Métodos de Classificação por Árvores de Decisão
Disciplina de Projeto e Análise de Algoritmos

Juliana Moreira Barbosa, Tiago Garcia de Senna Carneiro, Andrea Iabrudi Tavares
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: {julianamb.analista, tiagogsc, andrea.iabrudi}@gmail.com

***Resumo*—Este artigo apresenta um problema de classificação de dados, envolvendo a indução de uma árvore de decisão para auxílio da tomada de decisão na atualização dos valores do IPTU (Imposto Predial Territorial Urbano) de uma cidade. São apresentados os algoritmos CART e C4.5, e uma análise de complexidade dos mesmos. Nos experimentos é comparado o desempenho dos dois, em induzir uma árvore de decisão para o problema do IPTU.**

***Keywords*-CART, C4.5, Classificação, IPTU.**

I. Introdução

O problema de classificação abordado nesse artigo consiste em induzir uma árvore de decisão para ajudar na tomada de decisão referente a cobrança do IPTU (Imposto Predial Territorial Urbano). De acordo com a lei municipal de Ouro Preto n° 537 de 21 de dezembro de 2009, que estabelece a planta genérica de valores imobiliários do Município, onde são estabelecidos os valores por m^2 dos terrenos e edificações, os critérios referentes a atualização de valores estão no artigo 4° que diz: "Os valores contidos nesta Planta Genérica serão revistos, anualmente, tomando-se como referência a média dos seguintes índices: IGPM, IPCA e INPC", e também, "Em casos especiais e considerando as variações do mercado, o Município poderá fazer correções em áreas específicas, com o objetivo de adequar seus valores à realidade imobiliária". Podemos identificar alguns atributos que podem determinar possíveis mudanças na realidade imóbiliaria da cidade, dentre estes atributos está a localização que é de natureza espacial. O objetivo é identificar quais desses fatores podem influenciar na variação do mercado. Assim, o Município poderá verificar quais regiões atendem a esses critérios, e que podem ser possíveis candidatos a atualização do valor do m^2 na planta genérica de valores imobiliários, atualizando assim também o valor do IPTU dos imóveis nessas regiões.

Estimar a classe de um objeto através de um conjunto de medidas que o descreve é um problema típico de uma área científica normalmente identificada por reconhecimento de padrões. Classificação é uma técnica de aprendizagem supervisionada.

Um algoritmo para aprendizagem supervisionada determinística recebe como entrada o valor correto da função desconhecida para entradas específicas e deve tentar recuperar a função desconhecida ou algo perto disso. Podemos dizer que um exemplo é um par $(x, f(x))$, onde x é a entrada e $f(x)$ é a saída da função aplicada a x [1].

A tarefa da inferência indutiva pura (ou indução) é: dada uma coleção de exemplos f, retornar uma função h que se aproxime de f. A função h é chamada hipótese. A razão pela qual a aprendizagem é difícil, de um ponto de vista conceitual, é que não é fácil saber se uma h específica é uma boa aproximação de f. Uma boa hipótese irá generalizar bem - isto é, irá prever corretamente os exemplos ainda não-vistos [1].

A tarefa de indução pode ser resolvida por classificadores baseados em: exemplos, redes neurais, algoritmos genéticos, árvores de decisão, tendo todas elas as suas vantagens e desvantagens [2].

Foi escolhida para este artigo a técnica de classificação de dados por meio de árvores de decisão pelos seguintes motivos:

- Podem ser aplicadas a qualquer tipo de dados.
- Por ser uma técnica simples, mas ainda assim bem-sucedida.
- Pela grande inteligibilidade dos resultados que produz.
- Manipula de uma forma eficiente informação condicional subdividindo o espaço em sub-espaços que são manipulados individualmente.

Um dos trabalhos mais importantes de classificadores baseados em árvores de decisão é o de Breiman, onde o algoritmo CART é apresentado [3]. Também existe o trabalho de Quinlan onde o algoritmo C4.5 é apresentado [4].

O objetivo deste artigo é explicar detalhadamente o funcionamento dos algoritmos CART e C4.5, mostrar as diferenças e similaridade de ambos e fazer a análise de complexidade dos dois. Para isso teremos como experimento um problema de classificação de domínio urbano. Ao final será escolhido um dos dois algoritmos que obtiver o melhor

resultado de predição, ou seja, encontrar a hipótese h que melhor se aproxima do resultado esperado f.

O artigo está organizado da seguinte forma. A Seção II apresenta o funcionamento dos algoritmos de indução de árvores de decisão CART e C4.5 e a análise de complexidade. Os experimentos são mostrados na Seção III. E a conclusão é apresentada na seção IV.

II. Métodos

A. Indução de Árvores de Decisão

O processo de indução de árvores de decisão tem a função de particionar recursivamente um conjunto de treinamento até que cada subconjunto obtido deste particionamento contenha casos de uma única classe. Uma árvore de decisão toma como entrada um objeto ou situação descrito por um conjunto de atributos e retorna uma decisão - o valor de saída previsto, de acordo com a entrada. Os atributos de entrada podem ser discretos ou contínuos, mas no nosso caso teremos apenas atributos discretos [1].

Neste artigo daremos atenção a dois algoritmos de indução de árvores de decisão que são: CART, e C4.5.

Os algoritmos aplicam o critério de divisão e conquista na construção da árvore, e o critério guloso para escolha de melhores partições e melhores atributos.

A árvore de classificação é o resultado de se fazer uma sequência ordenada de perguntas, e as perguntas feitas a cada passo na sequência dependem das respostas às perguntas anteriores. A sequência termina em uma previsão da classe.

O ponto de partida de uma árvore de classificação é chamado de nó raiz e consiste em todo o conjunto de aprendizado, e está no topo da árvore. Um nó é um subconjunto do conjunto de atributos, e pode ser terminal ou não-terminal. Um nó não-terminal é um nó que se divide em nós filhos. Tal divisão é determinada por uma condição sobre o valor de um único atributo, e que vai dividir os exemplos de acordo com a condição, em outros nós. Um nó que não se divide é chamado de nó terminal, e a ele é atribuída uma classe. Cada exemplo no conjunto cai em um dos nós terminais [5].

A Figura 1 ilustra a classificação por árvore de decisão.

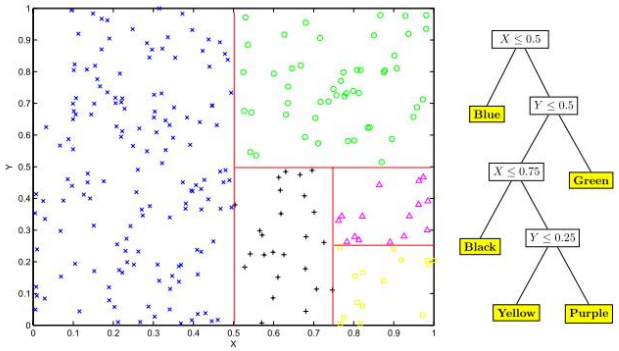

Figura 1. Exemplo de partições em árvores de decisão, adaptado de [5].

O processo de crescimento de uma árvore de classificação, necessita responder a quatro questões básicas: (1) Como escolher as condições para dividir cada nó? (2) Que critério devemos usar para dividir um nó pai em seus nós filhos? (3) Como vamos decidir quando um nó se tornar um nó terminal (parar a divisão)? (4) Como vamos atribuir uma classe a esse nó terminal?

O pseudocódigo 1 exemplifica os dois algoritmos.

```
INDUCAOCARTEC4.5(exemplos, subAtributos)
 1  if CRITERIOPARADA(exemplos)
 2      ESCOLHECLASSE(exemplos)
 3  else
 4      melhor = ESCOLHEATRIBUTO(subAtributos, exemplos)
 5      arvore =nova arvore com nó raiz= melhor
 6      particao = ESCOLHEPARTICAO(melhor)
 7      while particao
 8          ex_p =elementos de exemplos com melhor= p
 9          subAvr = INDUCAOCARTEC4.5(ex_p, subA − melhor)
10          ADICIONARAMOARVORE(p, subAvr)
11
12  PODAARVORE(arvore)
```

Serão explicadas com mais detalhes as diferenças entre CART e C4.5 com respeito aos métodos do pseudocódigo, **EscolheParticao**(), **EscolheAtributo**(), **CriterioParada**(), **EscolheClasse**(), e **PodaArvore**().

B. CART (Classification and Regression Trees)

O algoritmo CART é resultado de um trabalho publicado em 1984 [3].

O resultado deste algoritmo é sempre uma árvore binária que pode ser percorrida da sua raiz até às folhas respondendo apenas a questões simples do tipo sim/não [2].

Para o método **EscolheAtributo()**, o critério usado pelo algoritmo para medir a impureza de um nó, é o índice de Gini, criado por Conrado Gini em 1912, este índice mede a heterogeneidade dos dados.

Este índice num determinado nó é dado por:

$$G = 1 - \sum_{i=1}^{c} p_i^2$$

Onde:

p_i é a frequência relativa de cada classe em cada nó e c é o número de classes.

Quando este índice é igual a zero, o nó é puro. Por outro lado, quando ele se aproxima de um, o nó é impuro (aumenta o número de classes uniformemente distribuídas neste nó). Quando, o critério de Gini é utilizado tende-se a isolar num ramo os registros que representam a classe mais freqüente.

Para o método **EscolheParticao()** o algoritmo sempre faz uma divisão binária. Suponha que um determinado atributo é composto por M distintas categorias, o conjunto S de possíveis desdobramentos desse nó é dado por $2^{M-1} - 1$. A

melhor divisão S do nó é dada pela redução da impuridade em dividir o nó em filhos direito e esquerdo. A melhor divisão para um atributo, é a que tem a menor impuridade de todas as divisões possíveis $(2^{M-1}-1)$ para o atributo.

Para o método **CriterioParada()** o algoritmo cresce a árvore até a saturação. Depois de dividir o nó raiz, repetimos o mesmo processo para os nós filhos nesse caso 2, pois o CART só realiza partições binárias. Este processo de divisão sequencial de construir uma árvore de camada por camada, é chamado de particionamento recursivo. A árvore binária é dividida até que nenhum dos nós possa ser dividido mais, até que os nós sejam os mais puros possíveis, ou podemos definir um número mínimo de exemplos para que o nó pare, mas é preferível deixar crescer até a saturação. Mas esses dados podem estar super ajustados ao conjunto de treino, e conter nós com baixa significância estatística, por isso depois de gerada a árvore é necessário podá-la.

Para o método **EscolheClasse()** o algoritmo usa a chamada regra da pluralidade, onde a classe associada ao nó terminal é a classe que pertence a maioria dos exemplos atribuídos aquele nó.

Para o método de **PodaArvore()** O algoritmo usa uma medida chamada taxa de erro ajustada. Esta medida incrementa cada taxa de classificação errada de cada nó para o conjunto de treinamento pela imposição de uma penalidade baseada no número de folhas da árvore. O objetivo é podar primeiro os ramos que possuem um menor poder preditivo por folha. A fórmula da taxa de erro ajustada é:

$$EA(T) = E(T) + \alpha ContadorFolhas(T)$$

Onde α é um fator de ajuste incrementado gradualmente para criar novas subárvores.

Para encontrar a primeira subárvore, as taxas de erro ajustadas para todas as possíveis subárvores contidas no nó raiz são avaliadas com α sendo gradualmente aumentado. Quando a taxa de erro ajustada para alguma subárvore for menor do que a taxa da árvore completa, então tem-se a primeira subárvore candidata. Todos os ramos que não fazem parte desta subárvore são eliminados e então o processo se reinicia a partir desta subárvore para se obter uma segunda. O processo termina quando todos os caminhos até o nó raiz forem podados. Cada uma das subárvores são candidatas a fazerem parte do modelo final. O próximo passo é escoher dentre as subárvores, aquela que classifica melhor novos dados por meio de um conjunto de validação. Isto é, a árvore que classificar os novos registros com a menor taxa de erro é a escolhida.

C. C4.5

O algoritmo C4.5 é resultado de um trabalho publicado em 1993 por Quinlan [4].

O C4.5 é diferente do CART porque não é obrigatório fazer uma divisão binária.

Árvores menores são mais facilmente entendidas, e suscetíveis a ter maior precisão, então os algoritmos tem de se preocupar em induzir árvores o menor possível. Visto que é inviável garantir a minimalidade da árvore, o método **EscolheAtributo()** faz uma busca gulosa, selecionando a característica que maximiza a divisão dos dados por meio de entropia.

A entropia caracteriza a impureza dos dados, num conjunto de dados, ela caracteriza a falta de homogeneidade dos dados de entrada em relação a sua classificação. Por exemplo, a entropia é maxima (igual a 1) quando o conjunto de dados é heterogêneo [6].

Dado um conjunto de entrada S que pode ter c classes distintas, a entropia de S será dada por:

$$E(S) = \sum_{i=1}^{c} -p_i \lg p_i$$

Onde:

p_i é a proporção de dados em S que pertence a classe i.

O ganho de informação para um atributo A, de um conjunto de dados S, nos da a medida da diminuição da entropia esperada quando utilizamos o atributo A, para fazer a partição do conjunto de dados.

