Árvores de decisão

Uma árvore decisão é um dos classificadores/regressores mais simples de entender e usar. Basicamente consiste em dividir recursivamente o dado por meio de questões "se, então". Se o training set for um conjunto do tipo $\{(\bar{x}, y)\}$ com $\bar{x} \in \mathbb{R}^d$, sendo d o número de features e y o label; então, o método vai dividir o conjunto de dados baseado nos valores de $\bar{x}$, tal que em cada grupo o valor de y seja o mais homogêneo possível. Um novo dado é classificado ao percorrer a árvore.

No caso do classificador, seu score ou precisão é difinido como

$$\text{score} = \frac{TP + TN}{TP + TN + FN + FP}$$

TP → verdadeiro positivo
TN → verdadeiro negativo
FN → falso negativo
FP → falso positivo

Em cada ramo da árvore, podemos usar o coeficiente de Gini para medir a "pureza" da decisão até aquele ramo. Esse coeficiente é definido como

$$\text{Gini}_m = \sum_K p_{m,K}(1 - p_{m,K})$$

sendo

$$P_{m,K} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = K)$$

e $I(\cdots)$ a função indicador. Essa última quantidade é fração de observações que são classificadas como K no nó m, que contêm N_m observações. Se o ramo apresentar apenas uma classe, então

$$P_{m,K} = 1 \Rightarrow Gini_m = 0$$

Por outro lado, se o ramo tiver igualmente representado cada uma das classes, então

$$P_{m,K} = 1/N_m$$

e

$$Gini_m = \sum_K P_{m,K} - \sum_K P_{m,K}^2$$

$$= \frac{1}{N_m} \sum_K 1 - \frac{1}{N_m^2} \sum_K 1$$

$$= 1 - \frac{1}{N_m}$$

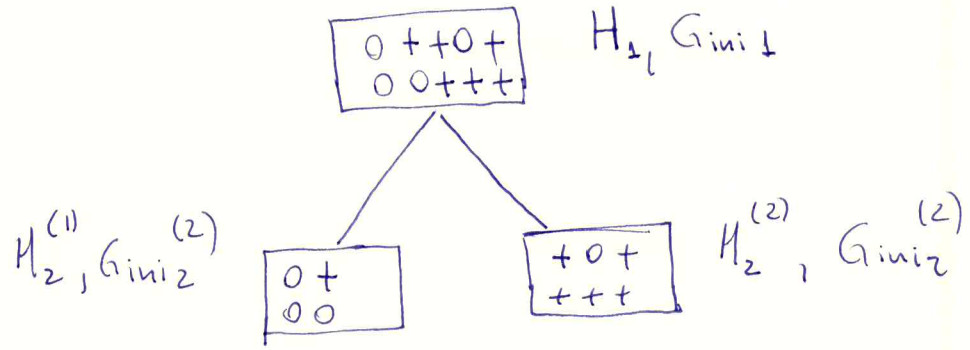
sendo N_m o número de observações no ramo m da árvore.

Uma outra medida de impureza é a entropia

$$H_m = - \sum_K P_{m,K} \log P_{m,K}$$

Nesse caso, se $P_{m,K} = 1$, $H_m = 0$. Por outro lado, se $P_{m,K} = 1/N_m$, então $H_m = \log N_m$

Os algoritmos envolvidos para produzir uma árvore de decisão escolhem os partições de modo a minimizar a impureza. Num exemplo de duas classes num dado ramo da árvore, teríamos algo do tipo

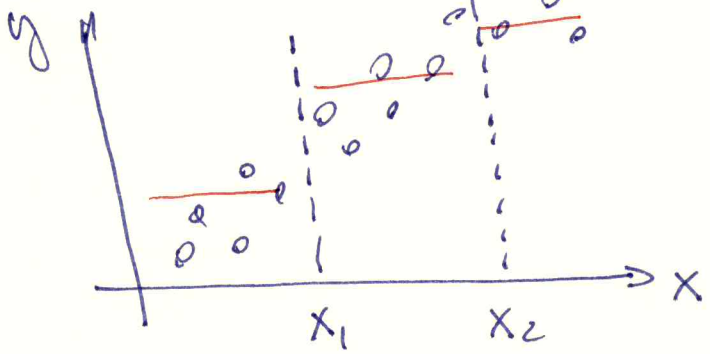


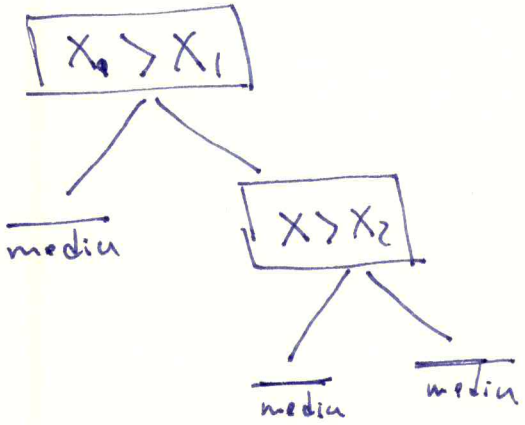
como $H_1 > H_2^{(1)}, H_2^{(2)}$

$Gini_1 > Gini^{(1)}, Gini^{(2)}$

Esse procedimento é repetido até os ramos conterem apenas uma classe. Vale notar que a depender da profundidade da árvore (número de ramificações), esse método pode aprender completamente o training set. Além disso, uma pequena variação no training set pode produzir uma árvore de decisão completamente diferente. Essas características fazem com que árvores de decisão sejam muito fáceis de conduzir a um overfitting dos dados, assim como apresentarem grande variância. Veremos que esses problemas podem ser contornados usando um procedimento chamado ensemble learning, que basicamente consiste em usar um conjunto de classificadores para realizar a classificação

O procedimento de regressão funciona de modo parecido ao de classificação. Nesse caso, a variável $y \in \mathbb{R}$ e a estimativa é usualmente baseada no valor médio de y dentro de cada ramo final da árvore. Num gráfico teríamos algo como

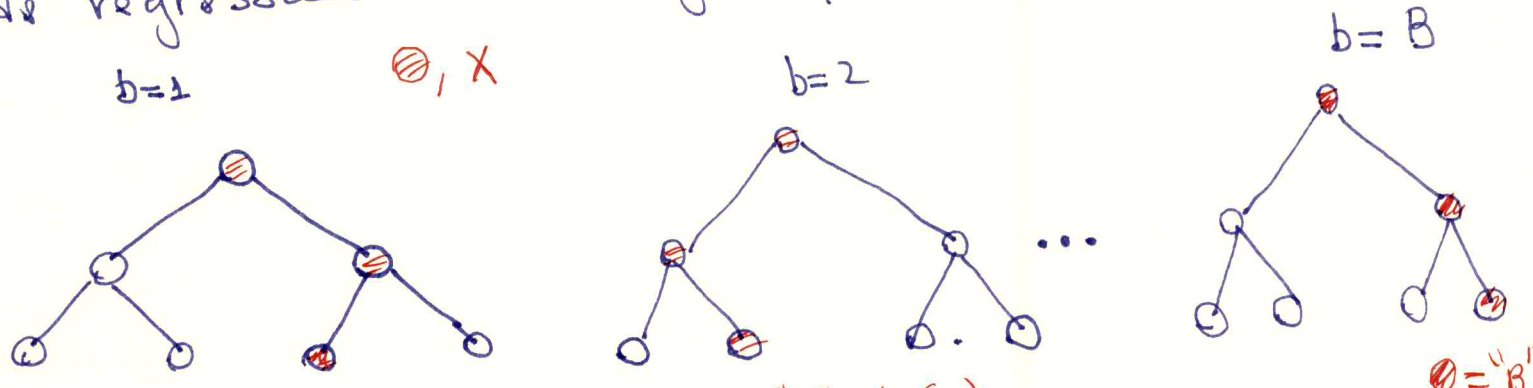




Exemplo notebook

Florestas aleatórias (random forests)

O método de random forest nada mais é do que uma combinação de árvores de decisão treinadas em subconjuntos aleatórios do training set. Desse conjunto de árvores, o resultado da classificação é obtido escolhendo o resultado mais comum entre as árvores (majority vote) ou tomando o valor médio das previsões de cada árvore, no caso de regressão. Num diagrama, teríamos algo como



Cada árvore é treinada, inclusive, com um conjunto diferente de features. Desse modo, é possível estimar a importância de uma feature baseada no número de vezes que ela é usada para dividir o dado e também na qualidade de cada separação. Podemos usar também o dado que não ajustado à árvore para estimar o erro, o chamado out-of-bag error. Essa quantidade também pode ser usada para quantificar a importância de uma feature. Nesse caso, temos a chamada Breiman importance, que consiste em permutar a j-ésima feature entre o dado usado para treinar a árvore (in-bag) e o dado não usado (out-of-bag), e estimar a diferença entre o valor médio do out-of-bag error sobre várias realizações. A ideia é que features que produzam grande variação desse erro sejam mais importantes para o processo de decisão. De modo geral, esse procedimento de combinar classificadores/regressores treinados em diferentes subconjuntos do training set é denominado de bagging ou bootstrap aggregating (bagging). Esse procedimento pode ser usado com outros métodos de machine learning, conduzindo uma menor variância (estabilidade) e prevenindo o over fitting.

Exemplo notebook