

Sistemas Baseados em Regras e Árvores de Decisão

Índice

Parte I – Sistemas baseados em regras

1.	Introdução.....	2
2.	Escopo de aplicação	4
3.	Exemplos de sistemas especialistas de grande sucesso	4
4.	Linguagens de programação e ambientes para o desenvolvimento de sistemas especialistas	8
5.	Projeto e composição de um sistema especialista	9
6.	Modelos de inferência	13
6.1	Encadeamento direto (forward chaining).....	15
6.2	Encadeamento reverso (backward chaining).....	17
7.	Resolução de conflito em encadeamento direto	20
8.	Vantagens e desvantagens de sistemas especialistas	21
8.1	Vantagens	21
8.2	Desvantagens.....	22
9.	Referências bibliográficas	23

Parte II – Árvores de Decisão

10.	Introdução.....	24
11.	Árvores de decisão e a tarefa de classificação	28
12.	Principais conceitos vinculados à indução de árvores de decisão	36
12.1	Top-Down Induction of Decision Tree (TDIDT)	38
12.2	Escolha dos atributos preditivos e critérios de parada	41
12.3	Ganho de informação	43
12.4	Razão de Ganho	45
12.5	<i>Gini</i>	56
12.6	Representação dos nós para atributos categóricos	59
12.7	Representação dos nós para atributos contínuos.....	62
12.8	Métodos de poda.....	66
13.	Algoritmos de indução de árvores de decisão	69
13.1	ID3.....	70
13.2	C4.5	72
13.3	CART.....	75
14.	Referências bibliográficas	78

Parte I – Sistemas baseados em regras

Obs: Este conteúdo está baseado em Notas de Aula vinculadas ao livro de NEGNEVITSKY (2002).

1. Introdução

- Parte significativa das aplicações de relevância prática em inteligência artificial (IA) está baseada na concepção de modelos computacionais do conhecimento empregado por um especialista humano.
- Modelo mais empregado em IA: **Sistema Especialista Baseado em Regras.**
- Algumas definições:
 - ✓ Conhecimento: Entendimento prático ou teórico de um assunto ou domínio. Somente quem conhece um processo ou fenômeno é capaz de controlá-lo e é denominado especialista naquele domínio. O especialista geralmente exerce algum tipo de poder, seja ele decisório ou apenas preditivo.

- ✓ Especialista: Pessoa que detém um conhecimento aprofundado e grande experiência prática em um determinado domínio. Expressa habilidades neste domínio que não estão presentes em outras pessoas.
- ✓ Regras como uma forma de representação do conhecimento: Têm uma estrutura na forma “SE ⟨antecedente⟩ ENTÃO ⟨consequente⟩” que relaciona informações ou fatos (no antecedente, também denominado de premissa ou condição) a alguma ação ou resultado (no consequente, também denominado de conclusão). São fáceis de criar e de interpretar.
- ✓ Antecedente e consequente: Ambos podem ser compostos pela conjunção e/ou disjunção de múltiplas proposições. Cada proposição é dada por um objeto linguístico e seu valor, ligados por um operador. Esse operador pode ser de atribuição (é, não é) ou de relação (é maior/menor que).
- ✓ Base de conhecimento: É dada pelo conjunto de regras associadas ao domínio.

2. Escopo de aplicação

- Indicado em situações em que a base de conhecimento pode ser bem caracterizada por um especialista humano, mas não pode ser facilmente transmitida a outros seres humanos.
- Indicado em situações em que devem ser tomadas ações repetitivas e em que erros humanos devem ser evitados.
- Muito empregado em tarefas como contabilidade, diagnóstico, controle de processos, operações financeiras e gerenciamento de recursos ou atividades produtivas.

3. Exemplos de sistemas especialistas de grande sucesso

- **DENDRAL**: É considerado o primeiro sistema especialista, por causa do seu modo automático de tomar decisões e resolver problemas relativos à química orgânica. Ele foi inteiramente escrito em LISP. DENDRAL é um sistema especialista e um projeto

pioneiro em inteligência artificial, que começou a ser desenvolvido em 1965, na Universidade de Stanford. O objetivo do projeto foi desenvolver soluções capazes de encontrar as estruturas moleculares orgânicas a partir da espectrometria de massa das ligações químicas presentes em uma molécula desconhecida.

- **MYCIN:** Ele surgiu no laboratório que havia criado anteriormente o sistema especialista DENDRAL, mas enfatizava a utilização de regras de julgamento em que havia elementos de incerteza (conhecidas como fatores de segurança) que lhes eram associados. Foi escrito em LISP, no início dos anos 70, para identificar bactérias causadoras de infecções graves, tais como bacteriemia e meningite, e para recomendar antibióticos, com a dose ajustada para o peso corporal do paciente. O nome derivou dos próprios antibióticos, uma vez que muitos antibióticos continham o sufixo “mycin”. O sistema MYCIN também foi utilizado para o diagnóstico de doenças da coagulação sanguínea, sendo composto por aproximadamente 450 regras.

- **CADUCEUS:** É um sistema especialista para diagnóstico em clínica médica. Foi iniciado nos anos 1970, mas só foi terminado em meados dos anos 1980, pois levou muito tempo para se construir a base de conhecimento. Foi desenvolvido por Harry Pople, da Universidade de Pittsburgh, com base em anos de entrevistas com o Dr. Jack Meyers, um dos principais diagnosticadores na área de clínica médica e professor da Universidade de Pittsburgh. CADUCEUS era capaz de diagnosticar em torno de mil doenças diferentes. Enquanto o CADUCEUS trabalhava usando um motor de inferência semelhante ao adotado pelo MYCIN, ele tinha uma série de mudanças (como a incorporação de raciocínio abduutivo) para lidar com a complexidade adicional da clínica médica, pois poderia haver uma série de doenças em simultâneo, e os dados eram escassos e geralmente incertos. O CADUCEUS é descrito como o sistema especialista “de maior conhecimento intensivo” existente.

- **PROSPECTOR:** É um sistema especialista concebido para ser utilizado em exploração mineral, tendo sido desenvolvido pelo Stanford Research Institute. Nove especialistas (geólogos) contribuíram para produzir a sua base de conhecimento, a qual usa uma estrutura híbrida que incorpora mais de 1000 regras.
- Número de sistemas especialistas bem-sucedidos implementados até 1986 (WATERMAN, 1986): perto de 200.
- Número de sistemas especialistas bem-sucedidos implementados até 1994 (DURKIN, 1994): mais de 2500.

4. Linguagens de programação e ambientes para o desenvolvimento de sistemas especialistas

- LISP, Prolog, OPS5: São linguagens computacionais para programação simbólica.
- CLIPS / Jess: Fornece um ambiente completo para o desenvolvimento de sistemas especialistas, com a disponibilização de uma máquina de inferência para a resolução de problemas de propósito geral.
- Exsys (www.exsys.com): É um ambiente de apoio ao desenvolvimento automático de sistemas especialistas, tendo sido utilizado por um grande número de empresas, em vários contextos de aplicação, indo desde sistemas para sugestão de vinhos até aconselhamento vocacional.
- ES/Kernel2: shell para o desenvolvimento de sistemas especialistas, vinculado à empresa japonesa Hitachi.

5. Projeto e composição de um sistema especialista

Definições relevantes para sistemas especialistas:

A computer system that emulates the decision-making ability of a human expert in a restricted domain.

Giarratano, J.C. & Riley, G.D. “Expert Systems: Principles and Programming”, 4th edition, Course Technology, 2004.

An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solutions.

Edward A. Feigenbaum (1994 Turing Award recipient)

- O termo “*Knowledge-Based System* (KBS)” pode ser utilizado como sinônimo de “Expert System”.

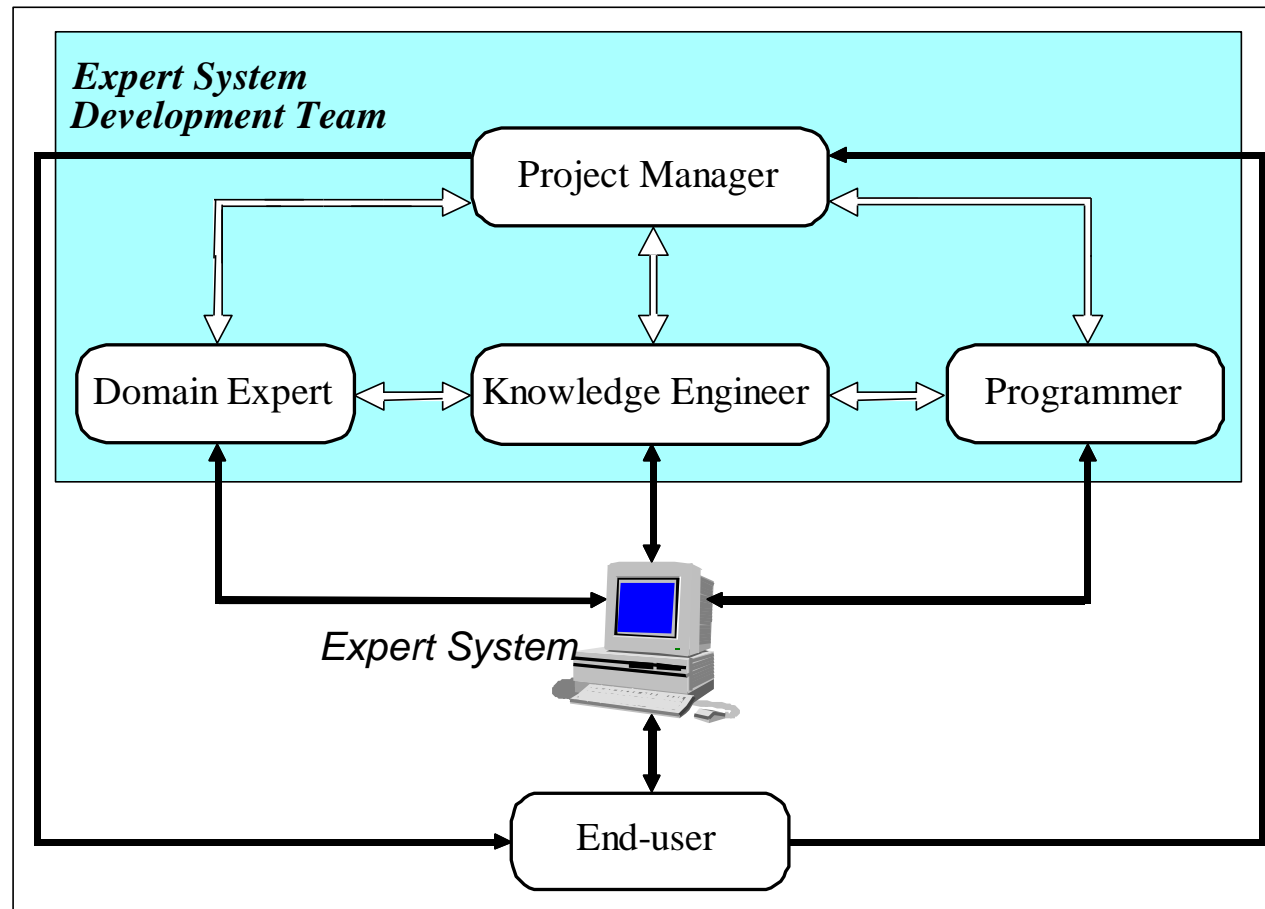


Figura 1 – Principais agentes participantes do projeto de um sistema especialista

- Componentes de um sistema especialista:
 - ✓ Conjunto de regras: compõe a base de conhecimento e é utilizada pela máquina de inferência para processar os dados de entrada.
 - ✓ Máquina de inferência: Infere conclusões a partir de fatos (dados de entrada) e da base de conhecimento.
- Esses componentes formam o sistema de produção, o qual foi proposto por Newell e Simon, da Universidade de Carnegie-Mellon, no início dos anos 1970. Neste modelo, o conhecimento é separado do seu processamento.
- O modelo do sistema de produção é baseado na concepção de que o ser humano resolve problemas pela aplicação de seu conhecimento (expresso na forma de regras de produção), dado um cenário que descreve uma situação ou estado de coisas (expresso na forma de fatos ou informações específicas sobre o problema).
- Os fatos são comparados com os antecedentes das regras e uma ou mais regras são ativadas, indicando ações contidas no consequente.

