Lógica Paraconsistente aplicada como Solução de Inconsistências entre Classificadores de Aprendizagem Simbólica

Luiz Gustavo Moro Senko[1]

[1]Instituto Federal Catarinense – Campus Ibirama
Rua Getúlio Vargas, 3006 – CEP 89140-000 Ibirama – SC – Brazil

gustavo.senko@ibirama.ifc.edu.br

Abstract. This paper presents a technique for handling inconsistencies in symbolic classifiers. Very often, ensemble-based techniques are used to improve performance and classification accuracy. However such methods compromise the knowledge understandability and decisions explanation because the knowledge is partitioned among the classifiers. Furthermore, the classifiers generated from data samples are likely inconsistent, demanding a complex ensemble combination strategy. On the other hand, knowledge integration techniques are used to merge symbolic classifiers into an understandable and global classifier, merging and selecting good classification rules from the local models. In this paper Paraconsistent Logic concepts are used to gather concepts from local models, generating a global ruleset with good accuracy avoiding data exchange and rules evaluations, saving a lot of computational resources. Whenever local models are gathered together the rules are ordered using Paraconsistent Logic operators and other ones are used for classification of new instances. We could observe good results in the experiments performed on a number of benchmark datasets.

Resumo. O desenvolvimento de algoritmos e técnicas em mineração distribuída de dados tem como principal interesse a análise de bases de dados fisicamente distribuídas. Essas abordagens produzem melhores soluções em termos de custos e complexidade computacional. Nesse cenário, a manipulação de regras de classificação inconsistentes é uma tarefa importante, pois inconsistências podem comprometer o desempenho dos classificadores. O objetivo deste trabalho foi o desenvolver uma nova metodologia para a integração de conhecimento (classificadores à base de regas) baseado em Lógica Paraconsistente. O método permite tomar decisões confiáveis na ocorrência de regras inconsistentes encontradas em diferentes classificadores. A vantagem do método proposto é evitar o alto fluxo de mensagens e dados entre processadores distribuídos durante a análise das regras geradas de diferentes bases de dados. Isso ocorre porque apenas informação local é utilizada para a geração de um conjunto de regras global. Lógica Paraconsistente é utilizada como um mecanismo de tratamento de incertezas para inferir a classe de um determinado exemplo de teste. Nós pudemos observar nos experimentos realizados sobre diferentes bases de dados que o método pode gerar excelentes resultados.

Keywords: Artificial Intelligence, Knowledge Acquisition, Knowledge based systems, Machine Learning

1. INTRODUÇÃO

Muitas pesquisas têm sido desenvolvidas para viabilizar a análise de grandes volumes de dados gerados diariamente em centros de pesquisas, organizações comerciais, industriais, etc. Os métodos desenvolvidos são geralmente baseados em mineração distribuída de dados. Entre os principais métodos desenvolvidos, destacam-se os de mineração distribuída orientada a dados [Freitas, A. and Lavington, S. H. 1998]. Esses métodos têm como princípio a divisão dos dados em subconjuntos menores, que são minerados individualmente, produzindo classificadores locais. Posteriormente, esses classificadores são combinados em um único classificador global. Nesse caso, um dos problemas que podem ocorrer é a existência de dados inconsistentes nos subconjuntos, levando o sistema a gerar classificadores com opiniões contraditórias. Essas inconsistências podem ser geradas pelo método de amostragem escolhido, ou podem ser inerentes às bases de dados distribuídas em diferentes sites (por exemplo, bases de dados de uma rede de lojas). Em mineração distribuída, a ocorrência de dados conflitantes ou inconsistentes compromete significativamente o desempenho dos algoritmos de aprendizagem da máquina. Não é difícil encontrar situações nas quais existam inconsistências explícitas. Por exemplo, considere uma técnica de combinação de classificadores que tem como objetivo unificar a decisão de diversos classificadores gerados a partir de bases de dados distribuídas. Assumindo que uma instância pode pertencer apenas a uma classe, pode haver opiniões contraditórias entre os classificadores se três deles previrem a classe "A", quatro deles previrem a classe "B" e os últimos três classificadores previrem a classe "C". Nesse caso, uma técnica como votação simples seria de difícil aplicação e poderia gerar erros de classificação. Os dados armazenados nas bases são considerados precisos se for possível assegurar que representam fielmente os dados do "mundo real", o que nem sempre ocorre.

