Problemas Clássicos da Computação

Árvores de Decisão

Felipe Augusto Lima Reis
felipe.reis@ifmg.edu.br

INSTITUTO FEDERAL
Minas Gerais

Sumário

Introdução

Introdução

INSTITUTO FEDERAL Minas Gerais

- Árvores de Decisão são estruturas destinada à tomada de decisões, comuns em Aprendizado de Máquinas
  - São fáceis de serem entendidas;
  - Tem poder de explicar as decisões tomadas;
  - O custo de criação é baixo;
  - Seu processamento é consideravelmente rápido;
- Múltiplas árvores, produzindo predições variadas dão origem a uma Floresta de Decisão.

Introdução

- Árvores de Decisão quebram o processo decisório em uma sequência de escolhas
  - Caminha-se da raíz até a folha e, a cada passo, são escolhidas as opções mais adequadas ao problema;
  - Árvores podem dar origem a um sistema de regras de indução[1] [Marsland, 2014].

[1] Técnica que cria regras do tipo *"if-else-then"* para produção de uma saída [Nettleton, 2014].

Introdução

- Existem diferentes algoritmos para construção de árvores de decisão;
  - Os algoritmos, em geral, são classificados como gulosos - a cada etapa escolhem o recurso que provê maior nível de informação [Marsland, 2014];
- Dentre os algoritmos, destacam-se:
  - ID3 (e sua extensão C4.5)
  - CART (Classification and Regression Trees);
- Para entender o funcionamento dos algoritmos é necessário conhecimento prévio de alguns conceitos de Teoria da Informação.

Teoria da Informação

Entropia

INSTITUTO FEDERAL Minas Gerais

- A Teoria da Informação surgiu em 1948, baseado no conceito de Entropia da Informação, proposta por Claude Shannon [Marsland, 2014];
  - Entropia da Informação mede o grau de impureza (incerteza) de um conjunto de *features*;
- A entropia H de um conjunto de probabilidades $p(x_i)$, onde b é uma unidade de informação (se informação binária, $b = 2$), é dada por:

$$\boxed{H(X) = -\sum_i p(x_i) \log_b p(x_i)}$$

Nota: Não confundir o conceito de Entropia, em Física (medida do grau de irreversibilidade de um determinado sistema), com o conceito de Entropia, em Teoria da Informação.

Entropia - Exemplo

- Considere um problema binário, onde amostras podem ter valores positivos ou negativos;
- Se todos os eventos forem positivos, um novo evento positivo não altera a informação prévia existente;
- Neste cenário, se uma amostra negativa for selecionada, haverá um ganho de informação
  - O evento imprevisto indica uma condição nova;
  - Pode-se dizer que o evento "alterou" a distribuição aparente e foi capaz de mudar o conhecimento acerca do problema.

Ganho de Informação

- O ganho de informação (*information gain*), pode ser definido como a entropia do conjunto S, subtraído da entropia de quando uma *feature* f particular, de um conjunto de todas as features F, é escolhida [Marsland, 2014] [Quinlan, 1986]

$$\boxed{Ganho(S,F) = Entropia(S) - \sum_{f \in F} \frac{|S_f|}{|S|} Entropia(S_f)}$$

Exemplo [Marsland, 2014]

- Considere um conjunto de dados $S = \{s_1 = true, s_2 = false, s_3 = false, s_4 = false\}$ e um conjunto de *features* $F = \{f_1, f_2, f_3\}$;
- Considere as seguintes relações entre dados e *features*:
  - Relação 1: $\{(s_1, f_2), (s_2, f_2)\}$;
  - Relação 2: $\{(s_3, f_3)\}$;
  - Relação 3: $\{(s_4, f_1)\}$.
- Podemos calcular a entropia para eventos positivos e negativos como:

$$H(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.811$$

Exemplo [Marsland, 2014]

- Podemos calcular a entropia de cada uma das *features* f:

$$\frac{|S_{f1}|}{|S|} Entropia(S_{f1}) = \frac{1}{4} \times \left(-\frac{0}{1} \log_2 \frac{0}{1} - \frac{1}{1} \log_2 \frac{1}{1} \right) = 0$$

$$\frac{|S_{f2}|}{|S|} Entropia(S_{f2}) = \frac{2}{4} \times \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \right) = \frac{1}{2}$$

$$\frac{|S_{f3}|}{|S|} Entropia(S_{f3}) = \frac{1}{4} \times \left(-\frac{0}{1} \log_2 \frac{0}{1} - \frac{1}{1} \log_2 \frac{1}{1} \right) = 0$$

- O ganho de informação é dado por:

$$\boxed{Ganho(S, F) = 0.811 - (0 + 0.5 + 0) = 0.311}$$

Índice Gini

- O Índice Gini ou Impureza de Gini, criado em 1912 por Conrado Gini, mede o grau de heterogeneidade dos dados [da Silva, 2005];
- O Índice Gini é frequentemente associado ao conceito de pureza de nós de árvores
  - Pureza está relacionado ao agrupamento de dados semelhantes em uma só classe;
  - Se todos os elementos pertencem a uma única classe, eles são chamados de puros [Marsland, 2014].