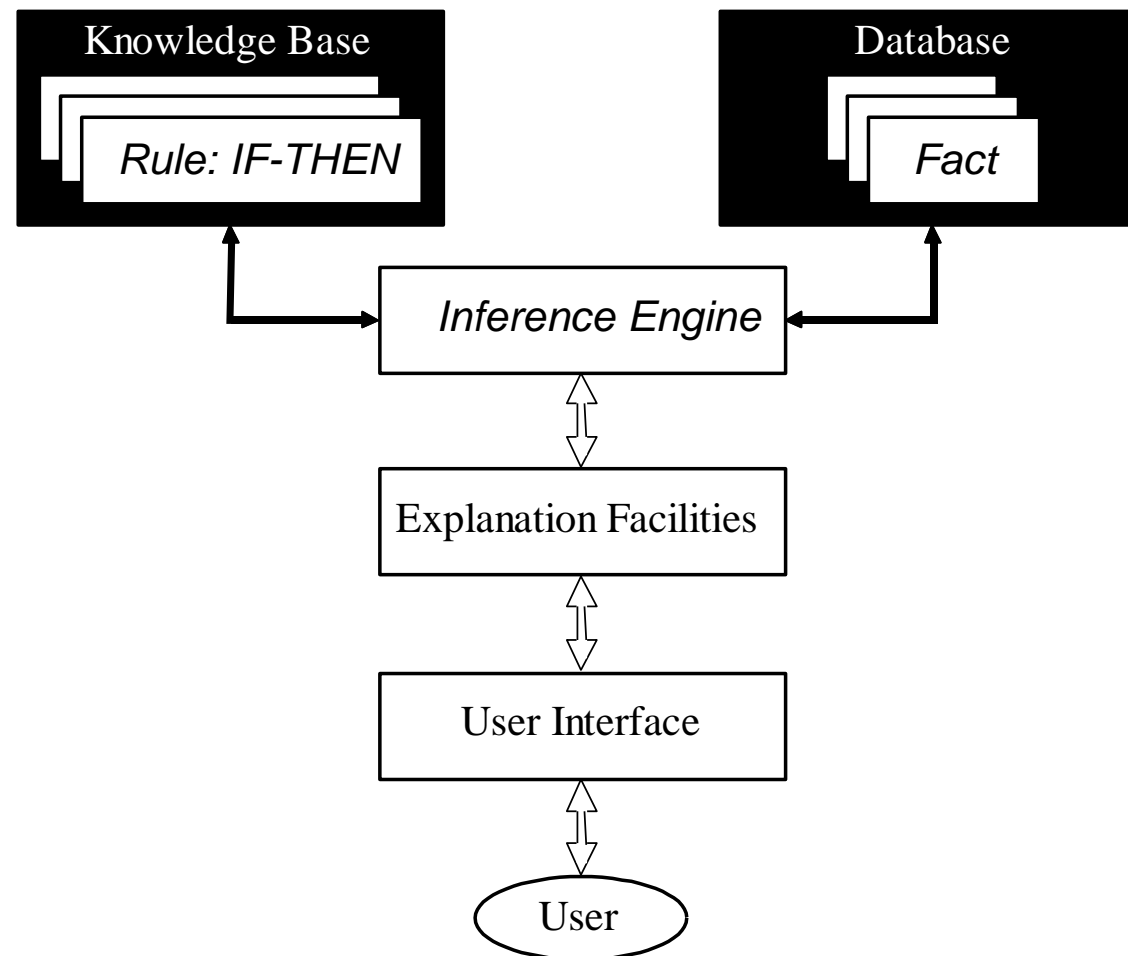


Figura 2 – Estrutura básica de um sistema especialista baseado em regras

6. Modelos de inferência

- A máquina de inferência (*inference engine*) realiza o “raciocínio lógico” que permite que o sistema especialista resolva um problema. Esse é o processo de vinculação das regras, presentes na base de conhecimento (*Knowledge Base*), aos fatos contidos na base de dados (*Database*).
- O encadeamento do processo de inferência indica como o sistema especialista aplica as regras para chegar a alguma conclusão.
- O usuário inclusive pode requisitar do sistema especialista uma explicação acerca de:
 - ✓ Como uma conclusão particular é produzida?
 - ✓ Qual é a motivação para que uma informação específica (fato) seja solicitado do usuário?
- Em outras palavras, o sistema especialista deve ser capaz de explicitar o seu processo de inferência e de justificar a sua conclusão.

- Um ciclo de inferência é ilustrado na Figura 3, onde um fato leva à ativação de uma regra que, por sua vez, produz um fato novo.

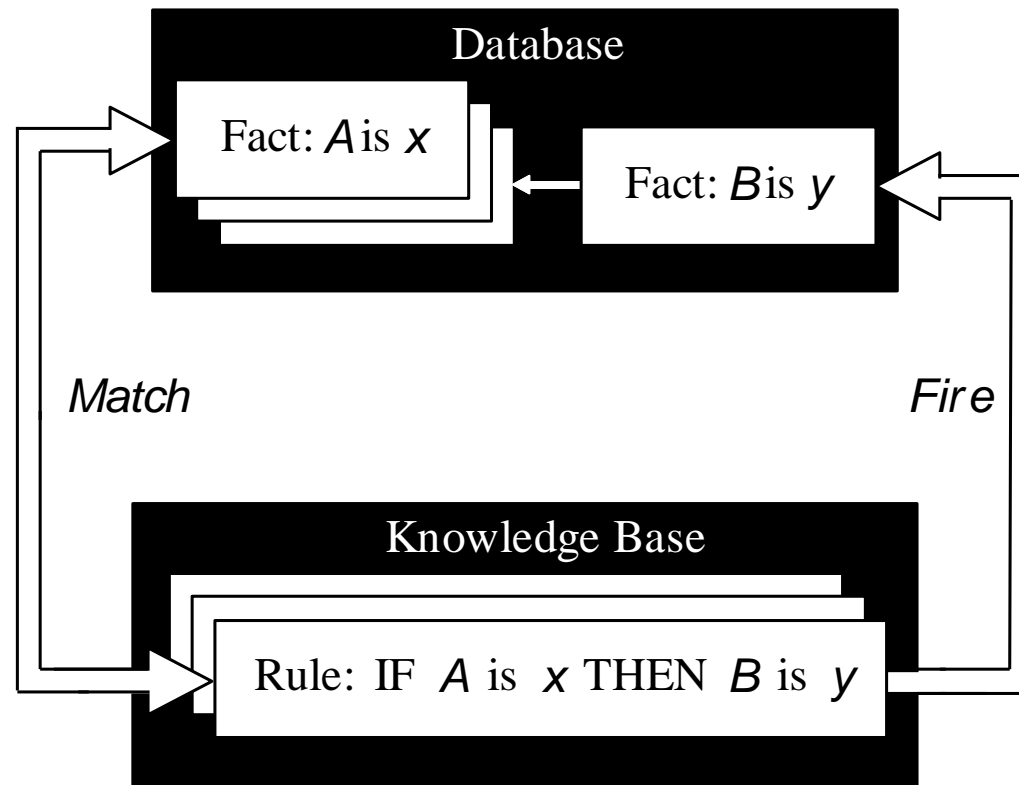


Figura 3 – Exemplo de um ciclo de inferência em um sistema especialista baseado em regras

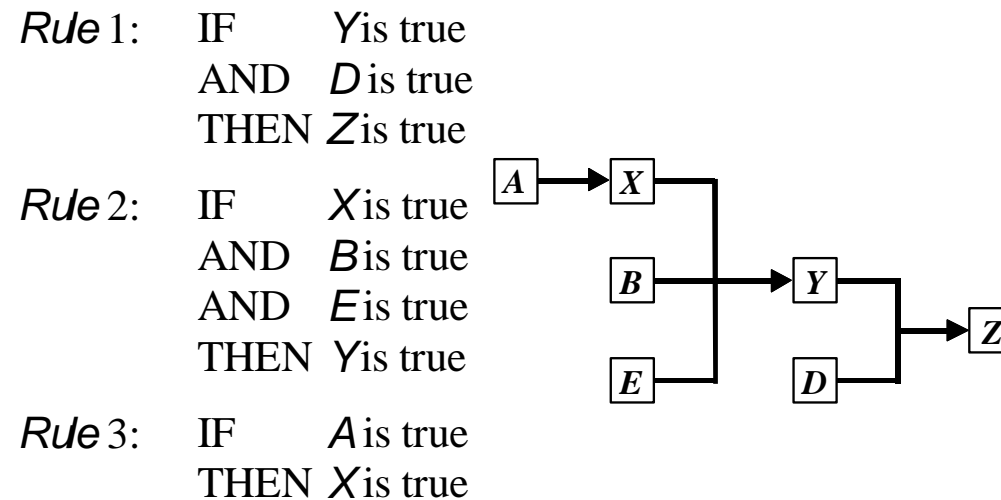


Figura 4 – Exemplo de um encadeamento de inferência (*inference chain*)

6.1 Encadeamento direto (forward chaining)

- É o raciocínio guiado por fatos. Parte-se dos fatos existentes e se executa apenas a regra mais ativa, o que causa a adição de um fato novo à base de dados. Cada regra só pode ser executada uma vez e o ciclo de inferência termina quando não houver mais regras que podem ser ativadas.

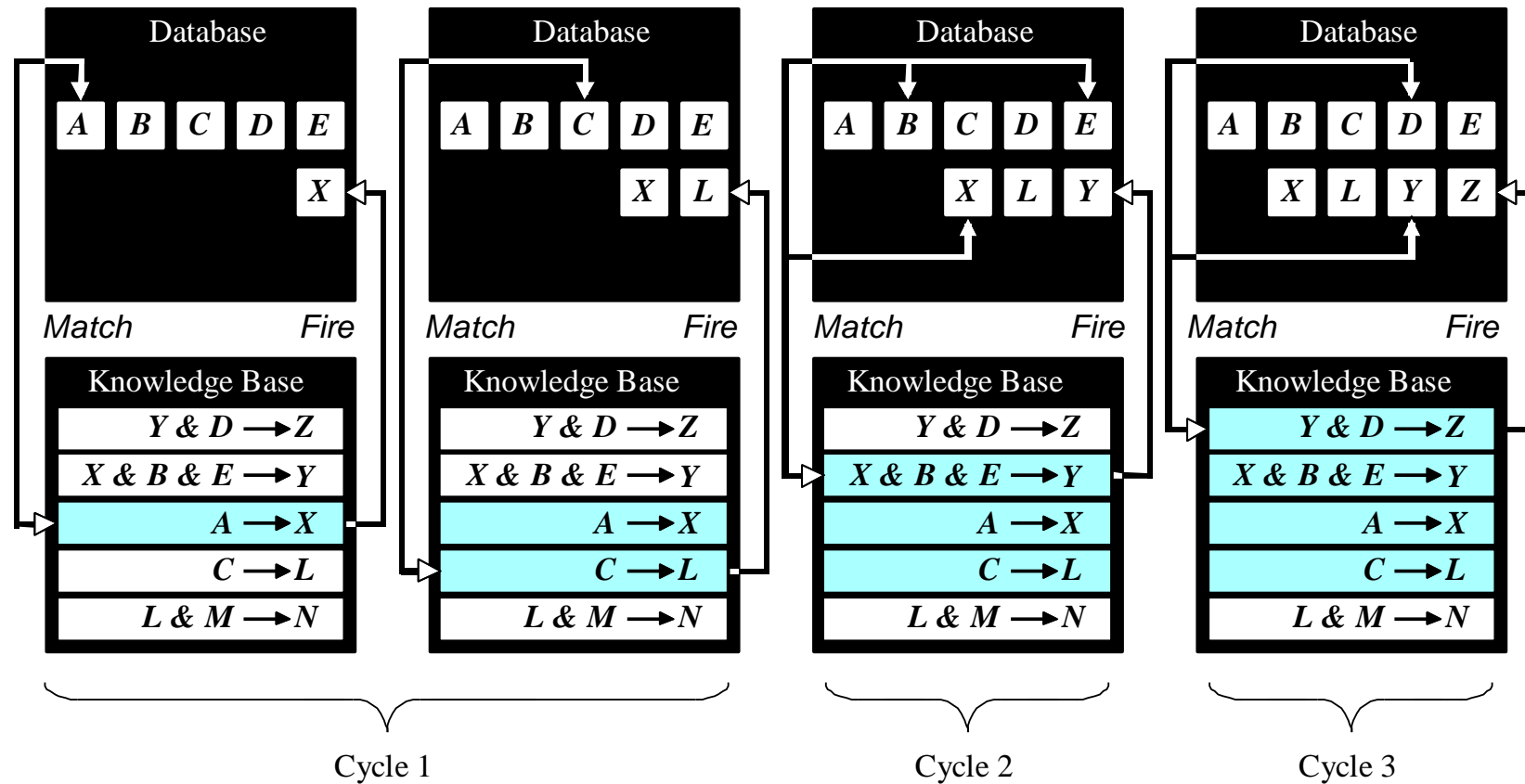


Figura 5 – Exemplo de encadeamento direto (*forward chaining*)

- O encadeamento direto é indicado quando se parte de fatos e o objetivo é inferir fatos novos, quaisquer que sejam eles.
- No entanto, se houver um objetivo previamente estabelecido para o processo de inferência, muitas regras podem ser executadas sem que tenham vínculo algum com o objetivo estabelecido.
- Logo, se o objetivo for inferir um fato específico, o encadeamento direto tende a ser ineficiente.