Segundo da Costa [da Costa, N. C. A and Abe, J. M. 2000], inconsistências surgem naturalmente no mundo real. Elas podem ocorrer em vários contextos e a automatização de um raciocínio adequado requer o desenvolvimento de teorias formais apropriadas. A existência de informações conflitantes pode estar relacionada a diversas razões, como, por exemplo, à união de bases de dados geradas a partir de diferentes fontes distribuídas, ruído nos dados, atributos com valores faltantes, erro na coleta de dados, entre outras razões, notadamente a falta de variáveis necessárias para a aquisição de conhecimento.

O método proposto neste trabalho utiliza o conhecimento obtido a partir de algoritmos de indução de regras do tipo "SE (condições) ENTÃO classe", nas quais o antecedente (condições) contém conjunções de condições formadas por pares de atributos e valores e o conseqüente (classe) contém a classe definida para um conjunto de exemplos que satisfazem as condições do antecedente da regra.

Em mineração distribuída, a união dos modelos obtidos por diversos classificadores pode resultar em um conhecimento global inconsistente, se as regras obtidas em classificadores diferentes são mutuamente exclusivas, mas prevêem a mesma classe. Imagine, por exemplo, que um classificador A possui a seguinte regra: SE idade > 25 ENTÃO classe = homem; enquanto o classificador B possui a seguinte

regra: SE idade < 12 ENTÃO classe = homem. Uma vez que os classificadores foram gerados a partir de dados da mesma natureza, essa contradição não é aceitável. A inconsistência também pode ser gerada em função do conseqüente das regras. Por exemplo, se um classificador A possui a regra "SE idade > 15 ENTÃO classe = homem" e o classificador B possui a regra "SE idade > 12 ENTÃO classe = mulher", pode-se dizer que existe uma contradição entre os conceitos das classes homem e mulher, pois um exemplo de teste com "idade = 21" seria classificado ao mesmo tempo como "homem" e "mulher".

Nesse caso, um critério de ponderação de regra ou classificador deveria ser utilizado para a escolha final. O objetivo deste trabalho foi desenvolver uma metodologia, utilizando conceitos de Lógica Paraconsistente, para auxiliar na obtenção de melhores interpretações sobre conjuntos de regras em que situações como as descritas no parágrafo anterior podem ocorrer, sem, entretanto, prejudicar o desempenho do sistema. A utilização dos formalismos da Lógica Paraconsistente permitiu associar a cada regra fatores evidenciais anotados que representam, respectivamente, o grau de crença (o quanto se acredita que a regra seja verdadeira) e o grau de descrença (o quanto a regra pode ser considerada falsa). Os fatores evidenciais serviram de base para aplicar o critério de decisão na escolha da regra que satisfaça as condições dos exemplos de teste e seja a mais próxima do estado lógico verdade da Lógica Paraconsistente.

2. LÓGICA PARACONSISTENTE

Informações contraditórias podem ser importantes no processo de raciocínio e tomada de decisão. Eliminá-las pode causar impacto negativo na solução de determinados problemas. A presença de informações inconsistentes em sistemas computadorizados [Enembreck, F. 1999] pode ser facilmente observada, uma vez que, em diversas aplicações, a inconsistência é inerente ao problema.

