Aula 04 - Classificação k-NN, Modelo de Núcleo e Árvores de Decisão

Clodoaldo A. M. Lima

09 de setembro de 2021

Programa de Pós-Graduação em Sistemas de Informação
Mestrado acadêmico - EACH - USP
http://ppgsi.each.usp.br

Classificação / Regressão Predição categórica / numérica

Racionínio indutivo e dedutivo

Argumento dedutivo

Procede de proposições (mais) universais para proposições particulares. Conclusões são extraídas a partir de premissas.

- Todo homem é mortal.
- João é homem.
- Logo, João é mortal

Tem-se um modelo (Todo homem é mortal) e um fato (João é homem) e a partir daí gera-se conclusões.

Argumento indutivo

Procede de proposições particulares para proposições (mais) universais.

- O ferro conduz eletricidade.
- O ouro conduz eletricidade.
- A prata conduz eletricidade.
- O chumbo conduz eletricidade.
- Logo, todo metal conduz eletricidade.

Tem-se uma série de fatos e a partir deles cria-se o modelo. O modelo, então, pode a partir de um novo fato, gerar conclusões que concordam com os fatos primeiros.

Indução e Aprendizado de Máquina

- **Aprendizado de Máquina:** Um programa de computador aprende se ele é capaz de melhorar seu desempenho em determinada tarefa, sob alguma medida de avaliação, a partir de experiências passadas. (Tom Mitchell)
- **Aprendizado de Máquina:** Um processo de indução de uma hipótese (ou aproximação de uma função) a partir da experiência passada. (André P. L. Carvalho)
- **Tarefa de indução:** Dada uma coleção de exemplos de uma função F, deve-se retornar uma função H que se aproxime de F. Nessa definição, a função H é uma hipótese e a função F é desconhecida. (Russel e Norvig)

Nesse sentido, pode-se afirmar que um processo algorítmico capaz de processar um conjunto de observações e formular uma regra geral que as explique, pode ser considerado um algoritmo de Aprendizado de Máquina.

Assim Aprendizado de Máquina pode ser considerado um paradigma. Neste paradigma ainda existem as subdivisões: aprendizado supervisionado e aprendizado não-supervisionado.

Importante notar que existem técnicas (as conhecidas técnicas de Aprendizado de Máquina) que, em sua concepção, implementam o raciocínio indutivo: redes neurais artificiais, k-NN, k-means ... Essas técnicas precisam de um conjunto de *dados de treinamento**.

Aprendizado supervisionado e a tarefa de Classificação

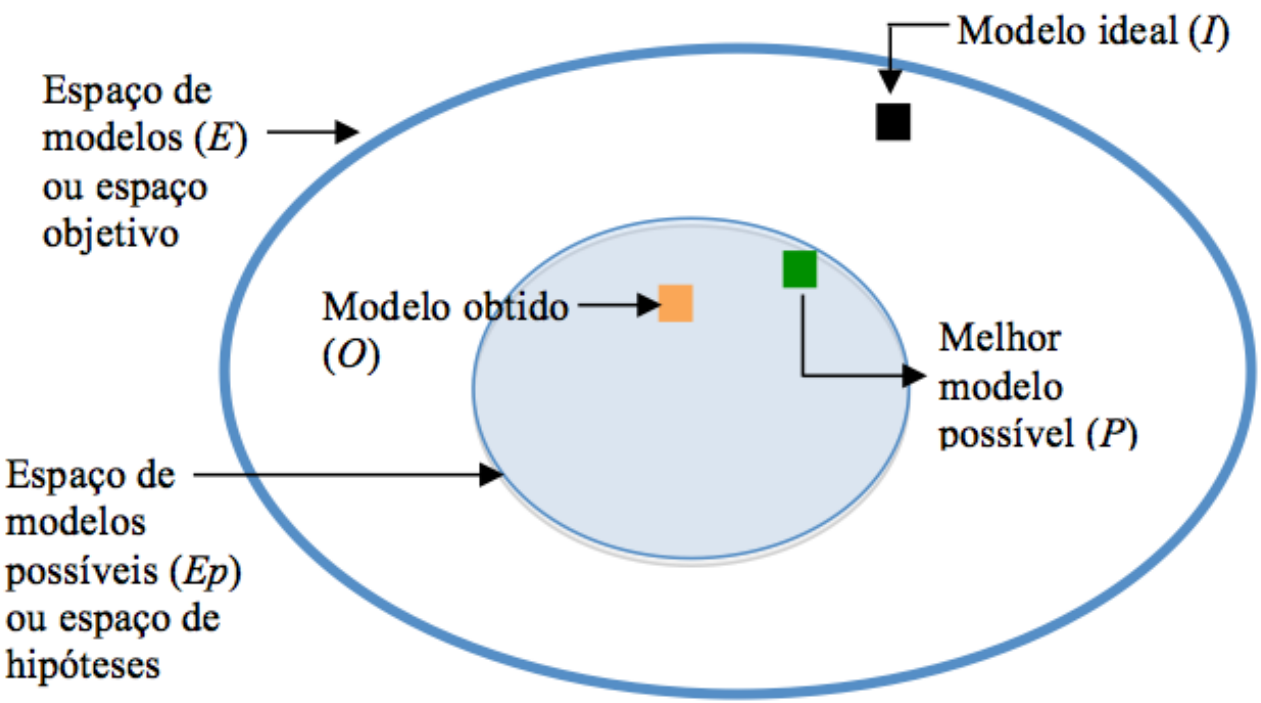
Classificação

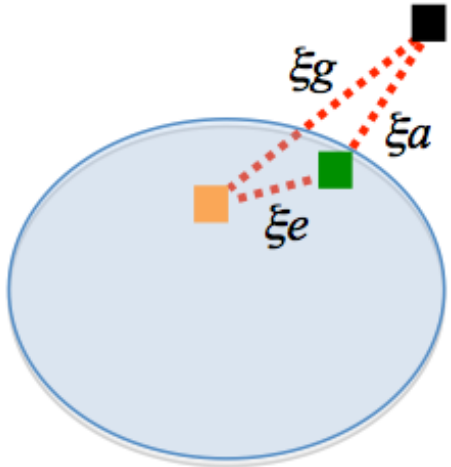
Comumente se denomina de classificação, o processo pelo qual se determina um mapeamento capaz de indicar a qual classe pertence qualquer exemplar de um domínio sob análise, com base em um conjunto de dados já classificado.

Os algoritmos que resolvem a tarefa de classificação pressupõem a correção do erro. Se há correção de erro então é porque há a SUPERVISÃO.

A tarefa de classificação pode ser dividida em, pelo menos, duas categorias: classificação binária e classificação multiclasse. Na classificação binária, a cardinalidade c é 2 e para os casos onde $c > 2$, o problema é dito ser de múltiplas classes.

Espaço de hipóteses e erros





ξg - Erro de generalização
ξa - Erro de aproximação
ξe - Erro de estimação

k-Nearest-Neighbor (K-NN)

k-Nearest-Neighbor

O método de classificação *k*-vizinhos-mais-próximos (*k*-NN do inglês *k-Nearest-Neighbor*) foi criado no início dos anos 50. Trata-se de um método de intenso trabalho quando um conjunto de treinamento muito grande é fornecido.

É classificado como um método de aprendizado *lazy*, pois ele gasta mais tempo no momento de fazer uma classificação ou predição do que no momento de aprendizado. Seu aprendizado é na realidade o estoque (armazenamento) das instâncias de treinamento.

Princípio

É baseado no aprendizado por analogia, ou seja, ele compara uma dada tupla de teste com as tuplas de treinamento que são similares a ela.

As tuplas de treinamento são descritas por n atributos, e assim, representam um ponto em um espaço n-dimensional. Todas as tuplas de treinamento são armazenadas em um espaço de padrões n-dimensional.