6.2 Encadeamento reverso (backward chaining)

- É o raciocínio guiado por objetivos. No encadeamento reverso, o sistema especialista recebe um fato a ser demonstrado como verdadeiro (esse é o objetivo) e a máquina de inferência busca encontrar evidências que provem a veracidade daquele fato.

- Para tal, a máquina de inferência deve localizar regras que têm aquele fato como consequente. Em seguida, ela tenta validar o antecedente de ao menos uma dessas regras localizadas. Geralmente há uma forma de ordenar as regras e tratar uma de cada vez.
- Se não houver fatos na base de dados capazes de validar o antecedente da regra que está sendo tratada, são criados objetivos intermediários, ou seja, os fatos intermediários a serem provados estarão associados ao antecedente desta regra.
- O processo recursivo se encerra quando não existem mais regras na base de conhecimento que tenham em seu consequente o objetivo intermediário pretendido (o que implica na impossibilidade de comprovar o fato inicial), ou então quando todos os objetivos intermediários e o objetivo inicial foram atendidos (o que implica que o fato inicial foi comprovado).

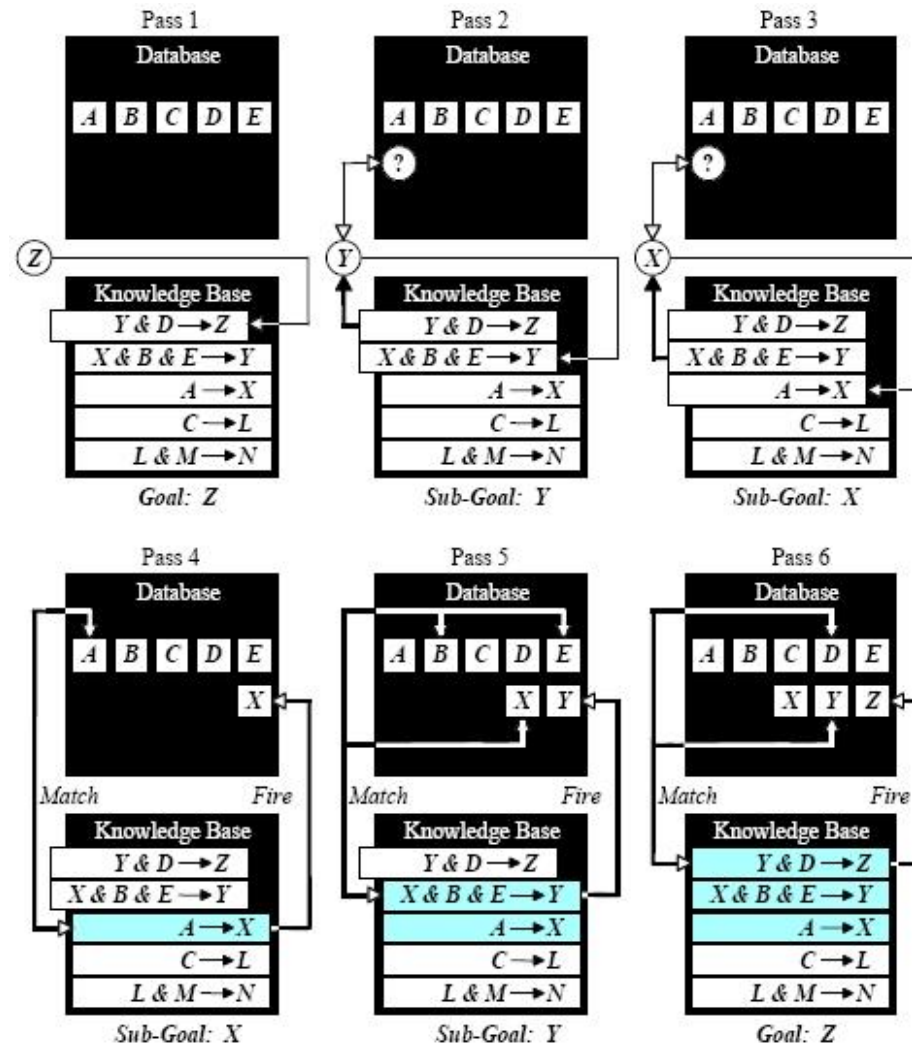


Figura 6 – Exemplo de encadeamento reverso (*backward chaining*)

7. Resolução de conflito em encadeamento direto

- Um método para escolha de qual regra terá o seu consequente executado, quando mais de uma regra é ativada em um dado ciclo de inferência, é chamado de resolução de conflito.
- Dentre os métodos mais empregados, tem-se:
 - ✓ Ative a regra de maior prioridade, o que implica em definir prioridades relativas entre as regras.
 - ✓ Ative a regra mais específica, supondo que essa regra mais específica carrega mais informação sobre o fato que a levou a disparar do que outras regras menos específicas.
 - ✓ Ative a regra que tem associados a si os fatos mais recentes da base de dados, o que implica em definir um rótulo temporal a cada fato.
- Também existem métodos de resolução de conflito em encadeamento reverso, mas esses não serão abordados aqui.

8. Vantagens e desvantagens de sistemas especialistas

8.1 Vantagens

- Concebidos para emular a estratégia de tomada de decisão de especialistas humanos;
- Tendem a ser muito precisos em suas conclusões;
- Tendem a produzir uma conclusão em um tempo reduzido, embora testes exaustivos no caso de bases de conhecimento compostas por muitas regras possam ser custosos;
- Apresentam uma alta capacidade de explanação do processo de inferência;
- Apresentam uma estrutura uniforme, em que cada regra é um pedaço independente do conhecimento disponível;
- Ao separar o conhecimento do seu processamento, permitem que um mesmo ambiente de projeto de sistema especialista possa ser empregado em diferentes aplicações;
- Quando empregam regras nebulosas, são capazes de realizar um processo de inferência logicamente preciso a partir de um conhecimento inicial impreciso.

8.2 Desvantagens

- Atuam em um domínio restrito;
- O conhecimento especialista pode não ser de fácil acesso;
- Trabalham predominantemente com raciocínio simbólico apenas;
- Recorrem predominantemente a processos de inferência dedutiva, em detrimento das inferências indutivas e abdutivas;
- Promovem pouco relacionamento entre as regras, de modo que há poucos subsídios para se definir o papel de regras individuais num processo de inferência;
- O processo de inferência pode não ser sempre confiável, possivelmente motivado pela dificuldade de lidar com ambiguidades, pela presença de regras conflitantes na base de conhecimento e pela dificuldade de promover atualizações (regras podem ficar ultrapassadas / obsoletas) ou aprendizado em sua base de conhecimento;

- Em geral, não apresentam capacidade de aprendizado, não sendo assim capazes de modificar automaticamente a sua base de conhecimento, cabendo ao projetista realizar a manutenção e a revisão do sistema;
- Ao lidarem com problemas mais complexos, os quais requerem bases de conhecimento compostas por um grande número de regras, é sabido que as regras escalam mal, ou seja, a inserção de regras adicionais, com o aumento do número de regras, tende a produzir efeitos colaterais e conflitos com outras regras já presentes.

9. Referências bibliográficas

DURKIN, J. (1994). Expert systems: design and development. Macmillan Publishing.

NEGNEVITSKY, M. (2002). Artificial Intelligence. A Guide to Intelligent Systems. Pearson Education.

WATERMAN, D.A. (1986). A guide to expert systems. Addison-Wesley.

Parte II – Árvores de Decisão

Obs: Este conteúdo está baseado no capítulo de Árvores de Decisão constante da Tese de Doutorado de Márcio Porto Basgalupp (BASGALUPP, 2010) e também em FERREIRA (2008).

10. Introdução

- Na síntese de modelos de classificação, a associação entre as classes e o conjunto de atributos que caracterizam os objetos a serem classificados pode se dar de formas variadas, empregando processamento simbólico e/ou numérico.
- A construção dos modelos computacionais de classificação geralmente emprega um dentre dois paradigmas alternativos:
 - ✓ Top-down: obtenção do modelo de classificação a partir de informações fornecidas por especialistas;
 - ✓ Bottom-up: obtenção do modelo de classificação pela identificação de relacionamentos entre variáveis dependentes e independentes em bases de

dados rotuladas. O classificador é induzido por mecanismos de generalização fundamentados em exemplos específicos (conjunto finito de objetos rotulados). Existem propostas também para dados não-rotulados.

- As árvores de decisão, tema deste tópico do curso, estão fundamentadas no paradigma bottom-up e sua aplicação requer as seguintes condições:
 - ✓ Toda informação sobre cada objeto (caso) a ser classificado deve poder ser expressa em termos de uma coleção fixa de propriedades ou atributos. Dessa forma, objetos distintos não podem requerer coleções distintas de atributos. Bases de dados que atendem a este requisito são denominadas *flat files*.
 - ✓ O número de classes pode ser definido a priori, o que transforma a modelagem num processo de treinamento supervisionado, ou então será definido automaticamente a partir dos dados disponíveis, o que caracteriza um processo de treinamento não-supervisionado.

- ✓ Há duas possibilidades de classes: classes discretas e classes contínuas. Quando um objeto pertence ou não pertence a uma determinada classe, não havendo a possibilidade de pertinência gradual, nos referimos a classes discretas, em contrapartida a classes de valores contínuos. Algumas árvores de decisão vão trabalhar apenas com classes discretas, enquanto outras admitem classes contínuas (resolvem problemas de regressão).
- ✓ Deve haver uma quantidade bem maior de objetos do que classes, inclusive para permitir a aplicação de testes estatísticos. A quantidade adequada de objetos vai depender do número de atributos, do número de classes e da complexidade intrínseca ao modelo de classificação.
- ✓ A tarefa de classificação deve poder ser implementada de forma lógica, ou seja, empregando uma base de regras de decisão. Assim, a classificação de cada objeto pode ser descrita por uma expressão lógica. Em contrapartida a este requisito, podemos mencionar a classificação por operações aritméticas,

empregada em discriminantes lineares, por exemplo, que realizam a classificação por uma combinação linear dos atributos, seguida da comparação com um limiar.

- Dessa forma, serão consideradas neste estudo bases de dados constituídas por objetos descritos por um conjunto de atributos (propriedades, características), sendo que a cada objeto deve ser associada uma classe, dentre um conjunto de classes possíveis.
- Os atributos são variáveis observáveis e independentes, que assumem valores em variados domínios. Costuma-se especificar os atributos da seguinte forma:
 - ✓ Contínuos: assumem valores numéricos em intervalos no eixo dos números reais;
 - ✓ Catégoricos ordinais: assumem um conjunto finito de valores, que podem ser ordenados;
 - ✓ Catégoricos não-ordinais: assumem um conjunto finito de valores que não podem ser ordenados.

- A classe é uma variável dependente cujo valor é definido a partir das variáveis independentes, ou seja, a partir dos atributos.
- Árvores de decisão são geralmente aplicadas junto a grandes bases de dados. Para tanto, regularidades implícitas presentes na base de dados devem ser descobertas automaticamente e expressas, predominantemente, na forma de regras.
- Conhecimentos de Inteligência Artificial e Estatística são comumente empregados para a geração das árvores de decisão.

11. Árvores de decisão e a tarefa de classificação

- O aprendizado indutivo de árvores de decisão é geralmente dividido em aprendizado supervisionado e não-supervisionado, embora o aprendizado semi-supervisionado também tenha sido considerado ao longo dos últimos anos (CHAPELLE *et al.*, 2006).
- Uma tarefa de classificação bastante conhecida é o diagnóstico médico, em que para cada paciente são definidos atributos contínuos ou categóricos ordinais (Ex: idade,

altura, peso, temperatura do corpo, batimento cardíaco, pressão, condição social, etc.) e atributos categóricos não-ordinais (Ex: sexo, cor da pele, local da dor, etc.).

