GUSTAVO CESAR BAZZO

CLASSIFICAÇÃO AUTOMÁTICA DE ERROS DE APRENDIZES HUMANOS DO PROCESSO DE INDUÇÃO ANALÍTICA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne Co-orientador: Prof. Dr. Luiz Eduardo Soares Oliveira

CURITIBA

2011

GUSTAVO CESAR BAZZO

CLASSIFICAÇÃO AUTOMÁTICA DE ERROS DE APRENDIZES HUMANOS DO PROCESSO DE INDUÇÃO ANALÍTICA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne Co-orientador: Prof. Dr. Luiz Eduardo Soares Oliveira

CURITIBA

2011

AGRADECIMENTOS

A Deus por toda força, perseverança e disciplina durante toda minha vida.

Ao professor orientador Alexandre Direne que com qualidades de um verdadeiro lider guiou este trabalho até o fim. Gostaria de ratificar sua competência como profissional e ser humano. Sou profundamente grato pela oportunidade e por toda confiança depositada em mim.

Ao professor co-orientador Luiz Eduardo Soares Oliveira pela ajuda com ideias valiosas para atingir os objetivos deste trabalho.

A Diego Marczal pela sua colaboração em momentos importantes no decorrer desta pesquisa.

A minha esposa, família e amigos da empresa Youse Technologies, por toda paciência e apoio durante os momentos de ausência.

Por fim agradeço a todos os professores, funcionários, colegas da pós-graduação e brasileiros que direta ou indiretamente contribuíram para a minha formação acadêmica e realização deste trabalho.

Muito obrigado.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

O problema da classificação automática de erros humanos no processo de indução analítica é exposto de maneira crítica em relação ao contexto da aprendizagem formal da matemática. As principais limitações de outras pesquisas existentes são apontadas em relação à concepção de modelos dinâmicos de aprendizes em sistemas tutores inteligentes. Os aspectos metodológicos de uma solução factível para o problema proposto são identificadas com base em métricas genéricas que se aplicam às diferenças entre expressões matemáticas erradas e corretas. Uma implementação do modelo estudado é feita com base em dados experimentais coletados de aprendizes reais e submetidos à análise de especialistas no ensino de indução analítica. Os principais resultados apontam para índices de 87% de acerto na classificação automática. Metas futuras de pesquisa são delineadas no sentido de monitorar a evolução do desempenho acadêmico dos aprendizes por longos períodos.

ABSTRACT

The problem of automatic classification of human errors in the process of analytic induction is laid out as critical relations of formal learning contexts of mathematical concepts. The main limitations of past research are spotted in relation to the concepts of dynamic learner modelling in intelligent tutoring systems. Methodological aspects of a fisible solution to the proposed problem are identified on the basis of generic measures which apply to differences between wrong and correct mathematical expressions. An implementation of the model is carried out using experimental data collected from real learner and submitted to experts in analytic induction teaching to be analysed. The mais results point to an 87% hits in automatic classification. Future research aims are outlined as an effort to track learners' performance during long periods of time.

CAPÍTULO 1

INTRODUÇÃO

1.1 O problema central

O objetivo geral deste projeto de pesquisa e desenvolvimento está na utilização do estado-da-arte da modelagem dinâmica de aprendizes em sistemas tutores inteligente para oferecer apoio tecnológico e pedagógico adaptado ao trinômio aprendiz-professor-conteúdo. Neste contexto, o apoio significa o monitoramento de longo prazo do aprendiz em relação aos erros cometidos na solução de problemas. O domínio específico de problemas tratado pelo sistema tutor é o de classificação de erros no campo da matemática denominado *indução analítica*. É importante destacar desde já que um erro desse tipo pode ser mascarado por erros de naturezas algébrica, aritmética, geométrica e outras. Isso causa enormes dificuldades de representação e de operação, tanto para o aprendiz humano como para uma máquina. Para efeito de simplificação da classificação, apenas três tipos de erros de indução analítica serão considerados neste trabalho: *(a)* subgeneralização; *(b)* supergeneralização; *(c)* não se aplica (é um erro de outra natureza).

Para a expressão correta $\frac{l}{2^n}$, um exemplo de erro da classe de subgeneralização é $\frac{l}{n^2}$. Adicionalmente, para a expressão correta $\frac{l}{8}$, um exemplo de erro da classe de supergeneralização é $\frac{l}{8 \times n}$. Cabe ainda dizer que, para a expressão correta $\frac{l}{2^n}$, um exemplo de erro que não se aplica a nenhuma das duas classes citadas anteriormente é $\frac{l}{16 \times 2^n}$.

Vários estudos foram conduzidos ao longo dos anos a respeito das diferenças fundamentais entre aprendizes de conceitos matemáticos dos níveis inicial, intermediário e avançado [3, 9, 7]. Os estudos consideraram tanto a relação da indução analítica com conceitos mais básicos da matemática (*e.g.*, aritméticos, algébricos e geométricos) quanto a modelagem dinâmica de aprendizes onde a mobilidade em redes virtuais catalisa ati-

vidades de aprendizagem com apoio inteligente [35]. Trabalhos mais recentes apenas especulam sobre a adequação de certas ferramentas para apoiar à aprendizagem com mobilidade [22]. Na medida do possível, a incorporação de resultados de pesquisa básica em psicologia cognitivista tem sido tentada como forma de cobrir o grande espaço existente entre campos interdisciplinares de desenvolvimento da indução analítica em ambiente escolar. Tal cobertura pode ser atingida com base em semelhanças conceituais entre, por exemplo, o processo de aquisição de conhecimentos de matemática e de estratégias de jogos heurísticos [27].

Pesquisas passadas sugerem que um indivíduo passa por diversos estágios durante a aquisição de conhecimento e competência em indução analítica [2]. Iniciantes aprendem a associar sinais para guiar a face operativa do acesso aos conteúdos de uma expressão. Na medida em que se tornam mais experientes, aprendizes desenvolvem uma abordagem, baseada em métodos indutivos, mais sistemática de composição de operadores para sintetizar expressões que resolvem problemas. Tais procedimentos podem ser resumidos em três passos: (a) estimativa rápida das expressões constituintes dos passos intermediários da solução à procura de características comuns que indiquem um arranjo ou padrão operativo familiar; (b) formação de uma hipótese inicial a respeito da composição de operadores e variáveis da expressão final; (c) busca sistemática de novas características operativas que venham a reforçar ou desconfirmar a hipótese de síntese adotada.

Usuários avançados são capazes de usar a abordagem hipotético-dedutiva quando necessário mas também desenvolvem com o tempo uma enorme habilidade para rápidas comparações de padrões sintáticos dos dados de um enunciado. Tal habilidade se baseia em um volumoso estoque de esquemas mentais de natureza metacognitiva [21, 25], destinados a controlar de formas diferentes a busca por um conteúdo equivalente mais resumido. Ao contrário dos iniciantes, os usuários avançados usam esquemas fortemente associados às operações das expressões intermediárias da solução de um problema e têm sólidas representações mentais da estrutura de precedências dos operadores assim como de seus operandos, sejam eles variáveis ou constantes.

Os estudos sobre diferenças entre entre iniciantes e avançados no tema de mobilidade

educacional [35] indicam que os usuários de nível intermediário de competência necessitam de especial assistência na identificação dos operadores críticos da formação de uma hipótese a respeito de um enunciado de problema. Além disso, eles também carecem de ajuda na indicação de evidências para a confirmação da eficácia da opção pretendida para serem os operandos desses operadores.

A construção de sistemas tutores inteligentes capazes de apoiar essa integração de fatores ainda é relativamente rara, mas algumas tentativas anteriores já demonstraram resultados parciais animadores [12]. Os resultados positivos são advindos de trabalhos em software construído com alguma inteligência isolada para ajudar o ensino. Todavia, em geral, as visões mais modernas de contribuição da Inteligência Artificial aplicadas à Educação têm sido voltadas à aprendizagem muito mais do que ao ensino e, nesses casos, geralmente focalizam jogos intelectivos como objetos de estudo da solução de problemas por parte de um aprendiz [20, 33].

1.2 Contexto do projeto

Esta pesquisa será desenvolvida dentro do contexto do projeto de pesquisa e desenvolvimento CONDIGITAL do grupo do estado do Paraná. Este grupo é financiado pelo MEC, por meio do FNDE (Fundo Nacional de Desenvolvimento da Educação), em atendimento ao Edital conjunto 001/07 MEC/MCT. O objetivo deste projeto é contribuir para a melhoria e a modernização dos processos de ensino e aprendizagem da área da Matemática. Aborda a faixa etária do último ano do Ensino Fundamental e todo o Ensino Médio. Como metas, o projeto tem planejamento de produzir não apenas os chamados Objeto de Aprendizagem mas também fragmentos de software inteligente e métodos documentados de abordagem pedagógica e didática para as seguintes subáreas do conhecimento matemático: (a) progressões geométricas em fractais; (b) funções de primeiro grau; (c) matemática financeira; (d) funções cíclicas.

Por convenção interna do projeto, durante a execução de qualquer passo de solução de um problema, o aprendiz é orientado a inserir como resposta diversas expressões analíticas para o mesmo enunciado. Cada exercício resolvido implica no aumento da complexidade,

aumentando também a quantidade de conceitos envolvidos. Isso requer esforço cognitivo do aprendiz e portanto é provável que venham a ocorrer erros durante este processo. Visando auxiliar o aprendiz na resolução de exercícios de forma personalizada e especializada é que surgiu a necessidade de desenvolvimento deste trabalho. Os objetivos principais e secundários estão descritos na seção a seguir.

1.3 Objetivos do trabalho

O objetivo principal deste trabalho é desenvolver métodos e ferramentas de software capazes de realizar a classificação automática de um aprendiz humano em situações de erro durante a resolução de exercícios. Esta classificação facilitará futuras tarefas também automáticas de geração de mensagens com explicações mais especializadas, de acordo com o contexto no qual o erro se enquadra. A partir dos resultados deste trabalho e utilizando técnicas de modelagem dinâmica de aprendizes (um campo da Inteligência Artificial aplicada à educação), também será possível em algum ponto do futuro visível dar apoio personalizado a cada perfil de aprendiz. Mais ainda, tal apoio poderá ocorrer por longos períodos de monitoramento do processo de ensino e aprendizagem, diagnosticando e remediando erros automaticamente, além de distribuir a carga cognitiva gasta pelo aprendiz na resolução de tarefas.

Tendo em vista o objetivo principal, podemos destacar como objetivos secundários os que seguem:

1. Estudar detalhes da natureza do conhecimento matemático sobre indução analítica;

2. Coletar dados sobre erros cometidos por aprendizes humanos reais que possam conter evidências de má formação de conceitos de indução analítica (além de outros tipos de erro intercalados);

3. Rotular (anotar) os dados coletados com informações confiáveis da análise humana quanto à classificação mais plausível para cada registro de erro coletado;

4. Tentativa de criar automaticamente uma taxonomia inicial de erros dentro do domínio

indução analítica e de fora dessa classe, até onde for possível constar os detalhes gerados automaticamente;

5. Definir características (quantitativas e qualitativas) formais voltadas à representação das diferenças entre expressões erradas e corretas da solução de problemas;

6. Implementar funções capazes de extrair automaticamente as características citadas no item anterior a partir de dados de entrada constituídos apenas da estrutura das expressões erradas e corretas;

7. Desenvolver um protótipo de classificador automático que dê suporte aos aspectos cognitivos envolvidos com as tarefas típicas de aprendizagem;

8. Enfatizar os aspectos referentes ao contexto de erros cometidos pelos aprendizes humanos reais por meio do ajuste de pesos do conjunto de características diante do classificador automático;

9. Validar parte desta pesquisa por meio da investigação de abordagens da literatura que tornam o ensino mais direcionado através do erro do aprendiz;

10. Disseminar o conhecimento alcançado em publicações da área de Informática na Educação e de Inteligência Artificial;

11. Disponibilizar o referido arcabouço na forma de Software Livre para que outras pesquisas também venham a contribuir com o domínio em questão.

As vantagens geradas por esta aplicação pedagógica são várias. Primeiramente na intervenção imediata, detectando e remediando os erros dentro do contexto específico em que ocorrem, o que pode facilitar um iniciante a compreender melhor a mensagem de remediação do erro. Em segundo lugar, a cada mínima interação com a ferramenta, tenderá a acarretar uma intervenção prematura do tutor, reduzindo assim, de acordo com autores renomados da área de Psicologia Cognitivista [4], o número de erros que ocorrem concorrentemente. Por último, a análise dos erros do aprendiz é altamente valiosa por identificar conhecimentos inconsistentes e incompletos. Isso pode ainda estimular o

aprendiz a novas questões e explorações, as quais levam geralmente a novas descobertas matemáticas durante a aprendizagem [8]. Estes argumentos visam ao desenvolvimento da perícia do aprendiz.

