Classificação - k–NN, Árvores de Decisão

Sarajane Marques Peres

Abril de 2020

Material baseado em:
HAN, J. & KAMBER, M. Data Mining: Concepts and Techniques. 2nd. 2006
ROKACH, L. & MAIMON, O. Data Mining with Decision Trees: Theory and Applications. 2008
RUSSEL, S & NORVIG, P. Inteligência Artificial. 2a. ed. 2004

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 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 (do mesmo tipo, da mesma natureza).

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

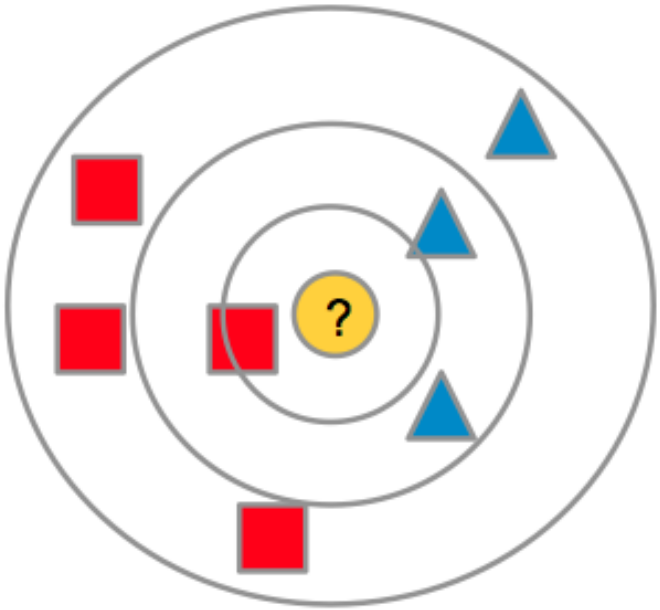
A "tupla mais próxima" é definida em termos de uma métrica de distância, por exemplo a distância Euclidiana. A distância Euclidiana entre dois pontos (ou tuplas) $X_1 = (x_{11}, x_{12}, ..., x_{1n})$ e $X_2 = (x_{21}, x_{22}, ..., x_{2n})$ é

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

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;

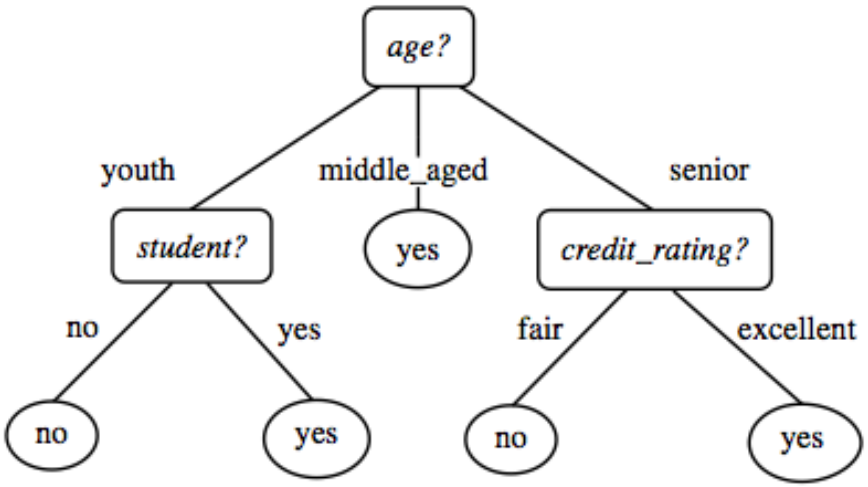
Árvores de Decisão

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

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

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*.