- A tarefa do classificador é realizar um mapeamento dos atributos para um diagnóstico (Ex: saudável, pneumonia, Influenza A, etc.).
- Na Figura 7, é ilustrado um diagrama do processo de indução de um classificador e, posteriormente, a sua utilização. Primeiro, o conjunto de treinamento, no qual os rótulos das classes dos exemplos são conhecidos, é utilizado por um algoritmo de aprendizado para construir um modelo.
- Após a construção, esse classificador pode ser aplicado para prever os rótulos das classes dos exemplos do conjunto de teste, ou seja, exemplos cujas classes são desconhecidas.
- De acordo com TAN *et al.* (2005), a classificação pode ser utilizada para os seguintes propósitos: modelagem descritiva e modelagem preditiva.



Figura 7 – Indução de um classificador e dedução das classes para novas amostras

- Na **modelagem descritiva**, um modelo de classificação é utilizado como uma ferramenta para distinguir exemplos de diferentes classes. Como exemplo, um médico

pode utilizar um modelo de classificação para identificar quais são as principais causas (sintomas) de uma determinada doença.

- A partir disso, é possível chegar a conclusões, por exemplo, de que na grande maioria dos casos o paciente que está com a doença Influenza A apresentou febre e pneumonia.
- Quando há o interesse em análise descritiva, é desejável que o modelo de classificação seja fácil de interpretar, ou seja, que fique claro ao usuário o porquê de um determinado exemplo pertencer a uma determinada classe.
- Na **modelagem preditiva**, um modelo de classificação é utilizado para classificar exemplos cujas classes são desconhecidas, ou seja, exemplos que não foram utilizados na construção do modelo. Como exemplo, um médico pode utilizar o histórico de seus pacientes, em que as classes (diagnóstico) já são conhecidas, para construir um modelo de classificação a ser utilizado para diagnosticar novos pacientes.

- Dependendo da área de aplicação, a interpretabilidade do modelo de classificação não é essencial para a predição. No exemplo anterior, seria interessante um modelo com boa interpretabilidade, tornando possível medicar o paciente para combater os sintomas apresentados e, conseqüentemente, tratar da doença diagnosticada.
- As árvores de decisão constituem uma técnica poderosa e amplamente utilizada em problemas de classificação. Uma das razões para a sua popularidade é o fato do conhecimento adquirido ser representado por meio de regras. Essas regras podem ser expressas em linguagem natural, facilitando o entendimento por parte das pessoas.
- Como exemplos de modelos de classificação que não são baseados em regras, temos: k -vizinhos mais próximos, máquinas de vetores-suporte (SVM) e redes neurais artificiais.
- Para ilustrar o funcionamento básico de uma árvore de decisão, pode ser considerado novamente o problema de diagnosticar pacientes (Figura 7). Suponha que um novo paciente chegue ao consultório. Como o médico poderia diagnosticar o paciente?

- A primeira pergunta que pode ser feita ao paciente é se ele tem sentido dor (corresponderia ao nó-raiz da árvore de decisão). A seguir, outras perguntas podem ser feitas, dependendo da resposta obtida. Por exemplo, se o paciente está tendo febre ou enjoos, ou ainda se tem notado alguma mancha no corpo.
- O exemplo anterior apresenta uma forma de solucionar um problema de classificação por meio de uma sequência de perguntas sobre uma série de características de um objeto (no caso, um paciente). Uma vez obtida a resposta àquela pergunta, outra pode ser realizada até que se chegue a uma conclusão sobre a classe a que pertence o objeto.
- Essa sequência de perguntas e suas possíveis respostas podem ser organizadas na forma de uma árvore de decisão, a qual é uma estrutura hierárquica composta por nós e arestas.
- Dessa forma, é possível utilizar uma árvore de decisão para classificar um novo paciente como saudável ou doente. Para isso, basta partir do nó raiz da árvore e ir percorrendo a

árvore, através das respostas aos testes dos nós internos (decisão), até chegar em um nó folha, o qual indica a classe correspondente do novo paciente.

- Além da obtenção da classe, a grande vantagem é que a trajetória percorrida até o nó folha representa uma regra, facilitando a interpretabilidade do modelo pelo usuário, no caso, um médico.
- A Figura 8 mostra uma árvore de decisão, que é uma estrutura que contém:
 - folha(s), indicando uma classe;
 - nó(s) de decisão, que define(m) algum teste sobre o valor de um atributo específico (ou de um subconjunto de atributos), com um ramo e sub-árvore para cada um dos valores possíveis do teste.
- Partindo da raiz, a cada nó de decisão o resultado do teste de decisão é determinado e se inicia o processo pela raiz da sub-árvore correspondente a esse resultado.
- Um mesmo conjunto de dados pode gerar várias árvores de decisão distintas. Assim, usando o exemplo da Figura 8, o nó raiz poderia ser “Aplicações” em vez de “Saldo em

conta corrente”, fazendo com que o nó “Saldo em conta corrente” passe a ocupar uma outra posição na árvore. Essa troca de nós faz com que seja necessário percorrer um caminho maior ou menor para se chegar a uma decisão.

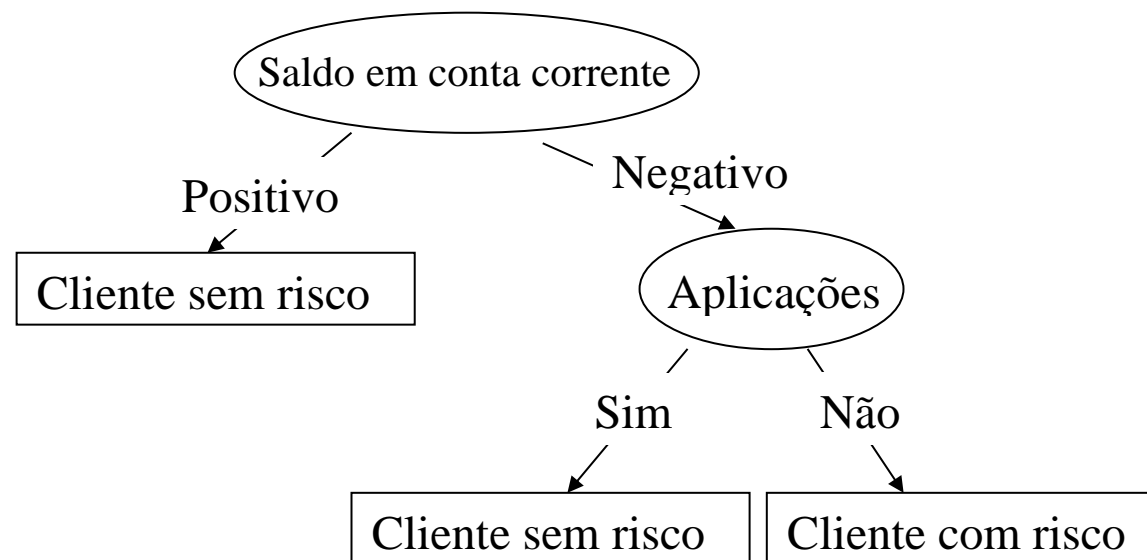


Figura 8 – Exemplo fictício de árvore de decisão, tomando atributos de clientes de alguma instituição financeira.

- Na construção da árvore de decisão, procura-se associar a cada nó de decisão o atributo “mais informativo” entre aqueles ainda não utilizados no caminho desde a raiz da árvore.
- No entanto, cada algoritmo tem a sua própria metodologia para distinguir o atributo mais informativo, fazendo com que a topologia da árvore e a qualidade da árvore variem em função do algoritmo utilizado.

12. Principais conceitos vinculados à indução de árvores de decisão

- Uma vez construída uma árvore de decisão, seu uso é imediato e muito rápido computacionalmente. Além disso, a interpretabilidade da árvore de decisão é um de seus pontos fortes.
- No entanto, a construção de uma árvore de decisão, chamado de processo de indução, pode ser uma tarefa de alta demanda computacional.

- Embora a indução de uma árvore de decisão possa ser realizada de forma manual, numa abordagem *top-down*, as principais demandas por árvores de decisão geralmente recorrem a processos automáticos de indução, numa abordagem *bottom-up*, a partir de dados disponíveis.
- **Todo processo de indução possui um bias indutivo, associado à preferência de uma hipótese sobre outras, supondo a existência de hipóteses que são igualmente consistentes.**
- Há muitas maneiras de uma árvore de decisão ser estruturada a partir de um conjunto de atributos. De forma exaustiva, o número de árvores de decisão atingíveis cresce fatorialmente à medida que o número de atributos aumenta.
- Logo, torna-se impraticável definir a estrutura da árvore de decisão ótima para um determinado problema, devido ao elevado custo computacional da busca (HYAFIL & RIVEST, 1976).

- Nesse sentido, algoritmos baseados em heurísticas gananciosas (*greedy*) têm sido desenvolvidos para a indução de árvores de decisão. Mesmo que eles não garantam uma solução ótima, apresentam resultados satisfatórios em tempo factível.

12.1 Top-Down Induction of Decision Tree (TDIDT)

- O *Top-Down Induction of Decision Tree* (TDIDT) é um algoritmo bem conhecido e é utilizado como base para muitos algoritmos de indução de árvores de decisão, dentre eles os mais conhecidos como ID3 (QUINLAN, 1986), C4.5 (QUINLAN, 1993) e CART (BREIMAN *et al.*, 1984).
- O TDIDT produz regras de decisão de forma implícita numa árvore de decisão, a qual é construída por sucessivas divisões dos exemplos de acordo com os valores de seus atributos preditivos. De acordo com BRAMER (2007), esse processo é conhecido como particionamento recursivo.

- O esqueleto do algoritmo de TDIDT é baseado em três possibilidades sobre um conjunto de treinamento T contendo classes C_1, C_2, \dots, C_k :
 1. T contém um ou mais objetos, sendo todos da classe C_j . Assim, a árvore de decisão para T é um nó folha que identifica a classe C_j .
 2. T não contém objetos. A árvore de decisão também é um nó folha, mas a classe associada deve ser determinada por uma informação externa. Por exemplo, pode-se utilizar o conhecimento do domínio do problema.
 3. T contém exemplos pertencentes a mais de uma classe. Neste caso, a ideia é dividir T em sub-conjuntos que são, ou tendem a se dirigir para, coleções de exemplos com classes únicas. Para isso, é escolhido um atributo preditivo A que possui um ou mais possíveis resultados O_1, O_2, \dots, O_n . T é particionado em sub-conjuntos T_1, T_2, \dots, T_n , onde T_i contém todos os exemplos de T que têm resultado O_i para o atributo A . A árvore de decisão para T consiste de um nó de decisão identificando

o teste sobre o atributo A , e uma aresta para cada possível resultado, ou seja, n arestas. No lugar de um único atributo A , pode também ser considerado um subconjunto de atributos.

- O mesmo algoritmo de indução de árvores de decisão (passos 1, 2 e 3) é aplicado recursivamente para cada sub-conjunto de exemplos T_i , com i variando de 1 até n .
- Basicamente, o algoritmo TDIDT é um algoritmo recursivo de busca gananciosa (*greedy*) que procura, sobre um conjunto de atributos, aquele(s) que “melhor” dividem o conjunto de exemplos em sub-conjuntos. Inicialmente, todos os exemplos são colocados em um único nó, chamado de raiz.
- A seguir, um atributo preditivo é escolhido para o teste desse nó e, assim, dividir os exemplos em sub-conjuntos. Esse processo se repete recursivamente até que todos os exemplos estejam classificados, sem erro, ou então até que todos os atributos preditivos já tenham sido testados ou não exibam mais poder discriminativo. O mesmo atributo

preditivo pode ser utilizado múltiplas vezes, se ainda houver múltiplos valores desse atributo no nó, permitindo estabelecer novas partições.

12.2 Escolha dos atributos preditivos e critérios de parada

- O critério de seleção define qual atributo preditivo é utilizado em cada nó da árvore.
- Existem diferentes tipos de critérios de seleção, sendo esta uma das variações entre os diversos algoritmos de indução de árvores de decisão. Esses critérios são definidos em termos da distribuição de classe dos exemplos antes e depois da divisão (TAN *et al.*, 2005).
- A maioria dos algoritmos de indução de árvores de decisão trabalha com funções de divisão univariável, ou seja, cada nó interno da árvore é dividido de acordo com um único atributo. Nesse caso, o algoritmo tenta encontrar o melhor atributo para realizar essa divisão.