Quando uma tupla desconhecida é apresentada, o *k*-NN busca, no espaço de padrões, pelas k tuplas de treinamento mais próximas à tupla apresentada. Estas k tuplas são os k vizinhos mais próximos.

Então, a tupla desconhecida é associada à classe mais comum entre as classes dos seus k vizinhos mais próximos.

K-Nearest-Neighbor

Quando $k = 1$, a tupla desconhecida é associada à classe da tupla de treinamento mais próxima à ela no espaço de padrões.

Regressão (ou Predição numérica)

K-NN pode ser usado para resolver a tarefa de regressão, ou seja, retornar um valor real para uma dada tupla desconhecida. Neste caso, o método retorna o valor médio dos valores reais associados com aos k vizinhos mais próximos da tupla desconhecida.

Proximidade ou similaridade

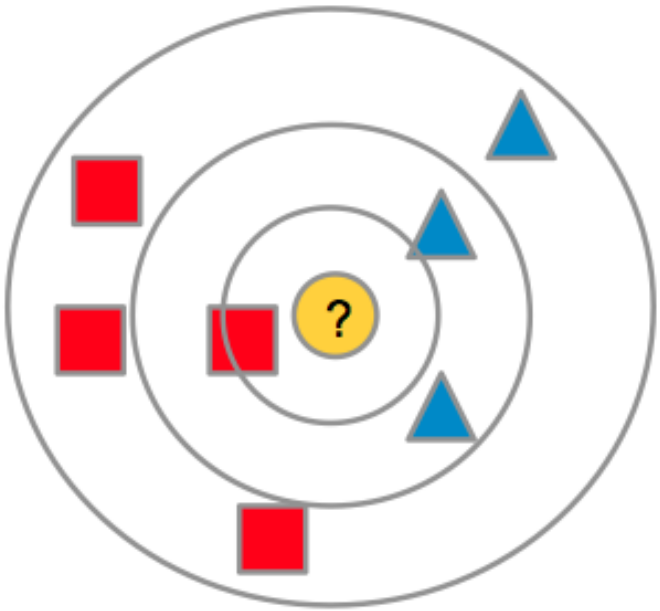
A "tupla mais próxima" é definida em termos de uma métrica de distância, por exemplo a distância Euclidiana. A distância Euclidiana entre dois pontos (ou tuplas) $X_1 = (x_{11}, x_{12}, ..., x_{1n})$ e $X_2 = (x_{21}, x_{22}, ..., x_{2n})$ é

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

K-Nearest-Neighbor

Qual é a classe?

- K = 1 : Pertence a classe de quadrados.
- K = 3 : Pertence a classe de triângulos.
- K = 7 : Pertence a classe de quadrados.



K-Nearest-Neighbor

E se os atributos não forem numéricos, mas sim categóricos?

Uma possibilidade é comparar os valores correspondentes do atributo na tupla X_1 com os valores na tupla X_2. Se os dois valores forem idênticos, então a diferença entre os dois é dita 0. Caso contrário, a diferença é considerada 1.

E se existirem valores faltantes?

Supondo que cada atributo tenha sido mapeado para o intervalo [0,1].

Em geral, se o valor de um dado atributo A é faltante em uma tupla ou nas duas, assume-se a máxima diferença possível.

Para atributos categóricos, a diferença máxima é 1.

Para valores numéricos, se o valor de A é faltante em ambas as tuplas, então a diferença máxima é 1. Se somente um dos valores é faltante, e o outro v está presente e normalizado, então a diferença é $|1 - v|$ ou $|0 - v|$, o que for maior.

K-Nearest-Neighbor

Como determinar k

Este parâmetro pode ser determinado experimentalmente. Iniciando com $k = 1$, usa-se um conjunto de teste para estimar a taxa de erro do classificador. Este processo pode ser repetido incrementando k para permitir mais vizinhos (um a mais a cada vez). O valor de k que fornecer a mínima taxa de erro deve ser selecionado.

Modificações

- incorporar pesos aos atributos;
- podar tuplas ruidosas (ou inúteis) do conjunto de treinamento;
- escolher outras métricas de distância;
- calcular a distância com base em um subconjunto de atributos (para melhorar a velocidade): se a distância no conjunto reduzido de atributos excede um limiar, então já não é mais necessário continuar computando a distância com os demais atributos, pois os dois vetores já estão suficientemente distantes e o processo segue para a comparação com o próximo vizinho;

Modelo de núcleo (by Russel e Norvig)

Modelo de núcleo

Em um modelo de núcleo, cada instância de treinamento é analisada a partir de uma geração de uma função de densidade - uma função de núcleo – de si mesma. A estimativa de densidade como um todo é a soma normalizada de todas as pequenas funções núcleo.

Cada instância de treinamento em (x_i) gera uma função núcleo $K(\mathbf{x}, \mathbf{x}_i)$ que atribui uma probabilidade para cada ponto $\mathbf{x}$, da seguinte forma:

$$P(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{x}, \mathbf{x}_i).$$

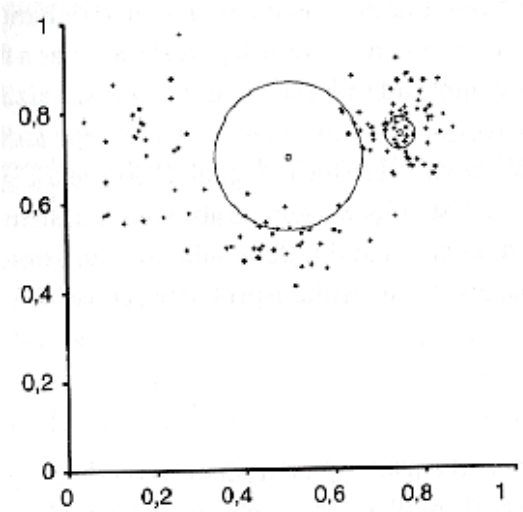
A função de núcleo normalmente depende da distância entre $\mathbf{x}$ e $\mathbf{x}_i$, D($\mathbf{x}$, $\mathbf{x}_i$). E uma função núcleo popular é o gaussiano. Supondo gaussianos esféricos com desvio-padrão σ ao longo de cada eixo, temos:

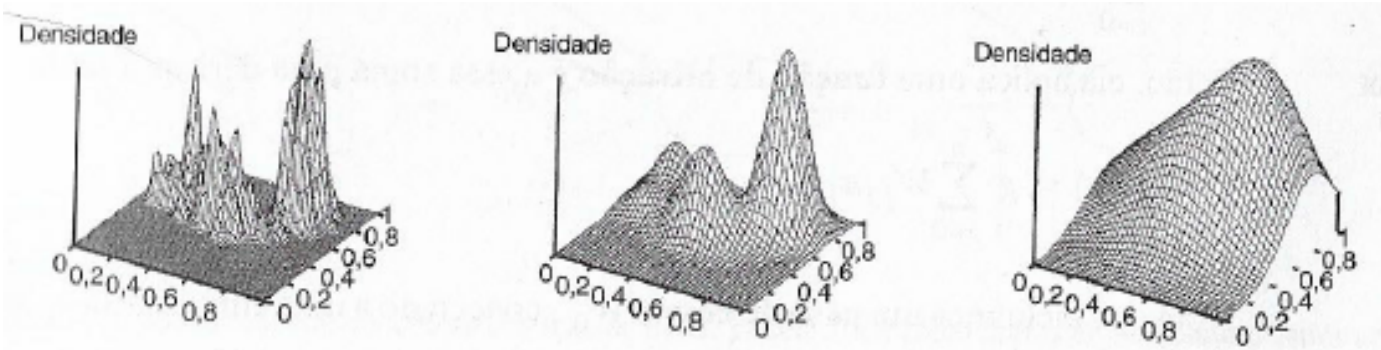
$$K(\mathbf{x}, \mathbf{x}_i) = \frac{1}{(\sigma^2\sqrt{2\pi})^d} e^{-\frac{D(\mathbf{x},\mathbf{x}_i)^2}{2\sigma^2}}$$

em que d é a dimensão de $\mathbf{x}$.

