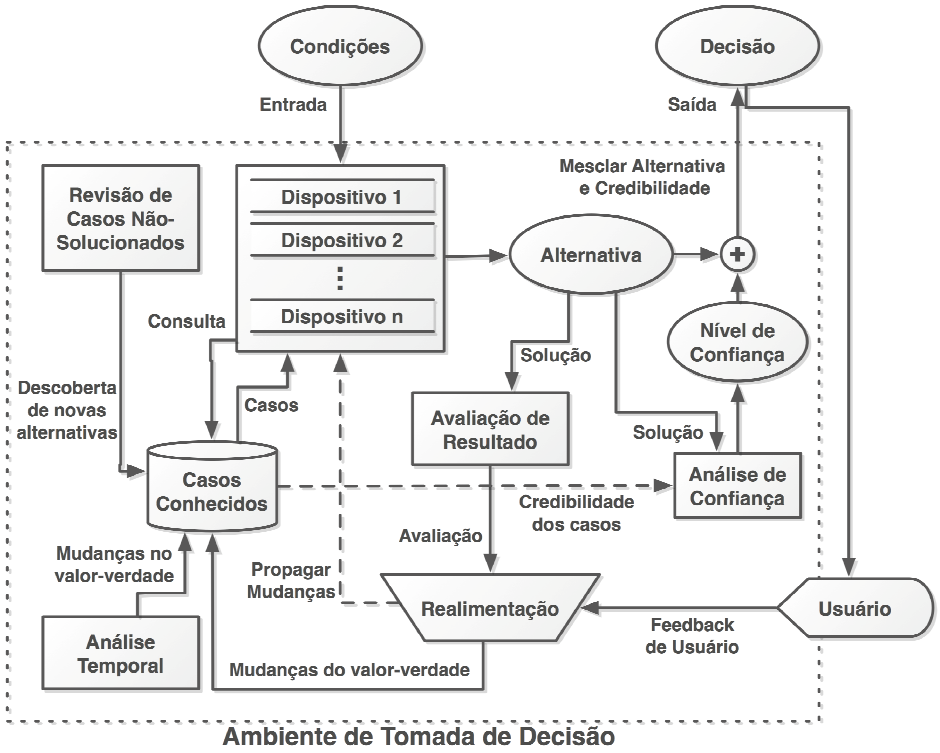

Figura 4: Visão Geral do Ambiente Adaptativo de Tomada de Decisão

alertam sobre a possibilidade de um erro [36].

Modelos que estimem a confiança baseada nas realimentações recebidas ainda não estão consolidados. Einhorn [37] sugere que a confiança C seja uma função baseado na soma F das realimentações:

$$C = f(F) \quad (1)$$

$$F = \beta_+ N_+ + \beta_- N_- \quad (2)$$

$$\beta_+ + \beta_- = 1.0, (\beta_+, \beta_- \geq 0.0) \quad (3)$$

Onde N_+ e N_- são as somas de realimentações positivas e negativas, respectivamente. Já β_+ e β_- são valores relativos de reforço das alimentações positivas e negativas. Note que a função f foi deixada em aberto em sua concepção original, mas espera-se um baixo valor para $F \sim 0$, e deve aumentar com o módulo de F. Note que se $\beta_+ N_+ > \beta_- N_-$, temos maior confiança que o resultado está correto, enquanto $\beta_+ N_+ < \beta_- N_-$ aumenta a confiança de o resultado estar incorreto.

Uma consideração deve ser feita com relação às somas das realimentações: imaginando-se que o treinamento – do ponto de vista das fórmulas acima – é uma espécie de "realimentação", podem-se utilizar valores iniciais elevados para N_+ ou N_-, pois se imagina que o treinamento tem grande certeza de que a resposta está correta ou não. Contudo, para novas soluções, devem-se utilizar valores menores, para que a confiança inicial também seja baixa. Os incrementos realizados pelo aprendizado também devem ser ponderados de acordo com a credibilidade do mesmo: como exemplo, realimentações vindas de usuário ou especialista são mais confiáveis que uma realimentação obtida por uma mera repetição de ações. O uso de realimentações já pode ser observado em exemplos individuais de sistemas de busca, como o CBIR (*content-based image retrieval*) [38].

Nota-se que a confiança e a veracidade de uma solução, apesar de serem utilizadas em conjunto, não necessariamente são correlacionadas. Nada impede que o tomador de decisão, baseado em evidências incorretas, confie que sua solução seja correta, ainda que seja inverídica.

III. Ambiente Adaptativo de Tomada de Decisão

A. Definições:

Primeiramente, para fins deste projeto, o termo *instância* será utilizado como sendo o conjunto de valores de entrada do sistema – um para cada critério – de acordo com o termo utilizado pelo IBL. Cada instância k é representada, internamente, como sendo um vetor x_k de m dimensões, onde o i-ésimo representa o valor associado ao i-ésimo critério.

$$x_k = (x_{k1}, x_{k2}, ..., x_{ki}, ..., x_{km}) \quad (4)$$

Para critérios *contínuos*, seus valores devem ser números reais, enquanto para critérios *nominais*, serão utilizados nomes – referenciados como *categorias*. O número de categorias possíveis para cada critério deve ser finito.

Em contrapartida, o termo *caso* será usado como um par instância-alternativa, representando o problema e uma possível solução, indicada pela alternativa. Considerando que a alternativa de um caso j seja representada por y_j, temos:

$$caso_j = (x_j, y_j) \quad (5)$$

O termo *dispositivo* será utilizado para designar os métodos de tomada de decisão a serem utilizados pelo sistema,

devendo, assim, ser adaptativos. Sua estrutura interna está descrita na seção C.

Por fim, *realimentação* será compreendida como uma reação à solução encontrada pelo sistema, capaz de alterar seu comportamento para futuras novas soluções, podendo ser gerada pelo usuário e/ou por um mecanismo inteligente. Sua descrição encontra-se na seção D.

B. Visão Geral:

A partir do modelo utilizado pela TDAE, em que um segundo dispositivo é utilizado para auxiliar o aprendizado do dispositivo adaptativo [13], pode-se expandi-lo para que utilize um número arbitrário de dispositivos, dispostos em uma estrutura de fila. Este modelo é ilustrado pela Figura 5. Sua ideia intuitiva é semelhante da TDAE: ao receber um problema, o sistema o envia para o primeiro dispositivo da fila. Caso consiga solucioná-lo de forma confiável, retorna sua decisão; caso contrário, o problema é repassado ao próximo da fila. Este procedimento deve se repetir até que um dos dispositivos consiga solucionar o problema, ou então, até esgotar a fila. O funcionamento interno de um dispositivo está descrito na subseção C.

Para cada alternativa gerada pelos dispositivos, deve-se estimar um nível de confiança para a decisão tomada, utilizando, como base, as realimentações recebidas pelo sistema, conforme introduzido na seção II. E. Para tanto, também se faz necessário utilizar uma base de conhecimento que armazene as realimentações recebidas para cada caso conhecido, permitindo calcular, de alguma forma, a equação (1). A saída do sistema, inclusive, é composta pela união entre a alternativa e as informações de confiança.