- Os critérios de seleção para a melhor divisão são baseados em diferentes medidas, tais como impureza, distância e dependência. A maior parte dos algoritmos de indução busca dividir os dados de um nó-pai de forma a minimizar o grau de impureza dos nós-filhos.
- Quanto menor o grau de impureza, mais desbalanceada é a distribuição de classes. Em um determinado nó, a impureza é nula se todos os exemplos nele pertencerem à mesma classe. Analogamente, o grau de impureza é máximo no nó se houver o mesmo número de exemplos para cada classe possível.
- Critérios de parada para a aplicação recursiva do algoritmo TDIDT em um certo nó podem ser: (1) número mínimo de exemplos; (2) máxima altura do nó; (3) queda de impureza acima de um limiar.
- A seguir, são apresentadas as medidas mais utilizadas para a seleção da melhor divisão.

12.3 Ganho de informação

- Uma das medidas baseadas em impureza é o **Ganho de Informação**, o qual usa a entropia como medida de impureza. O algoritmo ID3 (QUINLAN, 1986), pioneiro em indução de árvores de decisão, utiliza essa medida.
- Para determinar o quão boa é uma condição de teste realizada, é necessário comparar o grau de entropia do nó-pai (antes da divisão) com o grau de entropia dos nós-filhos (após a divisão). O atributo que gerar uma maior diferença é escolhido como condição de teste. O ganho é dado pela Equação (1), na forma:

$$\text{ganho} = \text{entropia}(\text{pai}) - \sum_{j=1}^n \left[\frac{N(v_j)}{N} \text{entropia}(v_j) \right] \quad (1)$$

onde n é o número de valores do atributo, ou seja, o número de nós-filhos, N é o número total de objetos do nó-pai e $N(v_j)$ é o número de exemplos associados ao nó-filho v_j .

- O grau de entropia é definido pela Equação (2) a seguir:

$$\text{entropia}(\text{nó}) = - \sum_{i=1}^c p(i / \text{nó}) \cdot \log_2[p(i / \text{nó})] \quad (2)$$

onde $p(i / \text{nó})$ é a fração dos registros pertencentes à classe i no nó, e c é o número de classes.

- O critério de ganho seleciona como atributo-teste aquele que maximiza o ganho de informação. O grande problema ao se utilizar o ganho de informação é que ele dá preferência a atributos com muitos valores possíveis (número de arestas).
- Um exemplo claro desse problema ocorreria ao utilizar um atributo totalmente irrelevante (por exemplo, um identificador único). Nesse caso, seria criado um nó para cada valor possível, e o número de nós seria igual ao número de identificadores. Cada um desses nós teria apenas um exemplo, o qual pertence a uma única classe, ou seja, os exemplos seriam totalmente discriminados.

- Assim, o valor da entropia seria mínima porque, em cada nó, todos os exemplos (no caso um só) pertencem à mesma classe. Essa divisão geraria um ganho máximo, embora seja totalmente inútil.

12.4 Razão de Ganho

- Para solucionar o problema do ganho de informação, foi proposto em QUINLAN (1993) a Razão de Ganho (do inglês *Gain Ratio*), que nada mais é do que o ganho de informação relativo (ponderado) como critério de avaliação.
- A razão de ganho é definida pela Equação (3), na forma:


$$\text{razão_de_ganho}(\text{nó}) = \frac{\text{ganho}}{\text{entropia}(\text{nó})}. \quad (3)$$

- Pela Equação (3), é possível perceber que a razão não é definida quando o denominador é igual a zero. Além disso, a razão de ganho favorece atributos cujo denominador, ou

seja, a entropia, possui valor pequeno. Em outras palavras, essa métrica cria um viés que favorece atributos caracterizados por um número menor de valores distintos.

- Em QUINLAN (1988), é sugerido que a razão de ganho seja realizada em duas etapas.
- Na primeira etapa, é calculado o ganho de informação para todos os atributos. Após isso, considera-se apenas aqueles atributos que obtiveram um ganho de informação acima da média, e então se escolhe aquele que apresentar a melhor razão de ganho.
- Dessa forma, Quinlan mostrou que a razão de ganho supera o ganho de informação tanto em termos de acurácia quanto em termos de complexidade das árvores de decisão geradas.
- Vamos considerar um exemplo didático, extraído de:

http://www.saedsayad.com/decision_tree.htm



The diagram shows a table with five columns. The first four columns are grouped under a green bracket labeled 'Predictors'. The fifth column is grouped under an orange bracket labeled 'Target'.

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Play Golf	
Yes	No
9	5



$$\begin{aligned}
 \text{Entropy}(\text{PlayGolf}) &= \text{Entropy}(5,9) \\
 &= \text{Entropy}(0.36, 0.64) \\
 &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
 &= 0.94
 \end{aligned}$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



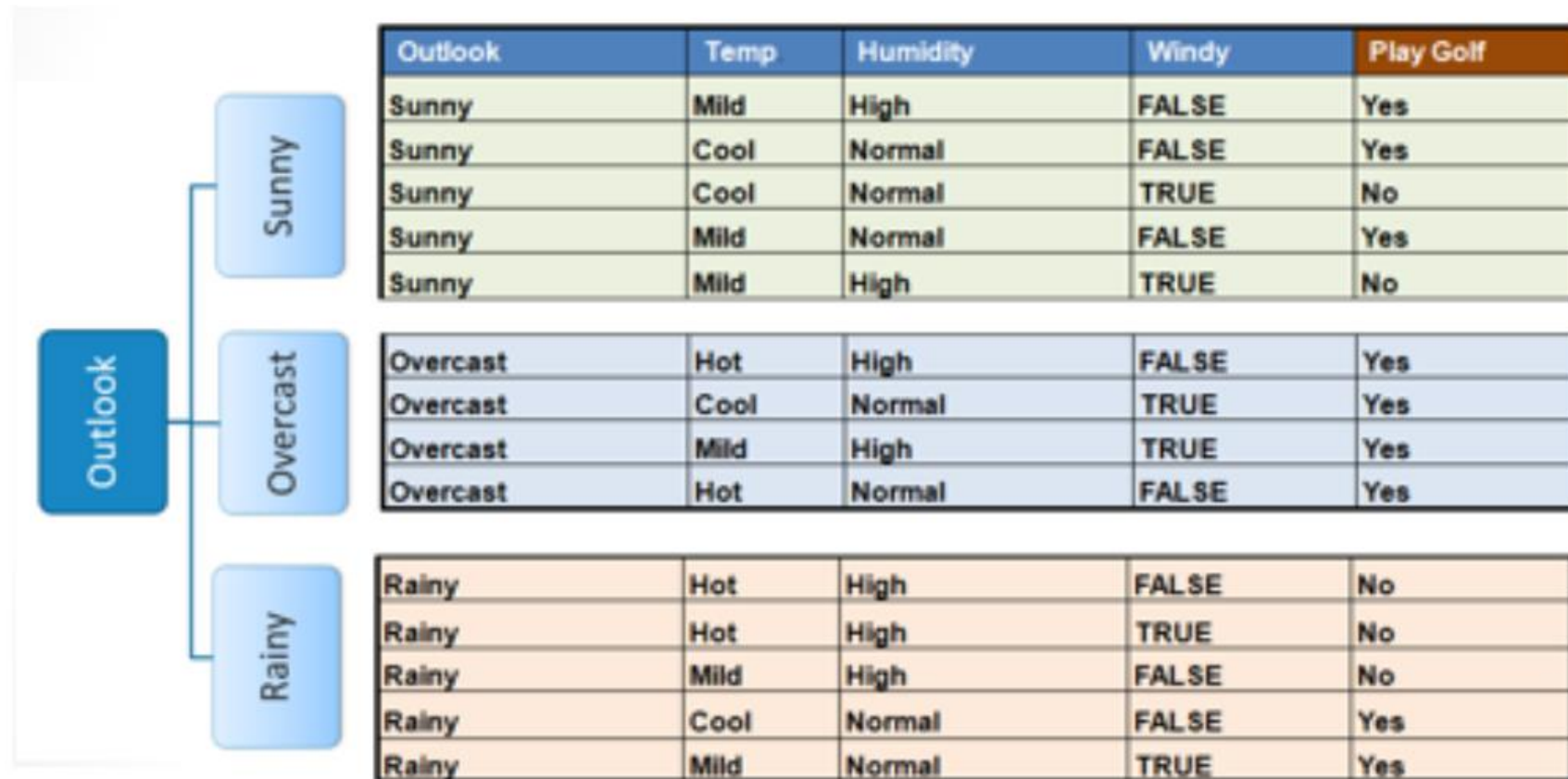
$$\begin{aligned}
 E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\
 &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\
 &= 0.693
 \end{aligned}$$

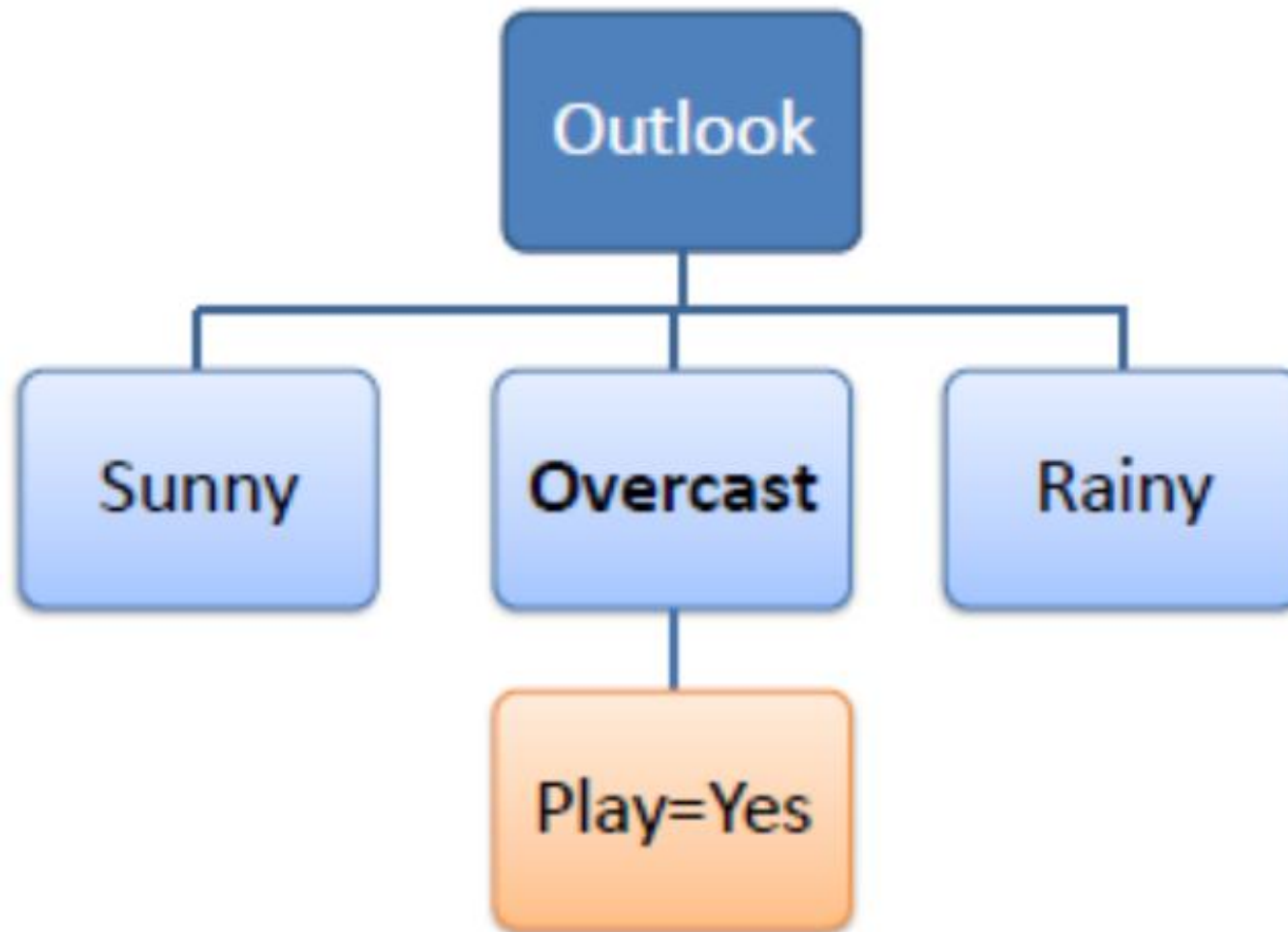
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