Modelo de núcleo

Para esse caso, há o problema de escolher o valor para σ. Veja exemplos com variando σ em $[0,02; 0,07; 0,2]$.





Modelo de núcleo

O aprendizado supervisionado neste contexto:

- o aprendizado surpervisionado com núcleos é feito tomando-se uma combinação ponderada de todas as previsões das instâncias de treinamento;
- o peso da i-ésima instância para um ponto de consulta $\mathbf{x}$ é dado pelo valor do núcleo $K(\mathbf{x}, \mathbf{x}_i)$;
- para uma previsão discreta, adota-se um voto ponderado;
- para uma previsão contínua, toma-se a média ponderada;

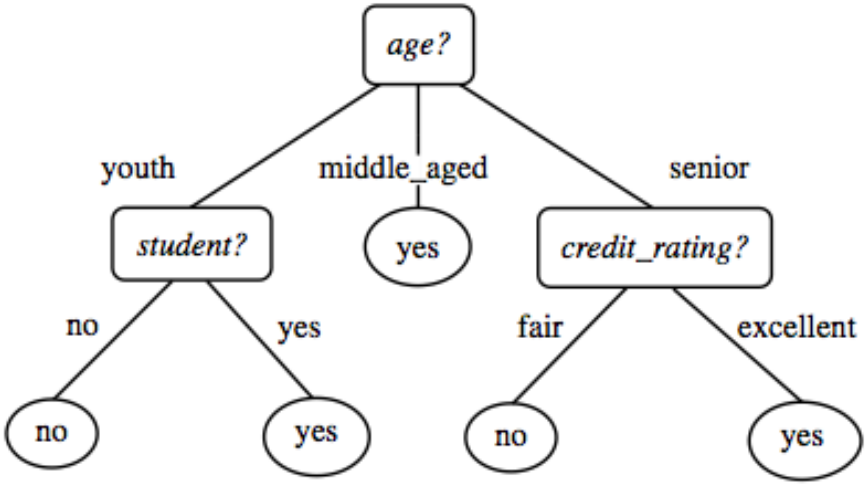
Árvores de Decisão

Árvores de Decisão

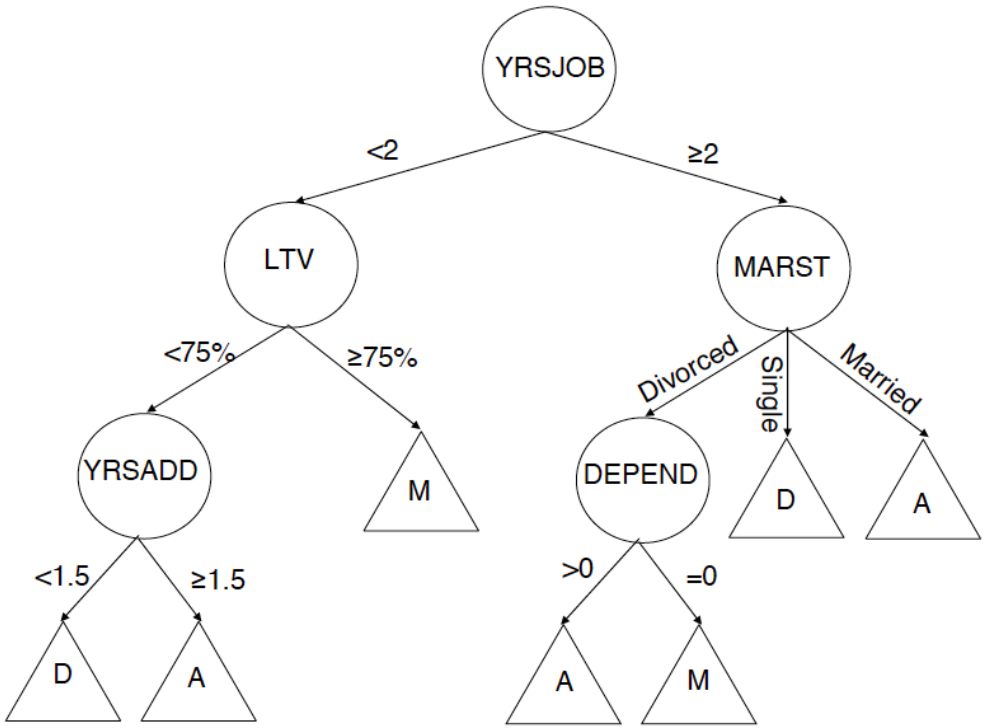
Árvore de decisão

Estrutura hierárquica onde cada **nó interno** (nó não folha) representa um teste em um atributo, cada **ramo** representa uma saída do teste, e cada **nó folha** (ou nó terminal) representa o rótulo de uma classe.

Indução de árvore de decisão é o aprendizado (construção) de árvores de decisão a partir de tuplas de treinamento rotuladas (com informação de classe).



Árvores de Decisão: pedido de hipoteca



- **DEPEND**: número de dependentes; **LTV**: razão *loan-to-value*; **MARST**: estado civil; **PAYINC**: razão *payment-to-income*; **RATE**: taxa de interesse; **YRSADD**: anos do endereço atual; **YRSJOB**: anos no emprego atual;
- **A**: aprovado; **D**: negado; **M**: análise requisitada

Árvores de Decisão

Como árvores de decisão são usadas para classificação?

Dada uma tupla, **X**, para a qual não se conhece o valor do rótulo (classe), os valores dos atributos na tupla são testados na (através da) árvore de decisão. Um caminho é traçado a partir da raíz até um nó folha, o qual tem a predição de classe para aquela tupla.

Árvores de decisão *versus* regras de classificação

Árvores de decisão podem ser facilmente convertidas em regras de classificação. Cada caminho da raíz da árvore até uma de suas folhas pode ser transformado em uma regra (conjunção dos testes como **premissas** e folha no final do caminho como **conclusão**).

Árvores de Decisão

Por que as árvores de decisão são tão populares?

- sua construção não exige conhecimento sobre o domínio, ou determinação (calibração) de parâmetros;
- podem lidar com dados em altas dimensões;
- o conhecimento construído na árvore é altamente acessível;
- os passos de indução e classificação são rápidos;
- em geral, têm boa acurácia (?).

Ávores de Decisão - indução (treinamento)

Considere:

- $B(A \cup y)$ como um esquema, sendo A um conjunto de atributos de entrada (candidatos) contendo n atributos, $A = \{a_1, ..., a_i, ..., a_n\}$, e y um atributo alvo (variável classe).
- $dom(a_i) = \{v_{i,1}, v_{i,2}, ..., v_{i,|dom(ai)|}\}$, em que $|dom(a_i)|$ tem cardinalidade finita para atributos categóricos e infinita para atributos numéricos.
- $dom(y) = \{c_1, ..., c_{|dom(y)|}\}$, o conjunto de classes possíveis.
- $X = dom(a_1) \times dom(a_2) \times ... \times dom(a_n)$, o espaço de instâncias.
- $U = X \times dom(y)$, o conjunto universo de instâncias rotuladas.
- $D(B) = (\langle x_1, y_1 \rangle, ..., \langle x_m, y_m \rangle)$, em que $x_q \in X$ e $y_q \in dom(y)$, é o conjunto de treinamento com m tuplas.

Usualmente, assumimos que o conjunto de treinamento é gerado randomicamente e independentemente de acordo com algum distribuição de probabilidade D sobre o conjunto universo U.