Seja $P(A)$ o conjunto de valores que o atributo A pode ter, seja x um elemento desse conjunto, e seja S_x um subconjunto de S formado pelos dados em que $A = x$, a entropia que se obtém ao particionar S em função do atributo A é dada por:

$$E(A) = \sum_{x \in P(A)} \frac{|S_x|}{|S|} EntropiaS_x$$

O ganho de informação será dado por:

$$ganho(S, A) = Entropia(S) - E(A)$$

Onde:

$Entropia(S)$ é uma medida de não homogeneidade do conjunto S.

$E(A)$ é uma medida de não homogeneidade estimada para o Conjunto S caso utilize o atributo A para fazer a próxima partição.

Para o método **EscolherPartição()** o algoritmo atribui a cada valor do atributo um ramo próprio. Este tipo de partição, embora permita extrair da característica todo o seu conteúdo informativo, tem como principal desvantagem a criação de um grande número de ramos muitas vezes completamente desnecessários, o que implica a formação de árvores de dimensões muitas vezes exageradas. Mas no final do algoritmo fazemos a poda onde esse problema será resolvido.

Para o método **CriterioParada()** o algoritmo só para de dividir se cada folha contém casos de uma única classe, ou até não ter como particionar mais porque os dois casos

têm os mesmos valores para cada atributo, mas pertencem a classes diferentes.

Para o método **EscolhaClasse()** o algoritmo atribui ao nó terminal a classe mais provável dentro dos exemplos.

Para o método **PodaArvore()** o algoritmo usa a poda baseada no erro, ele permite utilizar o próprio conjunto de treino para efetuar a poda da árvore e tem a vantagem de não obrigar a separação do conjunto de treino, em conjunto de treino e conjunto de teste.

Num dado nó que classifique N casos do conjunto de treino, k deles incorretamente, o erro aparente será $\frac{k}{N}$. Podemos olhar para estes números como representando uma experiência com k eventos em N experiências. Se optarmos por esta perspectiva podemos nos questionar sobre qual o significado destes valores em relação ao fenômeno que estamos tratando. Embora a partir destes dados não seja possível calcular a probabilidade de ocorrência do evento (ou seja, a probabilidade de erro) de uma forma exata, ela própria possui uma distribuição de probabilidades *a posteriori* que é normalmente representada por um par de limites de confiança.

Para um dado fator de confiança α, o limite superior desta probabilidade pode ser encontrado a partir dos limites de confiança da distribuição binomial. Estes limites inferior e superior que se denotam respectivamente por p_i e p_s representam os valores de probabilidade que permitem dizer que a probabilidade real de um acontecimento (que ocorreu k vezes em N experiências) estar fora do intervalo definido por estes valores é de $1 - \alpha$.

Com base nestes valores é possível calcular o limite superior do erro para cada nó. Sempre que o valor do erro por si só seja inferior à soma do erro dos seus descendentes o nó é podado, sendo substituído por uma folha cuja classificação será a da classe mais provável. Como é evidente, este cálculo deverá ser feito de baixo para cima, visitando sempre em primeiro lugar os descendentes de cada nó.

D. Análise de Complexidade

A complexidade dos dois é parecida, só difere no método EscolhePartição(), pois o CART faz uma divisão binária sempre e por isso tem que comparar os atributos $2^{A-1} - 1$ vezes.

A Tabela I mostra a análise de complexidade do CART e C4.5, onde m são os atributos, p é o numero diferente de valores para o atributo m, n os exemplos e a profundidade da árvore é $O(\log n)$.

Tabela I
ANÁLISE DE COMPLEXIDADE

	Crescer a árvore	Podar
CART	$O(m^{2^{p-1}} n \log n)$	$O(n(\log n)^2)$
C4.5	$O(mn \log n)$	$O(n(\log n)^2)$

III. EXPERIMENTOS

Foram utilizadas 66 amostras, envolvendo 7 bairros da cidade de Ouro Preto. As amostras contém 9 atributos que são mostrados na II.

Tabela II
ATRIBUTOS

Bairro	Nome do Bairro
Topografia	A - Aclive, D - Declive P - Plano I- Irregular
Utilização	R - Residencial C - Comercial T - Terreno
Estado Conservação	O - Ótimo B - Bom R - Regular M - Mau
Esgoto	Se existe rede de esgoto Sim ou Não
Pavimentação	P - Poliédrico A - Asfalto I - Irregular
Preço m^2	Preço médio do m^2 do terreno por bairro
Dist UFOP	Distância média da UFOP em km
Dist Praça	Distância média da Praça Tiradentes em km

Para fazer a classificação dos dados foi utilizado o *software* de domínio público, denominado *Waikato Environment for Knowledge Analysis (Weka)* [7], da Universidade de Waikato, Nova Zelândia. O Weka contém vários algoritmos de classificação de dados entre eles o C4.5 (sua versão em java é chamada J4.8), e o CART.