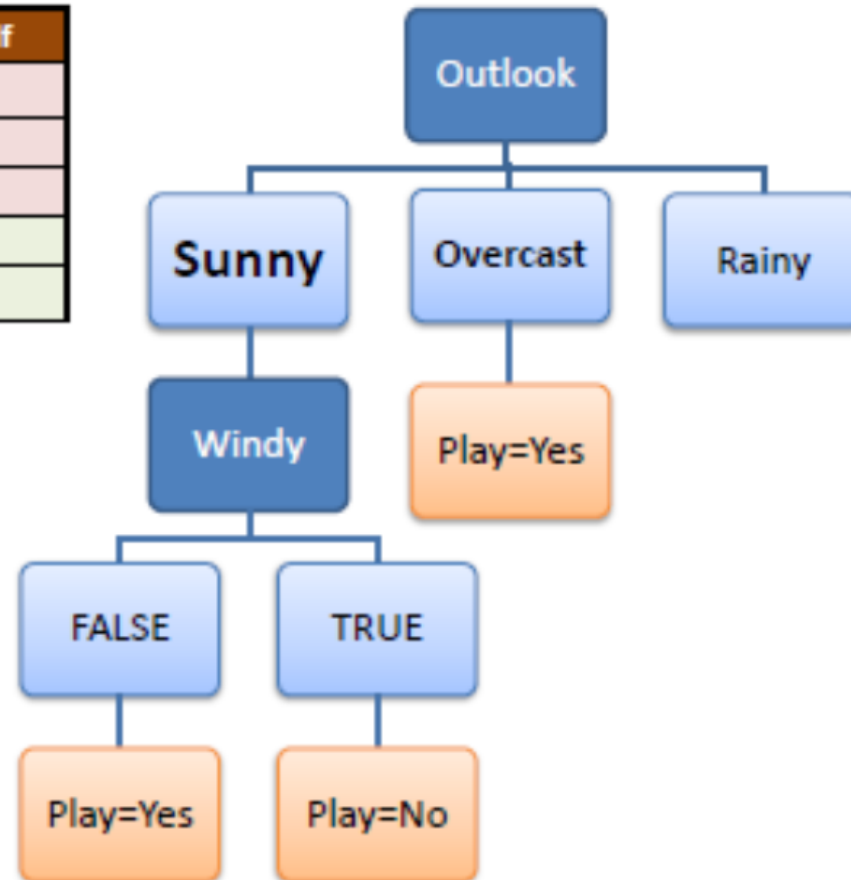
		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			





Temp.	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



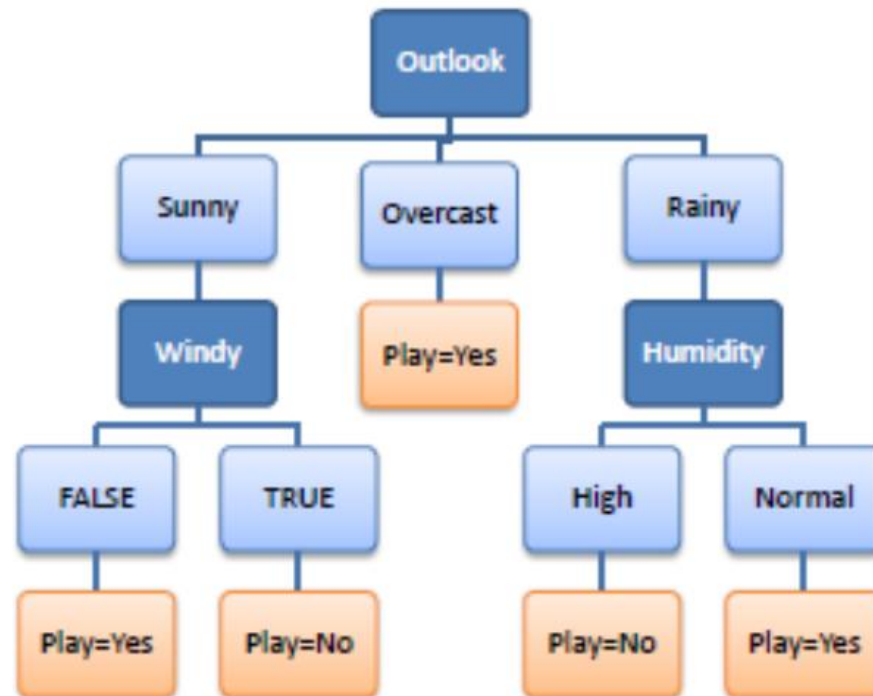
R_1 : IF (Outlook=Sunny) AND
(Windy=FALSE) THEN Play=Yes

R_2 : IF (Outlook=Sunny) AND
(Windy=TRUE) THEN Play=No

R_3 : IF (Outlook=Overcast) THEN
Play=Yes

R_4 : IF (Outlook=Rainy) AND
(Humidity=High) THEN Play=No

R_5 : IF (Outlook=Rain) AND
(Humidity=Normal) THEN
Play=Yes



$$\text{GainRatio}(T,X) = \frac{\text{Gain}(T,X)}{\text{SplitInformation}(T,X)}$$

$$\text{Split}(T, X) = - \sum_{c \in A} P(c) \log_2 P(c)$$

		Play Golf		
		Yes	No	total
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
Gain = 0.247				

$$\begin{aligned} \text{Split}(\text{Play}, \text{Outlook}) &= - (5/14 * \log_2(5/14) + 4/14 * \log_2(4/15) + 5/14 * \log_2(5/14)) \\ &= 1.577 \end{aligned}$$

$$\text{Gain Ratio}(\text{Play}, \text{Outlook}) = 0.247 / 1.577 = 0.156$$

		Play Golf		
		Yes	No	total
ID	id1	1	0	1
	id2	0	1	1
	id3	1	0	1
	id4	1	0	1
	id5	0	1	1
	id6	0	1	1
	id7	1	0	1
	id8	1	0	1
	id9	0	1	1
	id10	1	0	1
	id11	1	0	1
	id12	0	1	1
	id13	1	0	1
	id14	1	0	1

Entropy (Play Golf, ID) = 0



Gain (Play Golf, ID) = 0.94



Split (Play Golf, ID) = 3.81



Gain Ratio (Play Golf, ID) = $0.94 / 3.81 = 0.25$

- O ganho de informação é alto (= 0.94) sem a *split information* (chamado aqui de entropia do nó). No entanto, com a razão de ganho, o resultado vai para 0.25. Nesta caso, o **ID** ainda seria escolhido, quando comparado com **Outlook**, pois a divisão de amostras por nó com **Outlook** ficou quase uniforme também.

12.5 Gini

- Outra medida bastante conhecida é o *Gini*, a qual emprega um índice de dispersão proposto em 1912 pelo estatístico italiano Corrado Gini. Este índice é muito utilizado em análises econômicas e sociais, por exemplo, para quantificar a distribuição de renda em um certo país.
- Ele é utilizado no algoritmo CART (BREIMAN et al., 1984). Para um problema de c classes, o $gini_{index}$ é definido pela Equação (4), na forma:

$$gini_{index}(\text{nó}) = 1 - \sum_{i=1}^c p(i / \text{nó})^2 \quad (4)$$

- Assim como no cálculo do ganho de informação, basta calcular a diferença entre o $gini_{index}$ antes e após a divisão. Essa diferença, *Gini*, é representada pela Equação (5):

$$Gini = gini_{index}(\text{pai}) - \sum_{j=1}^n \left[\frac{N(v_j)}{N} gini_{index}(v_j) \right] \quad (5)$$

onde n é o número de valores do atributo, ou seja, o número de nós-filhos, N é o número total de objetos do nó-pai e $N(v_j)$ é o número de exemplos associados ao nó-filho v_j .

- Assim, é selecionado o atributo que gerar um maior valor para $Gini$.
- Para outras medidas, consultar ROKACH & MAIMON (2008).
- O comportamento das equações (2) e (4) está ilustrado na Figura 9 a seguir.

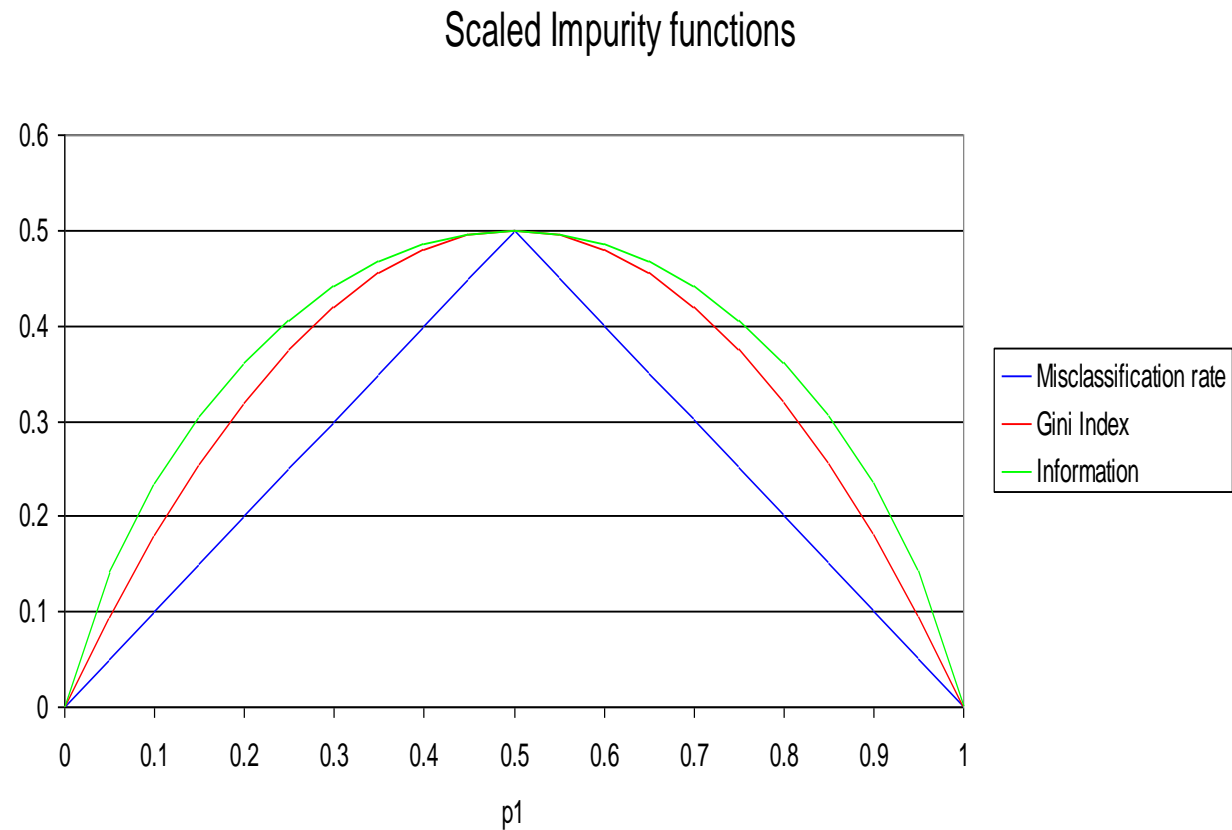


Figura 9 – Comportamento da entropia (curva verde) e do $gini_{index}$ (curva vermelha), supondo duas classes e considerando um único nó da árvore de decisão

12.6 Representação dos nós para atributos categóricos

- A forma de representação dos nós pode influenciar de maneira decisiva no desempenho das árvores de decisão induzidas.
- Dependendo do tipo de atributo, existem diferentes tipos de representação dos nós para o particionamento dos dados. A seguir, são apresentadas algumas formas de representação considerando atributos categóricos ordinais e não-ordinais. Na próxima subseção, será a vez dos atributos contínuos.

1. Um ramo para cada valor de atributo: É a partição mais comum, na qual é criada uma aresta para cada valor do atributo usado como condição de teste. Embora esse tipo de partição permita extrair do atributo todo o seu conteúdo informativo, possui a desvantagem de tornar a árvore de decisão mais complexa. O algoritmo C4.5 (QUINLAN, 1993) utiliza esse tipo de divisão para atributos categóricos não-ordinais.