Ávores de Decisão

Algoritmos para indução de árvores de decisão:

- ID3 - Iterative Dichotomiser (J. Ross Quinlan – 70s e 80s). Usa a medida **ganho de informação**.
- C4.5 - sucessor do ID3 (J. Ross Quinlan). Usa a medida **raio do ganho de informação**.
- CART - Classification and Regression Trees (L. Breiman, J. Friedman, R. Olshen e C. Stone). Usa a medida **índice Gini**.

Estes algoritmos adotam uma abordagem gulosa (i.e. sem backtracking), de forma top-down, recursiva, sob a estratégia dividir-para-conquistar.

Árvores de Decisão

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

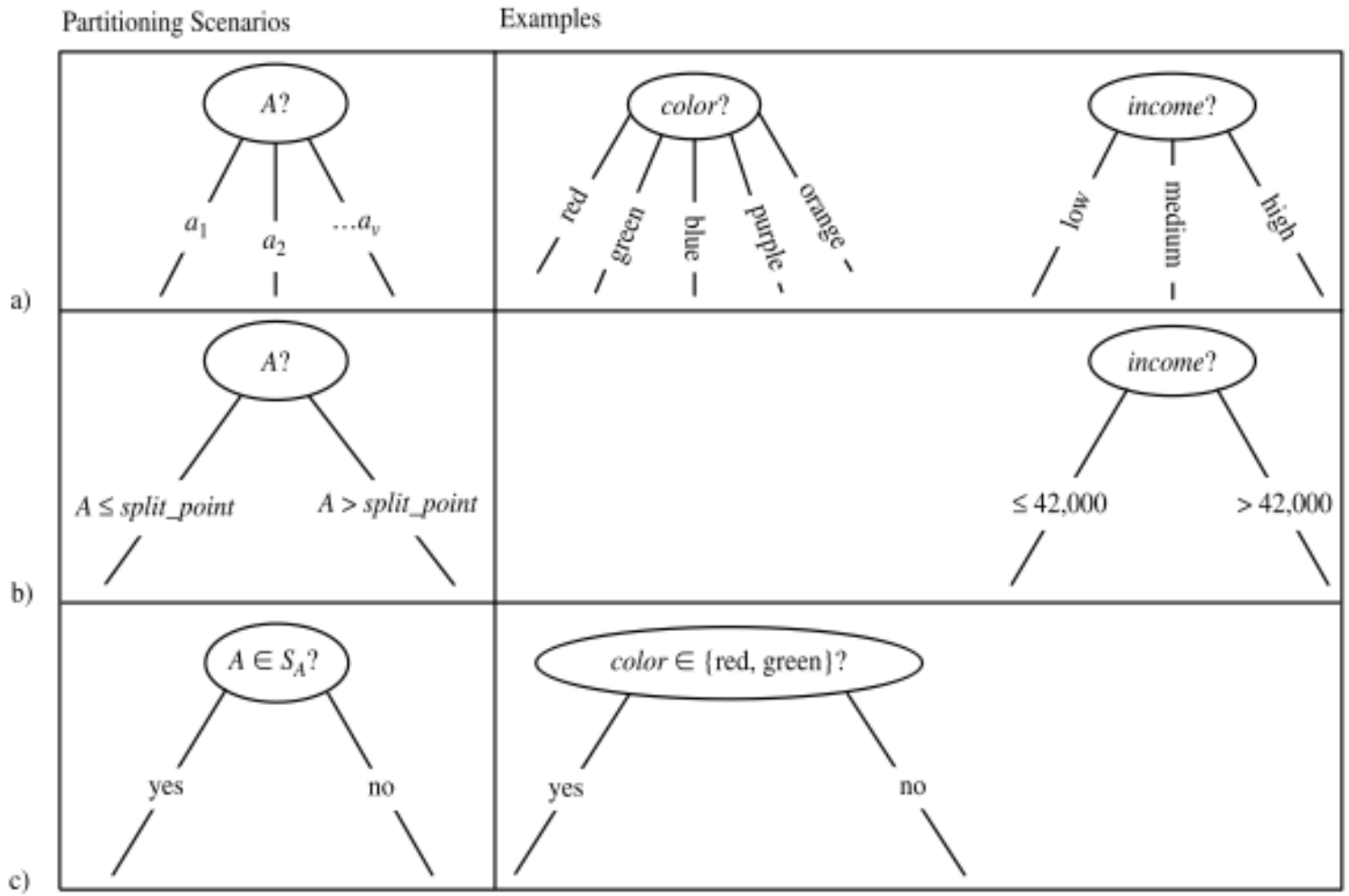
Output: A decision tree.

Árvores de Decisão

Method:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C **then**
(3) return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in D to node N;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
endfor
(15) return N;

Árvores de Decisão



Árvores de Decisão

Condições de parada do particionamento recursivo:

- Todas as tuplas no subconjunto D (representado no nó N) pertencem à mesma classe (passos 2 e 3), ou
- não existem atributos restantes no quais as tuplas devem ser ainda particionadas (passo 4). Neste caso, **voto majoritário** é aplicado (passo 5). Isto envolve converter o nó N em uma folha e nomeá-lo com a classe mais comum em D. Alternativamente, a distribuição de classes das tuplas do nó podem ser usadas.
- não existem tuplas para um dado ramo, isto é, um subconjunto D_j é vazio (passo 12). Neste caso, uma folha é criada com a classe majoritário em D (passo 13).

Outros critérios possíveis - para outras implementações

- a profundidade máxima da árvore foi alcançada;
- o número de casos em um nó terminal é menor do que o número mínimo de casos para nós pais (pode ser visto como um método de pré-poda);
- se um nó fosse dividido, o número de casos em um ou mais nós filhos seria menor do que o número mínimo de casos para nós filhos (pode ser visto como um método de pré-poda);
- o melhor valor para o critério de divisão não é maior do que um limiar estabelecido (pode ser visto como um método de pré-poda).

Árvores de Decisão

Um teste de mesa ...

Árvores de Decisão

Conjunto de dados

Class-labeled training tuples from the *AllElectronics* customer database.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

linha 1: criação de um nó (chamemos de *Nó 1*)
linhas 2 e 3: as tuplas em D não são todas da mesma classe, então continue ...
linhas 4 e 5: a attribute_ list não está vazia, então continue ...
linha 6: calculando o ganho de informação (age = 0.246; income = 0.029; student = 0.151; credit_ rating = 0.048) $\Rightarrow$ atributo de mais alto ganho é **age**;
linha 7: *Nó 1* $\Leftarrow$ **age**; **linha 8**: attribute_ list = { income, student, credit_ rating}
linha 9: os valores de **age** são: *youth, middle_ aged, senior*
linha 11: D_{youth} = 5 tuplas;
linha 12 e 13: D_{youth} não é vazio, então continue ...
linha 14: crie um filho chamando a função Generate_ decision_ tree(D_{youth},{ income, student, credit_ rating})
linha 11: D_{middle_aged} = 5 tuplas;
linha 12 e 13: D_{middle_aged} não é vazio, então continue ...
linha 14: crie um filho chamando a função Generate_ decision_ tree(D_{middle_aged},{ income, student, credit_ rating})
linha 11: D_{senior} = 5 tuplas;
linha 12 e 13: D_{senior} não é vazio, então continue ...
linha 14: crie um filho chamando a função Generate_ decision_ tree(D_{senior},{ income, student, credit_ rating})
linha 15: retorne a árvore.

Árvores de Decisão - Avaliação

Erro de Generalização

Seja $DT(S)$ uma árvore de decisão para classificação treinada no conjunto de dados S. O erro de generalização de $DT(S)$ é sua probabilidade de classificar erroneamente uma instância selecionada de acordo com a distribuição D do espaço de instâncias rotuladas. A acurácia de classificação de uma árvore é $1-$ o erro de generalização. Podemos definí-lo como:

$$\epsilon(DT(S), D) = \sum_{\langle x,y \rangle \in U} D(x, y) \cdot L(y, DT(S)(x))$$

onde x é uma instância; y é o rótulo (ou classe); U é o produto cartesiano de todos os domínios de atributos de entrada e domínios do atributo de rótulo; D é uma distribuição de probablidade sobre U; $L(y, DT(S)(x))$ é uma função de perda binária definida como:

$$L(y, DT(S)(x)) = \begin{cases} 0 \text{ if y} = DT(S)(x); \\ 1 \text{ if y} \neq DT(S)(x). \end{cases}$$

Árvores de Decisão - Avaliação

Erro de Treinamento

O erro de treinamento é definido como a porcentagem de exemplos no conjunto de treinamento corretamente classificados pela árvore.

Formalmente

$$\hat{\epsilon}(DT(S), S) = \sum_{<x,y>\in S} L(y, DT(S)(x))$$

Árvores de Decisão - Complexidade

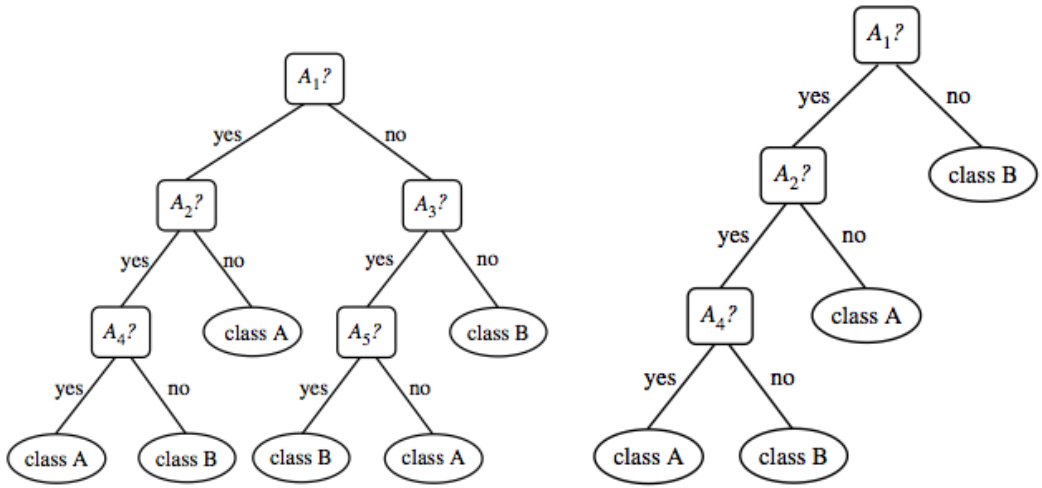
A complexidade da árvore de decisão tem efeito sobre a sua acurácia. Usualmente, os seguintes fatores são usados para medir a complexidade de uma árvore de decisão:

- o número total de nós;
- o número total de folhas;
- a profundidade da árvore;
- o número de atributos usados;

A complexidade da árvore de decisão é controlada pelos critérios de parada do seu processo de indução. Métodos de poda (*pruning*) podem ser usados.

Árvores de Decisão - *Pruning*

Usar critérios de parada muito restritos tendem a criar árvores pequenas, causando *underfitting*. Por outro lado, os ramos da árvore de decisão podem refletir anomalias existentes nos dados de treinamento – ruído e outliers, causando *overfitting*. Métodos de poda (*prunning*) previnem esses problemas.



(a) Sem poda (b) Com poda

Abordagens comuns: **pré-poda** e **pós-poda**.

Árvores de Decisão - pré-poda

Á árvore é podada por meio da interrupção da construção (ou seja, decide-se por não criar uma nova partição do subconjunto de tuplas de treinamento em um dado nó). Assim o nó se torna uma folha que deve conter (representar) a classe mais frequente no respectivo subconjunto de tuplas de treinamento.
Para isso, impõem-se um limiar para a medida que analisa a qualidade da (futura) partição. Se ela está abaixo do limiar, então poda-se a (futura) ramificação.

Dificuldade ....

... a escolha do limiar adequado. Limiares altos podem resultar em árvores muito simplificadas. Limiares baixo resultam em árvores pouco simplificadas.

Árvores de Decisão - pós-poda

É a abordagem mais comum - remove subárvores de um árvore totalmente construída.
A remoção de uma subárvore se dá por meio da remoção dos ramos de um nó e substituição dele por um nó folha. A folha é rotulada com a classe mais frequente dentre as presentes na subárvore (sob remoção).
A remoção é feita com base na análise de uma função de custo do nó. Se o custo de um nó se torna alto, a subárvore enraizada por ele é eliminada da árvore.
A função para o cálculo do custo pode ser baseada, por exemplo, em taxa de erro de classificação (C4.5) ou taxa de erro de classificação combinada ao número de folhas da subárvore sob análise (CART).

Árvores de Decisão - *Cost Complexity Pruning*

Também conhecida como *weakest link pruning* ou *error complexity pruning*, possui dois estágios:

1. uma sequência de árvores T_0, T_1, ..., T_k é construída sobre os dados de treinamento, onde T_0 é a árvore original antes da poda e T_k é a raíz da árvore.
2. uma destas árvores é escolhida como "árvore de poda", com base na estimativa de seu erro de generalização.

A árvore T_{i+1} é obtida pela substituição de uma ou mais subárvores na árvore predecessora T_i com as folhas adequadas.

As subárvores que serão podadas são aquelas que produzem o menor aumento na taxa de erro por folha podada:

$$\alpha = \frac{\epsilon(pruned(T,t),S) - \epsilon(T,S)}{|leaves(T)| - |leaves(pruned(T,t))|}$$

onde $\epsilon(T,S)$ indica a taxa de erro da árvore T para os exemplos S e $|leaves(T)|$ é o número de folhas em T; $pruned(T,t)$ é a árvore obtida na substituição do nó t em T com uma folha adequada.

Árvores de Decisão - *Reduced Error Pruning*

Enquanto caminha pelos nós internos da árvore, de baixo para cima (das folhas para a raíz) e de forma transversal, o procedimento checa cada nó interno e determina se substituí-lo com a classe mais frequente reduziria a acurácia da árvore. O nó é podado se a acurácia não é reduzida. O procedimento continua até que nenhuma poda diminua a acurácia.

Árvores de Decisão

Outras medidas ....

- *Minimum Error Pruning (MEP)*: realiza a poda com base em uma medida de erro que leva em consideração a probabilidade da classe majoritária na folha ocorrer no conjunto de dados.
- *Pessimistic Pruning*: realiza a poda se o erro obtido com a poda não ultrapassa um limite superior do erro (considera o número de folhas na árvore como um adicional no erro) da árvore de referência.
- *Error-Based Pruning*: realiza a poda se o erro obtido com ela é menor que o limite superior do erro de referência ponderado por um intervalo de confiança; na poda pode decidir por assumir a classe do filho que possui mais dados ou fazer a poda clássica.

Árvores de Decisão - *Minimum Error Pruning (MEP)*

Também envolve o caminho na forma transversal pelos nós internos da árvore, de baixo para cima. A técnica compara, em cada nó, a estimativa da taxa de erro de probabilidade *l* com e sem a poda.

