Aprendizado de regras de classificação

Marcelo K. Albertini

14 de Maio de 2014

Aprendizado de regras de decisão

Regras são fáceis de entender e populares em mineração de dados

- Hipóteses de tamanho variável
  - Qualquer função booleana pode ser representada

Algoritmos de aprendizado para regras podem ser

- Busca construtiva
  - Conjunto de regras é construído pela adição de novas regras
  - Cada regra é construída pela adição de condições
- Impaciente (constrói hipótese explicitamente)
- Em lote (usualmente)

Espaço de hipóteses de regras

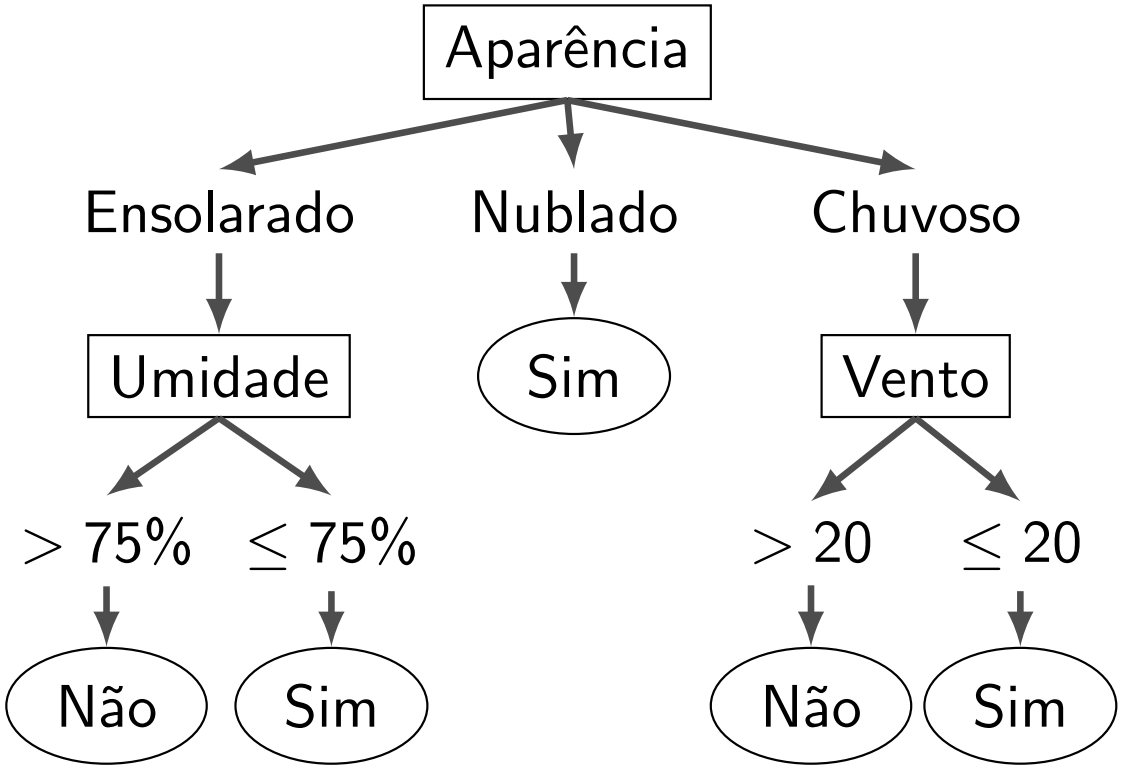
- Uma regra é uma conjunção de condições
  - Cada condição é na forma de um teste: $x_j = v$, ou $x_j \leq v$, ou $x_j \geq v$, ...
    onde v é um valor para o atributo x_j

Exemplo: $(x_1 = \texttt{Ensolarado}) \wedge (x_2 \leq 75\%) \Rightarrow y = 1$

- Um conjunto de regras é uma disjunção de regras
  - Tipicamente, todas as regras são para uma classe (para $y = 1$)
  - Um exemplo é classificado em $y = 1$ se qualquer regra é satisfeita

Exemplo: $(x_1 = \texttt{Ensolarado}) \wedge (x_2 \leq 75\%) \Rightarrow y = 1$
$(x_1 = \texttt{Nublado}) \Rightarrow y = 1$
$(x_1 = \texttt{Chuvoso}) \wedge (x_3 \leq 20) \Rightarrow y = 1$

Relação com árvores de decisão



Qualquer árvore de decisão pode ser convertida em um conjunto de regras.

Relação com árvores de decisão

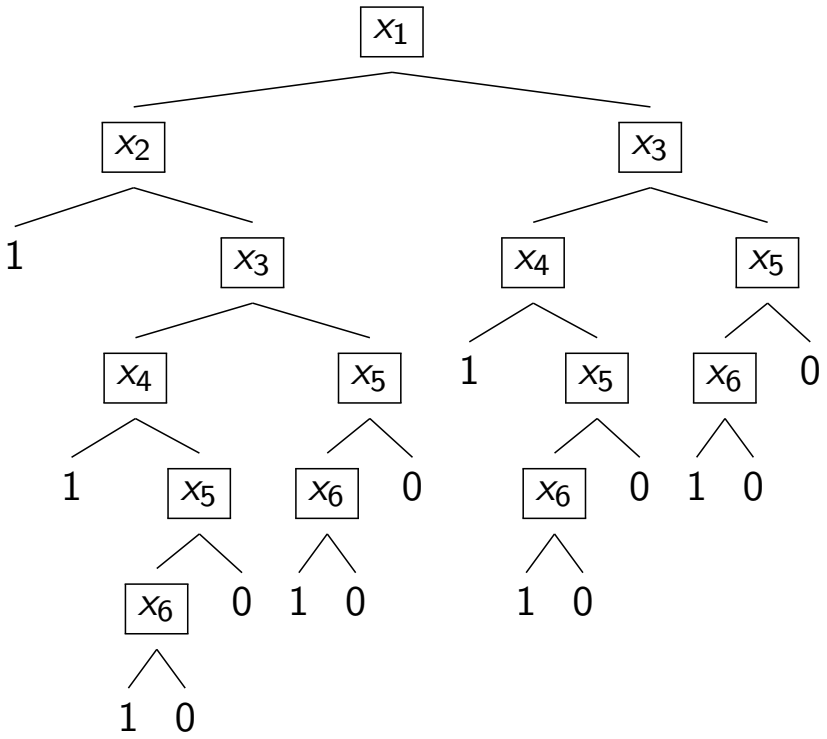
Problema da replicação

Um conj. de regras pequeno pode corresponder a uma árvore de decisão grande.

$x_1 \wedge x_2 \Rightarrow y = 1$

$x_3 \wedge x_4 \Rightarrow y = 1$

$x_5 \wedge x_6 \Rightarrow y = 1$



Algoritmo 1R

- Obtém um classificador com apenas um atributo
- Usa o atributo de menor erro para classificação
- Discretiza (encontra intervalos) atributos numéricos
- Discretização é guiada por um número mínimo de representantes da classe *ótima* por intervalo

R.C. Holte (1993). *Very simple classification rules perform well on most commonly used datasets*. Machine Learning. 11:63-91.

Simulador

`http://saedsayad.com/flash/OneR.html`

Exemplo: 1R

Para o conjunto de dados Íris:

```
petalwidth:
    < 0.8   -> Iris-setosa
    < 1.75  -> Iris-versicolor
    >= 1.75 -> Iris-virginica
```

usa regra com apenas um atributo que é separado em três intervalos.

```
1  /** Algoritmo 1R.
2  * Entrada: dataset com exemplos de treino
3  * Saída: regra do tipo 1R
4  */
5  Regra UmR(Dataset dataset, int minimoOtimo) {
6   Regra melhorRegra = new Regra();// Zero Regra
7
8   for (Atributo atr: dataset.atributos()) {
9    Regra regraNova = new Regra();
10
11    for (Valor val: valores(atr, minimoOtimo)) {
12     int cFreq = val.classeMaisFrequente();
13     // criar condição p/ classe +freq nesse valor do atr
14     regraNova.criarCondicao(atr, val, cFreq);
15    }
16
17    regraNova.aplicar(dataset);
18
19    if(melhorRegra.acuracia() < regraNova.acuracia())
20       melhorRegra = regraNova;
21   }
22   return melhorRegra;
23 }
```

Avaliação de Valor de atributos nominais/categóricos

`valores(atr, dataset, minimoOtimo)` retorna array com cada `Valor` possível do atributo e frequências de cada classe.

```
1  Valor[] valores(Atributo atr,
2                  Dataset dataset,
3                  int minOtimo) {
4    if (atr.CATEGORICO == true) {
5      Valor[] valores = new Valor[atr.numValores]();
6
7      // obter freq de cada classe p/ achar a +freq
8      for (Exemplo e: dataset.exemplos()) {
9        int idValorAtr = e.getIdDoValorAtributo(atr);
10       valores[idValorAtr].freqClasses[e.classeId()]++;
11     }
12
13     return valores;
14   }else
15   return valoresNumericos(atr,dataset,minimoOtimo);
16 }
```

Intervalos de `Valor` de atributos numéricos/contínuos

`valoresNumericos(atr, dataset, minimoOtimo)` divide `Atributo atr` numéricos em intervalos de `Valor`.

Regras

- ordena-se valores do atributo
- cada intervalo tem que ter no mínimo o número de `minimoOtimo` de exemplos da classe ótima, senão é necessário aumentar o intervalo
- o último intervalo não precisa respeitar o `minimoOtimo`
- o limite de um intervalo para outro constitui a mudança da classe ótima
- a classe ótima é a classe usada para classificação

Exemplo: intervalos de `Valor` de atributos numéricos/contínuos

Entradas

- Classe: 0 ou 1
- Valores de atributo x: de 10 a 41
- minimoOtimo = 3

0	1	1	0	0	0	1	1	1	0	0	0	0	0
23	12	20	32	23	35	25	27	28	12	32	37	10	41

Saída: intervalos

- $x < 24 \Rightarrow y = 0$
- $x < 30 \Rightarrow y = 1$
- $x >= 30 \Rightarrow y = 0$

Como encontrar os intervalos?

1º ordenamos pelos valores dos atributos

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

Como encontrar os intervalos?

1º ordenamos pelos valores dos atributos

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

2º cada valor define seu próprio intervalo

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

Como encontrar os intervalos?

1º ordenamos pelos valores dos atributos

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

2º cada valor define seu próprio intervalo

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

3º do menor para o maior, juntar intervalos para respeitar `minimoOtimo=3` e intervalos que concordam na classe

Como encontrar os intervalos?

1º ordenamos pelos valores dos atributos

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

2º cada valor define seu próprio intervalo

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

3º do menor para o maior, juntar intervalos para respeitar `minimoOtimo=3` e intervalos que concordam na classe

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

Como encontrar os intervalos?

1º ordenamos pelos valores dos atributos

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

2º cada valor define seu próprio intervalo

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

3º do menor para o maior, juntar intervalos para respeitar `minimoOtimo=3` e intervalos que concordam na classe

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

Como encontrar os intervalos?

1º ordenamos pelos valores dos atributos

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

2º cada valor define seu próprio intervalo

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

3º do menor para o maior, juntar intervalos para respeitar `minimoOtimo=3` e intervalos que concordam na classe

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

0	0	1	1	0	0	1	1	1	0	0	0	0	0
10	12	12	20	23	23	25	27	28	32	32	35	37	41

Intervalos de Valor de atributos numéricos/contínuos

`valoresNumericos(atr, dataset, minimoOtimo)` divide `Atributo atr` numérico em intervalos de valores.

```
1  Valor[] valoresNumericos(Atributo atr, Dataset dataset,
       int minimoOtimo) {
2    int[] valores = atr.valores();
3    int[] classes = dataset.classesDosExemplos();
4
5    sort(valores, classes);
6    Lista<Integer> intervalos
7              = new Lista<Integer>(valores, classes);
8
9    int prox = 0;
10   while (++prox < intervalos.size()) {
11     if (possivelJuntar(intervalos, prox-1, prox) ||
12         obrigatorioJuntar(intervalos, prox-1, prox)) {
13       juntar(intervalos, prox-1, prox);
14     }
15   }
16   return converterIntervalosEmValor(intervalos);
17 }
```

Aprendizado de uma regra: maneira geral

Criação de uma regra

Criamos uma regra começando com uma regra vazia e depois adicionamos mais testes, um por vez, até que a regra cubra somente exemplos positivos.

```
CriarUmaRegra(S) {
  R = {}
  while (S contém exemplos negativos) {
    escolher melhor teste x_j Θ v para adicionar a R, onde
        Θ pode ser {=, !=, =<, >=}
    R = R + x_j Θ v
    S = S - {exemplos que não satisfazem R}
  }
  return R
}
```

Escolha do melhor teste

- Regra atual R cobre m_0 exemplos negativos e m_1 exemplos positivos.
  - Seja $p = \frac{m_1}{m_0+m_1}$
- Regra proposta $R \cup \{x_j \Theta v\}$ cobre exemplos m'_0 e m'_1
- Ganho $= m'_1[(-p \log p) - (-p' \log p')]$

Objetivo

Queremos reduzir nossa surpresa (ao ponto em que temos certeza), mas queremos ao mesmo tempo que a regra cubra muito exemplos. A fórmula de Ganho é uma solução para esse dilema.

Aprendizado de um conjunto de regras (dividir-e-conquistar)

```
CriarConjuntoRegras(S) {
  A = {}

  while (S tem exemplos) {
    R = CriarUmaRegra(S)
    A = A + R // aumentar conjunto de regras
    S = S - {exemplos positivos cobertos por R}
  }

  return A;
}
```

Mais sobre procedimentos de busca

Encontrar o menor conjunto de regras é NP-Hard

Até agora vimos heurísticas gananciosas. Existem opções para melhorar os conjuntos de regras obtidos.

- Substituição em rodízio
  1. Após obter conjunto de regras completo, remover primeira regra.
  2. Obter exemplos não cobertos por ninguém
  3. Adicionar novas regras para cobrir exemplos
  4. (Opcional) Repetir para cada regra original

Mais sobre procedimentos de busca

Encontrar o menor conjunto de regras é NP-Hard

Até agora vimos heurísticas gananciosas. Existem opções para melhorar os conjuntos de regras obtidos.

- Substituição em rodízio
  1. Após obter conjunto de regras completo, remover primeira regra.
  2. Obter exemplos não cobertos por ninguém
  3. Adicionar novas regras para cobrir exemplos
  4. (Opcional) Repetir para cada regra original
- Backfitting
  1. Aplicar substituição em rodízio após a criação de algumas regras

Mais sobre procedimentos de busca

Encontrar o menor conjunto de regras é NP-Hard

Até agora vimos heurísticas gananciosas. Existem opções para melhorar os conjuntos de regras obtidos.

- Substituição em rodízio
  1. Após obter conjunto de regras completo, remover primeira regra.
  2. Obter exemplos não cobertos por ninguém
  3. Adicionar novas regras para cobrir exemplos
  4. (Opcional) Repetir para cada regra original
- Backfitting
  1. Aplicar substituição em rodízio após a criação de algumas regras
- Busca direcional
  1. Em vez de criar apenas uma regra por vez, criar B regras
  2. Para criá-las, adicionar cada teste possível a cada regra e manter as B melhores regras

Estimação de probabilidades a partir de números pequenos

Quando m_0 e m_1 são pequenos, a equação

$$p = \frac{m_1}{m_0 + m_1}$$

torna-se não confiável.

Alternativas

- Estimativa de Laplace
  1. Somar $1/2$ ao numerador e 1 ao denominador
     $p = \frac{m_1+0.5}{m_0+m_1+1}$
  2. Ou seja, na falta de mais evidências, esperamos $p = 1/2$
- Estimativa a priori geral
  1. Se crença prévia é de $p = 0.25$, somar k ao numerador e $4k$ ao denominador
     $p = \frac{m_1+k}{m_0+m_1+4k}$

Aprendizado de regras para múltiplas classes

E se temos mais de uma classe?

- Ordenar regras (lista de decisões)
- Voto ponderado

Exemplo: peso = acurácia $\times$ cobertura
cobertura é o número de exemplos em que a regra se baseia