Índice Gini

- O Índice Gini é calculado com base na probabilidade relativa p_i de um objeto pertencer a uma determinada classe em um nó da árvore [da Silva, 2005].

$$\boxed{Gini = 1 - \sum_{i=1}^{n} p_i}$$

- O grau do Índice Gini varia entre 0 e 1
  - 0: denota que todos os elementos pertencem a um certa classe ou somente existe uma classe;
  - 1: denota que os elementos estão distribuídos aleatoriamente entre as classes;
  - 0.5: denota que elementos estão igualmente distribuídos entre as classes [Tahsildar, 2019].

Impureza de Gini

- No contexto de árvores de decisão a pureza é utilizada para indicar que todos os dados em um folha pertencem a uma mesma classe;
- A Impureza de Gini pode ser associada à taxa de erro esperado se a classificação for selecionada de acordo com a distribuição de classes [Marsland, 2014].

Árvores de Decisão

ID3 e C4.5

ID3 - Exemplo [Marsland, 2014]

- O algoritmo parte da ideia de que a melhor *feature* para classificação é aquela que fornece mais informações
  - Ao prover mais informação, sua entropia será mais alta;
  - Depois de usar a *feature* de maior entropia, o objetivo é escolher a próxima *feature* de entropia mais alta.
- Outro conceito importante do algoritmo é o decrescimento da entropia
  - Ao enumerar múltiplas *features*, a ganho de informação decresce em cada passo do classificador;
  - Pode-se inferir essa característica à classificação prévia dos dados, resultando em novas classificações sobre dados já pré classificados [Marsland, 2014].

ID3

INSTITUTO FEDERAL
Minas Gerais

- Algoritmo ID3:
  1. Computar o ganho de informação para cada *feature*;
  2. Escolher, de maneira gulosa, a *feature* que possui o maior valor;
  3. Repetir até que somente reste uma classe de dados;
- O ID3 pode ser programado de forma recursiva e, ao final, produzirá uma árvore de decisão [Marsland, 2014].

ID3

- O algoritmo ID3 pode ser resumido em:

The ID3 Algorithm

- If all examples have the same label:
  - return a leaf with that label
- Else if there are no features left to test:
  - return a leaf with the most common label
- Else:
  - choose the feature $\hat{F}$ that maximises the information gain of S to be the next node
  - add a branch from the node for each possible value f in $\hat{F}$
  - for each branch:
    * calculate S_f by removing $\hat{F}$ from the set of features
    * recursively call the algorithm with S_f, to compute the gain relative to the current set of examples

Fonte: [Marsland, 2014]

ID3

- O ID3 generaliza (cria a árvore) a partir do conjunto de treinamento;
- O método de construção da árvore, com o ganho máximo de informação a cada etapa, tende a produzir árvores de tamanhos pequenos
  - Isso acontece pois o algoritmo tenta minimizar a quantidade informação restante para o próximo passo;
  - O algoritmo, entretanto, acaba por adicionar um viés[2], à tomada de decisão.

[2] Definido como viés induzido (*indutive bias*)

ID3 - Princípios Associados

- A definição de árvores de tamanhos pequenos pode ser associada a 3 princípios semelhantes:
  - KISS (Keep it Simple, Stupid):
    - Estabelece que as soluções devem ser a mais simples possíveis;
  - Navalha de Occam:
    - Nomeado em homenagem a Guilherme de Ockham (1288-1347);
    - Postula que de múltiplas explicações possíveis para o mesmo conjunto de fatos, deve-se optar pela mais simples delas;
  - MDS (Minimum Description Lenght):
    - Proposto por Rissanem, em 1989;
    - Afirma que a melhor descrição para algo é a menor, aquela que foi mais condensada / comprimida [Marsland, 2014].

C4.5

- Em alguns casos, o algoritmo ID3 pode gerar árvores profundas, quando o número de *features* é grande;
  - Essa situação pode causar, inclusive, *overfitting*;
- A evolução do algoritmo ID3, chamada de C4.5 adiciona um elemento para evitar essa situação: a poda (*pruning*)
  - A poda possibilita redução de *overfitting*, uma vez que são desconsiderados detalhes do conjunto de treinamento;
  - A poda também aumenta a legibilidade da árvore, tornando-a mais fácil de ser interpretada.

Métodos de poda

1. Poda completa da árvore, com verificação do erro no conjunto de validação;
   - Redução da árvore, pela substituição de nós por subárvores;
   - Verifica-se o erro no conjunto de validação - se o erro foi reduzido ou manteve-se igual, a poda é aceita;
2. Post-pruning (usada pelo C4.5)
   - Obtém a árvore gerada pela ID3 e a converte em um conjunto de regras *"if-else-then"*;
   - Remove pré-condições, se a acurácia da regra (no nó) aumentar sem as pré-condições;
   - Regras são ordenadas de acordo com a acurácia no conjunto de treino e aplicadas em ordem [Marsland, 2014].

Árvores e Variáveis Contínuas

INSTITUTO FEDERAL Minas Gerais

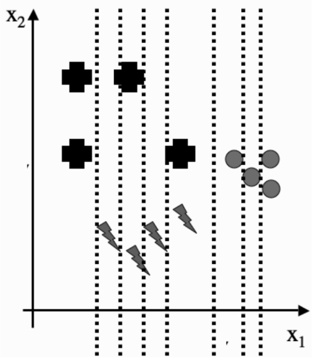
- Até o momento, os algoritmos vistos de árvores de decisão consideraram apenas variáveis discretas
  - No entanto, árvores podem também trabalhar com variáveis contínuas, após algumas transformações;
- Para trabalhar com variáveis contínuas, a solução mais simples seria discretizar essas variáveis
  - Exemplo: Suponha que um conjunto de valores contínuos no intervalo 0-100, $\{x \in \mathbb{R} \mid 0 <= x <= 100\}$;
    - Definimos conjuntos de valores discretos: A, B, C e D;
    - Para A, temos valores no intervalo 0<=x<=25;
    - Para B, temos valores no intervalo 25<x<=50;
    - Para C, temos valores no intervalo 50<x<=75;
    - Para D, temos valores no intervalo 75<x<=100.

Árvores e Variáveis Contínuas

- Apesar de simples, essa divisão pode ser ineficaz, uma vez que pode dividir elementos semelhantes
  - Suponha que dois elementos, α e β, com valores 24.9 e 25.1, respectivamente;
  - O elemento α pertenceria à classe A, enquanto β pertenceria à classe B;
  - Outro valor, $\gamma = 49.9$, pertenceria também à classe B;
  - No entanto, a similaridade (distância) entre elementos α e β é menor que a similaridade entre elementos β e γ.

Árvores e Variáveis Contínuas

- Um exemplo semelhante ao anterior pode ser representado pela imagem abaixo
  - As linhas tracejadas indicam possíveis pontos de divisão do conjunto.



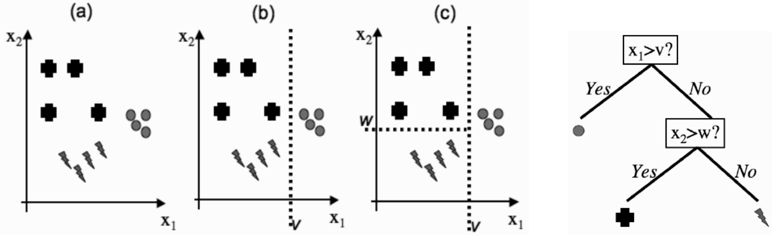
Fonte: [Marsland, 2014]

Árvores e Variáveis Contínuas

- As árvores geradas pelo método de discretização anterior são chamadas árvores univariadas
  - Definidas como árvores no qual o espaço é dividido em hiperplanos paralelos a um eixo;
- No entanto, podemos também ter divisões que geram árvores multivariadas
  - Essas árvores podem utilizar combinações de *features*;
  - Os hiperplanos não precisam ser necessariamente paralelos aos eixos;
  - Essas árvores devem ser utilizadas em casos em que a divisão univariada apresenta baixo desempenho.

Árvores e Variáveis Contínuas

- Um exemplo de divisão multivariada de um conjunto e a árvore resultante da divisão pode ser vista na figura abaixo.



Fonte: [Marsland, 2014]

CART

CART

- CART, acrônimo de *Classification and Regression Tree*, é uma algoritmo de construção de árvores de decisão
  - Como o nome indica, pode ser usado tanto para problemas de classificação quanto regressão;
  - De forma semelhante aos demais algoritmos, o CART é capaz de trabalhar com árvores multidecisão;
  - Para classificação basta transformar a decisão sobre as classes em questões binárias
    - A árvore de decisão gerada, então, é binária;
  - Para regressão (e consequentemente, variáveis contínuas), o algoritmo é definido com objetivo de minimizar a função de erro[3] [Marsland, 2014].

[3] Frequentemente é utilizada a soma de erros quadráticos.

CART

INSTITUTO FEDERAL Minas Gerais

- O algoritmo CART utiliza o Índice Gini para minimizar a função de custo em cada nó .
  - A seleção de *features* é usada de maneira gulosa para divisão de pontos;
  - A escolha é feita com base no valor mínimo do Índice Gini.
  - O processo é repetivo recursivamente para todos os sub-nós da árvore [Patel, 2020].
- Os algoritmos CART e ID3 possuem um funcionamento similar.

Floresta de Decisão

Floresta de Decisão

- Florestas de Decisão correspondem a um método de decisão por um comitê, usando árvores de decisão;
  - Esses algoritmos são muito populares na área de Aprendizado de Máquinas;
- A ideia do algoritmo é criar múltiplas árvores de decisão, com certa variedade, de modo que a decisão delas forme um comitê, responsável pela decisão final [Marsland, 2014].

Bagging

- Bagging é uma técnica para redução do erro de generalização, por meio da criação de várias versões de um preditor;
  - Esse preditores são usados para criar um preditor agregado;
  - Os valores de previsão são usados para um único resultado numérico baseado em votos individuais;
  - Técnicas que usam essa estratégia são chamados *ensemble methods* [Breiman, 1996] [Goodfellow et al., 2016].

Bagging é a abreviatura de *bootstrap aggregating*.

Floresta de Decisão

- Florestas de Decisão utilizam *bagging* para criação de múltiplas árvores
- O método realiza o treinamento em diferentes subconjuntos de dados;
  - São obtidas amostras de *bootstrap* para cada árvore;
- Para aumento da variabilidade, são adicionados limites às escolhas que as árvores podem tomar
  - Amostras aleatórias dos dados são fornecidas à cada nó da árvore;
- As duas técnicas previamente citadas permitem adição de aleatoriedade sem aumento do viés [Marsland, 2014].

Floresta de Decisão

- Após treinada, a saída de uma floresta de decisão é dada de forma diferente, de acordo com o objetivo:
  - Classificação: maioria dos votos (*majority vote*)
  - Regressão: resposta média para regressão;
- Vantagens das florestas de decisão em relação a árvores de decisão (árvores únicas):
  - Menor tamanho das árvores;
  - Altamente paralelizável, uma vez que cada árvore é independente uma da outra [Marsland, 2014].

Referências I

Breiman, L. (1996).
Bagging predictors.
Machine Learning, 24(2):123–140.

da Silva, L. M. O. (2005).
Uma Aplicação de Árvores de Decisão, Redes Neurais e KNN para a Identificação de Modelos ARMA Não Sazonais e Sazonais.
PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
http://www.deeplearningbook.org.

Kopec, D. (2019).
Classic Computer Science Problems in Python.
Manning Publications Co, 1 edition.

Marsland, S. (2014).
Machine Learning: An Algorithm Perspective.
CRC Press, 2 edition.
Disponível em: https://homepages.ecs.vuw.ac.nz/ marslast/MLbook.html.

Nettleton, D. (2014).
Chapter 6 - selection of variables and factor derivation.
In Nettleton, D., editor, Commercial Data Mining, pages 79–104. Morgan Kaufmann, Boston.

Referências II

INSTITUTO FEDERAL
Minas Gerais

Patel, F. (2020).
Decision tree the cart algorithm.
Disponível em: https://medium.com/analytics-vidhya/decision-tree-the-cart-algorithm-28c481d28813.

Quinlan, J. R. (1986).
Induction of decision trees.
1(1):81â106.

Richert, W. and Coelho, L. P. (2013).
Building Machine Learning Systems with Python.
Packt Publishing Ltd., 1 edition.

Shalev-Shwartz, S. and Ben-David, S. (2014).
Understanding Machine Learning: From Theory to Algorithms.
Cambridge University Press, 1 edition.
Disponível em: http://www.cs.huji.ac.il/ shais/UnderstandingMachineLearning.

Tahsildar, S. (2019).
Gini index for decision trees.
Disponível em: https://blog.quantinsti.com/gini-index/.