Estimativa da taxa de erro de probabilidade *l*

É uma correção para a estimativa de probabilidade simples usando frequências. Se S_t são as instâncias que alcançaram a folha t, então a taxa de erro esperada nesta folha é:

$$\epsilon'(t) = 1 - max_{c_i \in dom(y)} \frac{|\sigma_{y=c_i}| + l \cdot p_{apr}(y=c_i)}{|S_t| + l}$$

onde $p_{apr}(y = c_i)$ é a probabilidade *a-priori* de y dado o valor de c_i, e l denota o peso dado para a probabilidade *a-priori*

Árvores de Decisão - *Minimum Error Pruning (MEP)*

Estimativa da taxa de erro de probabilidade *I*

A taxa de erro de um nó interno é a média ponderada das taxas de erros de seus ramos.
A ponderação é realizada com base na proporção de instâncias ao longo de cada ramo.
Se um nó interno é podado, então ele se torna a folha e sua taxa de erro é calculada diretamente usando a equação do slide anterior.
Se a poda de um nó não aumentar a taxa de erro, a poda será aceita.

Árvores de Decisão - *Pessimistic Pruning*

Evita a necessidade de ter um **conjunto de pruning** ou de realizar um **procedimento de cross-validation**.

A idéia básica é que o erro, estimado usando um conjunto de treinamento, não é realístico o suficiente. Para uma medida mais realística, sugere-se:

$$\epsilon'(T,S) = \epsilon(T,S) + \frac{|leaves(T)|}{2 \cdot |S|}$$

Contudo, essa taxa é ainda otimista. Então sugere-se podar um no t interno se sua taxa de erro está dentro de um erro padrão com respeito a uma árvore de referência:

$$\epsilon'(pruned(T,t),S) \leq \epsilon'(T,S) + \sqrt{\frac{\epsilon'(T,S) \cdot (1-\epsilon'(T,S))}{|S|}}$$

A poda pessimista executa um caminho transversal **top-down** nos nós internos da árvore.

Árvores de Decisão - *Error-Based Pruning*

É uma evolução da poda pessimista, e é implementado no algoritmo C4.5.

$$\epsilon_{UB}(T, S) = \epsilon(T, S) + Z_\alpha \cdot \sqrt{\frac{\epsilon(T,S)\cdot(1-\epsilon(T,S))}{|S|}}$$

onde $\epsilon(T, S)$ designa a taxa de classificações erradas da árvore T no conjunto de treinamento S; Z é o inverso da distribuição cumulativa normal padrão; e α é o nível de significância desejado.

Seja $subtree(T, t)$ a subárvore enraizada no nó t. Seja $maxchild(T, t)$ o nó filho mais frequente em t (o que possui mais instâncias de S) e seja S_t todas as instâncias de S que chegam em t. O procedimento percorre transversalmente todos os nós no sentido **bottom-up** e compara:

1. $\epsilon_{UB}(subtree(T, t), S_t)$
2. $\epsilon_{UB}(pruned(subtree(T, t), t), S_t)$
3. $\epsilon_{UB}(subtree(T, maxchild(T, t)), S_{maxchild(T,t)}))$

De acordo com o menor valor, o procedimento: deixa a árvore como está; poda o nó t; substitui o no t com a subárvore de raiz $maxchild(T, t)$.

Random Forest

Random Forest...

... é formada por muitas árvores de decisão. Para classificar um dado (desconhecido), ele é apresentado a cada uma das árvores da floresta. Cada uma delas fornece uma classificação para o dado e um método de decisão é usado para dar a resposta final: por exemplo - voto da maioria.

De forma resumida: cada árvore é construída usando as estratégias abaixo:

- Se o número de dados de treinamento é N, uma amostragem randômica de tamanho N, com reposição, do conjunto de treinamento original é realizada. Sobre cada amostragem é construída uma árvore.
- Se existem M atributos descritivos no conjunto de dados, um número $m << M$ é escolhido tal que para cada nó, m atributos sejam randomicamente selecionados de M a fim de serem submetidos ao critério de escolha de atributos para construção das futuras partições.
- Cada árvore cresce em toda a sua extensão possível - não há poda.

Escolha do atributo - Entropia / Ganho de informação / Raio do Ganho de informação / Gini

Alinhando os conceitos ....

Redução do conjunto de dados e redução de dimensionalidade

Mineração e análise de dados em montantes muito grandes de dados podem tomar muito tempo e dificultar o trabalho de criação dos modelos, tornando tal análise impraticável e/ou ineficiente.
Técnicas de redução de dados e dimensões devem ser aplicadas para obter uma representação reduzida do conjunto de dados, porém mantendo a integridade dos dados originais – preservando informação.

Estratégias:

- Agregação de dados: aplicação de operações sobre os dados de forma a construir os 'cubos";
- **Discretização** e geração de hierarquias: os valores dos atributos são substituídos por intervalos ou conceitos.
- **Seleção de atributos**: atributos (dimensões) irrelevantes, fracamente relevantes ou redundantes devem ser detectados e removidos;
- Redução de dimensionalidade: mecanismos para transformação são usados para diminuir a quantidade de dimensões dos dados;
- Redução de "número" (*numerosity*): substituição ou estimativa por representações alternativas, tais como *clustering*, amostragem ou histogramas;

Discretização baseada em Entropia

Entropia é uma medida que poder ser usada para discretização. Introduzida por Claude Shannon em um trabalho sobre Teoria da Informação e Ganho de Informação.

Discretização baseada em Entropia é uma técnica supervisionada, de divisão top-down. Ela explora a informação sobre a distribuição de classes no cálculo e determinação de pontos de divisão (*split*) – valores dos dados usados para determinar um intervalo para o atributo.

Para discretizar um atributo, o método seleciona como ponto de divisão o valor do atributo que tem entropia mínima. Recursivamente, particiona os intervalos resultantes até chegar em uma discretização hierárquica satisfatória.

Entropia

A entropia é uma grandeza termodinâmica que mede o grau de irreversibilidade de um sistema. É comumente associada ao que se entende por "desordem" (não em senso comum) de um sistema termodinâmico.

Físicos ... é só uma piadinha!!!

Exemplo

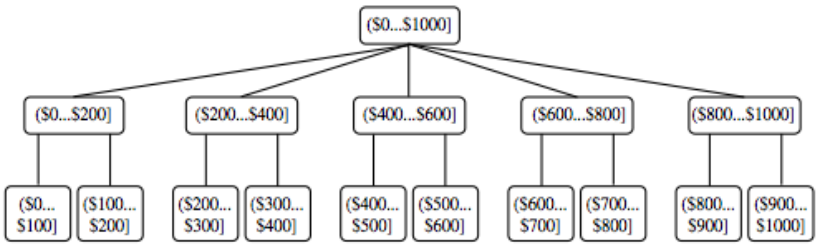
- antes de se misturarem tem-se
  - café
  - leite
  - açucar/adoçante
- depois de se misturarem teremos apenas uma informação
  - café com leite adoçado

(c) Baixa entropia - muita informação

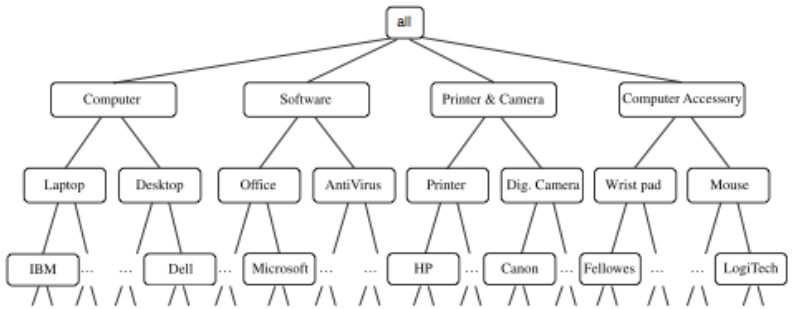
(d) Alta entropia - pouca informação

Discretização

Reduz a quantidade de valores para um atributo contínuo dividindo os valores em intervalos. É muito útil para a criação de conceitos em diferentes níveis de granularidade (valores numéricos para *idade* podem ser transformados em *jovem*, *meia-idade*, *senior*).