Os testes foram feitos através do mecanismo de *Cross-validation* disponibilizado pelo Weka, com o parâmetro $folds = 10$. De acordo com o manual do Weka esse mecanismos funciona da seguinte forma:

- Defini-se o parâmetro $folds$ que por *default* é 10;
- O conjunto de dados é aleatoriamente reordenado e depois divide-se em conjuntos de tamanho = $folds$;
- Em cada iteração, um conjunto é usado para testes e os outros $n - 1$ são utilizado para o treinamento do classificador;
- Os resultados do teste são coletados e calculados sobre todos os conjuntos.

Este método permite estimar a acurácia da árvore. Se o parâmetro $folds$ for o número de exemplos no conjunto de dados o método tende a dar resultados menos confiáveis. No entanto, ainda é bastante útil para lidar com conjuntos de dados pequenos, uma vez que utiliza a maior quantidade de dados de treinamento a partir do conjunto de dados.

A Figura 2 mostra a árvore gerada pelo algoritmo CART.

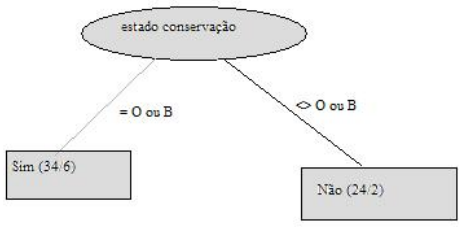

Figura 2. Árvore gerada pelo algoritmo CART

A Figura 3 mostra a árvore gerada pelo algoritmo C4.5.

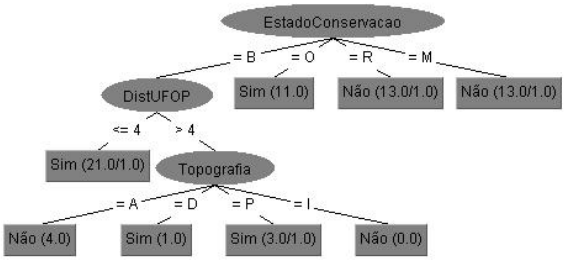

Figura 3. Árvore gerada pelo algoritmo C4.5

A Tabela III apresenta o resultado dos experimentos, quanto ao tempo de execução do algoritmo, e a acurácia do método, ou seja, qual a porcentagem de acerto da hipótese (h) em responder a pergunta (f).

Tabela III
EXPERIMENTOS

	Tempo (s)	Acurácia (%)
CART	0.18	89.39
C4.5	0.03	89.39

Podemos observar que o tempo de execução do CART foi maior do que o C4.5, isso se dá pelo fato dele realizar mais testes com os atributos na hora de realizar a partição que tem que ser binária.

Podemos observar que a acurácia dos dois algoritmos foi a mesma, mas eles geraram árvores diferentes, isso pode se dar por um super ajustamento ao conjunto de treino, pois para o CART apenas testando o estado de conservação se têm a mesma significância estatística em responder a pergunta, do que se testarmos também os atributos distância UFOP e topografia. O fato do CART só fazer divisões binárias também pode ter influenciado na árvore ter somente um nó.

IV. CONCLUSÃO

A conclusão que cheguei é que essa árvore ainda é um começo, seria necessário mais instâncias para fazer a classificação e também a ajuda de um especialista no assunto para saber a real necessidade das variáveis usadas nesse processo de classificação, e se há algum atributo mais importante para ser considerado. No âmbito espacial do problema, seria interessante também acrescentar aos atributos dados referentes as relações de vizinhança.

REFERÊNCIAS

[1] P. N. Stuart Russell, *Inteligência Artificial*, 2nd ed. Elsevier, 2004.

[2] J. M. M. R. da Fonseca, "Indução de Árvores de decisão (hist-class - proposta de um algoritmo não paramétrico)," Master's thesis, Universidade Nova de Lisboa, 1994.

[3] F. J. H. O. R. A. e. S. C. J. Breiman, L., *Classification and Regression Trees*. Wadsworth Press, 1984.

[4] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[5] A. J.Izenman, *Modern Multivariate Statistical Techniques*. Springer, 2008.

[6] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[7] G. H. B. P. P. R. I. H. W. Mark Hall, Eibe Frank, "The weka data mining software: An update," SIGKDD Explorations, Volume 11, Issue 1, 2009. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/