De acordo com Enembreck [Enembreck, F. 1999], existem basicamente duas formas para tratar inconsistências: i) atribuir ao algoritmo de aprendizado a habilidade de manipular adequadamente as informações contraditórias durante o processo de aprendizado, com a finalidade de gerar conceitos consistentes e confiáveis; ou ii) aplicar sobre o conhecimento, obtido por meio de um algoritmo de aprendizado qualquer, um método de raciocínio que possibilite a inferência confiável de informações: transformar a base de conhecimento de modo a torná-la consistente, gerar outra base de conhecimento a partir dos dados consistentes existentes na base inicial, ou ainda aplicar técnicas de gerenciamento de incerteza que possibilitem o raciocínio sobre conceitos inconsistentes. Neste trabalho, optou-se por essa última hipótese. Esta escolha possibilita o raciocínio sobre quaisquer conjuntos de regras do tipo SE-ENTÃO, tornando o método independente do algoritmo de aprendizado simbólico utilizado. Além disso, podem-se inferir decisões a partir de conjuntos de regras existentes em diferentes locais, gerados a partir de diferentes bases de dados, mesmo que existam variações de distribuição.

O modelo de inferência proposto está baseado no modelo da Lógica Paraconsistente [da Costa, N. C. A and J. M. Abe 2000] [Blair, H.A. and Subrahmanian V. S. 1998]. A Lógica Paraconsistente, diferentemente da Lógica Clássica, permite representar e realizar inferências sobre informações contraditórias e

também distinguir as situações em que uma determinada proposição é realmente falsa daquelas em que não se tem conhecimento suficiente para se chegar a uma conclusão. Ela permite que informações que são contraditórias p e $\neg p$ estejam simultaneamente presentes e fornece mecanismos para o raciocínio sobre informações com essas características. A conclusão obtida pode ser muito útil para a tomada de decisões em que não há informações suficientes, ou elas são contraditórias.

Lógica Evidencial Paraconsistente (LEP) [Subrahmanian, V. S. 1987] é um formalismo desenvolvido que utiliza conceitos de Lógica Paraconsistente.

Os valores-verdade utilizados em LEP são compostos por dois fatores evidenciais pertencentes a um reticulado $\{x \in \Re \mid 0 \le x \le 1\} \times \{x \in \Re \mid 0 \le x \le 1\}$.

A cada item do conhecimento de um sistema, nesse caso, a cada regra em um subconjunto, são associados fatores evidenciais de crença e descrença. O grau de crença representa a força que apóia a verdade da evidência, ao passo que o grau de descrença denota a força associada à falsidade da evidência. Ambos os fatores pertencem ao intervalo [0.0, 1.0], ou seja, infinitos valores podem ser associados às premissas pertencentes ao sistema. Portanto, pode ser definido um reticulado infinito $\tau = \langle |\tau|, \le \rangle$, tal que:

$$|\tau| = \{x \in \Re \mid 0 \le x \le 1\} \times \{x \in \Re \mid 0 \le x \le 1\}$$

O reticulado τ que pode ser demonstrado pelo diagrama de Hasse, possui um ponto máximo $[1.0,1.0]$que representa a situação de inconsistência máxima, pois se acredita tanto na verdade quanto na falsidade da premissa, simultaneamente. A verdade de uma premissa é representada pelos fatores evidenciais [1.0,0.0], pois se acredita totalmente na verdade e nada é conhecido sobre a falsidade. Por outro lado, a falsidade é representada por [0.0,1.0], porque se acredita totalmente na falsidade e sobre a verdade da premissa nada é conhecido.

Além de ser possível representar a inconsistência, em LEP, diferentemente da Lógica Clássica, é possível representar o desconhecido ou indeterminado – [0.0, 0.0]. Nesse caso, não se tem informação nem sobre a verdade nem sobre a falsidade da premissa.

3. METODOLOGIA

Estudos realizados [Ladeira, M., Viccari, R. M. 1996] indicam que podem existir várias causas de incertezas em sistemas computacionais e de Inteligência Artificial. Entre elas, está a presença de informações imprecisas e inconsistentes. A inconsistência pode ocorrer, por exemplo, quando informações geradas a partir de fontes distribuídas são unidas para que inferências possam ser realizadas.

Com o objetivo de fornecer um raciocínio mais adequado nessas situações, este trabalho propõe a aplicação dos conceitos de Lógica Evidencial Paraconsistente [da Costa, N.C.A. et al. 1999] [da Costa, N. C. A and Abe, J. M. 2000] [Blair, H.A. and Subrahmanian, V. S. 1998] [Subrahmanian, V. S. 1987] em mineração distribuída [Freitas, A. and Lavington, S. H. 1998]. A primeira etapa para o desenvolvimento do

método baseado na linguagem Paralog_e [Ávila, B. C. 1996] é a preparação e mineração das bases de dados.

Os diferentes conjuntos de regras são obtidos por meio da aplicação do algoritmo RIPPER [Cohen,W.W. 1995] sobre os diferentes subconjuntos de dados. Utilizou-se a implementação do RIPPER existente no WEKA [Frank, E.]. Esse algoritmo executa a aprendizagem de regras ordenadas e, dessa forma, permite que um exemplo possa ser classificado por mais de uma regra. Ele utiliza critérios de poda incremental para reduzir erros e produzir regras de boa qualidade mesmo em domínios ruidosos.

A segunda etapa consiste na transformação das regras obtidas anteriormente para o formato Paralog_e [Ávila, B. C. 1996]. Este último utiliza conceitos de programação lógica evidencial e permite inferir a partir de regras anotadas. Cada regra está associada a um grau de crença e outro de descrença.

Dessa forma, todas as regras do tipo SE (condições) ENTÃO (classe) são transformadas em regras Paralog_e, que possuem o formato Corpo← Cabeça. A Cabeça representa a conclusão da regra e é formada pela classe e pelos fatores evidenciais de crença e descrença que representam a regra. O Corpo é composto por uma conjunção de condições. A conjunção é denotada pelo símbolo "&". O conjunto de conjunções forma as condições que compõem a regra. Cada condição da regra é representada por um predicado avaliador que possibilita posteriormente verificar se as condições das regras são verdadeiras quando exemplos de teste são submetidos à inferência. Para ilustrar o procedimento de transformação de regras no formato Paralog_e, foi criada a seguinte regra no formato RIPPER [Cohen, W. W. 1995] sobre uma base de dados hipotética:

(producaolacrimal = normal) and (astigmatismo = sim) and

(espectropia = hipermetropia) => class=nenhuma (2.0/1.0)

Ao submeter a regra à transformação, obtém-se a mesma regra no seguinte formato:

class('nenhuma'):[2.0,1.0] <--

avaliador(producaolacrimal,V_0):[1.0,0.0] &

V_0 = normal &

avaliador(astigmatismo,V_1):[1.0,0.0] &

V_1 = sim &

avaliador(espectropia,V_2):[1.0,0.0] &

V_2 = hipermetropia.

É importante observar, nesta etapa, que os fatores evidenciais associados ainda não correspondem ao intervalo [1.0,0.0], pois na etapa seguinte esses valores serão modificados.

Na terceira etapa os valores dos fatores evidenciais são modificados a partir de uma ponderação linear da regra na forma de uma progressão aritmética. Essa atualização é realizada localmente, ou seja, antes da integração dos conjuntos de regras.

O fator evidencial favorável (c) (o grau de crença) é atualizado com o valor obtido por meio da ponderação linear do subconjunto de regras. O fator evidencial desfavorável (d) (o grau de descrença) é o complemento da ponderação linear no subconjunto de regras (d = 1 – c). Para se obter a razão da progressão aritmética, é necessário aplicar a seguinte Equação 1.

$$r = \frac{1}{número\_de\_regras\_no\_subconjunto} \quad (1)$$

O valor de crença atribuído à i_ésima regra corresponde à progressão aritmética de i por r partindo a última regra para a primeira. A última regra sempre possui $c = r$ e a primeira regra sempre possui $c = 1,0$. Um exemplo desta etapa pode ser visto.

Em Lógica Paraconsistente, os valores dos graus de crença e descrença são independentes entre si, pois não são valores complementares. Uma vez que os valores de crença *(c)* e descrença *(d)* calculados até então são complementares, eles devem ser modificados para se tornarem comparáveis à verdade absoluta, caracterizada pelo par ordenado (1.0,0.0). A partir da crença e descrença, são obtidos $valor_1$ e $valor_2$ da seguinte forma:

• $valor_1$ é denominado de Grau de Certeza e é calculado como o grau de crença diminuído do grau de descrença da regra, $(valor_1 = c - d)$;

• $valor_2$ é calculado pela multiplicação do grau de descrença da regra por 2, ($valor_2$ = 2 x d).

O grau de certeza $valor_1$ mede quanto de certeza está associado à verdade da proposição. Por outro lado, empiricamente foi definido o $valor_2$ como o dobro da descrença associada à regra. Acredita-se que toda regra produziria um erro maior (portanto, uma maior descrença) caso fosse avaliada sobre conjuntos de dados diferentes.

Como não há indícios de quanto seria essa propagação, o dobro da descrença foi utilizado empiricamente. Após o cálculo dos fatores evidenciais, os subconjuntos de regras são unidos em um único conjunto. Nesta fase, as regras são ordenadas de acordo com a menor distância euclidiana – conforme Equação 2 – em relação aos fatores evidenciais (1.0,0.0) que representam a verdade. onde x_i é o par ordenado formado por ($valor_1$, $valor_2$) de uma regra e x_j é (1.0,0.0).

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \quad (2)$$

Considera-se que quanto menor a distância obtida entre os valores ($valor_1$, $valor_2$), que denotam os fatores evidenciais da regra e o estado lógico que representa a verdade (1.0,0.0), mais próxima a regra está da verdade em termos quantitativos.

Na última etapa, cada exemplo de teste é transformado em um conjunto de fatos submetidos à prova a partir da base de conhecimento formada pelo conjunto de regras. A representação das regras em cláusulas Horn [Casanova, M.A; Giorno, F.A.C.; Furtado, A.L.F. 1987] permite verificar se as condições das regras são verdadeiras em relação ao exemplo de teste fornecido. A linguagem Paralog_e trabalha de maneira semelhante à linguagem Prolog, utilizando um procedimento de Resolução SLD [Casanova, M.A; Giorno, F.A.C.; Furtado, A.L.F. 1987] para encontrar a verdade ou falsidade de cada condição de uma regra.

Em seguida, é feito um questionamento *Q* para cada classe pertencente ao domínio da base de dados. As respostas são as evidências obtidas para cada classe. Como um questionamento é feito para cada classe pertencente ao domínio, o critério de decisão adotado elege a regra capaz de cobrir o exemplo cujos fatores evidenciais possuem a menor distância euclidiana em relação ao estado lógico da verdade — par ordenado (1.0,0.0). Para avaliar o método baseado na linguagem Paralog_e, a classe eleita é comparada à classe que rotula o exemplo de teste. Se a classe eleita for a mesma do exemplo, houve acerto na classificação.

4. EXPERIMENTOS, RESULTADOS E DISCUSSÕES

Nos experimentos, foram utilizadas dez bases de dados públicas, pertencentes ao diretório de bases para aprendizagem de máquina e mineração de dados, mantido pela Universidade da Califórnia [Blake, C.L.; Newman, D.J.; Hettich, S.; Merz, C.J. 1998].