(e)



(f)

Discretização baseada em Entropia

Considere D um conjunto de dados (tuplas/linhas) definido por um conjunto de atributos e um atributo de rótulo de classe. O atributo de rótulo de classe fornece a informação sobre a classe, por dado (tupla/linha). O método básico para discretização baseada em entropia de um atributo A dentro do conjunto consiste de:

1. Cada valor de A pode ser considerado como um potencial limite de intervalo ou ponto de divisão (*split-point*) para particionar os valores de A. Ou seja, o ponto de divisão para A divide os dados de D em dois subconjuntos que satisfazem as condições $A \leq split\_point$ e $A > split\_point$, criando uma discretização binária.

Discretização baseada em Entropia

2 Suponha que nós queremos classificar as tuplas de D particionando-as no atributo A usando algum *split-point*. Idealmente, nós gostaríamos que esta partição resultasse na classificação exata dos dados (tuplas/linhas). Ou seja, se nós tivéssemos duas classes, nós esperaríamos que todas as tuplas da classe C_1 caíssem em uma parte, e todas as tuplas de C_2 caíssem em outra parte.

Sendo isso improvável, **quanta informação é ainda necessária** para obter uma classificação perfeita após esse particionamento? Esse montante é chamado de **expected information requirement** para classificar um dado de D baseado no particionamento de A.

E é dado por ....

Discretização baseada em Entropia

$$Info_A(D) = \frac{|D_1|}{|D|} Entropy(D_1) + \frac{|D_2|}{|D|} Entropy(D_2),$$

em que

- D_1 e D_2 correspondem aos dados (tuplas/linhas) em D que satisfazem as condições $A \leq split\_point$ e $A > split\_point$, respectivamente;
- $|D|$ é o número de dados no conjunto D ($|.|$ é o número de dados);
- a função $Entropy(.)$ para um dado conjunto é calculada com base na distribuição das classes dentro deste conjunto. Assim, dado m classes, C_1, C_2, ..., C_m, a entropia de D_1 é ...

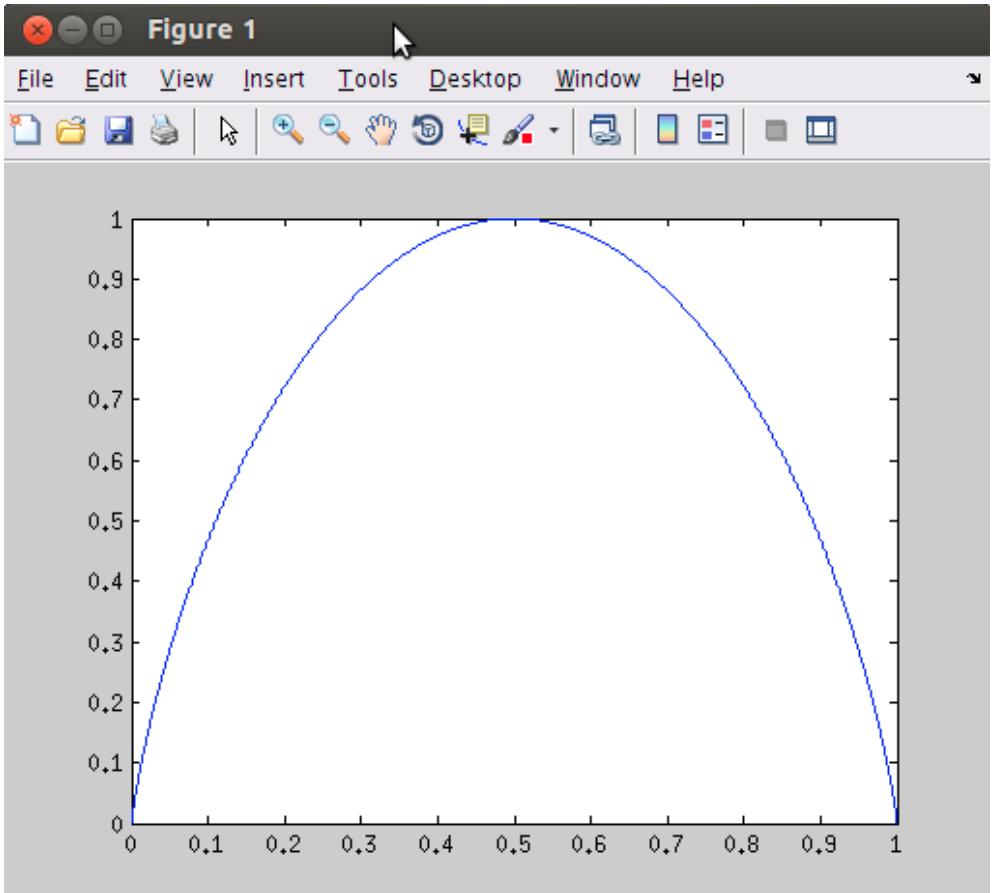
Discretização baseada em Entropia

$$Entropy(D_1) = -\sum_{i=1}^{m} p_i log_2(p_i),$$

em que

- p_i é a probabilidade da classe C_i em D_1, determinada dividindo-se o número de dados da classe C_i em D_1 por $|D_1|$.

Portanto, quando selecionamos um ponto de divisão para o atributo A, nós queremos escolher o valor de atributo que nos dá o mínimo *expected information requirement* (i.e., $\min(Info_A(D))$).
Isto resultaria no montante mínimo de informação esperada AINDA necessária para classificar perfeitamente os dados depois de usar esta partição. **Isto é equivalente a escolher o par "valor-atributo" com o máximo ganho de informação.**




```
x = 0:0.0001:1;
plot(x, -x.*log2(x) - (1-x).*log2(1-x))
```

Discretização baseada em Entropia

3 o processo de determinar o ponto de divisão é recursivamente aplicado para cada partição obtida, até algum critério de parada ser alcançado (por exemplo, quando a mínima *expected information requirement* de todos os pontos de divisão é menor do que um limiar; ou quando o número de intervalos é maior do que um limiar).

Discretização baseada em entropia simplifica os dados e cria um conceito de hierarquia, usando a informação de classe. Ela ajuda a produzir resultados de classificação mais precisos, já que embute informação na representação dos dados RESUMIDOS.

Seleção de atributos

Uma forma de selecionar atributos é usando o conhecimento do especialista sobre quais informações podem ser úteis para um determinado contexto. Entretanto, na maioria das vezes, isso não será eficiente. Manter atributos irrelevantes pode confundir o algoritmo de mineração empregado e levar a descoberta de padrões de qualidade baixa.

Meta ...

... encontrar um conjunto mínimo de atributos que resultem em uma distribuição de probabilidade de classes tão próxima quanto possível da distribuição original.

Como encontrar um bom subconjunto de atributos?

Para n atributos originais, existem 2^n possíveis subconjuntos. Logo, uma busca exaustiva pode ser proibitiva conforme n aumenta.