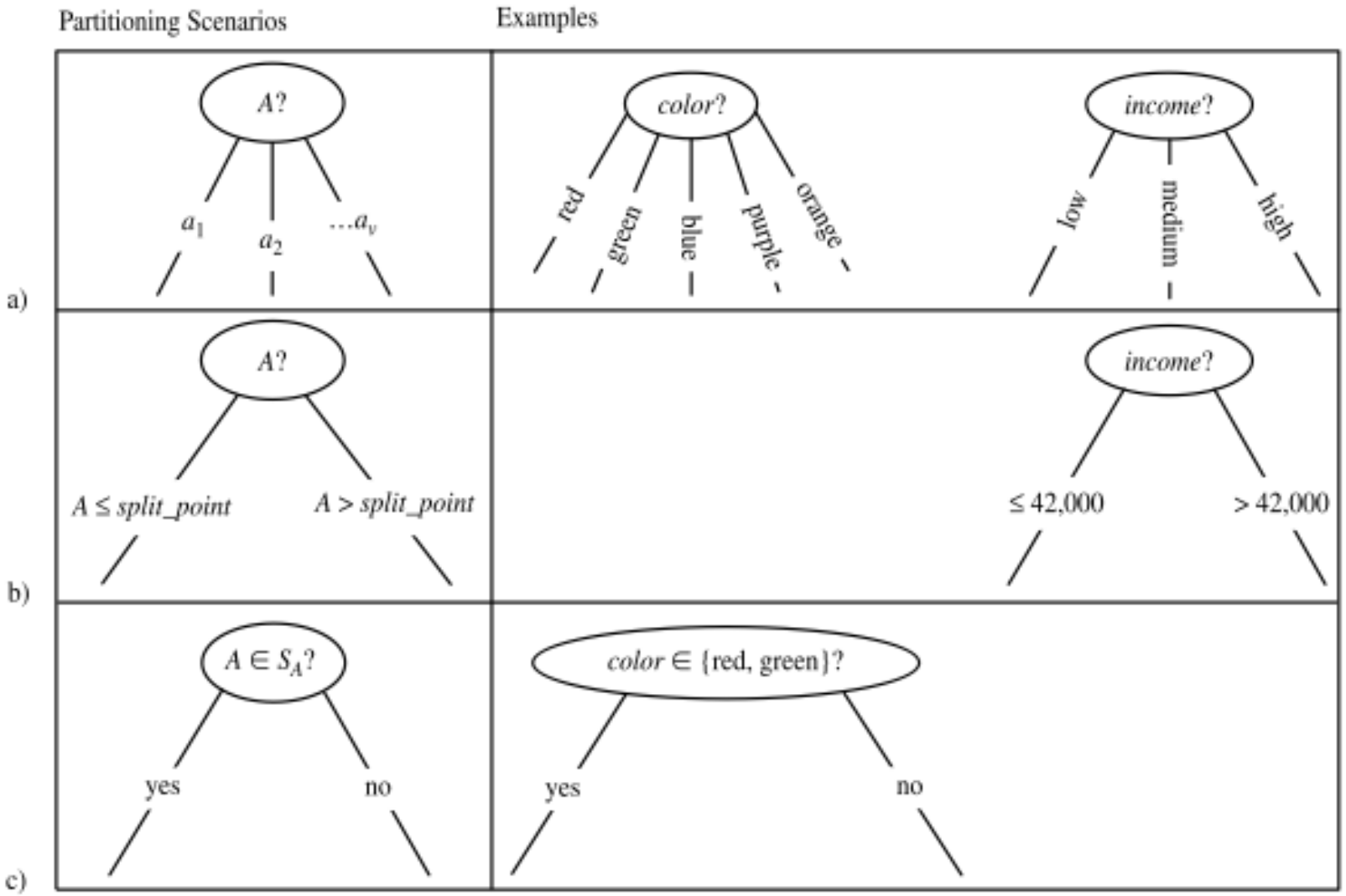
Output: A decision tree.

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 - Complexidade

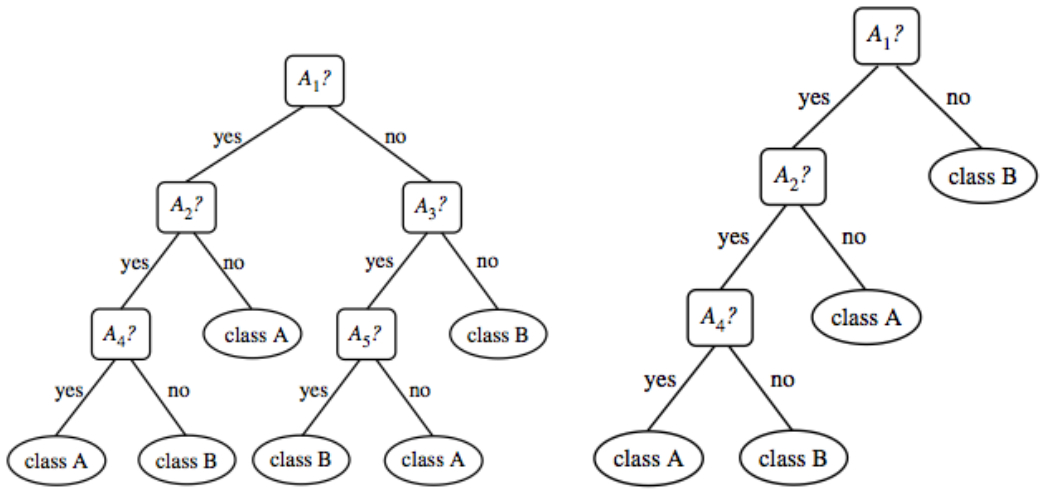
A complexidade da árvore de decisão tem efeito sobre a sua acurácia. Usualmente, os seguintes fatores são usados para medir a complexidade de uma árvore de decisão:

- 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).

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

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

- Sarajane M. Peres - sarajane@usp.br

Bacharelado em Sistemas de Informação - BSI
Escola de Artes, Ciências e Humanidades - EACH
Universidade de São Paulo - USP