CAPÍTULO 2

RESENHA LITERÁRIA

2.1 Classificação automática sub-simbólica

Os classificadores automáticos sub-simbólicos possuem grande capacidade de adaptação e aprendizagem de acordo com o domínio em que atuam. Estes importantes atributos tornam os algoritmos de classificação capazes de manipular dados imprecisos e com ruídos em diferentes tipos de situações. Portanto, desde que as condições necessárias sejam satisfeitas (*i.e.*, um bom conjunto de dados de treinamento), é possível que uma rede neural, um exemplo de classificador sub-simbólico, seja capaz de generalizar conceitos sobre determinado conjunto de conhecimento representado formalmente, mesmo quando recebe como entrada instâncias de dados não conhecidas previamente [41].

Além da habilidade de adaptação necessária durante o ciclo de vida do sistema, Russel e Norvig em [32] citam mais duas vantagens na utilização de inteligência artifical sub-simbólica para atividades de classificação automática. Primeiramente é levantada a questão do desconhecimento do domínio do problema, havendo portanto tipos de domínios nos quais é impossível identificar todas as classes de conceitos, ou seja, criar uma taxonomia universal [36]. A outra questão é com relação ao próprio programador em relação à modelagem da solução. Dependendo da complexidade do domínio de aplicação, a tarefa de modelar o problema e sua solução exige uma capacidade tão grande de abstração que torna difícil a finalização do processo de criação de uma taxonomia.

2.2 Classificação automática simbólica

Os classificadores simbólicos, diferentemente dos classificadores sub-simbólicos, fazem uso de recursos inteligíveis aos seres humanos. Um bom exemplo de uma classe de sistemas que utilizam representação simbólica são os sistemas especialistas (*expert systems*).

São programas que possuem uma parte do conhecimento empírico de um perito em determinado domínio de conhecimento aplicado, ou seja, um domínio em que as habilidades são desenvolvidas com a prática exemplarista de solução de problemas [28]. Esses sistemas usam uma base de fatos para representar o conhecimento do domínio e uma coleção de regras para prover mecanismos de raciocíonio e classificação através de inferência lógica [29].

Um exemplo de sistema especialista famoso na literatura é o MyCIN. Sua função é aconselhar o especialista humano no diagnóstico de doenças e no tratamento de pacientes com doenças infecto-contagiosas. Possui capacidade de assimilar novo conhecimento por meio da separação modular interna entre regras de perícia e regras de inferência. Além disso, MyCIN também é capaz de explicar seu raciocínio ao usuário, fornecendo o diagnóstico e suas causas com fatores de incerteza [29, 11].

Segundo Sloman [41], a lógica simbólica é a forma mais poderosa de representação. Ela possui aplicabilidade em diversos tipos de problemas e domínios, especialmente quando as informações envolvidas possuem características como negação, condição, disjunção e quantificação. Porém, devido à capacidade dos seres humanos em utilizar diversas formas de representação, tais como linguagem natural, gestos, mapas, notação musical entre outras, a lógica simbólica possui limites. Devido a isso, novas representações surgiram para ajudar a superar tais limites, como por exemplo as redes neurais [41, 40].

2.3 Taxonomia de erros na matemática

Uma taxonomia de erros descreve um modelo capaz de classificar a maior quantidade possível de erros de determinado domínio de aplicação. Este modelo serve como base para identificar os problemas mais comuns encontrados pelos aprendizes como também auxiliar o tutor a escolher melhor as suas estratégias pedagógicas [24]. Pode beneficiar tanto a transferência de conhecimento como a remediação de erros.

Basicamente, uma taxonomia pode ser representada lógicamente como uma estrutura de dados em árvore. Cada nó da árvore representa um conceito ou uma entidade de um objetivo particular. O caminho das folhas até a raiz representa a ordenação de entidades,

dentro de categorias, e compreende um crescente nível de abstração do domínio. Cada nível da árvore representa uma categoria que possui o mesmo grau de abstração e tende a ser mutuamente exclusivo. Como um todo, a taxonomia representa um esquema de classificação não ambíguo, capaz de descrever o domínio em que estamos trabalhando [30, 23].

Uma taxonomia permite fazer inferências e descobrir relacionamentos entre as entidades [23]. A partir desses relacionamentos é possível identificar classes de entidades. Mais adiante será visto que esse nível de granulação alcançado por uma taxonomia traz melhorias aos STI. São exemplos de benefícios: Mensagens de explicação pontuais, dicas mais apropriadas e descoberta do nível de aprendizado do estudante [24].

A criação de uma taxonomia de erros de indução analítica da Matemática, partindo de um ponto muito primitivo, não possui um método universal e pode ser bastante complexa. Ela exigiria de seu criador muita perícia no tema abordado e alta capacidade de generalização sobre as classes de erros. As abordagens encontradas na literatura demonstram que as taxonomias foram criadas com base no conjunto de erros coletados empiricamente pelos pesquisadores. Após a análise desses dados foi possível a alguns pesquisadores formular um modelo capaz de representar o domínio do problema [24, 8, 30].

Borasi em [8] conduziu uma pesquisa com seus alunos cujo objetivo era entender como os erros poderiam ajudar no ensino da matemática. O domínio abordado foi sobre definições de conceitos matemáticos tais como polígono, círculos, triangulos, entre outros. Experimentos controlados foram realizados e a partir do conteúdo gerado houve uma tentativa de categorizar os erros identificados. O resultado final foi uma taxonomia de erros que o estudante poderia utilizar para enriquecer seu aprendizado.

Em sua pesquisa, Isotani, McLaren e Altman [24] desenvolveram uma taxonomia inicial sobre equívocos na área de decimais da matemática. A base para a construção da taxonomia foi uma revisão de literatura relacionada. Alguns trabalhos datavam com mais de 80 anos de existência. Grande parte desta literatura já tratava de pequenos conjuntos de equívocos relacionados. A taxonomia é utilizada para orientar a criação de um STI capaz de identificar erros dos estudantes e apoiar o ensino. A Figura 2.1 exibe uma fração

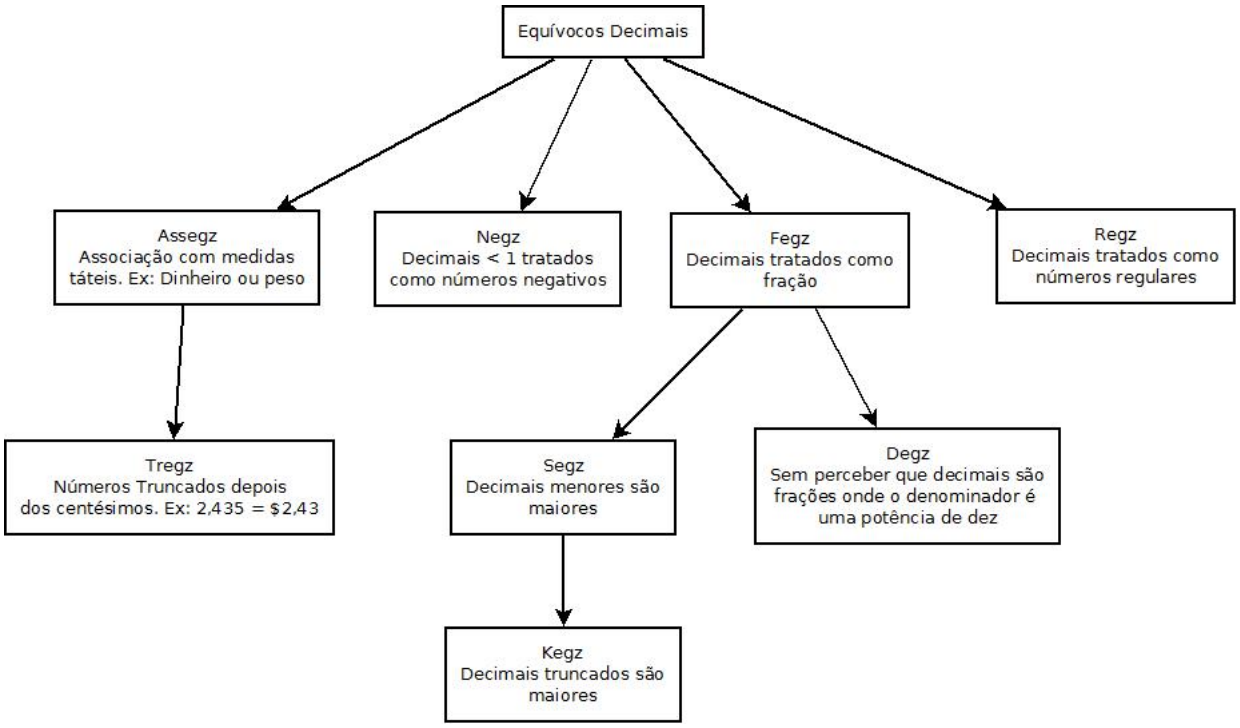

Figura 2.1: Fração da taxonomia de equívocos na área de decimais.

da taxonomia proposta por Isotani, McLaren e Altman[24].

Não foram encontrados trabalhos que tratam do domínio de erros de indução analítica. Tampouco foi encontrado um método capaz de orientar a construção de uma taxonomia desses erros. É importante também ressaltar que nenhum dos trabalhados encontrados sobre taxonomias [24, 8, 30] são da área de informática na educação matemática. Este fato representa a carência de pesquisa na área.

2.4 Modelagem de estudantes e aprendizagem de máquinas

Os sistemas tutores inteligentes são programas de computador que podem auxiliar aprendizes e professores humanos no processo educacional. Possuem caráter multidisciplinar por envolverem técnicas de áreas como ciência cognitiva, inteligência artificial e pedagogia. Através da modelagem de aprendizes são potencialmente capazes de diagnosticar e remediar dificuldades individuais [37].

A modelagem de aprendizes é constituída da representação e da interpretação do conhecimento de determinado estudante sobre um domínio em particular. É um processo que envolve a análise de todo o conhecimento teórico como também categorizar as soluções erradas do aprendiz. Isso requer uma modelagem dinâmica do processo defectivo de solução de problemas, a qual se baseia em uma biblioteca de erros típicos. O sucesso desse processo exige que a máquina tenha conhecimentos detalhados sobre o domínio, sendo um problema de alta complexidade humana de projeto e engenharia do sistema tutor [37, 36].

A modelagem de aprendizes é composta por três partes: comportamento, conhecimento base e o modelo do estudante. O comportamento é a entrada da modelagem do aprendiz e representa o retorno do estudante a determinado estímulo, como por exemplo: a resposta de alguma questão. O conhecimento base é todo o conhecimento do domínio do problema. Compreende todos os conceitos (teoria) como os erros, mantidos na biblioteca de erros. O modelo do estudante é a saída da modelagem de aprendiz e compreende a representação qualitativa do conhecimento do aprendiz sobre o domínio em particular. É realizado um confronto entre o comportamento do estudante e o conhecimento base para gerar o modelo do aprendiz.

A aprendizagem de máquina, uma subárea da inteligência artificial, já foi utilizada por alguns pesquisadores no passado para a implementação e automatização da modelagem de aprendiz. Ela enriquece o STI através da capacidade de compilação do conhecimento existente e indução de novo conhecimento. Isto equipa a máquina com recursos de predição e mantém os dados teóricos e a biblioteca de erros típicos do aprendiz atualizada. Com estes recursos é possível construir e manter o modelo de aprendiz de longo prazo, agregando informações sobre o histórico e desempenho de aprendizagem do aprendiz [37, 36, 6, 38].

A expansão da perícia do STI é realizada através do processo de aprendizagem que pode ser supervisionado ou não supervisionado. A aprendizagem supervisionada é aplicada quando os objetos a serem classificados são previamente rotulados por um supervisor. São exemplos de STI que utilizam este processo ACM [26], ASSERT [6], PIXIE [38, 39]. Já a aprendizagem não supervisionada é destinada à classificação de objetos que não pos-

suem qualquer informação sobre sua natureza. O programa MEDD é um exemplo de STI que utiliza este formato de aprendizagem [37, 36].

A construção da modelagem do aprendiz utilizando aprendizagem de máquinas possui característica particular e justifica parcialmente sua complexidade. Ao capturar a massa de dados sobre o aprendiz é bastante provável que exista um subconjunto de dados inconsistentes e incompletos. Estes dados são instâncias de "deslizes" específicos. São exemplos: enganos ou falhas ortográficas, recorrência de antigos conceitos incorretos, súbito aparecimento de novos conceitos incorretos, entre outros. Estes "deslizes" são frequentes na modelagem de aprendiz e podem surgir a qualquer momento, mesmo após uma remediação do STI [8, 36].

2.5 Diagnóstico automático de erros do aprendiz

Em teoria, os erros cometidos por aprendizes podem ser visualizados em uma estrutura de três níveis. Os níveis são, em ordem crescente de generalização, nomeados por: comportamento, conhecimento e aprendizagem. No nível comportamental são capturadas divergências entre o comportamento desejado e o comportamento do aprendiz, com base em erros sintáticos. O nível dois visa analisar as causas das divergências geradas no nível um. Através dessa análise é que classes de erros e outros tipos de equívocos são identificados. Ainda nesse segundo nível, os equívocos podem representar tanto conceitos incorretos e/ou inconsistentes, falta de conhecimento, como deslizes causados por fadiga, tédio e distração. No nível de aprendizagem o objetivo é descobrir e explicar as origens e ramificações dos erros identificados nos níveis inferiores. A Figura 2.2 exibe a estrutura de níveis dos erros descrito por Sison e Shimura [36].

A separação dos erros dos aprendizes em níveis é importante para melhor entender o tipo do erro. Dessa forma é possível definir uma abordagem especial para a correção dos erros de acordo com seu nível. Por exemplo: ao invés de remediar caso a caso todos os erros que surgem dos aprendizes, tais erros poderiam ser agrupados em uma classe

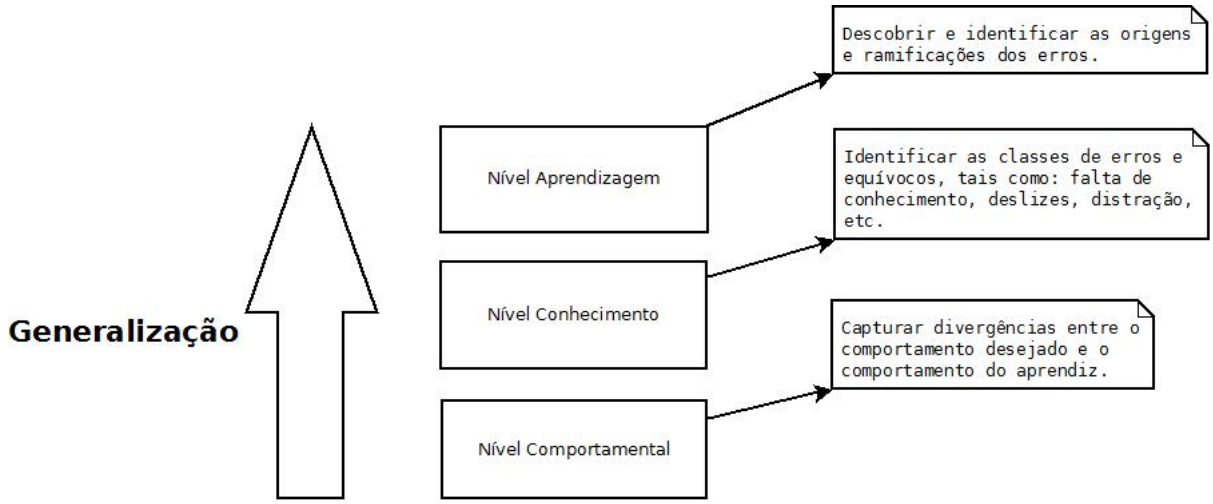

Figura 2.2: Estrutura de níveis de erros descrito por Sison e Shimura [36].

(nível dois) bem definida, identificados em todas as suas causas. Assim, uma solução poderia ser composta para esse grupo de erros. Desta forma, um conjunto inteiro de erros seria erradicado. Este é um dos principais objetivos dos especialistas de modelagem de estudantes [34].

O STI de ensino da linguagem LISP de John Anderson [5] é um bom exemplo de sistemas com diagnóstico automático de erros de aprendiz. O programa dispõe de um editor de texto apropriado para a codificação de linguagem de programação, o qual também é utilizado pelo tutor para interação com o aprendiz. Seu conhecimento é baseado em regras de produção. Não possui modelagem dinâmica de aprendizes porém utiliza um modelo ideal, que representa todo o conhecimento que supostamente o aprendiz deve adquirir. Possui uma interação passo-a-passo rica com o aprendiz através de mensagens apropriadas ao diagnóstico realizado, retorno imediato após identificação de erros e dicas e códigos para auxilio na resolução de problemas.

O ACT-R é uma teoria cognitivista cujo objetivo é predizer o comportamento humano na aquisição e processamento de conhecimento. Possui mecanismos de raciocínio, aprendizagem e rejeição de modelos não plausíveis ao comportamento humano. Por estas características torna-se útil sua aplicação na camada de aprendizagem dos níveis de erros pois é capaz de explicar o motivo do comportamento do aprendiz. Trabalha com representação de conhecimento simbólico e subsimbólico, utilizando técnicas de aprendizagem

de máquinas baseada em redes neurais. Alcançou sucesso na comunidade científica e atualmente dispõe de diversos pesquisadores espalhados pelo mundo. A utilização do ACT-R vai além de sistemas tutores inteligentes, existindo pesquisas de sua aplicação na área de interface homem-máquina [36, 1, 13].

Outra teoria que pode ser utilizada para explicar a formação dos erros é a REPAIR de VanLehn. Como o ACT-R, também pode ser aplicada na camada de aprendizagem dos níveis de erros. Ela visa explicar como os erros são gerados, porque certos erros acontecem e prever quais erros o aprendiz irá cometer no aprendizado de determinada habilidade. É embasada no fato do aprendiz chegar a um impasse na resolução de determinado problema e aplicar estratégias (repairs) para superar a situação. Para simular o cenário anterior são definidos conjuntos de resoluções incompletas, heurísticas e restrições. Através destes recursos é possível descrever a origem e ramificações dos erros. A teoria REPAIR foi implementada no programa SIERRA [36, 10].

Apesar do sucesso atingido pelo ACT-R e REPAIR, tais teorias não utilizam o histórico de comunicação com o aluno na modelagem do aprendiz. Isto acarreta no diagnóstico isolado dos erros. Por exemplo: Imagine um sistema em uma estação de metro, cuja função é exibir os dias e horários disponíveis dos trens ao ser questinado sobre os destinos existentes. Uma pessoa pode perguntar diversas vezes sobre determinado destino. O sistema irá responder sempre da mesma forma. Caso este sistema fosse equipado com a modelagem de aprendizes de longo prazo ele teria o conhecimento que uma mesma pessoa está perguntando sempre a mesma questão, e poderia fornecer outro tipo de resposta como: "Você já me perguntou isto. Será que sua dúvida não seria outra?" A modelagem de aprendizes de longo prazo pode trazer benefícios para o dignóstico automático indisponíveis sem o histórico da comunicação, como por exemplo a identificação de erros sintáticos do aprendiz (erros típicos de sub e supergeneralização e não os deslizes por falta de atenção eventual).

CAPÍTULO 3

CONCEITOS BÁSICOS DA SOLUÇÃO ADOTADA

3.1 Hipóteses simplificadoras

3.1.1 Redutibilidade da caraterização de erros analíticos

Devido à grande amplitude do campo de estudo, surge a necessidade de criar fronteiras visando determinar o que está dentro e fora da pesquisa. Neste trabalho foram estudados os erros cometidos por seres humanos durante o processo de aprendizagem de conceitos de indução analítica do campo da Matemática. Apenas os erros de indução gerados por um aprendiz durante os passos de resolução de exercícios foram levados em consideração. Optou-se por agrupar tais erros de generalização nas seguintes classes:

- Subgeneralização: Ocorre quando o aprendiz não consegue classificar determinado elemento como pertencente a determinada classe conceitual a qual ele realmente se enquadra;
- Supergeneralização: Ocorre quando o aprendiz classifica indevidamente determinado elemento como pertencente a determinada classe conceitual, quando na verdade ele não pertence;
- Miscelânea: Representa a ocorrência dos dois erros anteriores alternadamente;
- Não se aplica: Representa todos os outros erros cometidos pelo aprendiz que não se encaixam em erros analíticos, tais como os erros aritméticos, algébricos e geométricos;

Após um estudo mais aprofundado foi verificado que não é possível identificar claramente a classe Miscelânea em simples instâncias de erros. Para sua correta identificação foi preciso analisar vários erros em uma mesma sessão de resolução de exercício. Portanto

devido ao escopo desta pesquisa (análise de instâncias elementares de erros analíticos), esta classe foi retirada e sua abordagem foi postergada para trabalhos futuros.

Há várias maneiras de se exemplificar casos de erros por subgeneralização e supergeneralização. Seja a seguinte solicitação feita a dois alunos A e B: defina um triângulo. A resposta bastante completa seria: "polígono plano, fechado, de três lados, no qual qualquer lado é sempre menor que a soma dos dois outros e a soma de seus três ângulos é igual a 180 graus". Considere, porém, que o aluno A sofresse de problemas de SUBgeneralização. Assim, uma possível resposta de A seria: "uma figura de três lados iguais". De maneira oposta, considere que o aluno B sofresse de problemas de SUPERgeneralização. Assim uma possível resposta de B seria: "uma figura de três lados de qualquer tamanho e três ângulos de qualquer tamanho".

Para identificar e reconhecer os erros analíticos presentes nas expressões matemáticas foi utilizado um conjunto de características. Este conjunto de característica é específico do domínio de aplicação e são medidas extraídas de cada expressão. Com isso é possível criar um modelo (descritor) de cada erro analítico. O modelo criado traz benefícios como a eliminação de instâncias de erros que não pertencem a nenhuma classificação bem definida (*i.e.*, ruídos) e a redução da complexidade na manipulação da massa de dados formadas pelas expressões matemáticas. Na Seção 3.2 serão fornecidos maiores detalhes sobre o conjunto de características utilizadas nos estudos desta pesquisa.

3.1.2 Adoção de equivalências algébricas

As expressões matemáticas, principalmente as que possuem termos algébricos, podem possuir diversos componentes (constantes, termos algébricos e operadores). Esses componentes relacionam-se entre si através de regras aritméticas, algébricas, entre outras. Isto faz com que a manipulação ideal das expressões seja de alta complexidade, necessitando de um sistema especializado apenas para lidar com tais equivalências. Portanto para facilitar o processamento das expressões matemáticas e manter o foco da pesquisa, optou-se por uma abordagem heurística aproximada (não aprofundada) que não abrange equivalências algébricas.

As heurísticas adotadas são pontuais e representam o pré-processamento das expressões matemáticas para posterior extração das características de forma mais confiável. São elas:

- Eliminação das potências de valor 0 ou 1. Exemplos: $2^1 \Rightarrow 2$ e $2^0 \Rightarrow 1$
- Eliminação de parênteses inúteis. Exemplo: $(2) \Rightarrow 2$
- Manipulação de parênteses com potenciação ou exponenciação. Exemplos: $\left(\frac{n}{2}\right)^2 \Rightarrow \frac{n^2}{2^2}$ e $\left(\frac{n}{2}\right)^n \Rightarrow \frac{n^n}{2^n}$
- Conversão de literais com potências numéricas. Exemplo: $n^3 \Rightarrow n * n * n$

Com o detalhamento de cada característica, juntamento com a forma de extração, na Seção 3.2, o pré-processamento aplicado ficará menos sujeito a possíveis efeitos negativos de ruídos, ou seja, de dados irreleventes que, por estarem presentes na expressão, podem acabar sendo computados de alguma maneira no totais calculados pelo módulo de extração de características (ver Seção 4.1).

3.2 O conjunto de características estudadas

Nesta seção serão detalhadas as características utilizadas para identificar as classes definidas na seção anterior (subgeneralização, supergeneralização e não se aplica). Toda informação é extraída com base na resolução incorreta de exercícios, analisando duas expressões: a expressão correta (resposta do exercício) e a expressão errada (resposta do aprendiz). É importante ressaltar que todos os atributos extraídos são intrínsecos das expressões matemáticas e foram criados com o objetivo de entender a natureza destes, visando aumentar o grau de representatividade.

Os literais ou termos algébricos são os atributos de mais alta prioridade da expressão pois apresentam os pontos chaves de cada análise. Tais termos possuem prioridade sobre as outras características pois são decisivos na classificação. Um axioma criado durante esta pesquisa é:

Axioma 1: "Todo literal a mais na expressão errada caracteriza supergeneralização."

Juntamente com os termos constantes, os literais formam o conjunto de aspectos simbólicos da expressão.

Os operadores que relacionam os termos algébricos e constantes formam o conjunto de relações da expressão. Estes estão na segunda posição de prioridade na classificação. Cada operador possui uma influência diferente pois sua aplicação envolve diferentes cargas de indução analítica. Em ordem crescente de importância os operadores são organizados da seguinte forma: potenciação, multiplicação/divisão e adição/subtração/funções.

Na terceira posição de importância vem o conjunto de características semânticas e matemáticas. O conjunto semântico foi criado para representar a posição correta dos literais na expressão. Por exemplo, seja a fórmula da área do triângulo: $A = \frac{(b*h)}{2}$. A ideia deste conjunto é identificar que a base (b) e a altura (h) estão no numerador da resposta. Porém este tipo de característica é de difícil processamento quando não há manipulação de equivalências algébricas, visto que podem existir diferentes posições para o mesmo literal e com a mesma semântica. Já o conjunto de características matemáticas foi criado para tentar extrair informações em casos que não possuem literais na expressão. Sua extração é feita através da subtração das expressões e posterior análise do sinal. Na subseção 3.2.8 é dado um exemplo concreto deste tipo de característica. Foi através deste conjunto que foi criado outro importante axioma assumido durante a pesquisa:

Axioma 2: "Caso não exista literal na expressão correta e na errada, então a classificação é Não se aplica."

Ambos os conjuntos de características semânticas e matemáticas foram retirados da pesquisa por não servirem ao propósito de identificação de erros de indução analítica.

Após esta introdução sobre a conceituação dos tipos das características extraídas, as subseções seguintes irão abordar as 8 características criadas inicialmente (C1, C2, C3, C4, C5, C6, C7 e C8), detalhando a forma de extração, objetivos e exemplos. Vale ressaltar que cada característica possui diferentes níveis de aplicação da carga cognitiva para lidar com indução analítica. A numeração de 1 a 8 representa esta carga, sendo o número 1 a

mais alta carga e prosseguindo em ordem decrescente. Portanto a C1 possui uma carga cognitiva maior que a C2, que possui uma carga maior que a C3 e assim por diante.

3.2.1 Característica C1

A característica C1 aborda os literais e expressa a diferença do número de literais das expressões. Sua extração é realizada através da fórmula:

$$C1 = \mathrm{f}_1(E_2) - \mathrm{f}_1(E_1)$$

Em que E_1 é a expressão correta, E_2 é a expressão errada e $\mathrm{f}_1(E_n)$ é uma função que retorna a quantidade de literais existentes na expressão.

Vale ressaltar que, conforme as equivalências algébricas definidas na Subseção 3.1.2, é somado ao valor de C1 todas as potências numéricas presentes nos literais.

Exemplos:

Seja:

$$E_1 = \frac{l}{2^n} \quad \text{e} \quad E_2 = \frac{l}{2}$$

Logo:

$$C1 = \mathrm{f}_1(E_2) - \mathrm{f}_1(E_1)$$
$$C1 = 1 - 2$$
$$C1 = -1$$

Seja:

$$E_1 = \frac{\left(n^2 * \sqrt{3}\right)}{4} \quad \text{e} \quad E_2 = \frac{\left(n^3 * \sqrt{3}\right)}{4}$$

Logo:

$$C1 = \mathrm{f}_1(E_2) - \mathrm{f}_1(E_1)$$

$$C1 = 3 - 2$$

$$C1 = 1$$

3.2.2 Característica C2

Sendo a característica mais importante do conjunto de relações, a característica C2 representa os operadores de potenciação exponencial, ou seja, o valor da potência é uma expressão que contém uma variável. Ela expressa a diferença do número de operadores de exponenciação das expressões. Sua extração é realizada através da fórmula:

$$C2 = \mathrm{f}_2(E_2) - \mathrm{f}_2(E_1)$$

Em que E_1 é a expressão correta, E_2 é a expressão errada e $\mathrm{f}_2(E_n)$ é uma função que retorna a quantidade de operadores de exponenciação existentes na expressão.

Exemplos:

Seja:

$$E_1 = \frac{l}{2^n} \quad \text{e} \quad E_2 = \frac{l^n}{2^n}$$

Logo:

$$C2 = \mathrm{f}_2(E_2) - \mathrm{f}_2(E_1)$$

$$C2 = 2 - 1$$

$$C2 = 1$$

Seja:

$$E_1 = \frac{\left(2^n * \sqrt{3}\right)}{4} \quad \text{e} \quad E_2 = \frac{\left(2 * \sqrt{3}\right)}{4}$$

Logo:

$$C2 = \mathrm{f}_2(E_2) - \mathrm{f}_2(E_1)$$

$$C2 = 0 - 1$$

$$C2 = -1$$

3.2.3 Característica C3

A característica C3 também pertence ao conjunto de relações e aborda os operadores de potenciação numérica. Tal atributo exprime a diferença da quantidade de operadores de potenciação das expressões. Porém, como foi relatado na subseção 3.1.2, devido a equivalências algébricas a C3 foi retirada do conjunto de características utilizados na classificação. Após algumas análises, foi verificado que para o classificador é mais importante saber o número total de literais presente na expressão do que o número de operadores de potenciação. Por exemplo: considerando o termo algébrico n^2, o ideal é traduzi-lo para $n * n$. Após esta modificação a acurácia do classificador melhorou em média 4%.

3.2.4 Característica C4

É a terceira característica pertencente ao conjunto de relações e representa ao mesmo tempo os operadores de divisão e multiplicação. Ela expressa a diferença do número destes operadores presentes nas expressões. Sua extração é realizada através da fórmula:

$$C4 = \mathrm{f}_4(E_2) - \mathrm{f}_4(E_1)$$

Em que E_1 é a expressão correta, E_2 é a expressão errada e $\mathrm{f}_4(E_n)$ é uma função que tem como parâmetro uma expressão e retorna a quantidade de operadores de divisão e multiplicação existentes na expressão.

Exemplos:

Seja:

$$E_1 = \frac{l}{2^n} \quad \text{e} \quad E_2 = \frac{l * 4}{2 * n}$$

Logo:

$$C4 = \mathrm{f}_4(E_2) - \mathrm{f}_4(E_1)$$

$$C4 = 3 - 1$$

$$C4 = 2$$

Seja:

$$E_1 = \frac{\left(2^n * \sqrt{3}\right)}{4} \quad \text{e} \quad E_2 = 2^n * \sqrt{3}$$

Logo :

$$C4 = \mathrm{f}_4(E_2) - \mathrm{f}_4(E_1)$$

$$C4 = 1 - 2$$

$$C4 = -1$$

3.2.5 Característica C5

A característica C5 é a quarta e última do conjunto de relações da expressão. Ela aborda ao mesmo tempo os operadores de adição e subtração. Seu objetivo é exibir a diferença da quantidade destes operadores contidos nas expressões. Sua extração é realizada através da fórmula:

$$C5 = \mathrm{f}_5(E_2) - \mathrm{f}_5(E_1)$$

Em que E_1 é a expressão correta, E_2 é a expressão errada e $\mathrm{f}_5(E_n)$ é uma função que retorna a quantidade de operadores de adição e subtração existentes na expressão.

Exemplos:

Seja:

$$E_1 = \frac{l}{2^n} \quad \text{e} \quad E_2 = \frac{l+1}{n}$$

Logo:

$$C5 = \mathrm{f}_5(E_2) - \mathrm{f}_5(E_1)$$

$$C5 = 1 - 0$$

$$C5 = 1$$

Seja:

$$E_1 = \frac{\left(2^n + 1 * \sqrt{3}\right)}{4} \quad \text{e} \quad E_2 = \frac{\left(2^n * \sqrt{3}\right)}{4}$$

Logo:

$$C5 = \mathrm{f}_5(E_2) - \mathrm{f}_5(E_1)$$

$$C5 = 0 - 1$$

$$C5 = -1$$

3.2.6 Característica C6

Esta característica representa os termos constantes e expressa a diferença do número de constantes presentes nas expressões. Após a execução de procedimentos que exibem a qualidade das características, foi verificado que a C6 não era utilizada pelo classificador. Logo ela não determina se o erro em questão é de subgeneralização ou supergeneralização. Porém, como um dos objetivos desta pesquisa é identificar agrupamentos dentro das classes principais, esta característica foi mantida visando sua futura utilização por parte de outros classificadores. Sua extração é realizada através da fórmula:

$$C6 = \mathrm{f}_6(E_2) - \mathrm{f}_6(E_1)$$

Em que E_1 é a expressão correta, E_2 é a expressão errada e $\mathrm{f}_6(E_n)$ é uma função que retorna a quantidade de constantes existentes na expressão.

Exemplos:

Seja:

$$E_1 = \frac{l}{2^n} \quad \text{e} \quad E_2 = \frac{l+1}{2^n}$$

Logo:

$$C6 = \mathrm{f}_6(E_2) - \mathrm{f}_6(E_1)$$

$$C6 = 2 - 1$$

$$C6 = 1$$

Seja:

$$E_1 = \frac{\left((2^n + 1) * \sqrt{3}\right)}{4} \quad \text{e} \quad E_2 = 2^n * \sqrt{3}$$

Logo:

$$C6 = \mathrm{f}_6(E_2) - \mathrm{f}_6(E_1)$$

$$C6 = 2 - 4$$

$$C6 = -2$$

3.2.7 Característica C7

A característica C7 pertence ao conjunto de características semânticas da expressão. É do tipo booleano e visa expressar se o significado da posição dos literais nas expressões correta e errada são equivalentes. Em caso afirmativo o valor é verdadeiro, caso contrário é falso. Sua extração é complexa pois necessita manipular equivalências algébricas. Após análises foi verificado que esta característica não serve ao propósito de identificação de erros de indução analítica, ou seja, caso a posição semântica dos literais das expressões sejam divergentes, isto não acarreta erros de subgeneralização ou supergeneralização. Logo ela foi removida do conjunto de características utilizados na classificação.

Exemplos:

Seja:

$$E_1 = \frac{l}{2^n} \quad \text{e} \quad E_2 = \frac{n}{2^l}$$

Logo:

$$C7 = Falso$$

Seja:

$$E_1 = \frac{\left(2^n * \sqrt{3}\right)}{4} \quad \text{e} \quad E_2 = 2^n * \sqrt{3}$$

Logo:

$$C7 = Verdade$$

3.2.8 Característica C8

Por fim a característica C8 é a responsável por representar os erros que não possuem literais nas expressões correta e errada. É do tipo booleano e expressa o sinal do resultado da subtração das expressões. Em caso de sinal positivo seu valor é *verdadeiro*, caso contrário é *falso*. Pertence ao conjunto de características matemáticas da expressão e foi criada antes da definição do axioma 3.2. Logo, como o próprio axioma define, sua existência é irrelevante. Não há informações nestas expressões para identificar erros analíticos, ou seja, a natureza do erro pode ser aritmética, algébrica, geométrica, etc.

Exemplos:

Seja:

$$E_1 = \frac{9}{2} \quad \text{e} \quad E_2 = \frac{9}{3}$$

Logo:

$$C8 = \frac{9}{3} - \frac{9}{2}$$

$$C8 = -1,5$$

$$C8 = Falso$$

Seja:

$$E_1 = \frac{9}{3} \quad \text{e} \quad E_2 = \frac{9}{2}$$

Logo:

$$C8 = \frac{9}{2} - \frac{9}{3}$$

$$C8 = 1,5$$

$$C8 = Verdade$$

CAPÍTULO 4

ASPECTOS DE IMPLEMENTAÇÃO

Neste capítulo serão abordados todos os aspectos relacionados à implementação do extrator de características, à coleta e rotulação da base de dados utilizada na pesquisa e, por fim, às classificações supervisionadas e não supervisionadas feitas com um software de apoio chamado WEKA.

4.1 O extrator de características

A finalidade do extrator de características é extrair os cinco atributos, definidos na subseção 3.2, que fazem parte do descritor dos erros analíticos. As características são extraídas através de expressões regulares que são responsáveis pelo processamento de identificação de literais, operadores de potenciação, multiplicação, divisão, adição, subtração e constantes. A cada extração finalizada, é gerada uma instância do erro com uma nova representação.

O extrator foi desenvolvido na linguagem PHP [31]. A cada execução o script PHP recebe como entrada duas expressões (Expressão correta e errada) no formato de texto (*i.e.*, String) e tem como saída um arquivo no formato ARFF [16] pronto para ser utilizado no framework WEKA [16].

Existe ainda um pré-processamento das expressões recebidas com o objetivo de melhor prepará-las para a extração. Para tal, são executados os procedimentos:

- Eliminação de espaços em branco;
- Eliminação das potências presentes em termos constantes e algébricos no valor de 0 ou 1;
- Adequação de funções, tais como *sqrt(), sen(), cos()*, etc., para não influênciar a contagem de literais;

- Eliminação de parênteses que não indicam nenhuma prioridade, como no exemplo $(2) = 2$.

Durante a etapa de extração há um tratamento particular de parênteses das expressões que são seguidos de operadores de potenciação (Exemplo: $\left(\frac{n}{2}\right)^3$). Neste caso foi utilizada uma função recursiva que rearranja o trecho em questão para um formato mais adequado ao extrator (Exemplo: $\left(\frac{n}{2}\right)^3$ é mapeado para $\frac{n^3}{2^3}$).

Após o término do processo, existe a fase de atribuir peso à instância gerada pelo erro. Isto ocorre para que o algoritmo de aprendizagem leve em consideração as instâncias com maiores pesos pois elas são mais importantes. A atribuição de peso é realizada se alguma das condições seguintes forem atendidas:

- A classe do erro for Subgeneralização e o valor da característica C1 for menor que 0
- A classe do erro for Supergeneralização e o valor da característica C1 for maior que 0

4.2 A base de dados reais

Esta seção retrata a captura e posterior rotulação da base de dados empregada nesta pesquisa. Vale ressaltar que a qualidade e o tamanho da base influencia diretamente na classificação e nos resultado obtidos. Apesar da quantidade de erros coletados ser relativamente pequena para ser utilizada em sistemas de classificação, julgamos que a qualidade dos dados é suficiente para conduzir o estudo e obter resultados plausíveis.

4.2.1 Coleta dos dados

A coleta dos dados foi efetuada através do software Progressões Geométricas em Fractais do projeto Condigital do MEC, com participação das seguintes instituições consorciadas do Paraná [18, 17, 14]: UFPR, LACTEC, CETEPAR e UEL. Neste software há uma sequência de 6 exercícios, cada um deles contendo vários subitens, que necessitam de perícia analítica para solução. Ele foi desenvolvido com o objetivo de oferecer uma dificuldade crescente ao aprendiz, justamente para que sua capacidade indutiva seja explorada

gradativamente. A equipe de desenvolvimento deste sistema também faz parte do grupo de pesquisadores da UFPR que estuda erros analíticos cometidos por aprendizes humanos, o que naturalmente levou a uma integação maior com o presente trabalho de Mestrado, sendo a escolha deste software a melhor opção encontrada para coletar os dados.

O período de captura dos dados durou em torno de 15 dias e teve a participação de 34 aprendizes com nível de escolaridade de início da graduação (primeiro ano de Engenharia), ou seja, recém saídos do ensino médio. Ao término da coleta, foram obtidos um total de 752 erros. Os erros estão armazenados em uma base relacional e disponíveis para utilização de outros pesquisadores.

4.2.2 Rotulação dos dados

A rotulação dos dados é um procedimento de análise e classificação de cada instância de erro individualmente. Ela é necessária devido à classificação supervisionada que utiliza o rótulo aplicado na sua etapa de aprendizagem. No caso desta pesquisa foi um processo longo, realizado e revisado pelos próprios pesquisadores da UFPR, o qual teve os 752 erros analisados manualmente e rotulados com as classes de erros analíticos (Subgeneralização, Supergeneralização e Não se aplica). Esta fase foi importante para melhorar o entendimento do domínio do problema como também para enunciar os axiomas citados no capítulo 3.

Ao término do processo de rotulação, a situação dos erros obtida foi a seguinte:

- Número total de erros rotulados como Subgeneralização: 398 erros
- Número total de erros rotulados como Supergeneralização: 126 erros
- Número total de erros rotulados como Não se aplica: 228 erros

4.3 Aprendizagem e classificação sub-simbólica

Esta seção aborda a classificação de instâncias de erros analíticos cujo formato é semelhante ao da etapa de classificação de um sistema de reconhecimento de padrões. O

processo é dividido em duas fases: treinamento e testes. A fase de treinamento é o momento que o algoritmo passa pelo estágio de aprendizagem através do processamento dos dados extraídos pelo extrator de características. É nesse passo que os pesos nas instâncias de erros são utilizados. A fase de testes qualifica a instância do erro atribuindo uma das classes de Subgeneralização, Supergeneralização ou Não se aplica.

Neste momento é importante ressaltar que a escolha de utilizar um sistema de classificação com base em modelo ao invés de uma taxonomia foi devido à pouca estruturação do conhecimento que se tem atualmente sobre o domínio de aplicação de erros de indução analítica. Uma taxonomia requer uma forte estruturação do conhecimento do domínio e portanto seria um paradoxo adotá-la nesta etapa. Por isso, os classificadores com base em aprendizagem de máquina foram escolhidos como o meio mais adequado de prosseguir na classificação.

Há diversos algoritmos de classificação nos planos simbólico e sub-simbólico. A diferença entre ambos os planos está na representação do conhecimento sub-simbólico, o qual possui um formato não inteligível ao ser humano. Nesta pesquisa foi escolhido utilizar o plano sub-simbólico, mais especificamente redes neurais. Os motivos são:

- As redes neurais possuem baixa dependência de domínio e portanto pode ser aplicada em vários tipos de problemas.
- A classificação realizada pelas redes neurais (MLP) é não linear, o que é uma vantagem quando o domínio da aplicação que se está trabalhando é pouco conhecido e a separação linear dos dados é uma incógnita;
- Tentar demonstrar a utilidade da inteligência artificial sub-simbólica na modelagem dinâmica de aprendizes, o que segundo Sison e Shimura [36] é pouco explorado nesta área.

4.3.1 O software de apoio WEKA

O WEKA é um *toolkit* de aprendizagem de máquinas e mineração de dados e foi desenvolvido na universidade de Waikato, na Nova Zelandia. É distribuido sob a licença

pública GNU e pode ser utilizado em pesquisas, educação e aplicações. Possui diversos tipos de classificadores supervisionados e não supervisionados e funcionalidades úteis para pré-processamento de dados, gráficos e ambiente de comparação de algoritmos de aprendizagem. Atualmente é um projeto ativo, com bastante documentação disponível e serve de base para diversos outros projetos.

É amplamente utilizado pela comunidade científica pela sua eficiência e facilidade de integração. Seu funcionamento é bastante simples e foi integrado nesta pesquisa através do arquivo de dados no formato ARFF [16]. O WEKA foi escolhido para suprir a necessidade de implementação da rede neural. Nas subseções 4.3.2 e 4.3.3 serão exibidos maiores detalhes sobre a utilização deste *toolkit*.

4.3.2 O classificador supervisionado

Conforme dito na seção 4.3 o classificador escolhido para esta pesquisa foi uma rede neural, mais especificamente a rede MLP (Multilayer Perceptron). No WEKA este classificador possui uma grande variedade de parâmetros e opções, além de utilizar aprendizagem supervisionada e o mecanismo *Back Propagation* durante a fase de treinamento. A aprendizagem ocorre através da alteração do peso das conexões entre os nós da rede após cada dado ser processado. Esta alteração é baseada na comparação entre o resultado esperado e o valor do erro gerado na saída da rede. O mecanismo de *Back Propagation* é responsável por propagar esta alteração desde a saída até entrada da rede, interferindo positivamente no processo de aprendizagem.

Dos 752 erros coletados, ver seção 4.2, 376 (50%) instâncias fizeram parte do conjunto de treinamento e a outra metade do conjunto de testes. Para preencher estes conjuntos, os dados foram selecionados aleatoriamente. Os resultados foram obtidos através da média de 5 execuções visando eliminar possíveis ruídos e condicionamento dos dados. A cada classificação foram executados os seguintes passos:

1. Obter o conjunto de dados de treinamento e testes aleatoriamente

2. Gerar o arquivo ARFF através do extrator de características

3. Abrir no WEKA o arquivo ARFF

4. Selecionar a aba Classify

5. Escolher o classificador Function, Multilayer Perceptron

6. Clicar na opção *Supplied test set* e selecionar o arquivo de testes

7. Rodar o classificador

No final da execução é possível gerar gráficos (ver Subseção 4.3.2.1) e obter um resumo geral da classificação realizada pela rede MLP (Anexo A). Neste resumo estão dois conceitos que são importantes na análise dos resultados: acurácia e matriz de confusão. A acurácia é o valor percentual obtido da divisão entre o total de elementos classificados corretamente sobre o total de elementos utilizados no procedimento. A matriz de confusão exibe o resultado final do classificador, dispondo na diagonal principal da matriz os elementos classificados corretamente e nos demais os elementos classificados incorretamente.

Outros classificadores foram testados no domínio de aplicação desta pesquisa, visto a grande facilidade de acesso a diferentes algoritmos dispostos pelo WEKA. O procedimento de execução utilizado com estes classificadores foi o mesmo adotado para a rede MLP. A acurácia média e o desvio padrão das cinco médias estão ilustrados na tabela 4.3.2.

Classificador	Acurácia Média	Desvio Padrão (5 Médias)
BayesNet	80%	1,49
KStar	86%	1,53
Decision Table	71%	4,73
J48 (C4.5)	86%	1,15

4.3.2.1 Resultados obtidos

A acurácia média obtida foi 85% e desvio padrão de 1,97, referente aos 5 valores das médias. Isto corresponde a, em média, 329 instâncias de erros classificadas corretamente e 47 classificadas incorretamente.

A acurácia máxima obtida pela rede neural MLP foi de 87%. Para ter uma melhor visualização desta classificação, segue a matriz de confusão gerada pelo classificador:

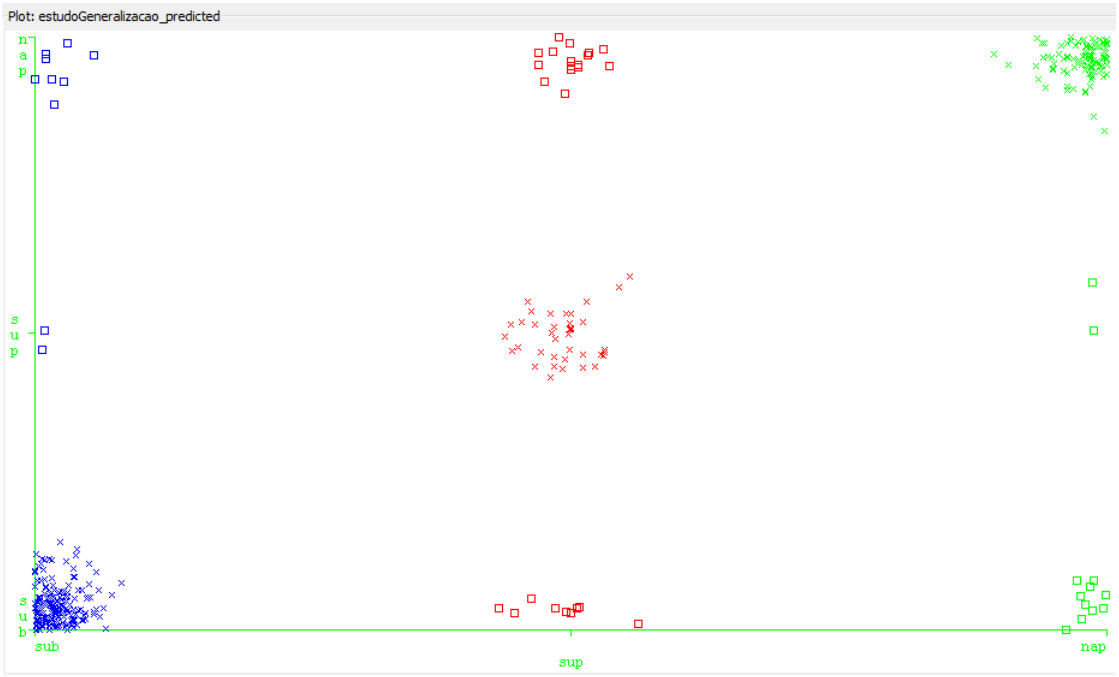

Figura 4.1: Classificação da rede neural MLP (Eixo X: rótulo da classe. Eixo Y: classificação feita pela rede MLP.)

Classificação	a	b	c
a = sub	189	2	8
b = sup	9	38	16
c = nap	10	2	102

Na figura 4.1 é possível analisar visualmente a classificação realizada pela rede neural MLP.

Para maiores detalhes sobre a classificação da rede MLP consultar o anexo A.