Métodos heurísticos

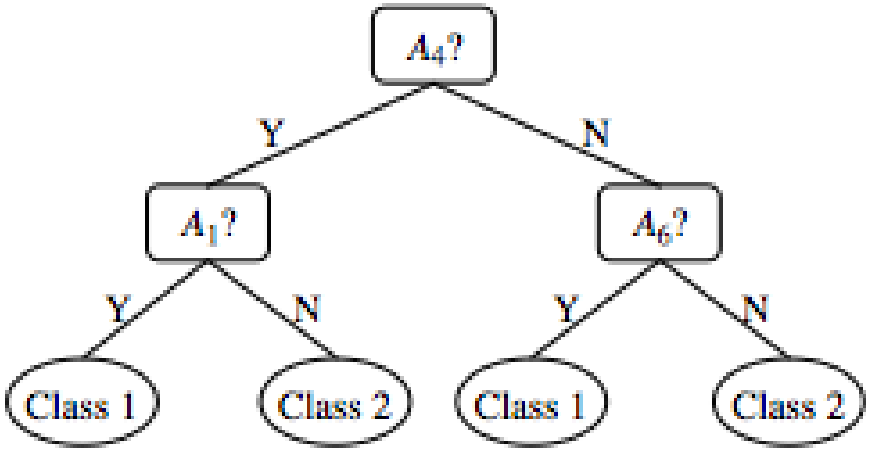
São métodos que exploram um espaço de busca reduzido. Geralmente são "gulosos" (*greedy*), ou seja, enquanto fazem a busca, tomam decisões que parecem ser a melhor escolha naquele instante de tempo, o naquela iteração da busca. A estratégia faz uma escolha ótima local **na esperança** de que isso leve a uma solução ótima global.

Os melhores (ou piores) atributos podem ser determinados usando testes de significância estatística, os quais analisam se os atributos são independentes um do outro. Medidas de avaliação dos atributos podem ser usadas, por exemplo, **medidas de ganho de informação**.

Algumas estratégias

- Seleção *stepwise forward*: o procedimento inicia com um conjunto vazio de atributos. O melhor dos atributos originais é determinado e adicionado ao conjunto. A cada iteração, o melhor dos atributos restantes é adicionado ao conjunto.
- Eliminação *stepwise backward*: o procedimento inicia com o conjunto total de atributos. A cada passo, o pior atributo dos restantes é eliminado.
- Combinação das anteriores: as estratégias anteriores são combinadas tal que, a cada passo, o procedimento seleciona o melhor atributo e elimina o pior atributo dentre os atributos restantes.
- Indução por Árvores de Decisão: na construção da árvore, para cada nó o algoritmo escolhe a "melhor" partição dos dados. Quando a árvore é usada para selecionar atributos, todos os atributos que não aparecem na árvore são considerados irrelevantes.

O critério de parada para cada estratégia pode variar. Os procedimentos devem aplicar um limiar nas medidas usadas para determinar quando parar o processo.

Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  => Reduced attribute set: $\{A_1, A_4, A_6\}$

Medidas para Seleção de Atributos

Quando dividimos o conjunto de dados D em partes, nós gostaríamos de dividi-lo em partes pequenas e, idealmente, que cada parte fosse pura (todas as tuplas que caem em uma dada parte pertencem à mesma classe). Conceitualmente, o melhor critério de particionamento é aquele que resulta em partições que mais se aproximem deste cenário. Medidas de seleção de atributos fornecem um *ranking* para cada atributo. O atributo com melhor pontuação é escolhido para particionar o conjunto D - ou para permanecer no conjunto de dados RESUMIDO.

Algumas medidas ....

- Ganho de Informação
- Razão de Ganho
- Índice Gini

Ganho de Informação

Minimiza a informação necessária para classificar os dados das partições resultantes e reflete a impureza ou a aleatoriedade das partições. A abordagem é minimizar o *expected information requirements* necessário para discriminar os dados a partir da existência da partição.

A informação esperada necessária para classificar um dado em D é dada por

$$Entropy(D) = -\sum_{i=1}^{m} p_i log_2(p_i),$$

onde p_i é a probabilidade da classe C_i em D, determinada dividindo-se o número de dados da classe C_i em D por $|D|$.

Note que, neste ponto, a informação que nós temos é baseada nas proporções das classes - temos a **entropia** do conjunto D.

Ganho de Informação

Agora suponha que nós queremos particionar as tuplas de D usando algum atributo A, tendo v valores distintos $a_1, a_2, ...a_v$. O atributo A pode ser usado para criar as partições $D_1, D_2, ..., D_v$. Se o atributo A é capaz de criar boas partições (próximas das ideais) ele é um atributo que contribui para a discriminação dos dados. Saberemos se ele é bom após calcular quanta informação é ainda necessária para chegar na classificação exata, após usar a informação de A.

$$Info_A(D) = \frac{|D_1|}{|D|} Entropy(D_1) + \frac{|D_2|}{|D|} Entropy(D_2)... + \frac{|D_v|}{|D|} Entropy(D_v),$$

* mesmo raciocínio usado na discretização do atributo, porém com um objetivo diferente.

Ganho de Informação

O ganho de informação é definido como a diferença entre a informação necessária para particionar os dados considerando o conjunto original e a informação necessária para particionar os dados considerando após o uso do atributo A no particionamento. Ou seja,

$$Gain(A) = Info(D) - Info_A(D)$$

Ganho de Informação - Exemplo

Class-labeled training tuples from the *AllElectronics* customer database.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Ganho de Informação - Exemplo

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$\begin{aligned} Info_{age}(D) = & \frac{5}{14} \times (-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) \\ & + \frac{4}{14} \times (-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}) \\ & + \frac{5}{14} \times (-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}) \\ = & 0.694 \text{ bits.} \end{aligned}$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Faça para os demais atributos e veja qual tem um ganho de informação maior!

Razão de Ganho

O ganho de informação é influenciado por atributos com muitas saídas. Por exemplo, considere um atributo que é um identificados único, como ID_produto. Usar este atributo levará a um grande número de partições puras e o ganho de informação será 1. Claramente, manter este atributo no conjunto de dados não é útil para a tarefa de classificação.

Na medida de Raio de Ganho, uma normalização é aplicada ao Ganho de Informação usando um "valor de informação da divisão", que representa a informação gerada pela divisão do conjunto de dados D em v partições.

$$SplitInfo_A(D) = (-\frac{|D_1|}{|D|} log_2(\frac{|D_1|}{|D|})) + (-\frac{|D_2|}{|D|} log_2(\frac{|D_2|}{|D|}))... + (-\frac{|D_v|}{|D|} log_2(\frac{|D_v|}{|D|})),$$

$$Raio\_Ganho = \frac{Ganho(A)}{SplitInfo(A)}$$

Índice Gini

Mede a impureza de D, de uma partição de dados ou de um conjunto de tuplas de treinamento.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

onde p_i é a probabilidade da tupla em D pertencer à classe C_i e é estimada por $|C_{i,D}|/|D|$. A soma é computada sobre m classes. Se os dados pertencem a uma mesma classe a impureza é $1 - 1 = 0$ (os dados são puros).

O índice Gini considera divisões binárias para cada atributo.
Deve-se examinar todos os possíveis subconjuntos que podem ser formados usando os valores conhecidos de um atributo, para determinar a melhor divisão para aquele atributo. Cada subconjunto é um teste binário para o atributo da forma $A \in S_A$? Dada uma tupla, este teste é satisfeito se o valor do atributo na tupla está entre os valores listados no conjunto S_A.

Índice Gini

Se uma divisão binária de A particiona D em D_1 e D_2, o indice Gini de D dado o particionamento é

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

O subconjunto que dá o mínimo índice Gini (menos impureza) para o atributo é selecionado como o subconjunto de divisão do atributo.
A redução de impureza se o atributo A é usado para particionar o conjunto é dado por:

$$\Delta GiniA = Gini(D) - Gini_A(D).$$

Assim, o atributo que **maximiza a redução** de impureza é escolhido como atributo de divisão do conjunto. Ou seja, quanto mais puro for a partição criada por A, menor é o $Gini_A(D)$ e maior será o $\Delta GiniA$.

Classificação - k–NN e Árvores de Decisão

- Clodoaldo A. M. Lima - c.lima@usp.br
- Sarajane M. Peres - sarajane@usp.br

Programa de Pós Graduação em Sistemas de Informação - PPgSI
Escola de Artes, Ciências e Humanidades - EACH
Universidade de São Paulo - USP