Cada base utilizada nos experimentos foi dividida aleatoriamente para compor o conjunto de treinamento e testes. O conjunto de treinamento possui 80% dos exemplos existentes na base e o conjunto de teste contém os 20% restantes. O conjunto de treinamento foi dividido em 10 partições de treinamento na forma de amostras de 10% com recobrimento. Portanto, cada conjunto de regras foi gerado em partições de cerca de (0,8 × 0,1) × 100 = 8% de todos os dados originais. Esse percentual foi escolhido para que a ocorrência de conflitos seja favorecida, uma vez que o algoritmo de aprendizagem utiliza um subespaço reduzido do conjunto de tuplas. Cada classificador gerado em uma partição é avaliado sobre os 20% dos exemplos de teste separados anteriormente. Esse procedimento foi repetido iterativamente 10 vezes para cada base de dados.

As características de cada base utilizada, bem como o número de exemplos presentes e a existência ou não de valores faltantes nos exemplos que compõem a base, entre outras características, foram detalhadas na Tabela I.

É importante considerar que, segundo Freitas [Freitas, A. and Lavington, S. H. 1998], em algoritmos de indução o produto cartesiano das variáveis e o número de

valores que estes atributos podem assumir aumentam exponencialmente o tamanho do espaço de tuplas e conseqüentemente o espaço de regras a ser pesquisado.

Tabela I. Característica das bases utilizadas

	Base de Dados	#Número de Exemplos	Valores Faltantes	#Total de Atributos	#Atributos Nominais	#Atributos Numéricos	#Classes	Distribuição de Classes
1	Zoo	101	não	17	16	1	7	desbalanceada
2	Audiology	226	sim	69	69	–	24	desbalanceada
3	Monk 1	432	não	6	6	–	2	desbalanceada
4	Monk 2	432	não	6	6	–	2	desbalanceada
5	Soybean	683	sim	35	35	–	19	desbalanceada
6	Vehicle	846	não	18	–	18	4	desbalanceada
7	Tic-Tac-Toe	958	não	9	9	–	2	desbalanceada
8	Vowel	990	não	13	3	10	11	balanceada
9	Car	1728	não	6	6	–	4	desbalanceada
10	Segment	2310	não	19	–	19	7	balanceada

Apesar disso, o número de atributos, bem como seus possíveis valores, contribuiu para gerar conjuntos de regras mais expressivos, gerando possivelmente regras formadas por um maior número de condições. Dessa forma, tais regras possivelmente representam melhor as características da base, cobrindo regiões específicas do espaço de tuplas e gerando menos interseções entre as regras. Quando os subconjuntos de regras são considerados individualmente, as interseções e inconsistências são ignoradas, gerando muitos erros de classificação. Por outro lado, o método proposto é capaz de identificar essa situação e selecionar a regra mais adequada para classificar o exemplo.

Deve-se notar que, à medida que o número de atributos diminui, o espaço de tuplas é drasticamente reduzido. As bases Tic-Tac-Toe e Car são densamente populadas (em termos do número de exemplos). Dessa forma, a amostragem dos dados não tem um forte impacto no desempenho dos classificadores simbólicos locais. Por outro lado, quando esses classificadores são unidos, o sistema não é capaz de diferenciar a melhor regra, gerando erros de classificação.

Acredita-se, portanto, que a técnica apresentada neste trabalho é indicada para mineração de conjuntos de dados distribuídos que representam visões parciais do espaço de *tuplas.*

O objetivo do método proposto foi tratar possíveis inconsistências provocadas pela união dos N subconjuntos de regras formados a partir da divisão de dados e evitar o fluxo de comunicação entre os processadores, reduzindo o tempo de processamento

e, principalmente, garantindo uma taxa de acerto aceitável. A utilização dos conceitos de Lógica Paraconsistente permitiu atribuir às regras fatores evidenciais de crença e descrença, quantificando o grau de importância da regra em relação aos subconjuntos. Com a ordenação do conjunto de regras, foi possível mensurar as regras em relação ao estado lógico que representa a verdade e determinar um critério de decisão na escolha entre as regras candidatas.

