Introdução à Mineração de Dados com Aplicações em Ciências Espaciais

Escola de Verão do Laboratório Associado de Computação e Matemática Aplicada

Rafael Santos

Programa

- ***Dia 1:*** Apresentação dos conceitos de mineração de dados, motivação e alguns exemplos.
- ***Dia 2:*** Algoritmos de classificação supervisionada e aplicações.
- ***Dia 3:*** Algoritmos de classificação não-supervisionada e aplicações. Algoritmos de mineração de associações.
- ***Dia 4:*** Visualização e mineração de dados. Outros algoritmos e idéias. Onde aprender mais.

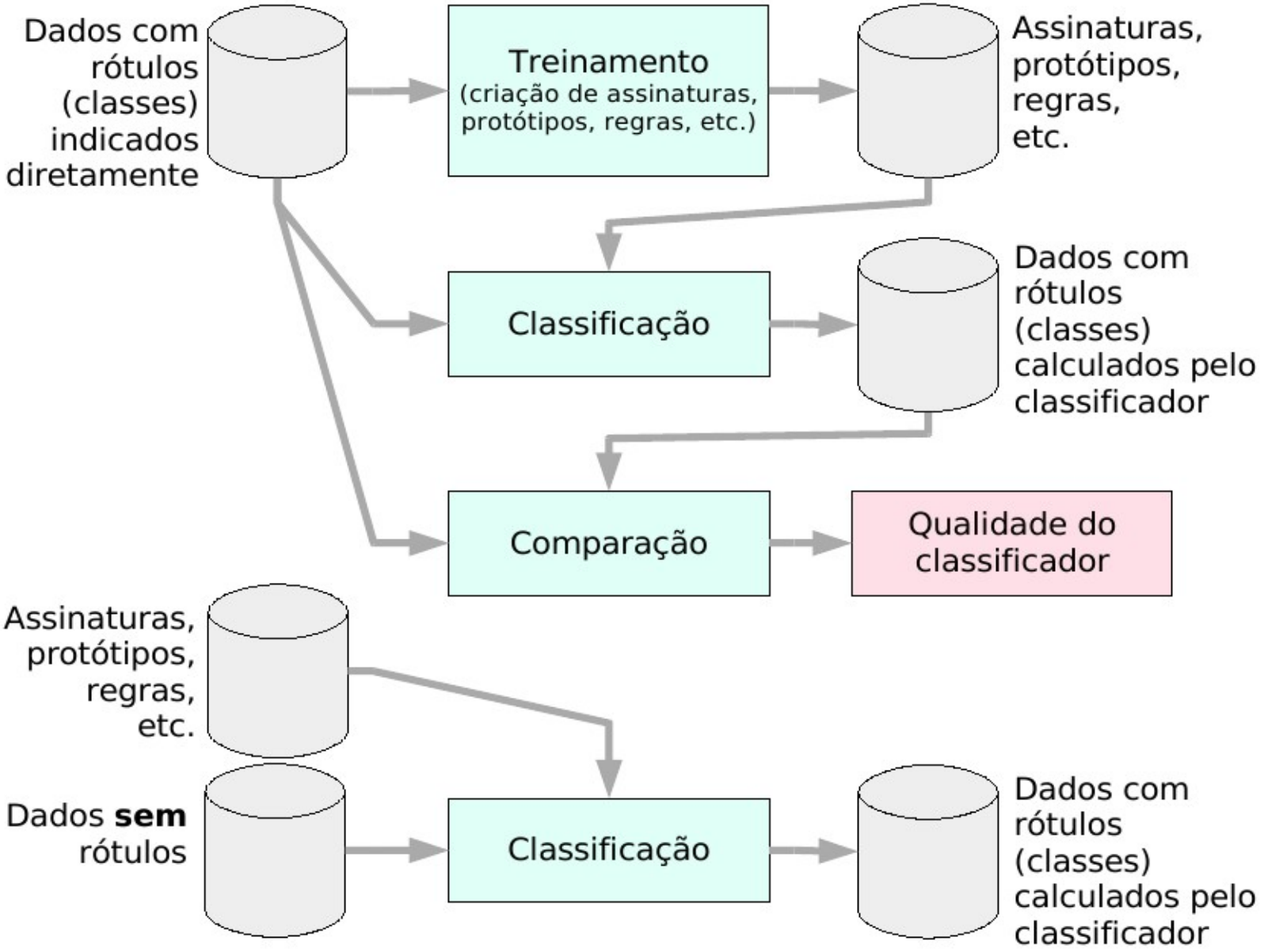