2. Solução de Hunt: A partição utilizada pelo algoritmo ID3 sugere uma partição binária. Nesse caso, um dos valores é atribuído a uma das arestas e todos os outros valores à outra aresta. A desvantagem desse tipo de partição é não aproveitar todo o poder de discriminação do atributo em questão.
3. Atributos categóricos ordinais: Como já definido, um atributo é ordinal quando há uma relação de ordem entre os seus possíveis valores. Por exemplo, tem-se um atributo altura que pode possuir os valores ⟨baixa⟩, ⟨média⟩ e ⟨alta⟩. Com atributos desse tipo, é possível realizar uma partição binária do tipo altura < ⟨média⟩, em que todos os exemplos cujo atributo altura tem valor ⟨baixa⟩ seguem por uma aresta e os outros seguem por outra aresta. Esse tipo de partição é uma das que foram implementadas para o algoritmo CART (BREIMAN *et al.*, 1984).
4. Agrupamento de valores em dois conjuntos: De acordo com BREIMAN *et al.* (1984), a divisão binária também pode ser realizada de uma forma mais complexa,

onde cada um dos dois subconjuntos pode ser formado por registros com mais de um valor para o atributo utilizado como condição de teste. O grande desafio desse tipo de partição é o elevado custo computacional para encontrar a melhor divisão, pois o número de combinações possíveis é $2^{n-1} - 1$, onde n é o número de valores possíveis para o atributo em questão.

5. Agrupamento de valores em vários conjuntos: Visando permitir o agrupamento de valores em vários conjuntos com uma complexidade de cálculo razoável, o algoritmo C4.5 (QUINLAN, 1993) permite encontrar uma solução de boa qualidade. Para isso, inicia criando uma aresta para cada valor do atributo em questão. Após, são testadas todas as combinações possíveis de dois valores e, caso nenhuma dessas combinações produza um ganho maior que a divisão anterior, o processo é interrompido e a divisão anterior é adotada como divisão final. Senão, é repetido o processo tendo como base a melhor das soluções anteriores. Nota-se que não se

pode garantir que a divisão encontrada seja a melhor possível, pois é verificado se houve melhoria apenas um passo à frente.

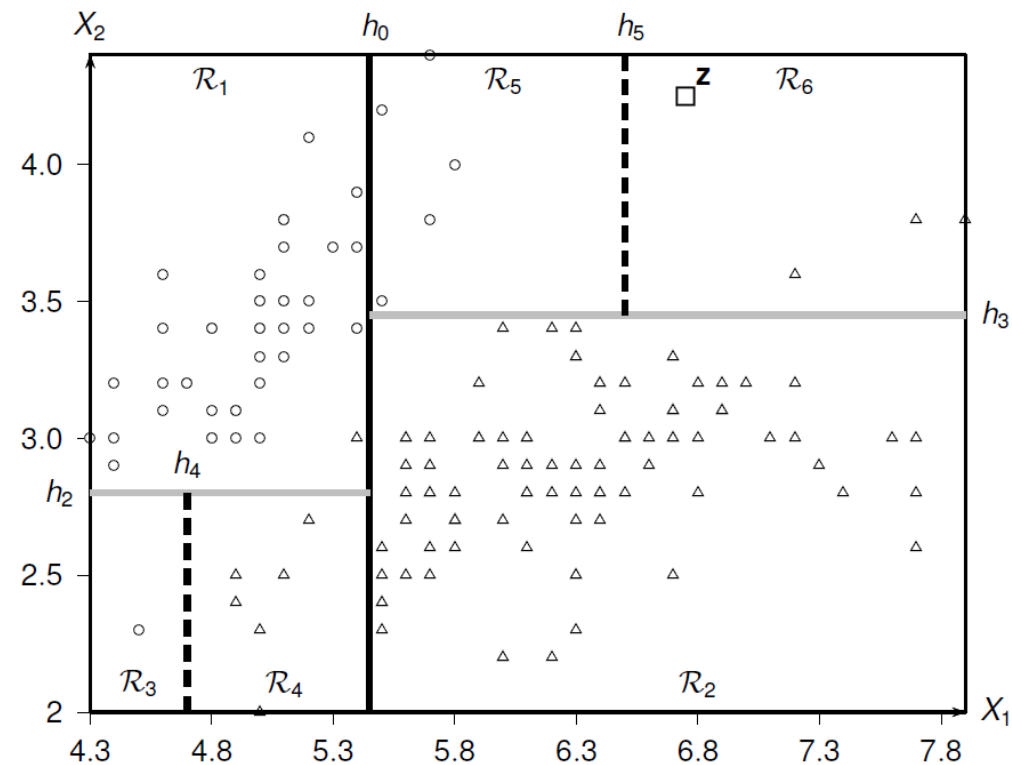
12.7 Representação dos nós para atributos contínuos

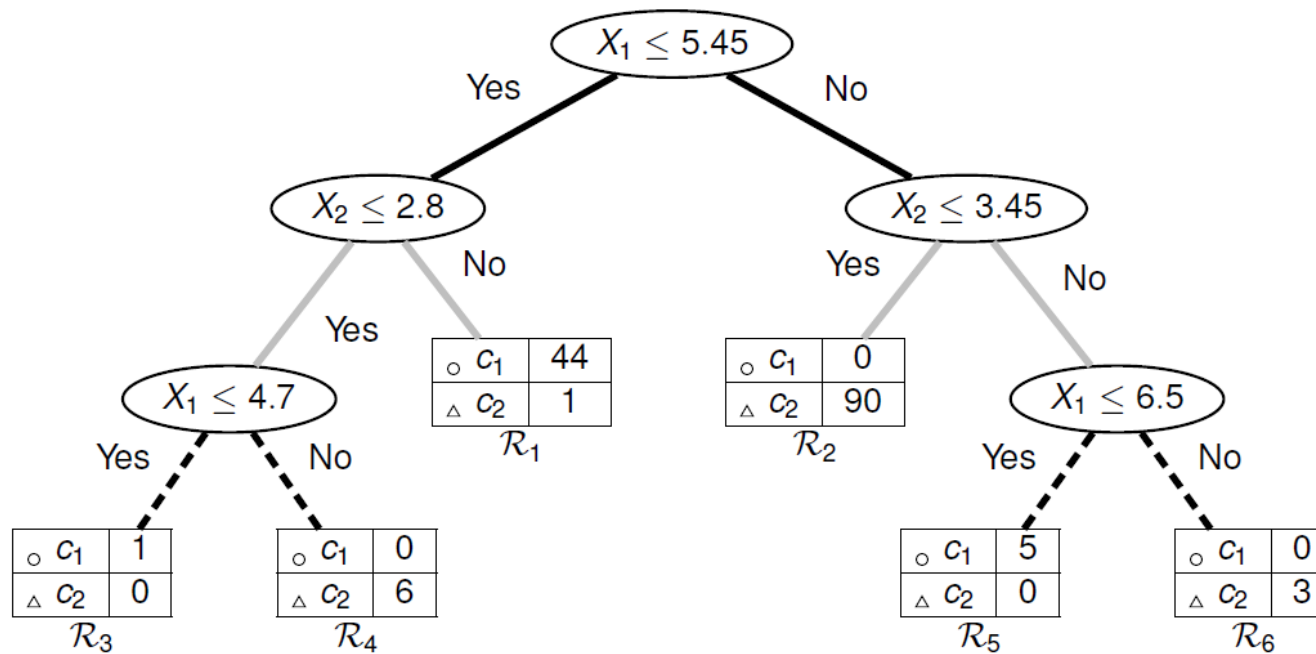
- Os atributos contínuos permitem uma maior variedade de testes e, conseqüentemente, implicam uma maior complexidade de cálculo. Segundo FONSECA (1994), alguns dos testes mais usados para partição de atributos contínuos são: testes simples ou pesquisa exaustiva, testes múltiplos (segmentação global e segmentação ao nível do nó) e combinação linear de características.
- O teste simples, também conhecido como pesquisa exaustiva, é o mais utilizado. Um dos algoritmos que o utiliza é o C4.5, e a divisão é sempre binária. Supondo um atributo contínuo X a ser utilizado como nó teste, mesmo que seu domínio seja infinito, o número de exemplos num conjunto de treinamento T é finito e, portanto, o número de valores diferentes para esse atributo também é finito.

- Assim, os exemplos do conjunto T são ordenados de acordo com seus valores para o atributo X . Supondo que os diferentes valores de X sejam, em ordem crescente, $\{a_1, a_2, \dots, a_m\}$, T é dividido em duas partes. São elas: T_1 , cujos exemplos possuem valores $\{a_1, a_2, \dots, a_i\}$ e T_2 , com valores $\{a_{i+1}, a_{i+2}, \dots, a_m\}$ para o atributo X .
- Para cada a_i , $i = 1, \dots, m-1$, é calculado o ganho (independente do critério utilizado) para a respectiva divisão. Após avaliar todas as divisões possíveis, é escolhida aquela que fornecer o maior ganho.
- Por fim, é necessário definir o valor que será usado como limiar (valor usado para dividir os exemplos no nó). Tendo posse do a_i que produziu o melhor ganho, o valor mais utilizado como limiar é $\frac{a_i + a_{i+1}}{2}$, pois assim espera-se que a árvore resultante apresente melhores resultados para exemplos que não participaram do conjunto de treinamento.

- Exemplo de resultado final para atributos contínuos, extraído de: Zaki, M.J. & Meira Jr., W. “Data Mining and Analysis: Fundamental Concepts and Algorithms”, Chapter 19, Cambridge University Press, 2014.

<http://www.dataminingbook.info/pmwiki.php/Main/HomePage>





- Já vimos no Tópico 2 (Parte 1) deste curso, que o número de partições possíveis de N pontos em k conjuntos não vazios é dado pelo número de Stirling do tipo 2 (KNUTH, 1992):

$$\frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^N$$

12.8 Métodos de poda

- Quando árvores de decisão são construídas, muitas das arestas ou sub-árvores podem refletir ruídos ou erros. Isso acarreta em um problema conhecido como sobreajuste, que significa um aprendizado muito específico do conjunto de treinamento, não permitindo ao modelo generalizar.
- Para detectar e excluir essas arestas e sub-árvores, são utilizados métodos de poda (*pruning*) da árvore, cujo objetivo é melhorar a taxa de acerto do modelo para novos exemplos, os quais não foram utilizados no conjunto de treinamento (HAN, 2001).
- Consequentemente, a árvore podada se torna mais simples, facilitando a sua interpretabilidade por parte do usuário. Junto ao método de seleção, o método de poda também varia de acordo com os diferentes algoritmos de indução de árvores de decisão.
- Existem diversas formas de realizar poda em uma árvore de decisão, e todas elas são classificadas como pré-poda ou pós-poda.

- O método pré-poda é realizado durante o processo de construção da árvore, em que o processo pode simplesmente parar de dividir o conjunto de elementos e transformar o nó corrente em um nó folha da árvore.
- O ganho de informação, por exemplo, pode ser utilizado como critério de poda. Caso todas as divisões possíveis utilizando um atributo A gerem ganhos menores que um valor pré-estabelecido, então esse nó vira folha, representando a classe mais frequente no conjunto de exemplos.
- A dificuldade é pré-estabelecer um valor adequado, visto que um valor muito alto pode gerar uma árvore super-simplificada, enquanto um valor muito baixo pode simplificar bem pouco a árvore.
- Já o pós-poda é realizado após a construção da árvore de decisão, removendo ramos completos, onde tudo que está abaixo de um nó interno é excluído e esse nó é transformado em folha, representando a classe mais frequente no ramo.

- Para cada nó interno da árvore, o algoritmo calcula a taxa de erro caso a sub-árvore abaixo desse nó seja podada. Em seguida, é calculada a taxa de erro caso não haja a poda. Se a diferença entre essas duas taxas de erro for menor que um valor pré-estabelecido, a árvore é podada. Caso contrário, não ocorre a poda.
- Esse processo se repete progressivamente, gerando um conjunto de árvores podadas. Por fim, para cada uma delas é calculada a acurácia na classificação de um conjunto de dados independente dos dados de treinamento (por exemplo, o conjunto de validação), e a árvore que obtiver a melhor acurácia será a escolhida.
- Embora a poda seja um método bastante utilizado e eficaz na solução do problema de sobreajuste, deve-se ter cuidado para não podar demais a árvore. Quando isso ocorre, tem-se o problema conhecido como sub-ajuste (*underfitting*), em que o modelo de classificação não aprendeu o suficiente sobre os dados de treinamento.