Tabela II. Comparação de resultados entre o método Paralog_e e o algoritmo RIPPER

Base de Dados	Média geral no Paralog_e (%)	Média Geral no Ripper (%)	Diferença Percentual entre os Métodos	Relação entre #Atributos e # Total de Exemplos
Audiology	**48,69 ± 2,94**	32,65 ± 3,88	1,49	0,305
Zoo	**61,43 ± 4,73**	49,14 ± 4,14	1,25	0,168
Soybean	**63,06 ± 7,59**	51,21 ± 2,32	1,23	0,051
Vowel	**43,89 ± 4,35**	33,29 ± 1,14	1,32	0,013
Segment	**87,87 ± 3,83**	82,83 ± 1,43	1,06	0,008
Monk2	**63,70 ±10,83**	60,48 ± 2,25	1,05	0,014
Vehicle	**52,35 ± 4,12**	52,24 ± 2,18	1,00	0,021
Monk1	55,23 ± 5,66	**55,50 ± 4,25**	1,00	0,014
Tic-Tac-Toe	65,22 ± 0,94	**67,76 ± 2,27**	0,96	0,009
Car	58,24 ± 4,45	**71,98 ± 1,36**	0,81	0,003

Na análise dos resultados, foi possível identificar que a relação entre o número de atributos que compõem a base pode ter impacto significativo no desempenho do método. Pode-se observar que o método apresenta um melhor desempenho quando a relação entre número de atributos e número de exemplos é elevada (o espaço de tuplas é esparso), pois os modelos locais gerados cobrem regiões distintas do espaço de tuplas.

5. REFERÊNCIAS

Freitas, A. and Lavington, S. H. Approaches to Speed Up Data Mining.Mining Very Large Databases with Parallel Processing. ISBN 0-79238048-7.Pages 89-108. Kluwer Academic Publishers, The Netherlands,1998.

da Costa, N. C. A and J. M. Abe. Paraconsistência em Informática e Inteligência Artificial. Revista Estudos Avançados n. 14, vol. 39 – USP. Estudos Avançados n. 14, vol. 39 – USP. São Paulo, maio/agosto 2000.

Enembreck, F. Um Sistema Paraconsistente para Verificação Automática de Assinaturas Manuscritas. Dissertação de Mestrado - PPGIA – PUCPR. Curitiba, 1999.

Blair, H.A. and Subrahmanian, V. S. Paraconsistent Foundations for Logic Programming. Journal of Non-Classical Logic, 5, 2, pag. 45-73, 1988

Subrahmanian, V. S. On the Semantics of Quantitative Logic Programs. Proceedings of 4th IEEE Symposium on Logic Programming, San Francisco, September, pp. 173-182, 1987.

Ladeira, M.; Viccari, R. M. Representação do Conhecimento Incerto. XIII Symposium on Artificial Intelligence SBIA'96, Curitiba, October,1996.

da Costa, N.C.A. et al., N. A. Lógica Paraconsistente Aplicada. Atlas, São Paulo, 1999.

Subrahmanian, V. S. On the Semantics of Quantitative Logic Programs.Proceedings of 4th IEEE Symposium on Logic Programming, SanvFrancisco, September, pp. 173-182, 1987.

Ávila, B. C. Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar de Exceções em Sistemas de Frames com Múltipla Herança. Tese de Doutorado, Escola Politécnica da Universidade de São Paulo. São Paulo, 1996.

Cohen, W. W. Fast Effective Rule Induction. In Proceedings of the 12th Int. Conference in Machine Learning (ICML '95). Pages 115-123. 1995.

Casanova, M.A; Giorno, F.A.C.; Furtado, A.L.F. Programação em Lógica e a Linguagem Prolog. Ed. Edgard Blücher Ltda. São Paulo, 1987.

Blake, C.L.; Newman, D.J.; Hettich, S.; Merz, C.J. UCI Repository of machine learning databases. Available on:

[http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

Frank, E. *WEKA Machine Learning Software*.

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