Classificação

Classificação

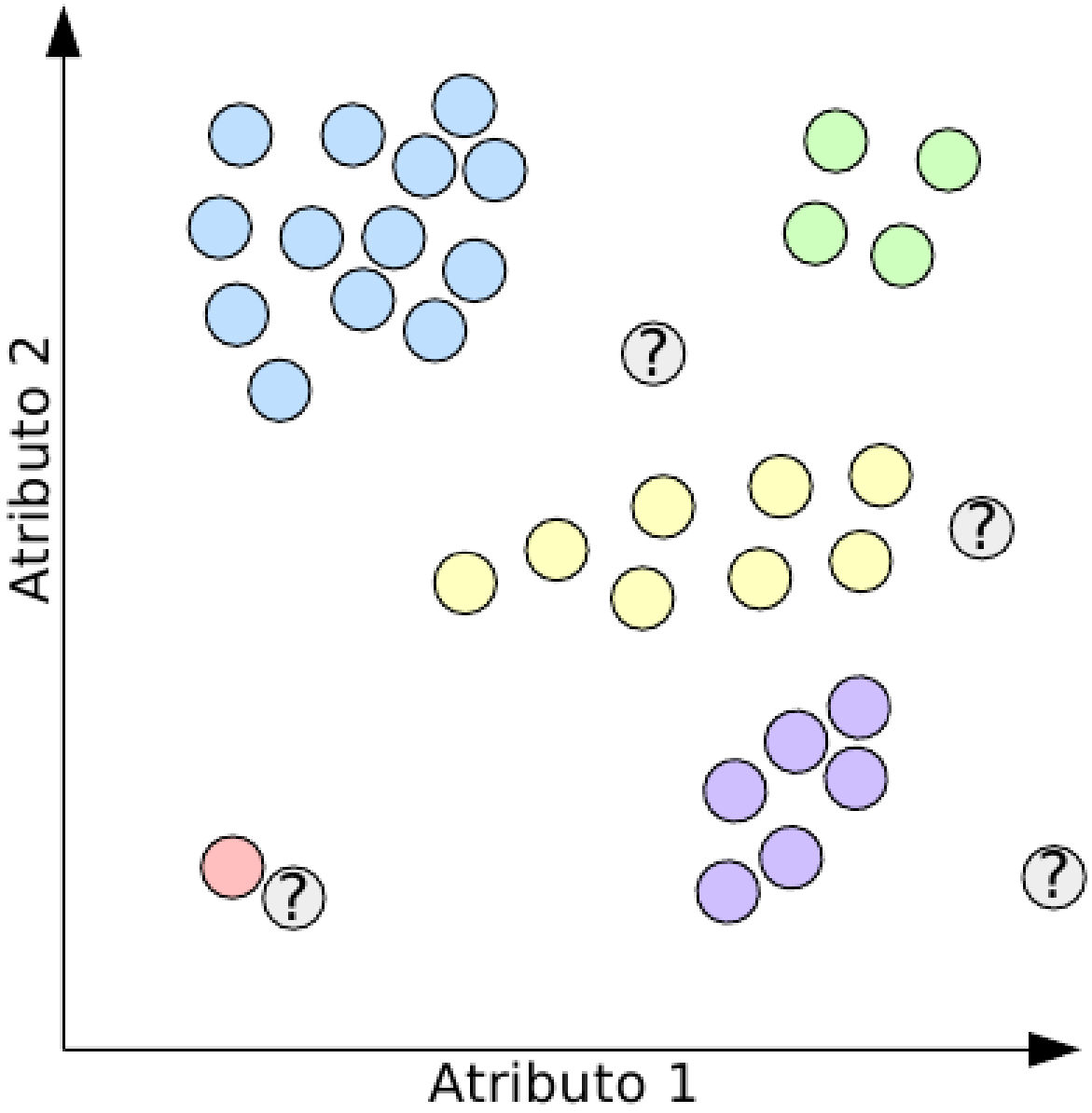
- **Predição de uma categoria ou classe discreta.**
- Como entrada: instâncias para as quais as classes são conhecidas.
  - Com isso criamos um **classificador** ou **modelo** (fase de treinamento).
- Como entrada em uma segunda fase, temos vários dados para os quais as classes não são conhecidas.
  - Usamos o classificador para indicar classes para estes dados.
  - Podemos avaliar o modelo classificando instâncias com classes conhecidas.

- Se temos como rotular instâncias, para que classificar?

Classificação

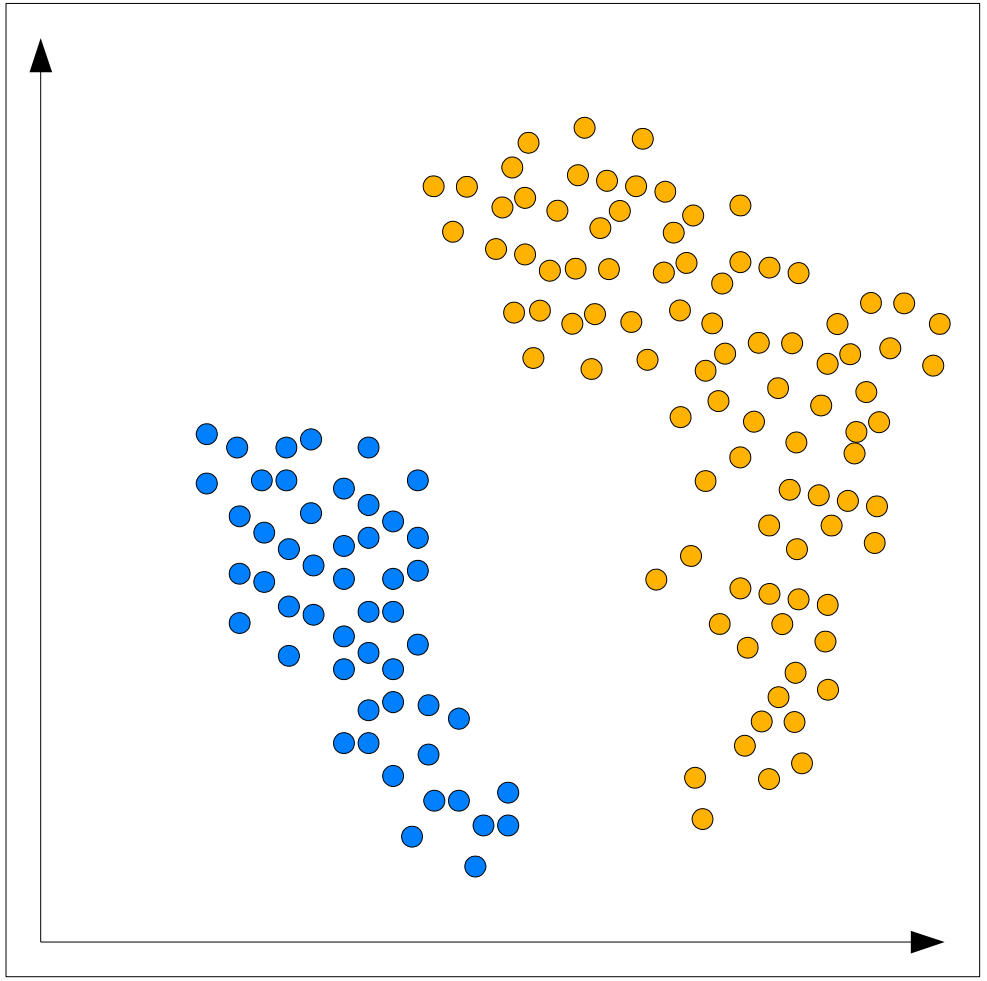


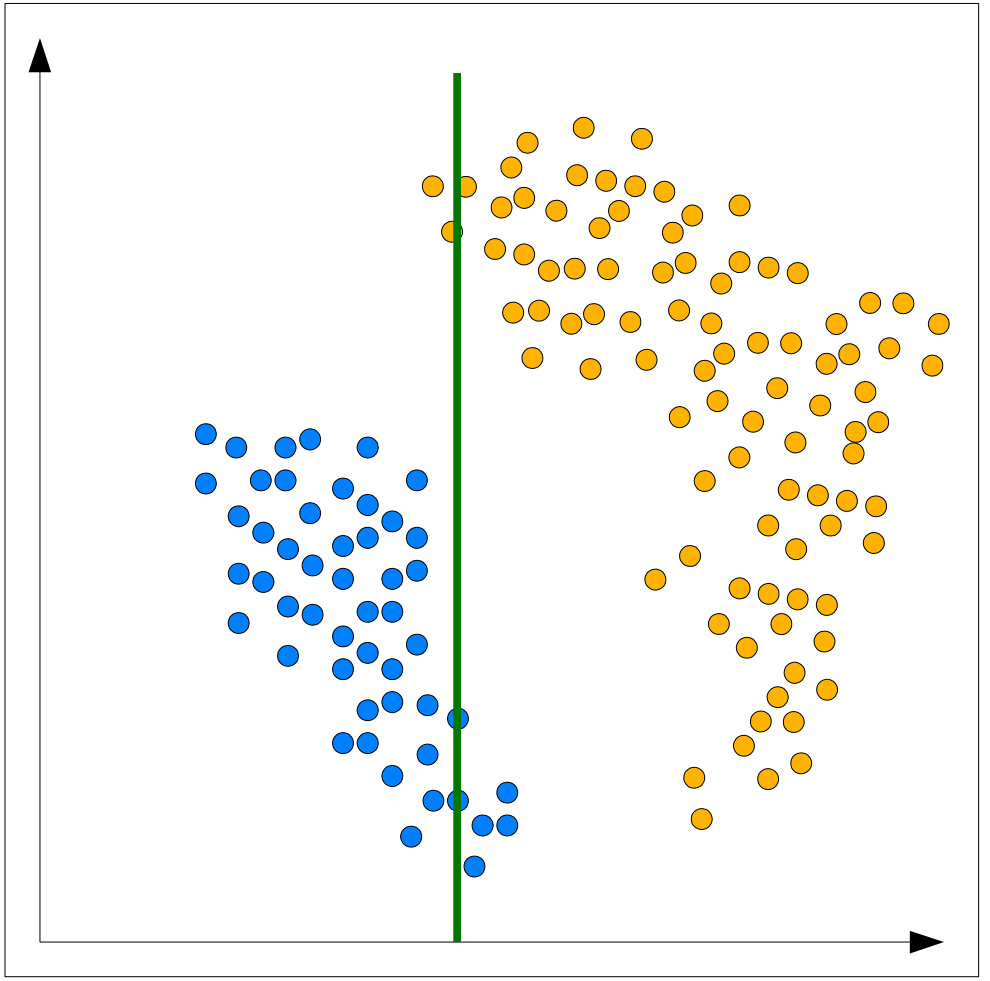
Classificação



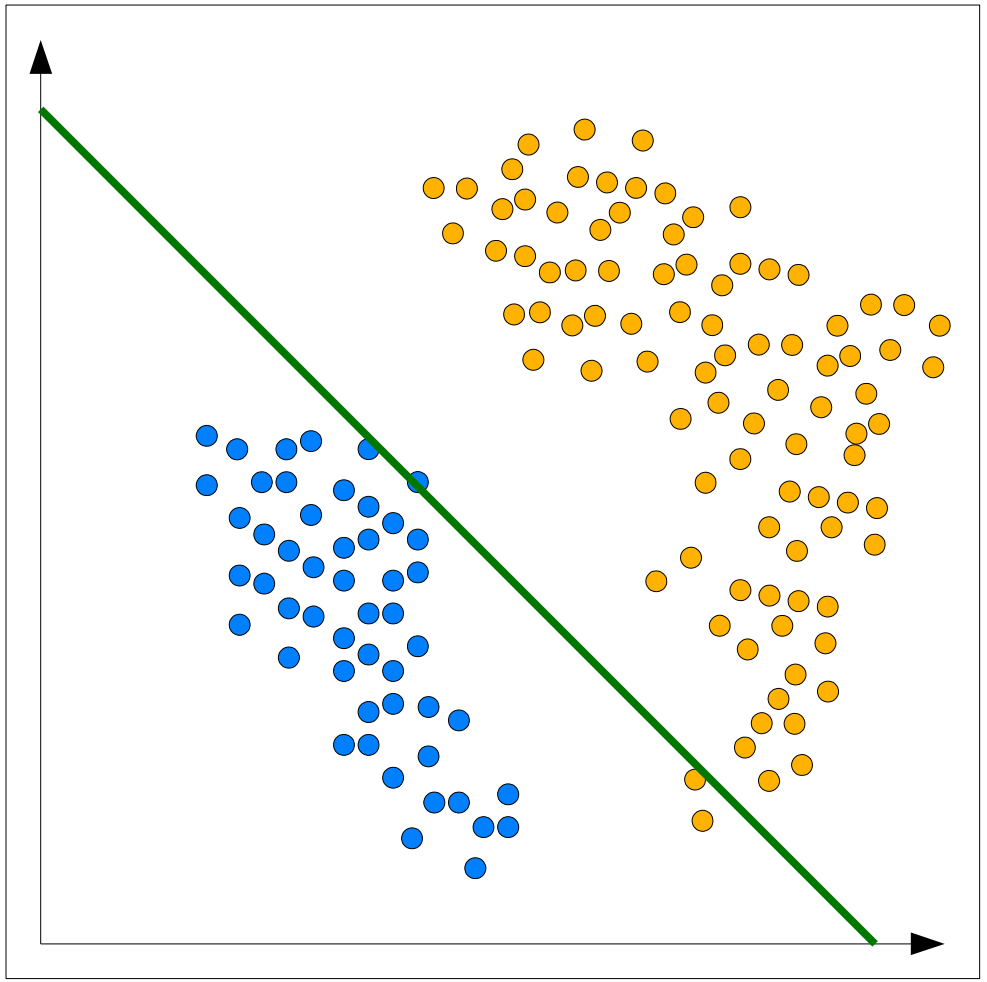
Devemos considerar a possibilidade de empate e/ou rejeição.

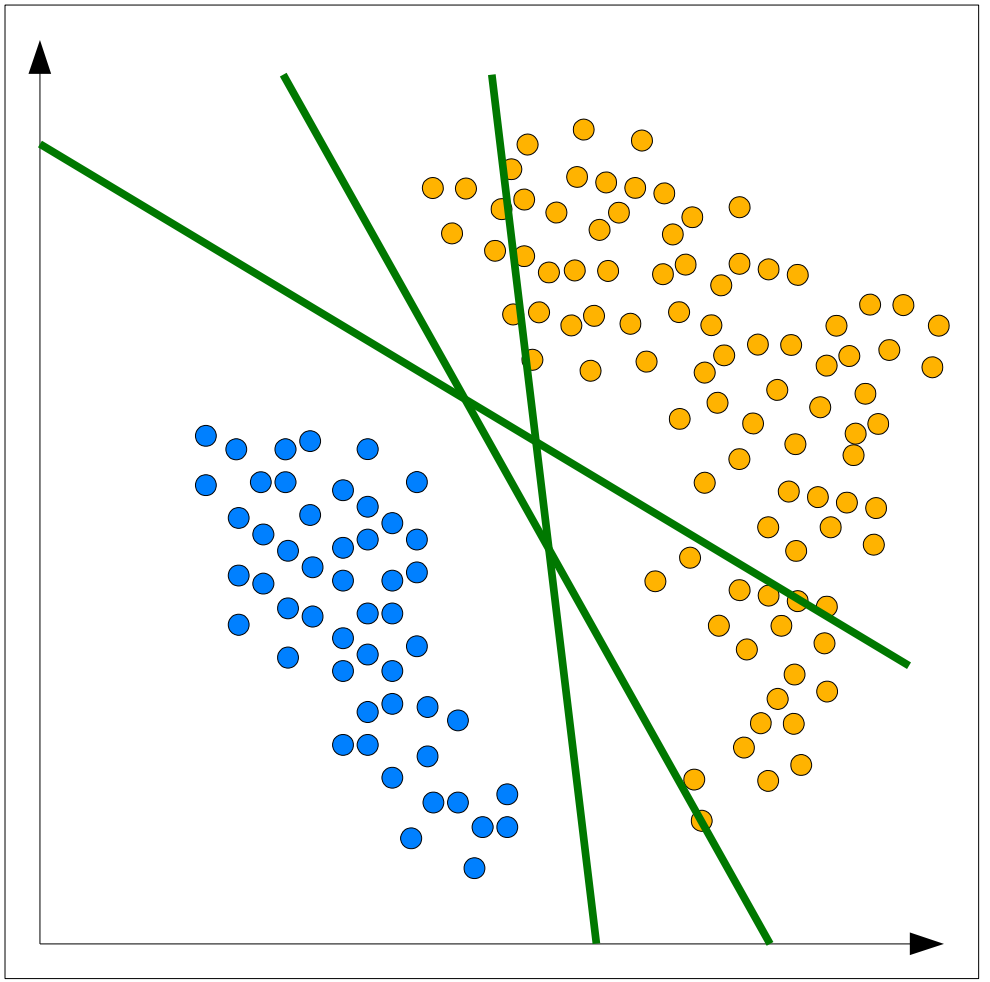
Classificação e Espaço de Atributos



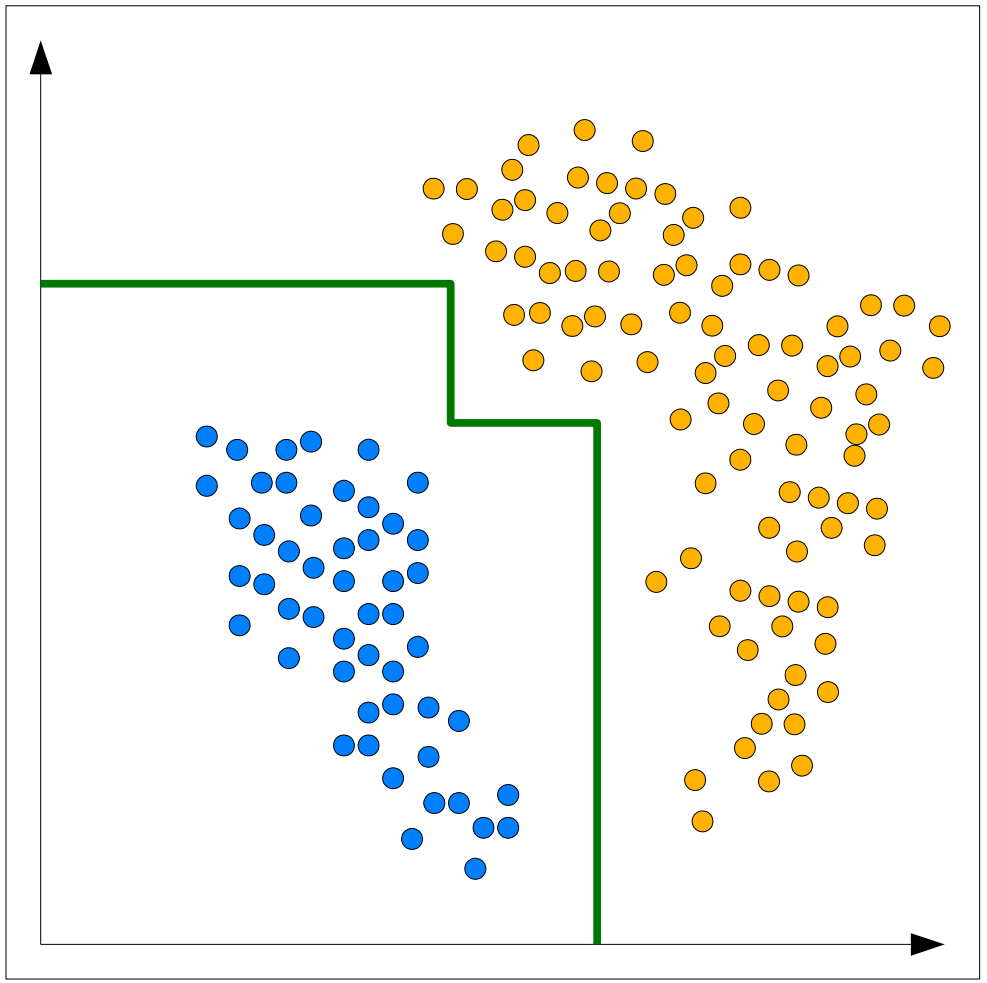


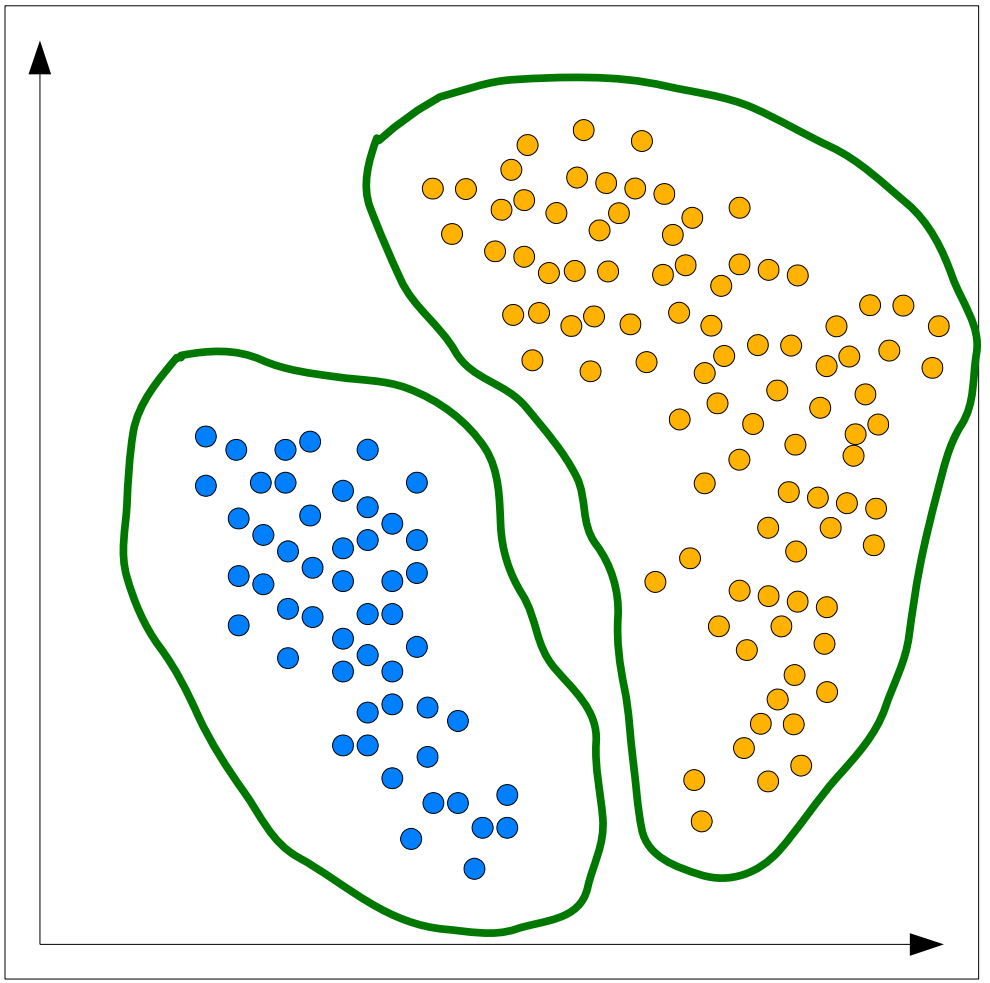
Classificação e Espaço de Atributos





Classificação e Espaço de Atributos



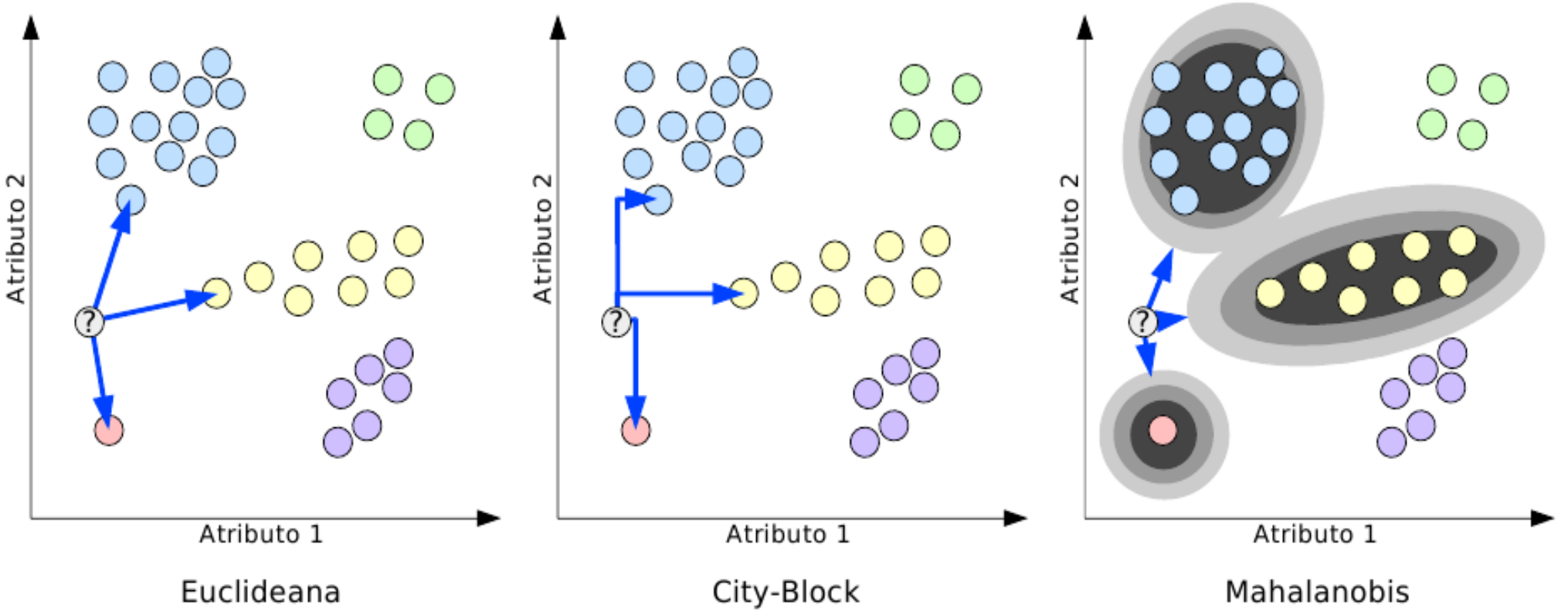


Classificação

- Métodos de classificação supervisionada:
  - Baseados em distâncias e diferenças, usando protótipos ou assinaturas: mínima distância euclideana e variantes.
  - Baseados em separabilidade (entropia): hiperparalelepípedo regular, árvores de decisão e variantes.
  - Baseados em particionamento: redes neurais (*back-propagation*), SVM (*support vector machines*).
  - Baseados diretamente nos dados: vizinhos mais próximos e similares.
- Existe superposição nesta taxonomia...

Classificação: Distâncias

- Medidas de distância

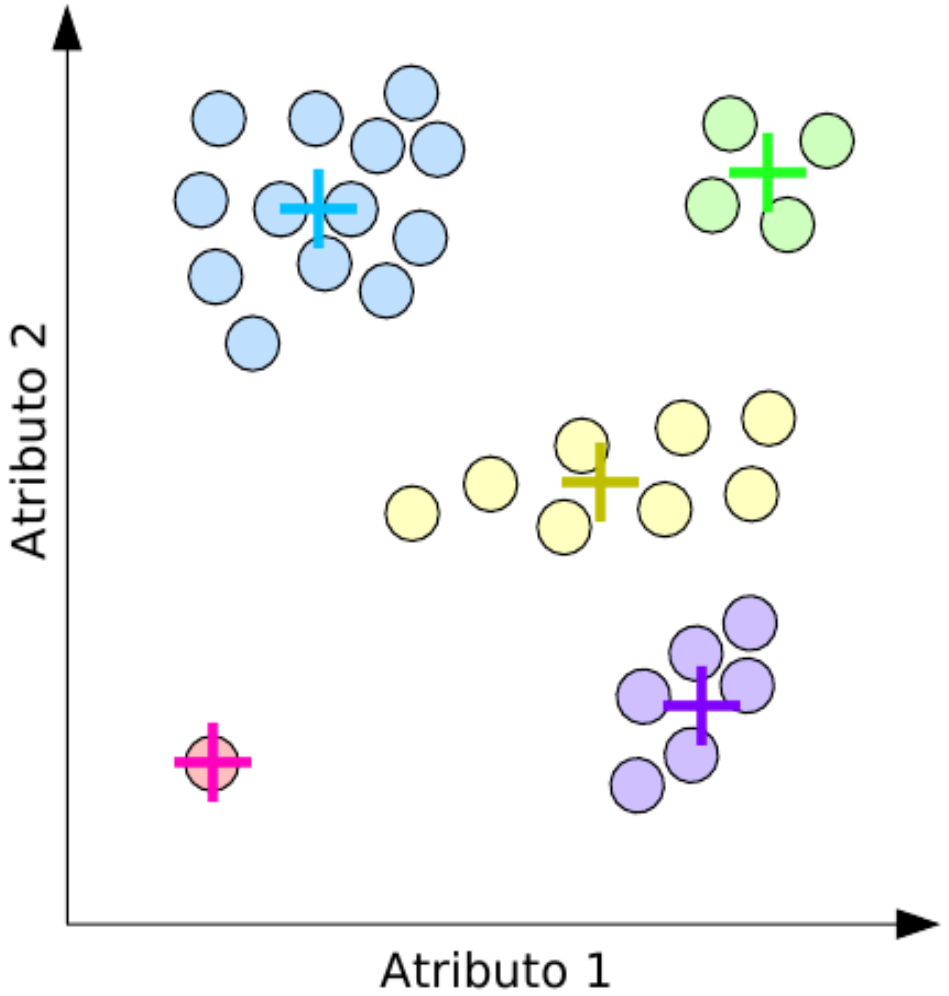


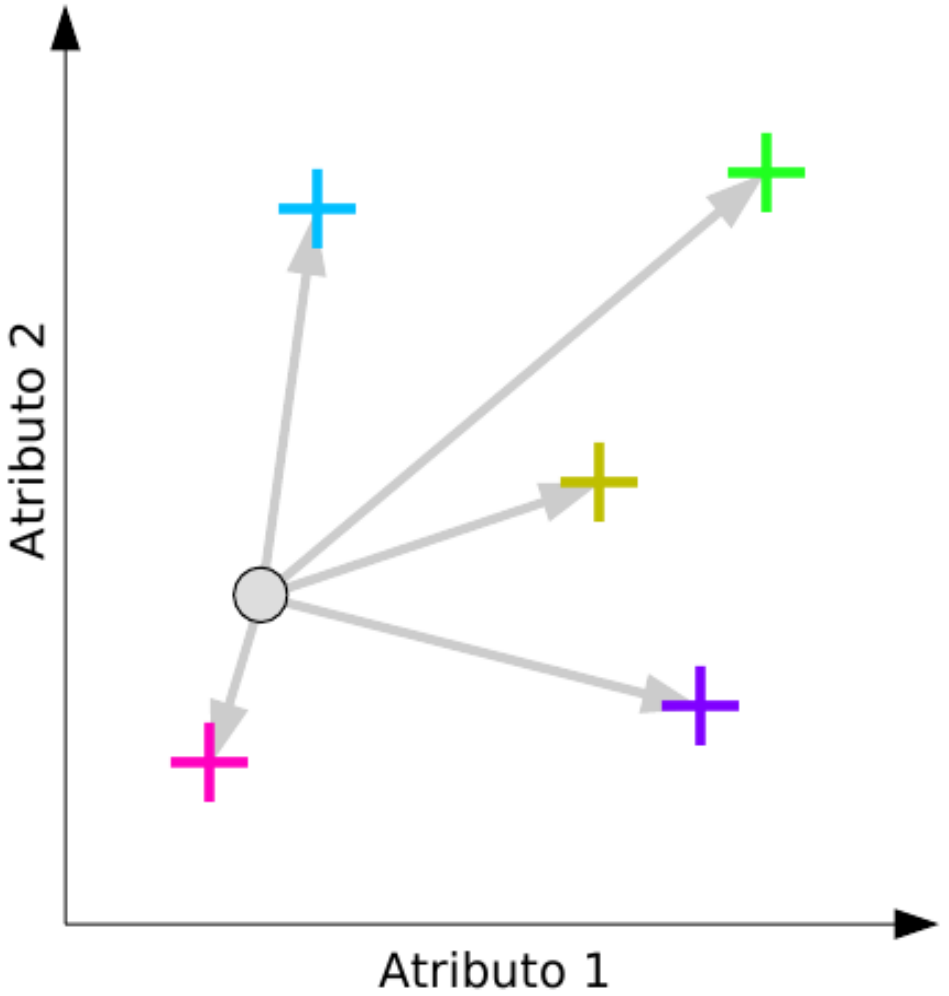
- Como usar atributos não-numéricos?