A realimentação armazenada na base pode ser originada no próprio sistema – através de um bloco de avaliação de decisão – ou então, fornecido pelo próprio usuário. O bloco de realimentação deve receber estas informações e enviá-las à base, bem como propagar as mudanças para os dispositivos, permitindo que alterem sua estrutura interna. Como exemplo, a TDAE original recebia esta informação propagada do AHP, o que resultava na geração de novas regras [13].

Junto à base de dados, deve haver um bloco de análise temporal, cuja finalidade é verificar se a validade de algum caso conhecido expirou – isto é, deixou de valer após um período de tempo – permitindo tratar conceitos quem variem no tempo. Também deve estar presente um bloco de revisão, que verifica a possibilidade de existirem novas alternativas, antes desconhecidas. Esta possibilidade, em princípio, ocorre ao detectar que há um grande número de casos semelhantes em que nenhuma das soluções é considerada válida.

C. Dispositivo Adaptativo de Tomada de Decisão:

Internamente, cada dispositivo adaptativo de tomada de decisão, a exemplo do modelo original ilustrado pela Figura 1, deve conter um dispositivo subjacente, capaz de solucionar o problema de decisão vindo do dispositivo anterior, confirme ilustra a Figura 5. Isso deve ser feito calculando-se um *índice de mérito* relacionado com a probabilidade de cada alternativa ser solução do problema, de acordo com o método de decisão utilizado.

A partir do índice calculado, o dispositivo deve analisar se a solução obtida pode ser considerada "adequada", verificando se a solução de melhor nota é um bom candidato para solucionar o problema. Uma técnica simples para realizar tal operação é utilizar um limiar: se a melhor nota estiver acima do limiar, retorna sua solução como sendo resposta ao sistema. Caso contrário, repassa o problema ao próximo dispositivo. A condição de saída da fila está detalhada na subseção G.

A camada adaptativa deve permitir que realimentações – geradas a partir das soluções encontradas pelos dispositivos – alterem seu comportamento. Detalhes sobre a realimentação se encontram nas seções D. e E.

Deve-se notar que o dispositivo subjacente pode ter acesso à base de conhecimento para realizar consultas, uma vez que métodos de aprendizado baseados em instâncias dependem do armazenamento dos casos conhecidos. Tal estrutura permite utilizar uma única base, mesmo que seja utilizado mais de um método baseado em casos.

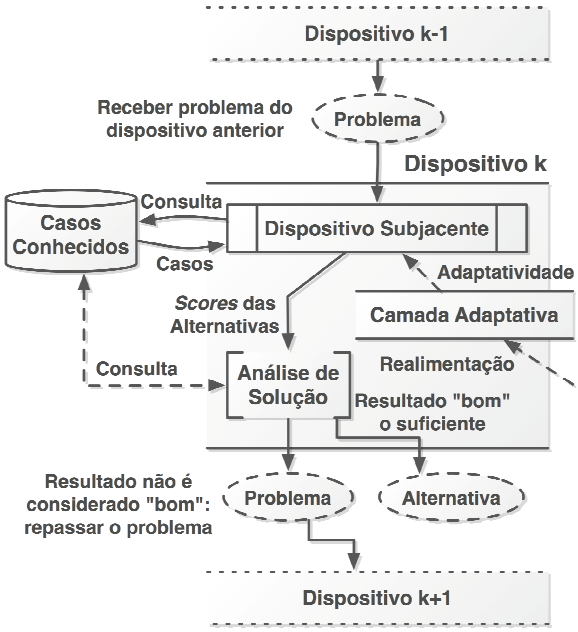

Figura 5: Modelo de um dispositivo adaptativo de tomada de decisão.

D. Realimentação:

Para este projeto, realimentação é uma relação que avalia um par instância-alternativa, informando se a alternativa é ou não uma solução para a instância do problema, de acordo com o analisador que a gerou. Uma instância k de realimentação é representada por uma tripla (v_k, β_k, c_k), tal que:

- $\boldsymbol{v_k}$ = valor verdade (0.0 ou 1.0) da dupla instância-alternativa avaliada por esta realimentação – valor 1.0 diz que a alternativa é solução, enquanto 0.0 diz o contrário;
- $\boldsymbol{\beta_k}$ = valor relativo de reforço, que depende se a realimentação é positiva (valor verdade 1.0) ou negativa (valor verdade 0.0) – ver equações (2) e (3);
- $\boldsymbol{c_k}$ = credibilidade do analisador que gerou esta

realimentação – deve ser um valor positivo maior que zero.

A realimentação, gerada após a tomada de uma decisão, pode ter origem em três tipos de fontes:

- **Usuários do sistema**: informam se as soluções foram o que esperavam inicialmente;
- **Analisadores Computacionais:** blocos dotados de "inteligência" (IA), capazes de verificar automaticamente a qualidade da resposta – representado pela "Avaliação de Resultado" da Figura 4;
- ***Repetition-bias*:** o sistema, naturalmente, envia realimentações com baixo valor de c_k ao repetir a execução de uma solução, acreditando que está correta (pois o sistema espera que, se estiver errado, alguém lhe dirá através de uma realimentação negativa);

Nota 1: apesar de não ser realimentação, o treinamento do sistema também pode ser compreendido como sendo uma tripla (v_k, β_k, c_k), exceto que a credibilidade c_k assume valores muito maiores. Para este projeto, ainda que seja conceitualmente diferente, o treinamento também utilizará esta notação.

Nota 2: o bloco de avaliação de resultado não possui uma implementação fixa, e deve ser especificamente desenvolvida para cada sistema em particular. A presença deste bloco no sistema é opcional (quando não estiver presente, não haverá realimentação através de IA).

A junção da credibilidade e do valor de reforço resulta no peso que uma realimentação terá frente às demais:

$$w_k = \beta_k \cdot c_k \tag{6}$$

Assim, de acordo com as definições dadas por Einhorn [37], deve-se estabelecer uma função que estime a confiança de uma solução a partir da soma das realimentações recebidas – no caso, utilizamos uma soma ponderada por w_k – o que se encontra descrito na seção seguinte.

E. Cálculo do Valor-Verdade e Intervalo de Confiança:

Um meio para calcular o valor-verdade de uma alternativa para uma instância, então, pode ser dado pela média ponderada dos valores verdades de todas as *n* realimentações recebidas que avaliaram este par instância-alternativa:

$$\bar{v} = \frac{V}{W}, \left(V = \sum_{k=1}^{n} v_k \cdot w_k, W = \sum_{k=1}^{n} w_k \right) \tag{7}$$

O valor *V* pode ser interpretado como número de vezes que a alternativa foi considerada correta dentre as *W* vezes que o par foi avaliado.

A partir destes valores, também se calcula um intervalo de confiança para este valor-verdade, fornecendo, assim, a medida de confiança comentada anteriormente.

Tipicamente, o cálculo do intervalo de confiança é realizado encontrando dois limites, v_{low} e, v_{upp} tais que a probabilidade de a verdadeira média μ estar neste intervalo seja $1-\alpha$. Assim:

$$P\left(v_{low} \le \mu \le v_{upp}\right) = 1 - \alpha \tag{8}$$

Como o valor-verdade, de acordo com o que fora definido anteriormente, só assume valores binários – verdadeiro (1.0) ou falso (0.0) – tem-se uma distribuição de Bernoulli, cujo intervalo de confiança, quando é aproximado por uma distribuição normal, é:

$$\bar{v} \pm z_{1-\alpha/2} \sqrt{\frac{\bar{v}(1-\bar{v})}{W}} \tag{9}$$

tal que $Z_{1-\alpha/2}$ é o $(100\times(1-\alpha/2))$-ésimo percentil de uma distribuição normal. Seu cálculo é feito pela inversa da função distribuição acumulada da distribuição normal – comumente referenciada como *probit function*. Nota-se que não existe uma forma fechada para esta função, devendo ser estimada por aproximações [39].

Este intervalo, contudo, pode não representar bem situações em que o número de amostras (no caso, o número de realimentações recebidas) é baixo, ou então, para situações em que a verdadeira média é muito próxima de 0 ou 1 [40].

Como exemplo, ao lançar uma moeda não-viciada três vezes, a possibilidade de sair três caras é factível. Caso fosse calculada, a partir desta amostra, a massa de probabilidade de um lançamento dar "cara", chega-se à conclusão que a probabilidade é 1.0 com intervalo zero, o que não representa a real situação.

Uma alternativa viável neste caso é utilizar outros métodos de cálculo de intervalo de confiança. Um exemplo conhecido é o método de *Agresti-Coull*, que insere, artificialmente, uma quantidade igual de amostras de valor 0 ou 1, movendo a média para mais próximo de 0.5, evitando assim que o intervalo seja zero. O efeito das amostras artificiais é reduzido conforme o número de amostras reais aumenta, fazendo com que a média de *Agresti-Coull* convirja para a média real.

Através deste método, a nova média e intervalos são calculados conforme as equações (10), (11) e (12):

$$\hat{W} = W + \left(z_{1-\alpha/2}\right)^2 \tag{10}$$

$$\hat{v} = \frac{V + z_{1-\alpha/2}^2 / 2}{\hat{W}} \tag{11}$$

$$\hat{v} \pm c_{1-\alpha/2} = \hat{v} \pm z_{1-\alpha/2} \sqrt{\frac{\hat{v}(1-\hat{v})}{\hat{W}}} \tag{12}$$

Nota-se que neste método, o número de amostras artificiais inseridas é $(Z_{1-\alpha/2})^2$, novamente, relativo ao percentil utilizado anteriormente. O efeito das amostras pode ser visualizado pelo gráfico da Figura 6. Assim, este sistema utilizará, como valor-verdade de um par instância-alternativa, a média desviada de *Agresti-Coull*. O intervalo de confiança será $C_{1-\alpha/2}$.

Nota 1: o método de Agresti-Coull é válido mesmo quando o número de amostras for zero, resultando uma média 0.5 e intervalo também igual a 0.5, englobando todo o espaço de valores possível.

Nota 2: o cálculo do valor-verdade e do intervalo de confiança pode ser realizado de forma incremental, armazenando apenas as somatórias W e V, incrementando-as conforme novas realimentações cheguem e aplicando as equações (11) *e* (12).

A cada recebimento de realimentação, a base de conhecimento deve notificar os dispositivos sobre a mudança

no valor-verdade para o caso analisado, permitindo que os mesmo se adaptem ao novo conhecimento.

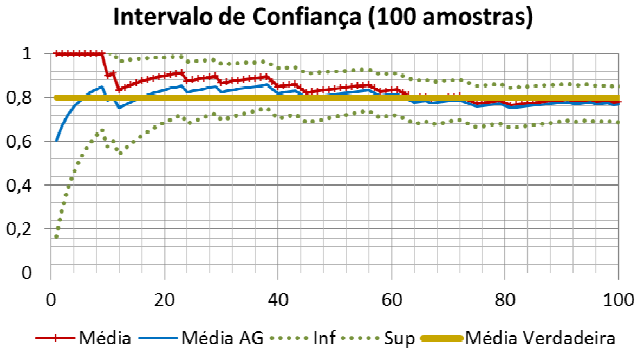

Figura 6: Gráfico da média de um valor que segue a distribuição de Bernoulli conforme o número de amostras aumenta. A média real é dada pela linha amarela (p=0.8). Nota-se que a média e o intervalo de Agresti-Coull (dado pelas linhas pontilhadas) se mantêm mais próximos da média real do que a média aritmética quando o número de amostras é baixo, mas convergem para o verdadeiro valor de p conforme o número de amostras aumenta.

F. *Base de Conhecimento:*

Conforme descrito pelo item anterior, é necessário armazenar os valores de V e W para cada caso possível para permitir estimar seu valor verdade. Como um caso é definido pela dupla instância-alternativa, pode-se utilizar a informação espacial da instância para rapidamente encontrar casos conhecidos, utilizando uma estrutura eficiente, como o *Locally Sensitive Hashing*, ou então, se todos os critérios forem numéricos, uma estrutura como o *R*-Tree* [41].

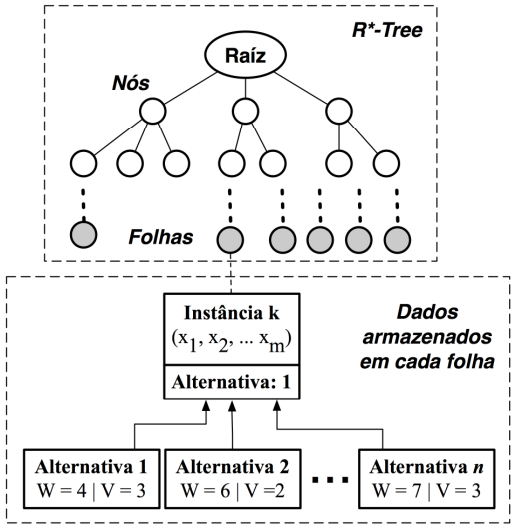

Figura 7: Exemplo de uma base utilizando uma *R*-Tree*. Em cada folha, deve estar armazenada a instância – utilizada como informação espacial, e todos os casos possíveis que utilizam tal instância – no caso, todas as alternativas possíveis, cada uma com seu valor W e V.

O uso destas estruturas se justifica para que consultas à base sejam eficientes, principalmente quanto for necessário encontrar os vizinhos mais próximos – uma necessidade para qualquer método de aprendizado baseado em instâncias. Nota-se que é possível armazenar estes dados em uma simples tabela, apesar de ser computacionalmente mais custoso.

Uma consideração, contudo, deve ser feita: critérios numéricos, ao contrário dos critérios nominais, podem assumir valores teoricamente infinitos. Para evitar um possível crescimento infinito da base de conhecimento, critérios numéricos devem ser discretizados, conforme as técnicas citadas na seção II. B.

G. *Condição de Saída da Fila de Dispositivos:*

A descrição do dispositivo utilizado pelo ambiente de decisão especifica a necessidade de decidir se a solução encontrada por um dispositivo é adequada, repassando o problema ao próximo em caso negativo. Para isso, o dispositivo deve calcular um *score* para cada alternativa.

Uma possível maneira de se decidir a parada é utilizar um *limiar de aceitação*, tal que a solução é aceita somente se o melhor *score* obtido está acima do limiar. Para isso, pode-se normalizar este escore para que a soma de todas as notas seja 1.0, o que permite escolher um limiar no intervalo [*0...1*]. A este índice de mérito normalizado, define-se o termo *probabilidade de classes* (baseado no termo utilizado pelo classificador de Bayes - *class probability*).

Supondo que p_i seja a probabilidade de classe da alternativa i, temos a seguinte lógica booleana para decidir a saída da fila:

$$aceita = \left(\max\left(p_1, p_2, ..., p_n\right) > thresh\right) \quad (13)$$

tal que *thresh* é o limiar escolhido (do inglês, *threshold*). Este método, porém, não considera a possibilidade de haver duas alternativas com probabilidades elevadas, o que poderia ocasionar ambiguidade. Para esta condição, poderia ser utilizado um segundo critério, em que se verifica a razão entre a segunda maior probabilidade e a primeira. Este valor deve estar abaixo de um limiar para que a solução seja considerada livre de ambiguidades. Assim:

$$ambig. = \left(\frac{\max_2\left(p_1, p_2, ..., p_n\right)}{\max\left(p_1, p_2, ..., p_n\right)} > thresh2\right) \quad (14)$$

$$aceita = \left(\max\left(p_1, p_2, ..., p_n\right) > thresh\right) \wedge \neg ambig. \quad (15)$$

tal que max_2 é uma função que calcula o segundo maior valor da lista e *thresh2* é o limiar de ambiguidade, devendo ser um valor menor que 1.0.

Nota: caso o dispositivo, por construção, não consiga resolver o problema, deve retornar probabilidade zero para todas as alternativas. Ainda que não cumpra com o quesito da normalização, não deve passar pela condição de saída.

H. *Treinamento:*

Inicialmente, deve-se treinar o sistema com um conjunto inicial de casos, informando-lhe quais são as alternativas corretas para um conjunto de instâncias conhecidas. A base de conhecimento deve armazenar, para cada instância, $V=W=w_0$ para a alternativa correta e $V=0$ para todas as demais. O valor w_0 representa um peso inicial para os casos de treinamento.

Nota: $w_0 = \infty$ faz o valor-verdade subir para 1.0 para a alternativa correta, e impede que este fato seja alterado por realimentações futuras.

Também deve ser definida a credibilidade c_k das realimentações de repetição – enviadas após cada decisão, reforçando a alternativa escolhida com $(v_k=1,\ \beta_k = \beta_+,\ c_k)$ e

negativando todas as demais com $(v_k=0, \beta_k = \beta_-, c_k)$. Para este trabalho, utiliza-se $c_k=0.01$. Nota-se que um valor muito elevado faz com que o sistema ganhe confiança rapidamente, diminuindo o impacto das realimentações de origem mais confiável.

IV. Dispositivos Adaptativos com Aprendizado Incremental

Para este projeto, foram desenvolvidos três dispositivos adaptativos, baseados em métodos desenvolvidos previamente, e que, por natureza, mostravam um bom potencial para adaptatividade: TDAE, k-Nearest Neighbor e classificador de Bayes. Cada um deles será descrito a seguir.

A. TDAE:

Como está descrito na seção II. , a tabela de decisão adaptativa estendida foi a base conceitual deste projeto, e é, também, um dos dispositivos com implementação disponível para uso pelo ambiente. Porém, diferentemente da formulação original, a tabela aqui utilizada faz uso de entradas estendidas (isto é, entradas podem assumir valores discretos arbitrários em lugar de valores binários). Os critérios numéricos, a exemplo da base de dados, também devem ser discretizados para serem inseridos na tabela.

A tabela de decisão, por definição, consegue tratar apenas instâncias já recebidas anteriormente, pois somente estas terão uma regra equivalente na tabela. Para estes casos, a tabela retorna probabilidade 1.0 para a alternativa prevista pela regra encontrada, e 0.0 para as demais.

Ao receber uma notificação de realimentação através de sua camada adaptativa, a TDAE pode prosseguir de duas maneiras:

- Se a instância avaliada já era conhecida pela base de dados, a TDAE deve verificar se a melhor alternativa mudou. Em caso positivo, deve alterar sua regra correspondente na tabela;
- Se a instância é nova, uma nova regra deve ser criada.

Nota-se que a tabela de decisão adaptativa permite executar uma adaptatividade ao executar uma regra. Neste projeto, não será utilizada a adaptatividade desta maneira, apesar de ser teoricamente possível.

B. k-Nearest Neighbor:

Um dos métodos de aprendizado baseado em instâncias utilizados foi uma extensão do popular *k-Nearest Neighbor* (k-NN). Originalmente, o classificador k-NN procura pelos k vizinhos mais próximos a uma nova instância t, cada um com uma solução y_i já conhecida. A solução da nova instância, então, é definida por [42]:

$$y_t = \underset{c\in\{c_1\ldots c_n\}}{\operatorname{argmax}} \sum_{i=1}^{k} I(y_i = c) \tag{16}$$

sendo que $I(x)$ é a função indicadora, retornando 1 se a alternativa correta da instância é c. Assim, a alternativa da nova instância é aquela que ocorre com maior frequência entre os vizinhos mais próximos. Esta função, porém, não considera a grau de semelhança entre os casos: instâncias mais próximas deveriam receber peso maior por, supostamente, terem maior influência na resposta final. Assim:

$$y_t = \underset{c\in\{c_1\ldots c_n\}}{\operatorname{argmax}} \sum_{i=1}^{k} I(y_i = c)\cdot w(x_t, x_i) \tag{17}$$

tal que $w(x_t,x_i)$ é uma função que pondera a instância x_i pela sua distância. Dentre os métodos utilizados, estão o método de Shepard [43] e a função de similaridade [42], dados pelas equações (18) e (19):

$$w(x_t, x_i) = \frac{1}{d(x_t, x_i)^p} \tag{18}$$

$$w(x_t, x_i) = 1 - \frac{d(x_t, x_i)}{d_{\max}} \tag{19}$$

onde $d(x_t,x_i)$ é uma função que calcula a distância entre duas instâncias, enquanto d_{max} é um fator de normalização para que a similaridade esteja no intervalo *[0...1]*.

Esta definição exige, contudo, que cada instância tenha uma única alternativa conhecida com 100% de confiança, explicitado pela função indicadora. No caso do ambiente adaptativo, as instâncias possuem valores-verdade ao invés de uma única verdade absoluta. Para adequar o k-NN a este problema, utiliza-se a equação (20):

$$y_t = \underset{c\in\{c_1\ldots c_n\}}{\operatorname{argmax}} \sum_{i=1}^{k} \hat{v}(x_i, c)\cdot w(x_t, x_i) \tag{20}$$

Esta nova definição substitui a função indicadora pelo valor-verdade de cada alternativa, conforme dado pela equação (11).

Já para a função de distância, classicamente, utiliza-se a distância Euclidiana. Para um caso com m critérios, temos:

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{m} \left| x_{1j} - x_{2j} \right|^2} \tag{21}$$

Contudo, esta função é somente válida em casos em que todos os critérios são numéricos. E ainda assim, como os valores de cada critério podem assumir ordens de grandeza diferentes, a influência de cada critério sobre o valor da distância acaba sendo desigual. Para mitigar este problema, normaliza-se a distância para que cada critério contribua com a distância de forma igualitária:

$$d(x_1, x_2) = \sqrt{\frac{\sum_{j=1}^{m} d_j(x_1, x_2)^2}{m}} \tag{22}$$

$$d_j(x_1, x_2) = \left| x_{1j} - x_{2j} \right| \Big/ d_{j\max} \tag{23}$$

onde $d_j(x_1,x_2)$ calcula a distância normalizada entre as instâncias x_1 e x_2 no critério j, enquanto $d_{j\ max}$ é a máxima distância possível neste critério.

Estas equações, contudo, continuam válidas apenas para critérios contínuos. Para tratar critérios nominais, pode-se estabelecer a seguinte equação:

$$d_j(x_1, x_2) = \delta(x_{1j}, x_{2j}) \tag{24}$$

A função $\delta(x_{1j}, x_{2j})$ retorna 0 se, e somente se, os valores x_{1j} e x_{2j} forem idênticos, retornando 1 caso contrário. Esta métrica é comumente definida como *Overlap Metric* [44]. Outra opção é utilizar uma métrica que considere as probabilidades de uma

alternativa aparecer com cada um dos valores da instância:

$$d_j(x_1, x_2) = \sum_{c=1}^{C} \left| P(c \mid x_{1j}) - P(c \mid x_{2j}) \right| \quad (25)$$

onde $P(c \mid x_{jj})$ é a probabilidade de a alternativa da instância x_i ser c, dado que seu valor para o j-ésimo critério é x_{ij}. Este valor pode ser obtido conhecendo-se a frequência com que cada alternativa aparece quando uma instância aparece com o valor x_j. Esta métrica recebe o nome de *Value Difference Metric* [44].

Note-se que o k-NN não requer manter uma abstração sobre as instâncias conhecidas, somente dependendo da base de dados.

C. Classificador de Bayes:

O classificador de Bayes é um método probabilístico de classificação que assume que os valores necessários dos critérios para definir uma alternativa são independentes entre si. Pela teoria de Bayes, a probabilidade de uma instância x_a ter a alternativa c como resposta é dada pela equação (26):

$$p(c \mid x_a) = \frac{1}{Z(x_a)} p(c) \prod_{i=1}^{m} p(x_{ai} \mid c) \quad (26)$$

onde $p(c)$ é a probabilidade a priori de uma instância x qualquer ter c como alternativa correta, enquanto $p(x_{ai}|c)$ é a probabilidade de o valor x_{ai} aparecer no i-ésimo critério quando uma instância tiver a alternativa c como solução. A princípio, Z é um fator de escala que denota a *evidência*, normalizando as probabilidades e fazendo com que sua soma seja 1.0:

$$Z(x_a) = \sum_{j=1}^{n} \left(p(c_j) \prod_{i=1}^{m} p(x_{ai} \mid c_j) \right) \quad (27)$$

Contudo, como Z é um valor comum a todas as probabilidades, chega-se à seguinte conclusão:

$$p(c \mid x_a) \propto p(c) \prod_{i=1}^{m} p(x_{ai} \mid c) \quad (28)$$

Supondo que a alternativa de uma instância seja aquela cuja probabilidade calculada seja a maior, temos:

$$y(x_a) = \underset{c \in (c_1, \ldots c_n)}{\operatorname{argmax}} \; p(c) \prod_{i=1}^{m} p(x_{ai} \mid c) \quad (29)$$

Para este trabalho, o valor de $p(c)$ é calculado através da soma dos valores-verdade de cada instância x para a alternativa c, e dividindo-o pelo total de valores verdades. Supondo que há s instâncias salvas na base, temos:

$$p(c) = \frac{\sum_{i=1}^{s} v(x_i, c)}{\sum_{j=1}^{n} \sum_{i=1}^{s} v(x_i, c_j)} \quad (30)$$

De forma similar, cada elemento $p(x_{ai}|c)$ pode ser calculado somando o número de vezes com que o valor x_i aparece quando uma instância tem a alternativa c como solução.

$$p(x_{ai} \mid c) = \frac{\sum_{j=1}^{s} u(x_j, x_a, i, c)}{\sum_{j=1}^{s} v(x_j, c)} \quad (31)$$

$$u(x_j, x_a, i, c) = v(x_j, c) \cdot I(x_{ji} = x_{ai}) \quad (32)$$

Alterações nos valores-verdade e no número de instâncias – informadas pela adaptatividade – devem alterar as probabilidades definidas acima. A atualização pode ser feita de forma eficiente se for armazenado o valor das três somatórias definidas nas equações (31) e (32), e apenas incrementá-las/decrementá-las conforme os valores-verdade utilizados por elas se alterem, utilizando a propriedade:

$$S_{t+1} = S_t - s_{ilt} + s_{slt+1} \quad (33)$$

onde S_t é uma somatória qualquer em um instante t, enquanto $s_{t \mid t}$ é o valor de um elemento da somatória em um instante t. Se este valor, em um instante $t+1$, for alterado, basta substituir o valor anterior e somar o novo.

Utilizando esta propriedade, tais somatórias acima podem ser atualizadas incrementalmente. Como exemplo, supondo que o valor de $v(x_b.c)_t$ seja o valor-verdade da alternativa c para a instância x_b em um instante t. Caso este valor seja alterado em um instante $t+1$:

$$p(c)_{t+1} = \frac{\left(\sum_{i=1}^{s} v(x_i, c) \right)_t - v(x_b, c)_t + v(x_b, c)_{t+1}}{\left(\sum_{j=1}^{n} \sum_{i=1}^{s} v(x_i, c_j) \right)_t - v(x_b, c)_t + v(x_b, c)_{t+1}} \quad (34)$$

O cálculo de $p(x_{ai}|c)_{t+1}$ pode ser feito de forma análoga, aplicando o mesmo conceito sobre $u(x_j,x_a,i,c)$ nos instantes t e $t+1$.

V. Resultados

A. Aprendizado Incremental:

Para verificar o funcionamento do aprendizado incremental, foi criado um sistema experimental, que utiliza dois dispositivos: TDAE, seguido por um k-NN, com $k=10$ e utilizando ponderação pelo método de Shepard, com $p=2$. Para calcular distâncias entre atributos nominais, utiliza-se o *Overlap Metric*, definido na equação 24. Para ponderar as realimentações positivas e negativas, é utilizado $\beta_+=\beta_-=0.5$. A discretização de valores numéricos foi realizada através do método *Equal Width Discretization*, descrito em [24], utilizando-se 100 valores discretos de igual comprimento.