- Dentre os métodos de poda existentes, destacam-se: *Cost Complexity Pruning*, *Reduced Error Pruning*, *Minimum Error Pruning* (MEP), *Pessimistic Pruning*, *Error-Based Pruning* (EBP), *Minimum Description Length* (MDL) *Pruning*, *Minimum Message Length* (MML) *Pruning*, *Critical Value Pruning* (CVP), OPT e OPT-2.
- Maiores detalhes sobre esses métodos podem ser encontrados em ROKACH & MAIMON (2008).

13. Algoritmos de indução de árvores de decisão

- Nesta seção, são apresentados sucintamente os três principais algoritmos para indução de árvores de decisão. São eles: ID3 (QUINLAN, 1986), C4.5 (QUINLAN, 1993) e CART (BREIMAN *et al.*, 1984).
- Na literatura, existem novos algoritmos para indução de árvores de decisão, inclusive alguns que fogem do algoritmo básico TDIDT. Como exemplos, pode-se mencionar:

NBTree (KOHAVI, 1996), ADTree (FREUND & MASON, 1999), LMT (LANDWEHR *et al.*, 2005) e BFTree (SHI, 2007). Essas novas propostas não serão cobertas neste curso.

- Para uma revisão, consultar KOTSIANTIS (2013).
- Cabe mencionar que a indução de regras pode ser feita diretamente, sem necessitar de uma árvore de decisão como um passo intermediário (DONALD, 1992), como no caso do RIPPER (COHEN, 1995).

13.1 ID3

- O ID3 (QUINLAN, 1986) é o algoritmo pioneiro em indução de árvores de decisão. Ele é um algoritmo recursivo e baseado em busca gananciosa, procurando, sobre um conjunto de atributos, aqueles que “melhor” dividem os exemplos, gerando sub-árvores.
- A principal limitação do ID3 é que ele só lida com atributos categóricos não-ordinais, não sendo possível apresentar a ele conjuntos de dados com atributos contínuos, por exemplo. Nesse caso, os atributos contínuos devem ser previamente discretizados.

- Além dessa limitação, o ID3 também não apresenta nenhuma forma para tratar valores desconhecidos, ou seja, todos os exemplos do conjunto de treinamento devem ter valores conhecidos para todos os seus atributos.
- É de conhecimento geral que, na prática, os conjuntos de dados tendem a apresentar muitos valores desconhecidos. Logo, para se utilizar o ID3, é necessário gastar um bom tempo com pré-processamento dos dados.
- O ID3 utiliza o ganho de informação para selecionar a melhor divisão. No entanto, esse critério não considera o número de divisões (número de arestas), e isso pode acarretar em árvores mais complexas.
- Somado a isso, o ID3 também não apresenta nenhum método de pós-poda, o que poderia amenizar esse problema de árvores mais complexas.

13.2 C4.5

- O algoritmo C4.5 (QUINLAN, 1993) representa uma significativa evolução do ID3 (QUINLAN, 1986). As principais contribuições em relação ao ID3 são:
 - Lida tanto com atributos categóricos (ordinais ou não-ordinais) como com atributos contínuos. Para lidar com atributos contínuos, o algoritmo C4.5 define um limiar e então divide os exemplos de forma binária: aqueles cujo valor do atributo é maior que o limiar e aqueles cujo valor do atributo é menor ou igual ao limiar. Atributos contínuos podem ser usados várias vezes (em nós diferentes, numa mesma sequência de decisões), para refinar a partição.
 - Trata valores desconhecidos. O algoritmo C4.5 permite que os valores desconhecidos para um determinado atributo sejam representados como ‘?’, e o algoritmo trata esses valores de forma especial. Esses valores não são utilizados nos cálculos de ganho e entropia.

- Utiliza a medida de razão de ganho para selecionar o atributo que melhor divide os exemplos. Essa medida se mostrou superior ao ganho de informação, gerando árvores mais precisas e menos complexas.
 - Lida com problemas em que os atributos possuem custos diferenciados.
 - Apresenta um método de pós-poda das árvores geradas. O algoritmo C4.5 faz uma busca na árvore, de baixo para cima, e transforma em nós-folha aqueles ramos que não apresentam nenhum ganho significativo.
- A ferramenta de mineração de dados WEKA (WITTEN & FRANK, 1999) (<http://www.cs.waikato.ac.nz/~ml/weka/index.html>) (*Waikato Environment for Knowledge Analysis*) disponibiliza a implementação do algoritmo C4.5, porém o mesmo é chamado de J48 nessa ferramenta.
 - Cabe destaque para o autoWEKA (<https://arxiv.org/abs/1208.3719>), que realiza a otimização de hiperparâmetros de todas as técnicas disponíveis.

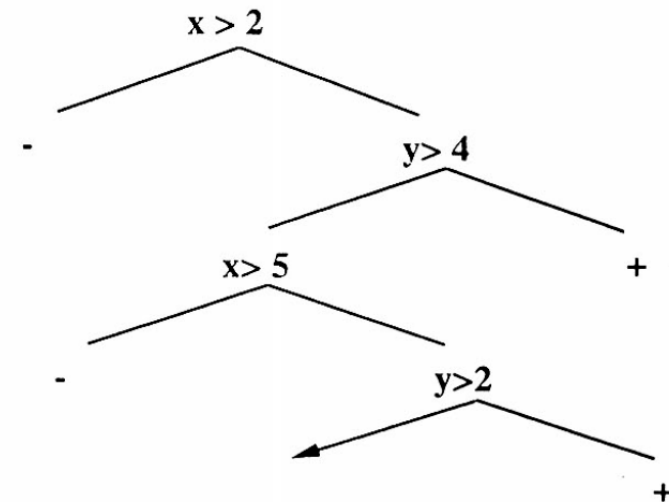
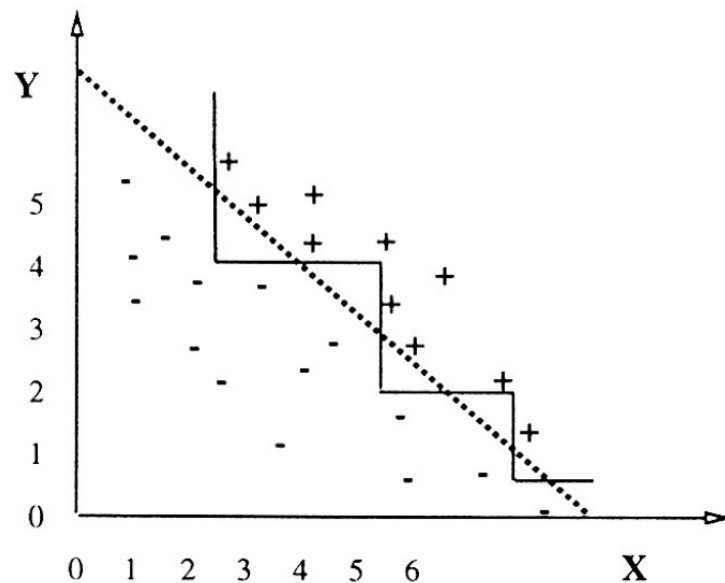
- O C4.5 é um dos algoritmos mais utilizados na literatura, por ter mostrado ótimos resultados em problemas de classificação. Embora já tenha sido lançado o C5.0, o C4.5 possui código-fonte disponível, enquanto o C5.0 é um software comercial.
- O C4.5 é do tipo:
 - Ganancioso: executa sempre o melhor passo avaliado localmente, sem se preocupar se este passo, junto à sequência completa de passos, vai produzir a melhor solução ao final;
 - “Dividir para conquistar”: partindo da raiz, criam-se sub-árvores até chegar nas folhas, o que implica em uma divisão hierárquica em múltiplos sub-problemas de decisão, os quais tendem a ser mais simples que o problema original.

13.3 CART

- O algoritmo CART (*Classification and Regression Trees*) foi proposto em BREIMAN *et al.* (1984) e consiste de uma técnica não-paramétrica que induz tanto árvores de classificação quanto árvores de regressão, dependendo se o atributo é nominal (classificação) ou contínuo (regressão).
- Dentre as principais virtudes do CART está a grande capacidade de pesquisa de relações entre os dados, mesmo quando elas não são evidentes, bem como a produção de resultados sob a forma de árvores de decisão de grande simplicidade e legibilidade (FONSECA, 1994).
- As árvores geradas pelo algoritmo CART são sempre binárias, as quais podem ser percorridas da sua raiz até as folhas respondendo apenas a questões simples do tipo “sim” ou “não”.

- Os nós que correspondem a atributos contínuos são representados por agrupamento de valores em dois conjuntos. Da mesma forma que no algoritmo C4.5, o CART utiliza a técnica de pesquisa exaustiva para definir os limiares a serem utilizados nos nós para dividir os atributos contínuos.
- Adicionalmente, o CART dispõe de um tratamento especial para atributos ordenados e também permite a utilização de combinações lineares entre atributos (agrupamento de valores em vários conjuntos).
- Diferente das abordagens adotadas por outros algoritmos, os quais utilizam pré-poda, o CART expande a árvore exaustivamente, realizando pós-poda por meio da redução do fator custo-complexidade (BREIMAN *et al.*, 1984).
- Segundo os autores, a técnica de poda utilizada é muito eficiente e produz árvores mais simples, precisas e com boa capacidade de generalização.

- A figura a seguir, extraída de BRODLEY & UTGOFF (1995), ilustra o grande potencial de redução do tamanho da árvore. Na abordagem multivariada (também chamada oblíqua), a árvore tem apenas um nó interno ($x + y \geq 8$?) e duas folhas, enquanto na abordagem univariada, a árvore da figura nem está completa e já apresenta 4 nós internos.



14. Referências bibliográficas

- BASGALUPP, M.P. (2010) LEGAL-Tree: Um algoritmo genético multi-objetivo lexicográfico para indução de árvores de decisão. Tese de Doutorado, ICMC-USP, São Carlos.
- BRAMER, M. (2007). Principles of data mining. Springer, London.
- BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., & STONE, C. J. (1984). Classification and Regression Trees. Wadsworth.
- BRODLEY, C.E. & UTGOFF, P.E. (1995). Multivariate Decision Trees. Machine Learning, 19: 45-77.
- CHAPELLE, O., SCHÖLKOPF, B., & ZIEN, A. (2006). Semi-supervised learning. MIT Press.
- COHEN, W.W. (1995). Fast Effective Rule Induction. Proceedings of the 12th International Conference on Machine Learning, pp. 115-123.
- DONALD, J.H. (1992). Rule Induction: Machine Learning Techniques for Data Analysis, Classification and Knowledge Elicitation, ERA Technology Ltd.

- FERREIRA, H.M. (2008) Uso de ferramentas de aprendizado de máquina para prospecção de perdas comerciais em distribuição de energia elétrica, Dissertação de Mestrado, FEEC/Unicamp.
- FONSECA, J. (1994). Indução de árvores de decisão. Tese de Mestrado, Lisboa.
- FREUND, Y. & MASON, L. (1999). The alternating decision tree learning algorithm. In Proc. 16th International Conf. on Machine Learning, pp. 124-133. Morgan Kaufmann, San Francisco, CA.
- HAN, J. (2001). Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- HYAFIL, L. & RIVEST, R.L. (1976). Constructing optimal binary decision trees is NP-complete, Information Processing Letters, 5(1): 15-17.
- KNUTH, D.E. (1992). Two notes on notation, The American Mathematical Monthly, 99(5): 403-422.
- KOHAVI, R. (1996). Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 202-207.

- KOTSIANTIS, S.B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261-283.
- LANDWEHR, N., HALL, M., & FRANK, E. (2005). Logistic model trees. *Machine Learning*, 59(1-2):161-205.
- QUINLAN, J. R. (1993). C4.5: programs for machine learning. Morgan Kaufmann Publishers.
- QUINLAN, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81-106.
- QUINLAN, J. (1988). Decision trees and multivalued attributes. *Machine Intelligence*, 11:305-318.
- ROKACH, L. & MAIMON, O. (2008). Data mining with decision trees. Theory and applications. World Scientific Publishing.
- SHI, H. (2007). Best-first decision tree learning. Master's Thesis, University of Waikato, COMP594.
- TAN, P.-N., STEINBACH, M., & KUMAR, V. (2005). Introduction to Data Mining, (First Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- WITTEN, I. H. & FRANK, E. (1999). Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann.