Classificação
Representação de *Conhecimento*

Eduardo R. Hruschka

Baseado no curso de Gregory Piatetsky-Shapiro, disponível no sítio http://www.kdnuggets.com

Tendo-se em mente que:

- KDD é o processo *não trivial* de identificar padrões válidos, novos, potencialmente úteis e compreensíveis em dados.
- Antes de abordarmos os algoritmos para classificação propriamente ditos, vamos estudar alguns exemplos de seus *resultados,* no contexto:

Entradas $\rightarrow$ processamento $\rightarrow$ resultados (saídas).

Noção Intuitiva sobre classificação:

- The weather problem (Witten & Frank, 2000)

Outlook	Temperature	Humidity	Windy	**Play**
sunny	85	85	false	**no**
sunny	80	90	true	**no**
overcast	83	86	false	**yes**
rainy	70	96	false	**yes**
rainy	68	80	false	**yes**
rainy	65	70	true	**no**
overcast	64	65	true	**yes**
sunny	72	95	false	**no**
sunny	69	70	false	**yes**
rainy	75	80	false	**yes**
sunny	75	70	true	**yes**
overcast	72	90	true	**yes**
overcast	81	75	false	**yes**
rainy	71	91	true	**no**
rainy	63	84	true	**?**

Considerando-se que a tabela ao lado representa dados passados, como estabelecer um modelo para os valores do atributo meta *play?*

Visão Geral:

1. Tabelas de Decisão;
2. Árvores de Decisão;
3. Regras;
4. Representação baseada em exemplos:
   - Protótipos, *clusters*.

Representando *padrões estruturais*:

→ *Representação de conhecimento*;

- Existem muitas formas diferentes para representá-los:
  - Árvores, regras, protótipos, ...
- A representação determina o método de inferência;
- Entender os *resultados (saídas)* é chave para entender os métodos de aprendizado;
- Diferentes tipos de saídas para diferentes problemas:
  - *Clustering;*
  - Classificação / Regressão;
  - Associação.

1. Tabelas de Decisão:

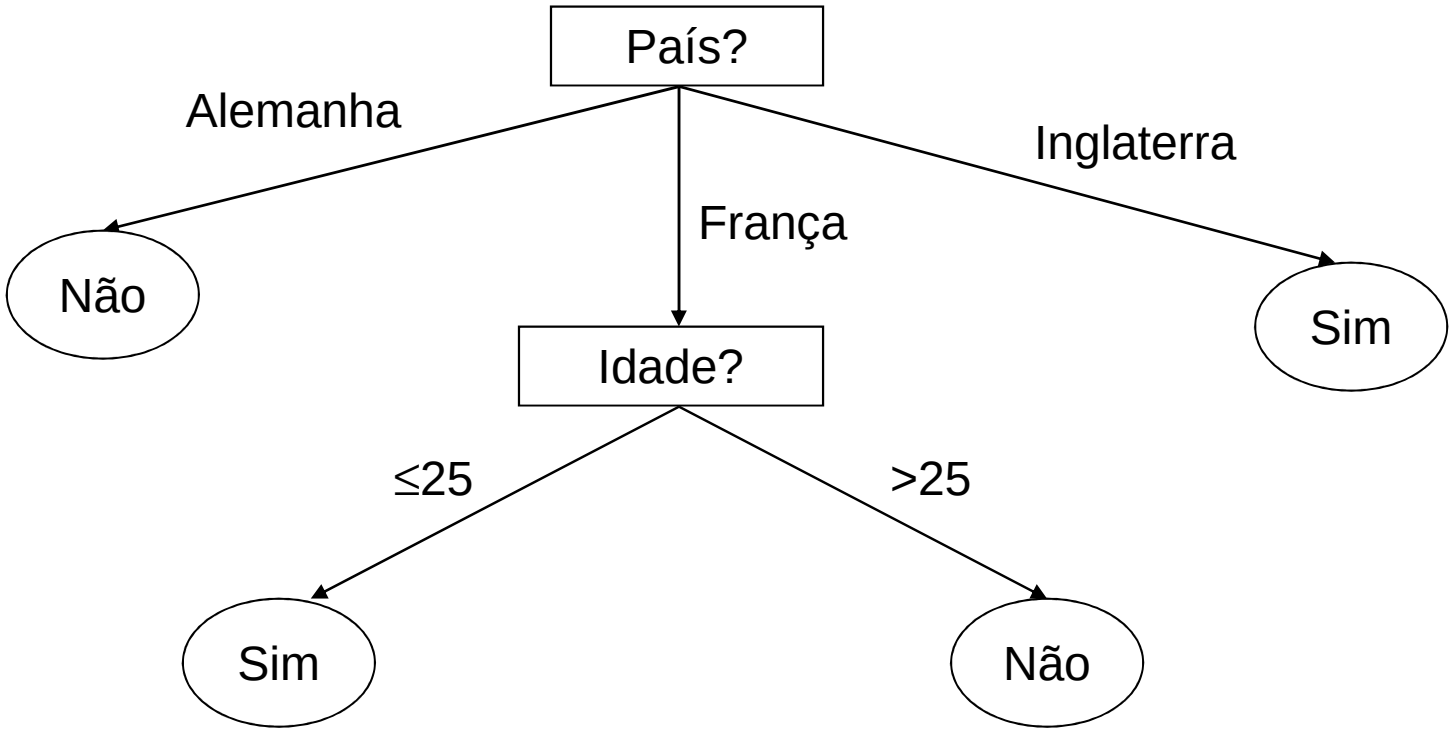
- Modo mais simples de representar o resultado da mineração:
  - Usar o mesmo formato das entradas.
- Tabela de decisão para o *weather problem*:

Outlook	Humidity	Play
Sunny	High	No
Sunny	Normal	Yes
Overcast	High	Yes
Overcast	Normal	Yes
Rainy	High	No
Rainy	Normal	No

- Principal problema: selecionar os atributos corretos;
- Pouca flexibilidade (relações entre atributos: >,<,etc.)

2. Árvores de Decisão

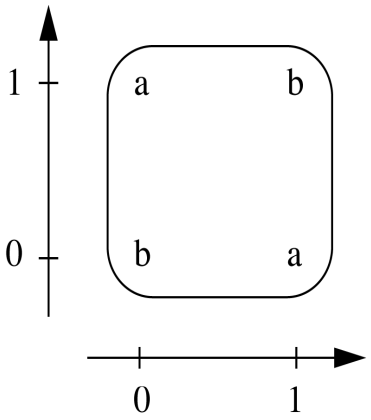
- Abordagem *dividir* para *conquistar;*
- Geralmente cada nó representa um teste para um atributo;
- Usualmente as folhas atribuem a classificação ou suas probabilidades;

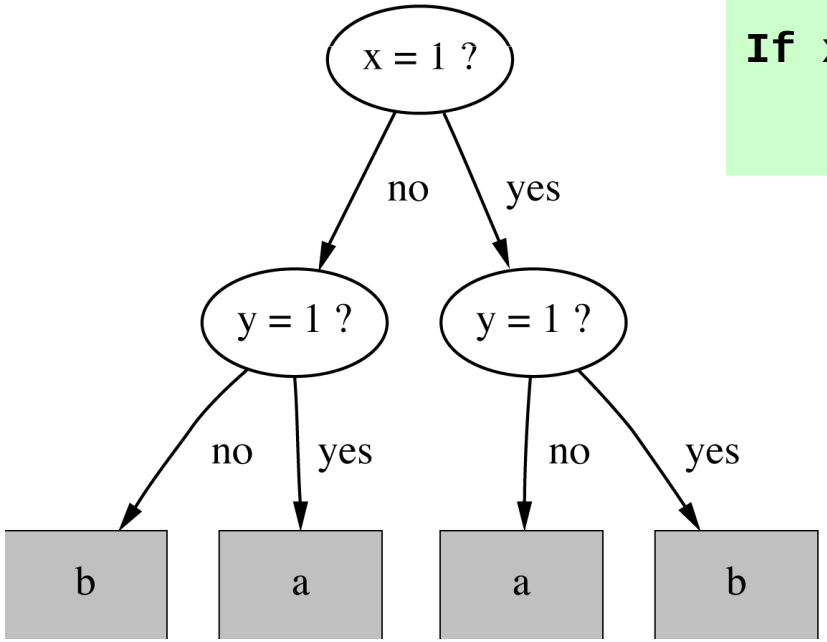


3. Regras de Classificação

- *Antecedente* (pré-condição): série de testes;
- Usualmente regras "Se ... Então ..."
- *Conseqüente* (conclusão): classes ou probabilidades associadas às classes;
- Pode-se *converter* uma árvore num conjunto de regras;
- Exemplo:
  - Se país=Alemanha Então classe=não;

Ilustrando relações entre regras e árvores:






```
If x = 1 and y = 0
   then class = a

If x = 0 and y = 1
   then class = a

If x = 0 and y = 0
   then class = b

If x = 1 and y = 1
   then class = b
```

Interpretando regras:

- O que fazer se duas ou mais regras são conflitantes?
  - Não concluir *nada* ?
  - Adotar a regra mais significativa?
  - …
- O que fazer se nenhuma regra se aplica ao exemplo de teste?
  - Não concluir *nada* ?
  - Escolher classe mais freqüente (conjunto de treinamento)?
  - …

Caso especial: classe *booleana*.

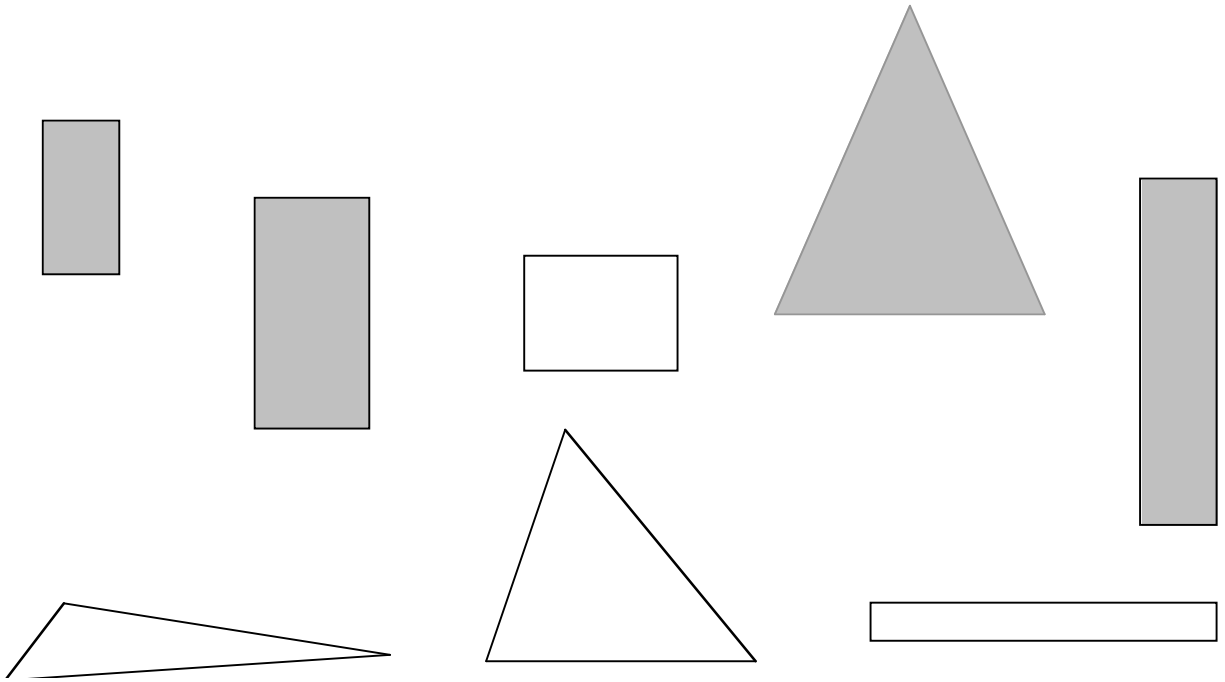
- Premissa: se um exemplo não pertence à classe "a", este pertence à classe "b";
- Aprender regras para a classe "a" e usar uma regra *default* para a classe "b";

```
Se x = 1 e y = 1 então classe = a
Se z = 1 e w = 1 então classe = a
Caso contrário classe = b
```

- Não gera conflitos, mas...

Problema das formas:

- Conceito alvo: *"em pé"*



Solução proposicional:

Largura	Altura	Lados	Classe
2	4	4	"em pé"
3	6	4	"em pé"
4	3	4	"deitado"
7	8	3	"em pé"
7	6	3	"deitado"
2	9	4	"em pé"
9	1	4	"deitado"
10	2	3	"deitado"

Se largura ≥ 3.5 e altura < 7.0
então "deitado"

Se altura ≥ 3.5 então "em pé"

Solução relacional:

- Comparar atributos entre si:

```
Se largura > altura então “deitado”
Se altura > largura então “em pé”
```

- Generaliza melhor;
- Relações: =, <, >, ...
- Custo computacional?

4. Representação baseada em exemplos:

- Forma simples de aprendizado:
  - Procurar por exemplos de treinamento mais parecidos com o exemplo de teste;
  - Conhecimento representado nos próprios exemplos;
  - Aprendizado baseado em exemplos (*instance-based learning / lazy learning*)
- Função de similaridade define o "aprendizado";
- Métodos: KNN (*k-nearest-neighbor*) e suas variações.
- Adaptar conceitos vistos em *clustering (funções de distância, normalização)* para classificação.
- *Filtrar* a base de dados (e.g. algoritmo de agrupamento de dados) e usar protótipos para classificação pode ser uma alternativa interessante.