As figuras 4.2, 4.3 e 4.4 ilustram a área abaixo da curva ROC (Receiver Operating Characteristic) [19]. Esta curva exibe o desempenho do classificador, com relação a uma determinada classe, sendo plotado de acordo com as classificações corretas e incorretas. Segue os resultados obtidos:

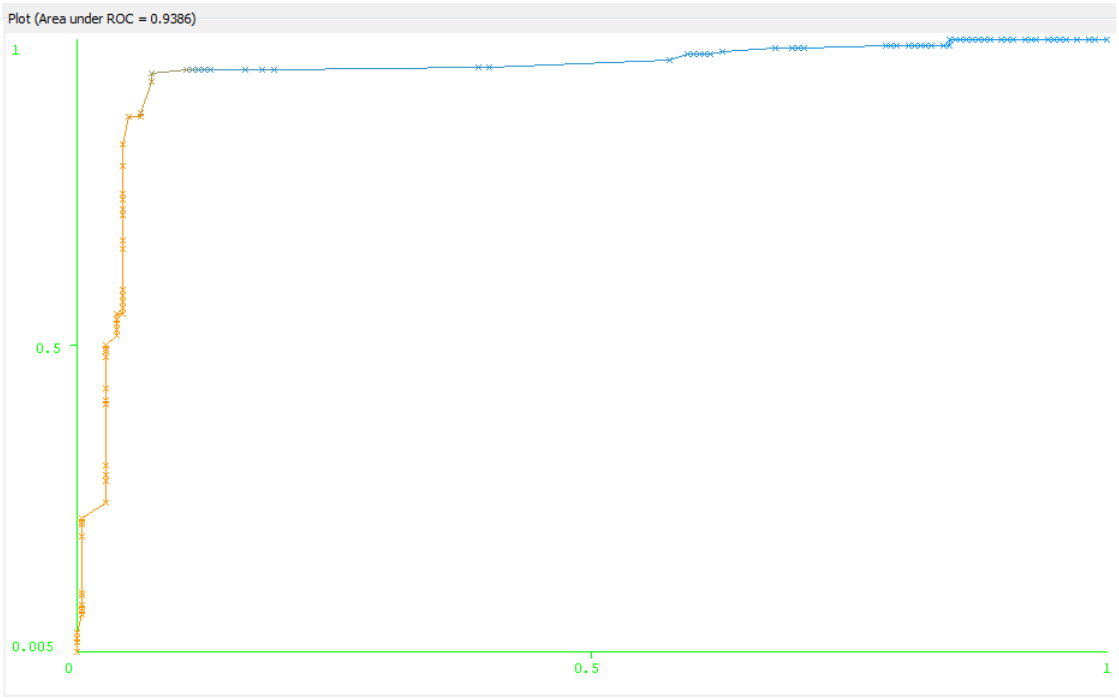

Figura 4.2: Classe de Subgeneralização, com a área abaixo da curva ROC de 93%. (Eixo X: False Positive e eixo Y: True Positive.)

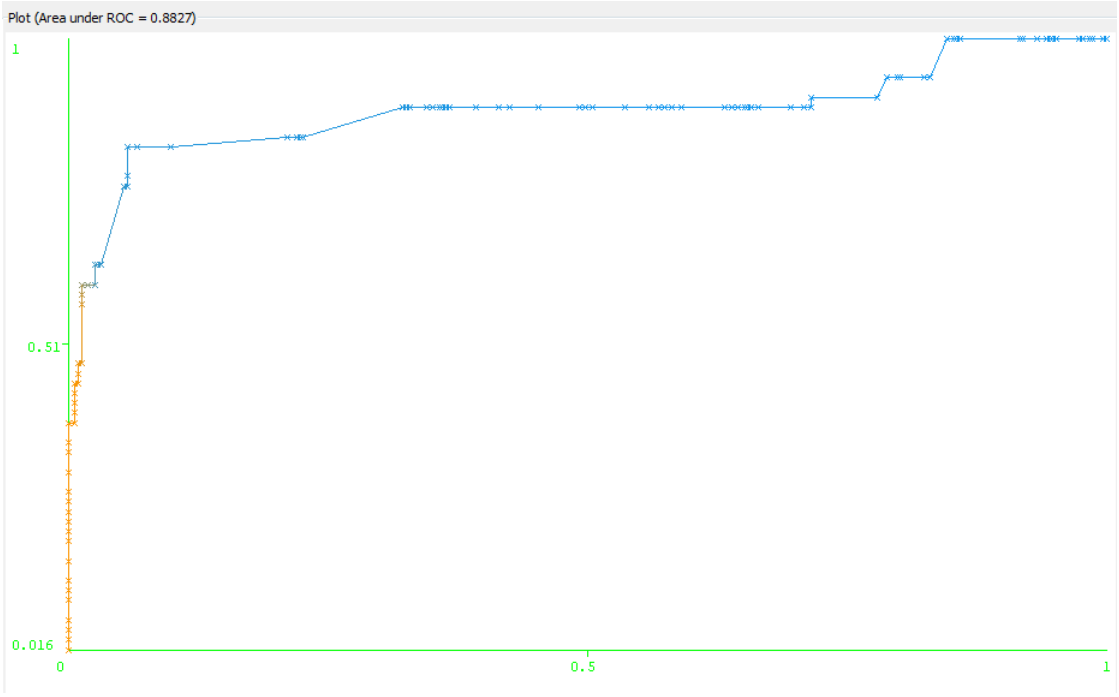

Figura 4.3: Classe de Supergeneralização, com a área abaixo da curva ROC de 88%. (Eixo X: False Positive e eixo Y: True Positive.)

4.3.2.2 Análise dos resultados

Os resultados obtidos foram considerados satisfatórios devido aos seguintes motivos:

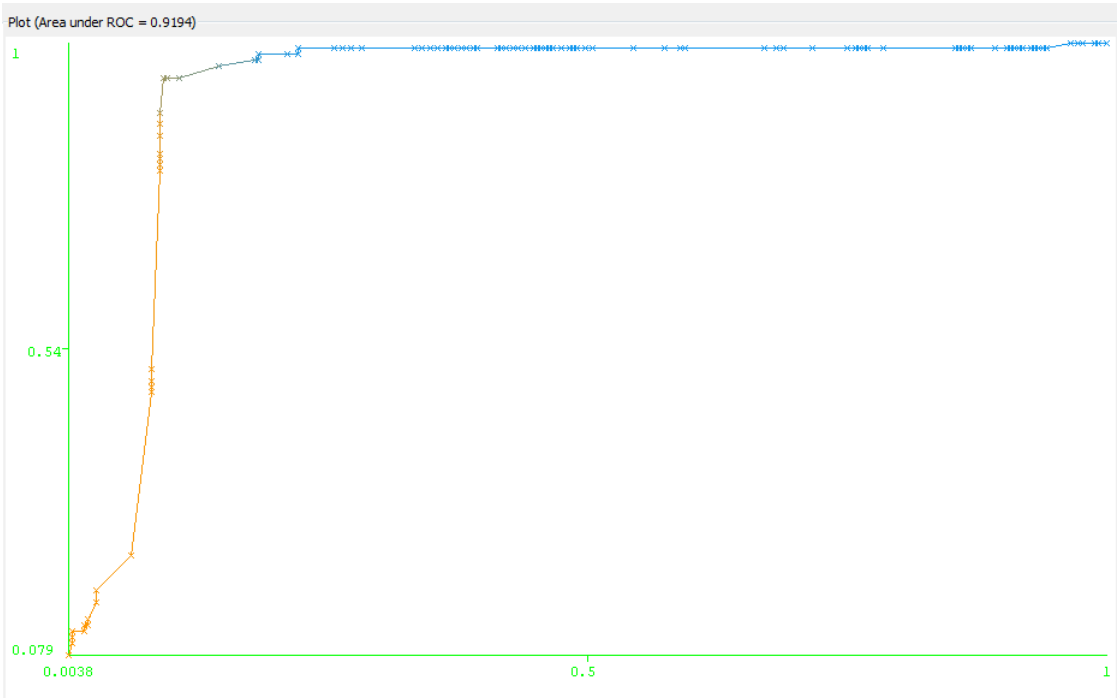

Figura 4.4: Classe de Não se aplica, com a área abaixo da curva ROC de 91%. (Eixo X: False Positive e eixo Y: True Positive.)

- O domínio de aplicação é inerentemente difícil para humanos classificarem (se estes receberem as mesmas entradas do classificador, i.e., duas expressões, sendo uma certa e uma errada);

- Do ponto de vista pedagógico (e talvez também didático), a recuperação conceitual do aprendiz humano que comete um erro depende muito mais de informações sobre a certeza de ocorrência do erro do que da classificação de tal erro como instância de sub- ou supergeneralização (i.e., a máquina só tentará fazer um pouco a mais pelo aprendiz quando for um caso confirmado de erro recorrente, inclusive aproximando o professor humano desse aprendiz por meio de um ambiente integrado);

- Esta pesquisa é apenas um passo em um longo e importante estudo que ainda deve ser trilhado no campo de aprendizagem de indução analítica;

- Como futuro trabalho, será encadeada ao classificador uma aplicação de monitoramento de longo prazo de um aprendiz humano (o monitorador é um modelador dinâmico que poderá acompanhar o aprendiz com realimentação constante de frequência de erros para que a máquina forme hipóteses mais exatas sobre as fragi-

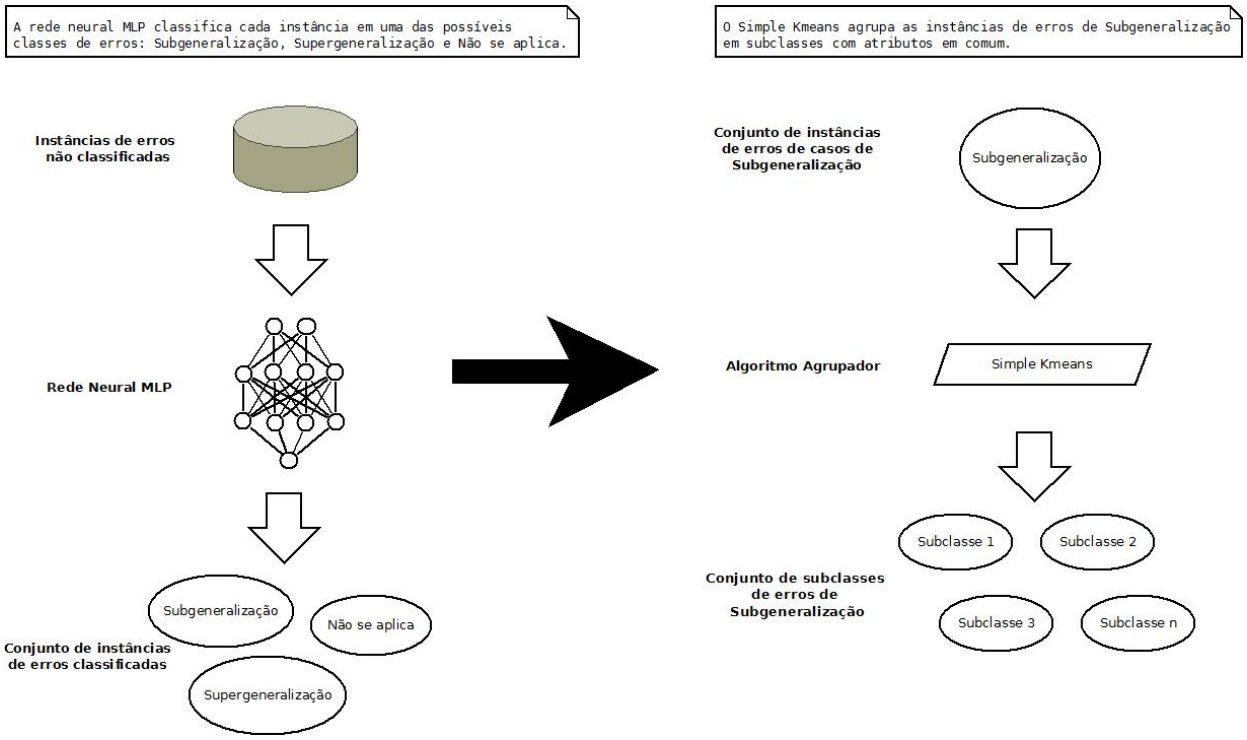

Figura 4.5: Subclasses de erros

lidades conceituais desse aprendiz);

4.3.3 O classificador não supervisionado

Após a classificação supervisionada finalizada com sucesso, todas as instâncias de erros rotuladas com uma das três classes utilizadas na Seção 3.1.1pelo classificador, surge a necessidade de descobrir a existência de subclasses de erros dentro das classes principais. A figura 4.5 ilustra a ideia.

Ao começar a explorar este domínio de aplicação, os pesquisadores possuiam muito pouco conhecimento prévio sobre as subclasses, logo não foi possível rotular as instâncias de erros como foi feito na Seção 4.3.2. Este fato levou à escolha de uma ferramenta capaz de suprir esta falta de informação e optou-se por utilizar classificadores não supervisionados. Estes agrupadores (*i.e.*, clustering), como são frequentemente conhecidos, são capazes de aprender através da observação das instâncias, agrupando-as conforme as semelhanças de determinadas características.

O agrupador utilizado nesta pesquisa foi o **Simple KMeans**. Disponibilizado pela WEKA na seção de *Cluster*. Este classificador utiliza o clássico algoritmo de agrupamento chamado *KMeans*. Assim como os outros classificadores disponíveis, possui diversas opções e parâmetros para execução. Os parâmetros mais importantes a ressaltar são o cálculo da distância Euclidiana e o número de agrupamentos a identificar.

A saída gerada pela rede MLP, a qual teve acurácia máxima, foi separada em dois arquivos ARFF diferentes, cada qual contendo apenas os dados de Subgeneralização e Supergeneralização. Isto foi possível através de um script de apoio, o qual também é capaz de permitir a visualização da expressão matemática em alto nível para estudo humano da classificação obtida. O arquivo ARFF de Subgeneralização possui 208 instâncias de erros contra 42 do arquivo de Supergeneralização. A classificação não supervisionada foi executada com 50% dos dados para treinamento e os outros 50% para testes e teve os seguintes passos:

1. Obter o resultado gerado pela classificação supervisionada
2. Gerar um novo arquivo ARFF contendo apenas dados da classe em questão (Exemplo: apenas instância de erros classificadas como Subgeneralização)
3. Abrir no WEKA o arquivo ARFF
4. Selecionar a aba Cluster
5. Escolher o agrupador Simple Kmeans
6. Clicar no nome do agrupador e escolher o número de agrupamentos
7. Selecionar o modo de agrupamento *Percentage split* e inserir o valor de 50%
8. Rodar o classificador

No final da execução foi possível gerar gráficos (ver ver Subseção 4.3.2.1) e obter um resumo geral do agrupamento realizado pelo *Simple Kmeans* (ver Seção B e C).

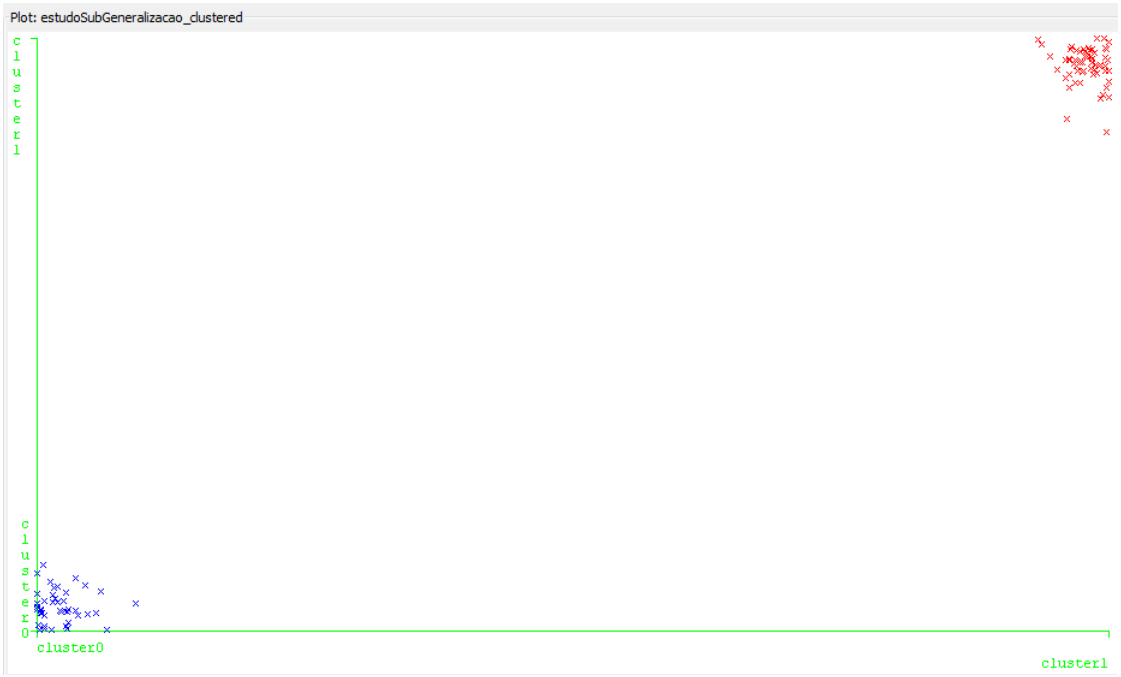

Figura 4.6: Agrupamento feito pelo Simple Kmeans no conjunto de dados Subgeneralização. Cada ponto representa uma instância de erro analítico.

4.3.3.1 Resultados obtidos

Para selecionar o melhor agrupamento foi utilizada a métrica *DB-Index* (Davies and Bouldin Index) [15]. Este é um método que calcula a dispersão dos elementos e visa obter os agrupamentos mais compactos e longe um dos outros. Para ambas as classes estudadas (Subgeneralização e Supergeneralização) o melhor resultado obtido foi de dois grupos distintos. O valor do DB-Index para o conjunto de dados de Subgeneralização foi de 4,4 e de Supergeneralização foi 0,66.

Para o conjunto de instâncias de erros de Subgeneralização, o algoritmo fez nove iterações até atingir o resultado final de: 59% de itens inseridos no **Grupo A** e 41% inseridos no **Grupo B**. Isto corresponde a 61 elementos pertencentes ao **Grupo A** contra 43 do **Grupo B**, totalizando 104 instâncias de erros.

Na figura 4.6 é possível analisar os agrupamentos gerados pelo Simples Kmeans para as instâncias de erros de Subgeneralização. Note que as figuras 4.6 e 4.7 possuem eixos adimensionais, isto é, seus valores e unidades dos eixos são resultados de rebatimentos de várias dimensões do mundo conceitual sobre uma única métrica artificial.

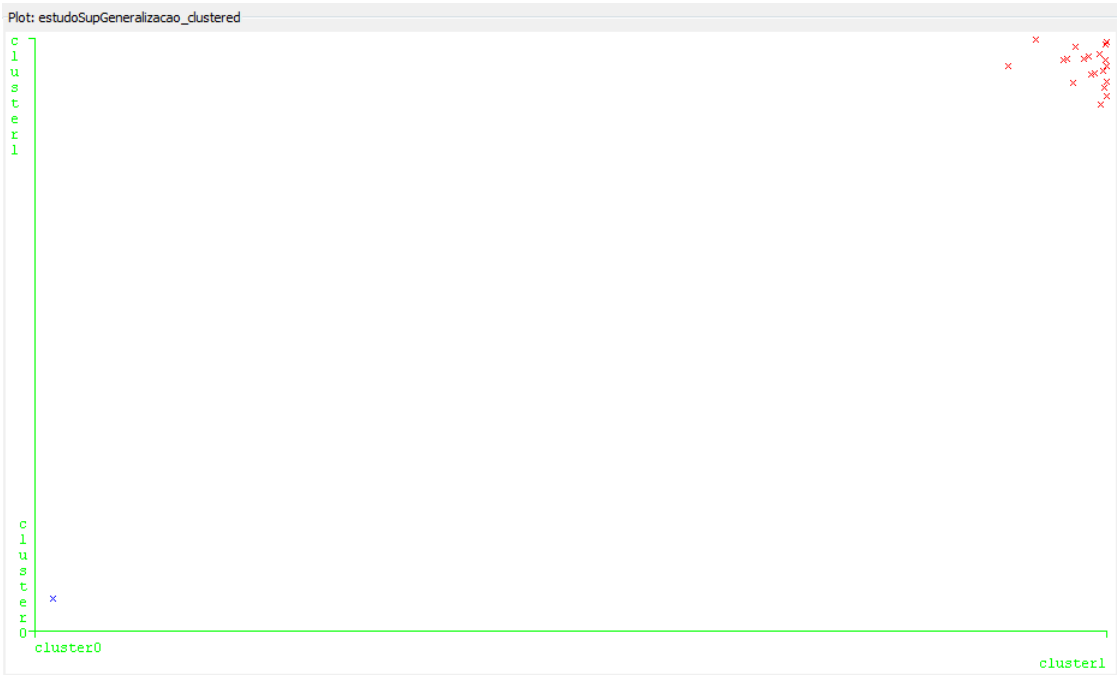

Figura 4.7: Agrupamento feito pelo SimpleKmeans no conjunto de dados Supergeneralização. Cada ponto representa uma instância de erro analítico.

Para maiores detalhes sobre este agrupamento consultar o anexo B.

Para o conjunto de instâncias de erros de Supergeneralização, o algoritmo fez duas iterações até atingir o resultado final de: 5% de itens inseridos no **Grupo A** e 95% inseridos no **Grupo B**. Isto corresponde a 1 elemento pertencente ao **Grupo A** contra 20 do **Grupo B**, totalizando 21 instâncias de erros.

Na figura 4.7 é possível analisar os agrupamentos gerados pelo Simples Kmeans para as instâncias de erros de Supergeneralização.

Para maiores detalhes sobre este agrupamento consultar o anexo C.

4.3.3.2 Análise dos resultados

Apesar de ser dois a quantidade de agrupamentos ideal indicada pelo *DB-Index*, para ambas as classes estudadas, tal informação representa apenas um índice de análise numérica. Esse artefato é importante e pode ser útil durante a análise, porém não se pode esquecer que a finalidade de médio e longo prazo do contexto geral deste projeto, é mo-

nitorar aprendizes humanos da forma mais granular possível. Portanto, necessariamente outros itens devem ser levados em consideração e ponderados para a escolha dos melhores agrupamentos.

Acreditamos que estes itens são exclusivos do domínio da aplicação. Nesta pesquisa é importante compreender as propriedades matemáticas das instâncias de erros pertencentes a um grupo. Seja qual for a natureza do atributo (*e.g.*, aritmético, algébrico, geométrico ou uma ocorrência híbrida), o mapeamento, com início nas características (C1, C2, C4, C5, C6) e término na expressão matemática (instância de erro gerada pelo aprendiz) deve ser traçado para efetivamente elucidar o motivo do agrupamento.

Segue um exemplo com os casos de Subgeneralização para ilustrar a questão:

- Considere a instância de erro $\frac{l}{2} * \sqrt{3}$ para a qual a expressão correta é $\frac{l^2 * \sqrt{3}}{4}$, pertencente ao **Grupo A**. A justificativa do erro é a falta de literal.

- Novamente considere uma outra instância de erro $\frac{l^2}{2}$ sendo que sua expressão correta correspondente é $\frac{l^2 * \sqrt{3}}{4}$, pertencente ao **Grupo B**. A explicação do erro é a falta de operadores de multiplicação/divisão.

- Agora seja uma terceira instância de erro $\frac{l}{2} * 9$, cuja expressão correta é $l * 9$, novamente pertencente ao **Grupo A**. A razão do erro também é a falta de operadores de multiplicação/divisão.

Nota-se que o segundo e o terceiro exemplo possuem a mesma justificativa de erro. Logo entende-se que deveriam pertencer ao mesmo Grupo mas não é o que acontece. Portanto o *DB-Index* indicado não é bom o suficiente para agrupar as expressões acima. Porém, ele foi o melhor índice obtido de maneira completamente automática. Apesar desta conclusão, acredita-se que o mesmo pode ser utilizado como um limitante inferior em termos de quantidade de especializações de mensagens de explicação a serem geradas dinamicamente para um aprendiz.

CAPÍTULO 5

CONCLUSÃO

A identificação e explicação matemática das subclasses encontradas compõem uma forte base para o futuro projeto de modelagem dinâmica de aprendizes. Acredita-se assim na factibilidade de rastreamento contínuo do conhecimento sobre a indução analítico do aprendiz por meio de suas subclasses de Subgeneralização e/ou Supergeneralização. Isso poderia permitir que esse aprendiz, de fato, pudesse até corrigir seus equívocos de maneira autônoma. Outro ponto positivo é o auxílio ao tutor humano tendo a máquina como mediadora entre ele e os aprendizes. O tutor poderia utilizar o diagnóstico realizado pelo classificador de instâncias de erros para visualizar a subclasse de erro mais frequente (e provável) do aprendiz, assim como as eventuais ramificações desses erros (ou de um grupo deles). Isso permitiria, por exemplo, uma intervenção direcionada do tutor sobre os aprendizes de maneira adequada e bem embasada.

Um possível exemplo de caso real de utilização do material produzido nesta pesquisa na modelagem dinâmica de aprendizes seria: seja a resposta correta $\frac{2}{2^n}$ e a sequência de erros produzida por um aprendiz em uma sessão de resolução de exercícios: $\frac{1}{n}$, $\frac{1}{2}$, $\frac{n}{2}$ e $2 * n$. Provavelmente o aprendiz possui dificuldade de generalizar a solução. Isto se verifica porque, apesar da inclusão correta do literal, ele não foi capaz de generalizar a exponenciação. Logo o classificador, já com as características extraídas e a rede neural treinada, identificaria que este aprendiz sofre de problemas de Subgeneralização.

Com o foco do problema sendo Subgeneralização, os erros cometidos passariam pelo processo de classificação em subclasses, obtendo maiores detalhes sobre possíveis motivos e correções. Suponhamos que exista uma subclasse com conhecimento e mensagens de suporte sobre eventuais dificuldades de exponenciação. Cada erro seria classificado como pertencente à respectiva subclasse e uma possível mensagem de explicação seria dada ao aprendiz: “Está faltando um operador de potenciação. Tente novamente.”. Como o

aprendiz está em uma sessão de resolução do exercício, a cada nova tentativa o sistema teria o conhecimento das mensagens de suporte já exibidas e poderia especializá-las: “Você tinha inserido o literal n e agora o retirou. Este termo é necessário e ainda falta especificar o operador de potenciação.”. Este diálogo seguiria até o eventual fornecimento da resposta correta.

Há diversos pontos de interação entre o aprendiz, ambiente de aprendizagem e a modelagem dinâmica que podem ser utilizados para enriquecer o diagnóstico e as mensagens de suporte. Estes pontos foram chamados de meta-características e representam os elementos que fazem parte do contexto da resolução, tais como: percentual de acerto, percentual de erro, índice de dificuldade aritmética, índice de dificuldade geométrica, descrição de detalhes do exercício em questão, entre outros. Porém pouco foi feito nesta pesquisa para desenvolvê-los, ficando assim como possíveis trabalhos futuros.

BIBLIOGRAFIA

[1] ACT-R. Act-r. http://act-r.psy.cmu.edu/.

[2] S. Ainsworth. Deft: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16(3):183–198, 2006.

[3] S. Ainsworth, P. Bibby, e D. Wood. Examining the effects of different multiple representational systems in learning primary mathematics. *Journal of the Learning Sciences*, 11(1):25–62, 2002.

[4] John R. Anderson, editor. *The Architecture of Cognition*. Harvard University Press, 1983.

[5] John R. Anderson e Brian J. Reiser. The lisp tutor. 1985.

[6] Paul Baffes e Raymond Mooney. Refinement-based student modeling and automated bug library construction. *Journal of Artificial Intelligence in Education*, 7:75–116, 1996.

[7] J. Bonar e E. Soloway. Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human-Computer Studies in Mathematics*, 20:293–316, 1985.

[8] Raffaella Borasi. Student´s constructive uses of mathematical errors: A taxonomy. *Annual Meeting of the American Educational Research Association*, 1989.

[9] J. S. Brown e R. R. Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, (2):155–191, 1978.

[10] John Seely Brown e Kurt VanLehn. Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, (4):379–426, 1980.

[11] Bruce G. Buchanan e Edward H. Shortliffe. *Rule-Based expert systems - The MYCIN experiments of the Stanford heuristic programming project*. Addison-Wesley, 1984.

[12] S. Bull e J. Kay. Student models that invite the learner in: The SMILI open learner modelling framework. *International Journal of Artificial Intelligence in Education*, 17(2):89–120, 2007.

[13] Taatgen N.A. Lebiere C. e Anderson J.R. *Modeling Paradigms in ACT-R*. Cambridge University Press, 2006.

[14] CONDIGITAL. Condigital. http://www.diaadia.pr.gov.br/condigital/.

[15] David L. Davies e Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence In Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[16] Remco R. Bouckaert Eibe Frank Mark Hall Richard Kirkby Peter Reutemann Alex Seewald e David Scuse. Weka manual for version 3.7.3, 2010.

[17] D. Marczal A. Direne G. Ramos A. Pimentel M. Castilho F. Silva L. Bona M. Sunyé e L. García. Um arcabouço para apoiar a generalização de conceitos através da pedagogia de representações externas. *Anais do XVI Workshop sobre Informática na Escola (WIE-2010)*, páginas 1–10, Belo Horizonte-MG, julho de 2010. SBC.

[18] A. Direne A. Pimentel G. Ramos D. Marczal E. Carvalho M. Castilho. L. García F. Silva L. Bona e M. Sunyé. Objetos de aprendizagem generalizáveis para o currículo de matemática do ensino médio. *Anais do XV Workshop sobre Informática na Escola (WIE-2009)*, páginas 1693–1702, Bento Gonçalves-RS, julho de 2009. SBC.

[19] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*.

[20] A. Feitosa e outros. Definição formal de táticas de xadrez por meio da autoria incremental de conceitos heurísticos. *XVIII SBIE - Simpósio Brasileiro de Informática na Educação*, páginas 244–253, São Paulo, Brasil, Novembro de 2007. SBC.

[21] F. Gobet e G. Campitelli. Education and chess: A critical review. T. Redman, editor, *Chess and education: Selected essays from the Koltanowski conference*. Chess Program at the University of Texas at Dallas, 2006.

[22] P. Graziola. Aprendizagem com mobilidade (m-learning) nos processos de ensino e de aprendizagem: reflexões e possibilidades. *Novas Tecnologias na Educação (CINTED)*, 7(1):1–10, 2009.

[23] Robert W. Howard. *Concepts and Schemas an Introduction*. 1987.

[24] Seiji Isotani, Bruce McLaren, e Max Altman. Towards intelligent tutoring with erroneous examples: A taxonomy of decimal misconceptions. *Intelligent Tutoring Systems*, páginas 346–348. Springer Berlin / Heidelberg, 2010.

[25] Carlos Klock, Renato Ribas, e André Reis. Karma: um ambiente para o aprendizado de síntese de funções booleanas. *RBIE - Revista Brasileira de Informática na Educação*, 18(2):33–42, 2010.

[26] Pat Langley e Stellan Ohlsson. Automated cognitive modeling. páginas 193–197, 1984.

[27] A. M. Lesgold. Acquiring expertise. J. R. Anderson e S. M. Kosslyn, editors, *Tutorials in Learning and Memory: Essays in Honor of Gordon Bower*. W. H. Freeman, 1984.

[28] A. M. Lesgold, H. Rubinson, P. Feltovich R. Glasser, D. Klopfer, e Y. Wang. Expertise in a complex skill: Diagnosing x-ray pictures. M. Chi, R. Glasser, e M. Farr, editors, *The Nature of Expertise*. Lawrence Erlbaum, 1989.

[29] John McCarthy. Some expert system need common sense. 1984.

[30] Raymond R. Panko. Revisiting the panko halverson taxonomy of spreadsheet errors.

[31] PHP. Php. http://php.net/.

[32] Stuart Russell e Peter Norvig, editors. *Artificial Intelligence A Modern Approach - Third Edition*. Pearson Education, Inc., 2010.

[33] E. Sá, J. Teixeira, e C. Fernandes. Design de atividades de aprendizagem que usam jogos como princípio para cooperação. *XVIII SBIE - Simpósio Brasileiro de In-*

formática na Educação (SBIE-2007), páginas 607–616, São Paulo, Brasil, Novembro de 2007. SBC.

[34] John Self, La Yr, e John A. Self. Bypassing the intractable problem of student modelling, 1990.

[35] M. Sharples, N. Jeffery, B. du Boulay, B. Teather, D. Teather, e G. du Boulay. Socio-cognitive engineering: a methodology for the design of human-centred technology. *European Journal of Operational Research*, 136(2):310–323, 2002.

[36] Raymund Sison e Masamichi Shimura. Student modeling and machine learning. *International Journal of Artificial Intelligence in Education*, (9):128–158, 1998.

[37] Raymund C. Sison, Masayuki Numao, e Masamichi Shimura. Multistrategy discovery and detection of novice programmer errors. *Machine Learning*, (38):157–180, 2000.

[38] D. Sleeman, Haym Hirsh, Ian Ellery, e In-Yung Kim. Extending domain theories: Two case studies in student modeling. *Machine Learning*, 5:11–37, 1990.

[39] D. H. Sleeman. PIXIE: a shell for developing intelligent tutoring systems. R. Lawler e M. Yazdani, editors, *AI and Education: Learning Environments and Intelligent Tutoring Systems*. Ablex Publishing, 1987.

[40] Aaron Sloman. Interactions between philosophy and a.i.: The role of intuition and non-logical reasoning in intelligence. páginas 209–225, London, 1971.

[41] Aaron Sloman. Musings on the roles of logical and non-logical representations in intelligence. *Computational and Cognitive Perspectives, AAAI Press*, 1995.

APÊNDICE A

RESULTADO DA CLASSIFICAÇÃO DA REDE NEURAL MLP

```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V
Relation:     estudoGeneralizacao
Instances:    376
Attributes:   6
              C1
              C2
              C4
              C5
              C6
              class
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    -13.18923556360542
    Node 3    6.999020159117681
    Node 4    9.687816683620287
    Node 5    11.184275898455228
    Node 6    7.4666068326875195
```

```
Sigmoid Node 1
    Inputs    Weights
    Threshold    9.039188256741427
    Node 3    -3.8167144043844914
    Node 4    -10.19267927948163
    Node 5    -11.52467911981968
    Node 6    -2.5589857111430954
Sigmoid Node 2
    Inputs    Weights
    Threshold    -5.141273452029551
    Node 3    -5.933200583393204
    Node 4    -10.04366963170524
    Node 5    7.0611823473231485
    Node 6    -7.693876169055739
Sigmoid Node 3
    Inputs    Weights
    Threshold    -17.682654837072374
    Attrib C1    -5.530972229086187
    Attrib C2    -18.174238765412944
    Attrib C4    -17.882310401536042
    Attrib C5    5.079291751634299
    Attrib C6    5.452676394142102
Sigmoid Node 4
    Inputs    Weights
    Threshold    -5.461220841595951
    Attrib C1    -17.195223012404032
    Attrib C2    1.2771854055163732
    Attrib C4    3.7168281179770797
    Attrib C5    7.106129751812957
```

```
    Attrib C6    -1.1168191393480391
Sigmoid Node 5
    Inputs    Weights
    Threshold    -13.697886481808892
    Attrib C1    -20.88522914596891
    Attrib C2    -3.702670845142402
    Attrib C4    -0.7802477600035158
    Attrib C5    -9.89910700964159
    Attrib C6    -6.410744671884
Sigmoid Node 6
    Inputs    Weights
    Threshold    0.8032712802222275
    Attrib C1    -1.1548484802282355
    Attrib C2    13.950386591677196
    Attrib C4    -0.25425874896859324
    Attrib C5    -3.063195126383825
    Attrib C6    1.653365569571232
Class sub
    Input
    Node 0
Class sup
    Input
    Node 1
Class nap
    Input
    Node 2



Time taken to build model: 1.98 seconds
```

```
=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances         329               87.5    %
Incorrectly Classified Instances        47               12.5    %
Kappa statistic                          0.787
Mean absolute error                      0.13
Root mean squared error                  0.267
Relative absolute error                 35.4355 %
Root relative squared error             53.7606 %
Total Number of Instances              376

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.95      0.107      0.909     0.95      0.929      0.939    sub
                 0.603     0.013      0.905     0.603     0.724      0.883    sup
                 0.895     0.092      0.81      0.895     0.85       0.919    nap
Weighted Avg.    0.875     0.087      0.878     0.875     0.871      0.923

=== Confusion Matrix ===

   a   b   c   <-- classified as
 189   2   8 |   a = sub
   9  38  16 |   b = sup
  10   2 102 |   c = nap
```

APÊNDICE B

RESULTADO DOS AGRUPAMENTOS - SUBGENERALIZAÇÃO

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R fi
Relation:     estudoSubGeneralizacao
Instances:    208
Attributes:   5
              C1
              C2
              C4
              C5
              C6
Test mode:    split 50% train, remainder test


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 32.93997703822984
Missing values globally replaced with mean/mode
```

```
Cluster centroids:
                         Cluster
Attribute    Full Data          0          1
                 (208)       (46)      (162)
=============================================
C1               -0.5913    -0.6304    -0.5802
C2               -0.3365    -0.6739    -0.2407
C4               -0.6394      0.587    -0.9877
C5                0.0288        0.5    -0.1049
C6               -0.9471     0.4565    -1.3457



=== Model and evaluation on test split ===

kMeans
======

Number of iterations: 9
Within cluster sum of squared errors: 14.212906142424622
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster
Attribute    Full Data          0          1
                 (104)       (47)       (57)
=============================================
C1               -0.5577          0    -1.0175
```

C2	-0.2788	-0.2128	-0.3333
C4	-0.6538	-0.766	-0.5614
C5	0.0096	-0.0851	0.0877
C6	-1.0192	-1.617	-0.5263

Clustered Instances

0	43 (41%)
1	61 (59%)

APÊNDICE C

RESULTADO DOS AGRUPAMENTOS - SUPERGENERALIZAÇÃO

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R fi
Relation:     estudoSupGeneralizacao
Instances:    42
Attributes:   5
              C1
              C2
              C4
              C5
              C6
Test mode:    split 50% train, remainder test


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 5
Within cluster sum of squared errors: 5.961850823045267
Missing values globally replaced with mean/mode
```

```
Cluster centroids:
                         Cluster
Attribute    Full Data          0          1
                  (42)       (30)       (12)
=============================================
C1                1.2857     0.6333     2.9167
C2               -0.1667    -0.3333       0.25
C4                1.4286        0.6        3.5
C5                1.2143        0.7        2.5
C6                 1.381        0.5     3.5833



=== Model and evaluation on test split ===

kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 4.066109182098766
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster
Attribute    Full Data          0          1
                  (21)        (5)       (16)
=============================================
C1                 1.619        3.8     0.9375
```

C2	-0.0952	0.6	-0.3125
C4	1.619	4.2	0.8125
C5	1.381	2.8	0.9375
C6	1.8095	5.6	0.625

Clustered Instances

0	1	(5%)
1	20	(95%)