Para testá-lo, foram utilizados *datasets* fornecidos pelo repositório de aprendizado de máquina da UCI [45]. Para cada conjunto de testes, 20% dos dados foram sorteados aleatoriamente para serem utilizados como treinamento inicial do sistema, utilizando $w_0 = \infty$. Após o treinamento, o sistema foi executado 100 vezes consecutivas, selecionando aleatoriamente 80% dos dados como entradas – não necessariamente o complemento dos 20% de treinamento. Em caso de o sistema retornar uma alternativa não esperada, há 2% de chance de o sistema receber uma realimentação $(v_k=0,\beta_k=\beta_-,c_k=1)$ para a alternativa retornada, e outra $(v_k=1,\beta_k=\beta_+,c_k=1)$ para a alternativa esperada.

A Tabela I mostra os resultados obtidos, ilustrando o aumento na taxa de acertos obtido para cada conjunto de dados.

TABELA I
EVOLUÇÃO DE APRENDIZADO APÓS 100 EXECUÇÕES CONSECUTIVAS

DATASET	TAXA DE ACERTOS INICIAL	TAXA DE ACERTOS FINAL
ABALONE	37.29%	87.13%
CAR EVALUATION	84.66%	96.96%
CREDIT	85.51%	96.92%
HABERMAN	72.95%	94.67%
IRIS	96.67%	99.17%
NURSERY	91.49%	98.37%
SHUTTLE	99.79%	99.97%
STATLOG	75.75%	94.50%
TIC-TAC-TOE	85.51%	97.52%
WINE	92.96%	99.30%
YEAST	61.92%	92.25%

Analisando os resultados individualmente, verifica-se que a maior parte dos erros ocorre com alternativas menos frequentes, e que, por consequência, possuem menos amostras no conjunto inicial de treinamento. Como exemplo, no *Car Evaluation*, a alternativa "*unacc*" domina as demais com uma frequência de 70%, e raramente é confundida com as demais, obtendo taxa de acertos superior a 99% logo após o treinamento. As demais alternativas, menos numerosas, possuem acertos iniciais inferiores, mas evoluem conforme recebem realimentações, conforme ilustra a Figura 8.

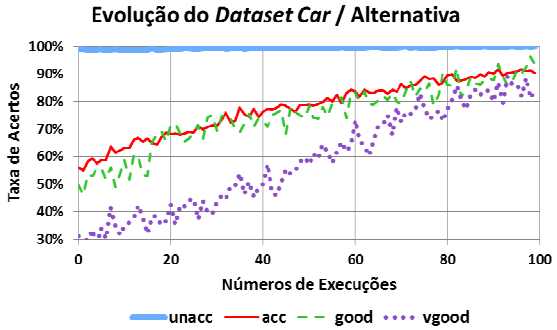

Figura 8: Evolução do *dataset Car Evaluation* para cada alternativa.

B. *Compromisso de Taxa de Acertos e Tempo de Resposta:*

Um aspecto analisado por este trabalho é o efeito do limiar de aceitação no tempo de resposta e acertos de um sistema de tomada de decisão, conforme definido na seção III. G.

Para efetuar esta análise, foi utilizado um sistema composto por três dispositivos, nesta ordem: TDAE, classificador de Bayes e k-Nearest Neighbor. A escolha destes dispositivos e de sua ordenação foi feita de forma que:

1. Casos conhecidos sejam solucionados de maneira imediata pela tabela de decisão;
2. Casos jamais vistos sejam solucionados de maneira rápida pelo classificador de Bayes;
3. Se o classificador de Bayes não conseguir um bom escore para a melhor solução, repassa o problema ao k-NN.

Note que o número de casos conhecidos, em geral, é muito superior ao número de critérios e de alternativas. Assim, por depender linearmente do tamanho da base de conhecimento, o k-NN deve ser mais lento que Bayes, que depende do número de critérios e alternativas.

Nota: exceto pela adição de Bayes e pelos dados de treinamento diferentes, este sistema possui a mesma configuração que o teste anterior, com k=10, p=2, etc.

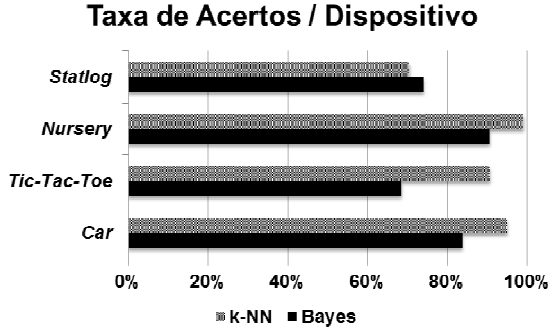

Figura 9: Taxa de acertos de 4 *datasets* quando testados com o classificador de Bayes e com o k-NN.

Como dados de testes, foram utilizados os mesmos *datasets* empregados anteriormente. Para cada conjunto, foi verificado o efeito do limiar sobre a taxa de acertos e tempo de resposta, variando-o de 0.3 até 1.0, em passos de 0.02. Para cada limiar, o sistema foi testado 10 vezes, sorteando aleatoriamente 70% dos dados como treinamento e utilizando os 30% remanescentes para validar o sistema, executando-o e verificando se a resposta foi aquela esperada pelo *dataset*.

Na maior parte destes conjuntos de testes, o k-NN – utilizando os parâmetros acima – obtém maior número de acertos do que o classificador de Bayes. Um dos raros exemplos contrários foi o *dataset Statlog*, conforme ilustra a Figura 9.

Os resultados do teste do limiar para quatro dos conjuntos utilizados podem ser visualizados pelos gráficos da Figura 10. De imediato, verifica-se que todos os exemplos apresentam uma piora no tempo de resposta com o aumento do limiar – comportamento esperado, uma vez que o k-NN passa a ser requisitado com maior frequência.

Outro resultado visível ocorre com os *datasets Car*, *Nursery* e *Tic-Tac-Toe*. Como k-NN consegue maior taxa de acertos do que Bayes nestes exemplos, seu uso faz com que os acertos do sistema cresçam conforme o limiar aumenta. Esta relação, então, pode ser utilizada para estimar um limiar que melhor satisfaça as necessidades do usuário (e.g. a escolha pode ser feita para "obter a maior número de acertos por unidade de tempo", ou "garantir uma taxa de acertos de x%").

Note-se também que, nestes exemplos, há um ponto de saturação, em que o aumento do uso do k-NN passa a não impactar em um aumento de acertos. Com isso, o uso deste limiar permite que o sistema consiga obter a mesma taxa de acertos do dispositivo mais capaz – o k-NN – utilizando, em média, apenas uma fração de seu tempo. O teste com o *dataset Car* é quem melhor evidencia este ganho: a taxa de acertos do sistema com limiar 0.5 é comparável ao k-NN, mas utilizando apenas 4ms por decisão.

O *Statlog* exemplifica um caso contrário, em que o uso

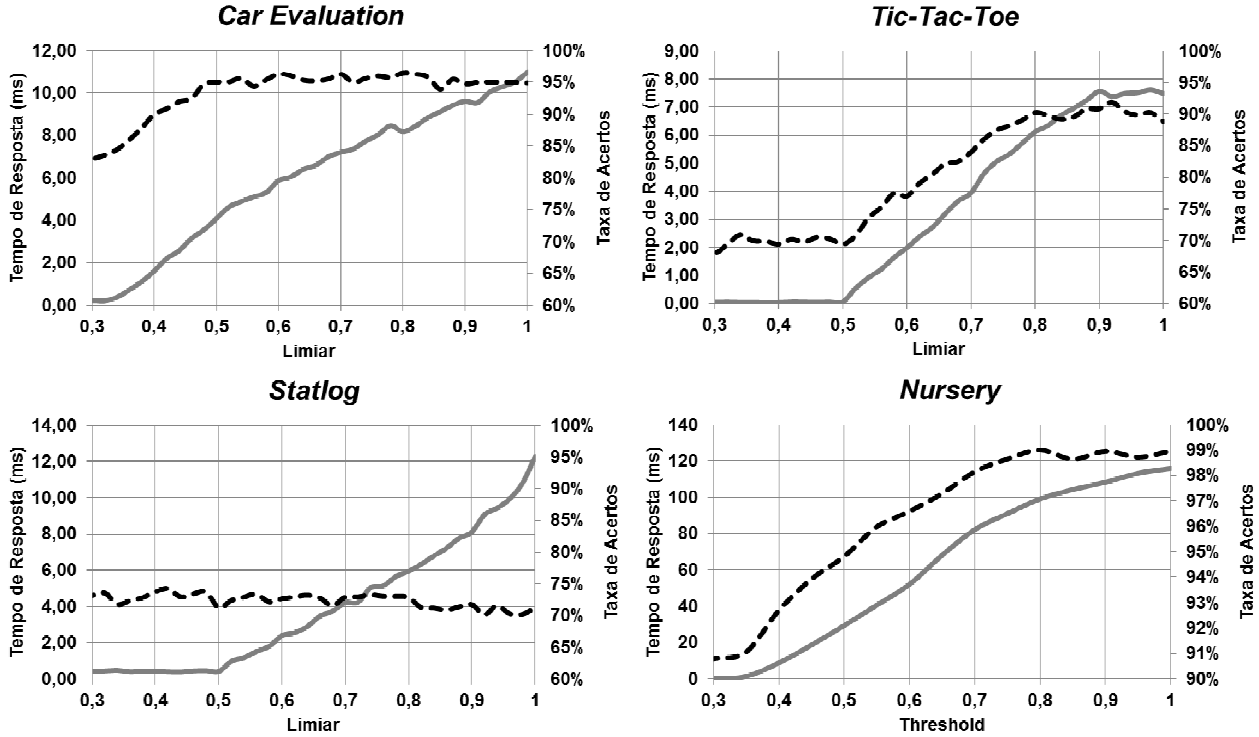

Figura 10: Resultados para o teste de compromisso entre acertos e tempo de resposta. A linha tracejada mostra a taxa de acertos do sistema para um determinado limiar, enquanto a linha em cinza indica, em média, o tempo de resposta utilizado para tomar cada decisão.

mais extensivo do k-NN resultou em resultados piores, não sendo, assim, vantagem utilizá-lo neste problema Deste resultado, pode-se verificar que a solução de um problema – ao invés de ficar a cargo do primeiro dispositivo que conseguir obter uma solução com índice de mérito acima do limiar – poderia ser determinada pela combinação das soluções obtidas por cada dispositivo, em uma tentativa de minimizar os erros via consenso.

VI. Conclusão

A. Contribuições:

Neste trabalho, foi criado um ambiente de tomada de decisão que, aliado às técnicas adaptativas, permite a criação de sistemas híbridos, que não só permitem utilizar um *trade-off* entre taxa de acertos e velocidade, mas também permitem utilizar métodos incrementais para que o sistema evolua de acordo com as entradas e realimentações recebidas.

Os resultados obtidos nestes primeiros experimentos se mostram promissores, permitindo o reuso de métodos clássicos. A junção dos métodos de k-Nearest Neighbor, tabela de decisão e Naive Bayes abrem a possibilidade de seu uso de maneira a tratar incertezas e aprender através de realimentações.

B. Trabalho Futuro:

Um aspecto a ser estudado é a validade temporal de uma decisão, para que seja possível conceber um ambiente capaz de tratar conceitos variáveis ao longo do tempo. Em especial, o processo de "esquecer" conhecimento anterior torna-se atraente para este fim. Também é desejável verificar uma maneira de o sistema verificar a possibilidade de haver novas alternativas – antes desconhecidas – que possam ser atribuídas para casos que não possuem solução definida.

Por fim, é desejável analisar mais uma série de dispositivos de diferentes naturezas, como exemplo, a árvore de decisão incremental [32] e o K* [23].

C. Mais Informações:

Informações relativas ao projeto – e.g. teorias e programas-exemplos – podem ser encontradas na seguinte URL:

http://sites.google.com/site/suzukiresearch/

Referências

[1] R. Crozier and R. Ranyard, "Decision Making: Cognitive Models and Explanations", in *Cognitive Process Models and Explanations of Decision Making*, *Decision Making*: *Cognitive Models and Explanations*, Saffron Walden: Routledge, 1997, pp. 5-20.

[2] D. E. Bell, H. Raiffa and A. Tversky, "Decision Making: Descriptive, Normative, and Prescriptive Interactions", in *Descriptive, Normative, and Prescriptive Interactions in Decision Making*, Cambridge: Cambridge University Press, 1988, pp. 9-30.

[3] B. D. Dunn, T. Dalgleish and A. D. Lawrence, "The Somatic Marker Hypothesis: A Critical Evaluation", Neuroscience and Biobehavioral Reviews, vol. 30, pp. 239–271, 2006.

[4] J. L. Kolodner, "Improving Human Decision Making Through Case-Based Decision Aiding", *AI Mag*, vol. 12, pp. 52-68, 1991.

[5] L. I. Kuncheva, "Using Control Charts for Detecting Concept Change in Streaming Data", Bangor University, UK, 2009.

[6] J. J. Neto, "Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa", *Revista IEEE América Latina*, vol. 5, (7), pp. 496-505, Nov. 2007.

[7] H. Pistori, "Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações", Ph.D. dissertation, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 2003.

[8] J. J. Neto, "Adaptive Rule-Driven Devices - General Formulation and Case Study", in *CIAA '01: Revised Papers from the 6th International*

Conference on Implementation and Application of Automata, 2001, pp. 234-250.
[9] G. A. Agasandjan, "Automata with a Variable Structure", *Doklady Akademii Nauk SSSR*, vol. 174, pp. 529-530, 1967.
[10] H. Christiansen, "Recognition of Generative Languages", *Programs as Data Objects*, pp. 63-81, 1985.
[11] J. J. Neto, "Contribuições à Metodologia de Construção de Compiladores", Ph.D. dissertation, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 1993.
[12] R. L. A. Rocha and J. J. Neto, "Autômato Adaptativo, Limites e Complexidade em Comparação com Máquina de Turing", in *Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000*, 2000, pp. 33-48.
[13] A. H. Tchemra, "Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério", Ph.D. dissertation, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 2009.
[14] C. E. D. Menezes and J. J. Neto, "Um Método Híbrido para a Construção de Etiquetadores Morfológicos, Aplicado à Língua Portuguesa, Baseado em Autômatos Adaptativos", in *Anais da Conferencia Iberoamericana em Sistemas, Cibernética e Informática*, 2002, pp. 19-21.
[15] B. A. Basseto and J. J. Neto, "A Stochastic Musical Composer Based on Adaptative Algorithms", in *Proceedings of the 6th Brazilian Symposium on Computer Music - SBC&M99*, 1999, pp. 105-113.
[16] J. Jesus, D. G. Santos, A. A. C. Junior and H. Pistori, "Adaptools 2.0: Aspectos de Implementação e Utilização", *Revista IEEE América Latina*, vol. 5, (7), pp. 527-532, 2007.
[17] H. G. Sol, C. A. T. Takkenberg and P. F. V. Robbé, "Expert Systems and Artificial Intelligence in Decision Support Systems", in *Proceedings of the Second Mini Euroconference*, 1985, pp. 17-20.
[18] R. H. Sprague and E. D. Carlson, *Building Effective Decision Support Systems*, Englewood Cliffs: Prentice Hall Professional Technical Reference, 1982, pp. 30-32.
[19] I. Hendrickx and A. Bosch, "Hybrid Algorithms with Instant-Based Classification", in *Machine learning - ECML 2005*, 2005, pp 158-169.
[20] J. R. Quinlan, "Induction of Decision Trees", *Machine Learning*, vol. 1, (1), pp. 81-106, Mar. 1986.
[21] C. Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning*, vol. 20, (3), pp. 273-297, 1995.
[22] J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5", *Journal of Artificial Intelligence Research*, vol. 4, pp. 77-90, Mar. 1996.
[23] J. G, Cleary and L. E. Trigg, "K*: An Instance-based Learner Using an Entropic Distance Measure", in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 108-114.
[24] Y. Yang and G. I. Webb, "A Comparative Study of Discretization Methods for Naive-Bayes Classifiers", in *Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop*, 2002, pp. 159-173.
[25] J. Dougherty, R. Kohavi and M Sahami, "Supervised and Unsupervised Discretization of Continuous Features", in *Machine Learning: Proceedings of the Twelfth International Conference*, 1995, pp. 194-202.
[26] I. H. Witten and E. Frank, *Data mining: Practical Machine Learning Tools and Techniques*, San Francisco: Morgan Kaufmann, 2005, pp. 393-399.
[27] T. Bui and J. Lee, "An Agent-Based Framework for Building Decision Support Systems", *Decision Support Systems*, vol. 25, (3), pp. 225-237, Apr. 2003.
[28] D. Arnott, "Decision Support Systems Evolution: Framework, Case Study and Research Agenda", *European Journal of Information Systems*, vol. 13, (4), pp. 247-259, Dec. 2004.
[29] D. A. Ross, J. Lim, R. S. Lin and M. H. Yang, "Incremental Learning for Robust Visual Tracking", *International Journal of Computer Vision*, vol. 77, (1-3), pp. 125-141, May 2008.
[30] A. Tsymbal, M. Pechenizkiy, P. Cunningham and S. Puuronen, "Dynamic Integration of Classifiers for Handling Concept Drift", *Information Fusion*, vol. 9, (1), pp. 56-68, Jan. 2008.
[31] D. W. Aha and D. Kibler, "Instance-based Learning Algorithms", in *Machine Learning*, 1999, pp. 37-66.
[32] P. E. Utgoff, N. C. Berkman and J. A. Clouse, "Decision Tree Induction Based on Efficient Tree Restructuring", *Machine Learning*, vol. 29, (1), pp. 5-44, Oct. 1997.
[33] J. L. Kolodner, "Improving Human Decision Making Through Case-Based Decision Aiding", *AI Mag.*, vol. 12, (2), pp. 52-68, 1991.
[34] A. Aamodt and E. Plaza, "Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", *AI Communications*, vol. 7, (1), pp. 39-59, 1994.
[35] B. E. Bakken, "Learning and Transfer of Understanding in Dynamic Decision Environments", Ph.D. dissertation, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA, 1993.
[36] E. Parsloe and M. J. Wray. *Coaching and Mentoring: Practical Methods to Improve Learning*. London: Kogan Page Publishers, 2000, pp. 33-40.
[37] H. J. Einhorn and R. M. Hogarth, "Confidence in Judgment: Persistence of the Illusion of Validity", *Psychological Review*, vol. 85, (5), pp. 395-416, Sep. 1978.
[38] D. Ding, Q. Li and J. Yang, "Multimedia Data Integration and Navigation Through Mediaview: Implementation, Evolution and Utilization", in *Lecture Notes in Computer Science*, 2004, vol. 2973, pp. 263-271.
[39] P. J. Acklam. "An algorithm for computing the inverse normal cumulative distribution function". Internet: home.online.no/~pjacklam/notes/invnorm/, Feb. 27, 2009 [Apr 5, 2011].
[40] L. D. Brown, T. T. Cai and A. DasGupta, "Interval Estimation for a Binomial Proportion", *Statistical Science*, vol. 16, (2), pp. 101-133, 2001.
[41] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, "The R*-Tree: an Efficient and Robust Access Method for Points and Rectangles", in *International Conference on Management of Data*, 1990, pp. 322-331.
[42] W. Liu and S. Chawla, "Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets", in *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 2011, pp. 345-356.
[43] D. Shepard, "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data", in *Proceedings of the 1968 ACM National Conference*, 1968, pp. 517-524.
[44] C. Li and H. Li, "A Survey of Distance Metrics for Nominal Attributes", *Journal of Software*, vol. 5, (11), pp. 1262-1269, 2010.
[45] A. Frank and A. Asuncion, "UCI Machine Learning Repository". Internet: http://archive.ics.uci.edu/ml, Sep. 13, 2011 [Sep. 29, 2011].

Rodrigo Suzuki Okada é graduado em Engenharia Elétrica pela Escola Politécnica da Universidade de São Paulo (USP), Brasil, em 2008. Entre os anos de 2007 a 2008, participou de pesquisas relacionadas à simulação de eventos estocásticos através do Laboratório de Arquitetura e Redes de Computadores (LARC). Desde 2010 é aluno de Mestrado em Engenharia Elétrica, na área de técnicas adaptativa através do Laboratório de Linguagens e Técnicas Adaptativas (LTA) da Universidade de São Paulo (USP). Suas pesquisas se concentram nas técnicas adaptativas voltadas para o aprendizado de máquina, cognição e reconhecimento de padrões.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas do PCS – Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.