Classificação: Mínima Distância Euclideana

- Menor distância a protótipo.
  - Mais exatamente: Mínima Distância Euclideana.
- Usa protótipo de uma classe como assinatura.
- Compara atributos de uma instância com os protótipos → o protótipo mais próximo (considerando a distância Euclideana) indica a classe.
- Raramente mostra empate, requer parâmetro adicional para rejeição.

Classificação: Mínima Distância Euclideana





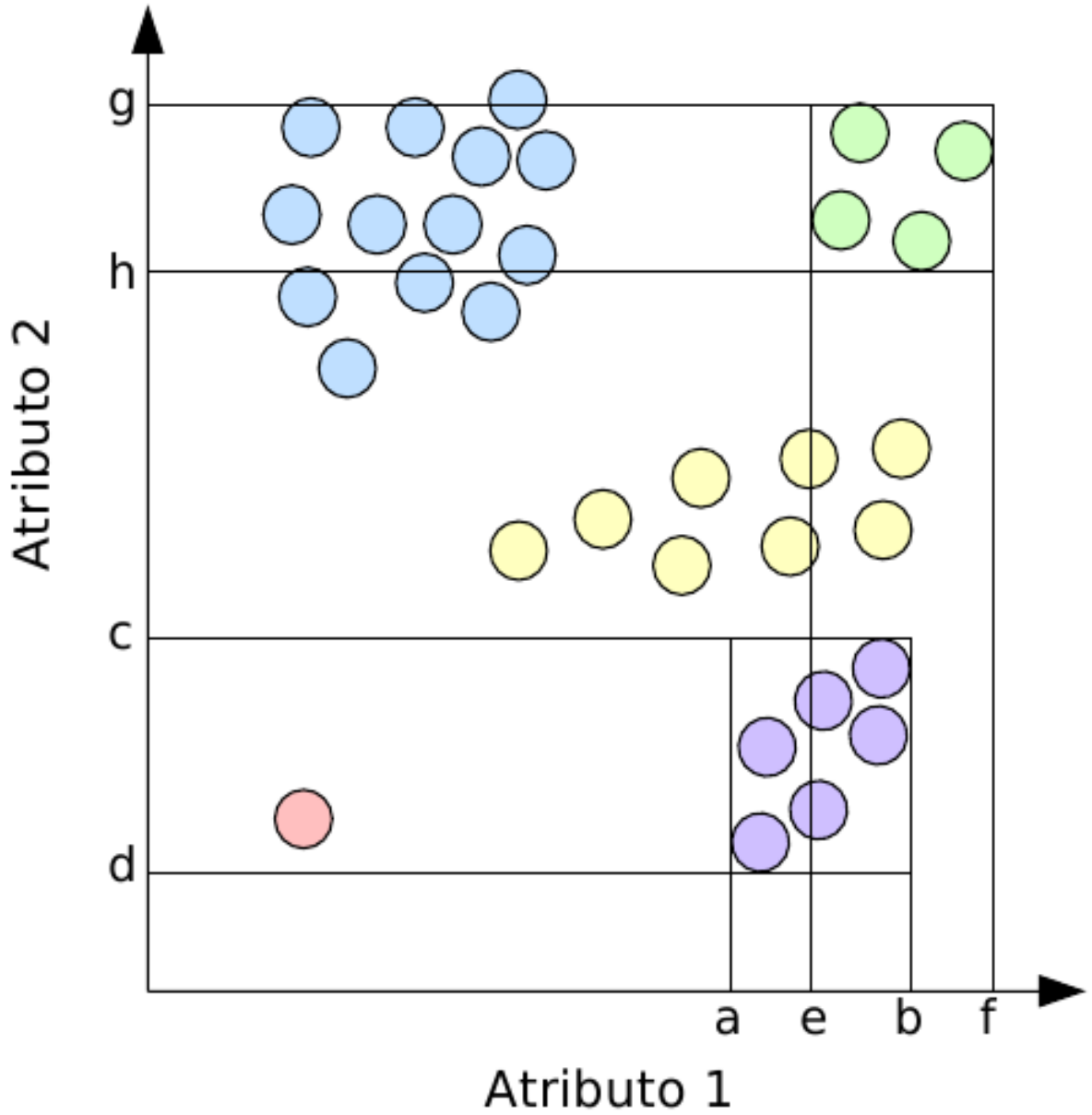
Classificação: Mínima Distância Euclideana

- Vantagens:
  - Simples de implementar.
  - Modelo de simples interpretação.
- Problemas:
  - Distribuição das classes nem sempre (quase nunca?) é hiperesférica.
  - Somente para atributos numéricos.
- Soluções:
  - Modelagem aproximada com mais de um protótipo.
  - Medidas de distância para outros tipos de atributos podem ser usadas...
    - ...mas como criar estas medidas?

Classificação: Hiperparalelepípedo regular

- Método do hiperparalelepípedo regular:
- Usa limiares ou extremos de cada classe como assinaturas.
- Classificação simples, permite rejeição.
- Atributos nominais e ordinais podem ser usados diretamente.
- Interoperabilidade com sistemas especialistas e árvores de decisão.
- **Pode haver empate!**

Classificação: Hiperparalelepípedo regular

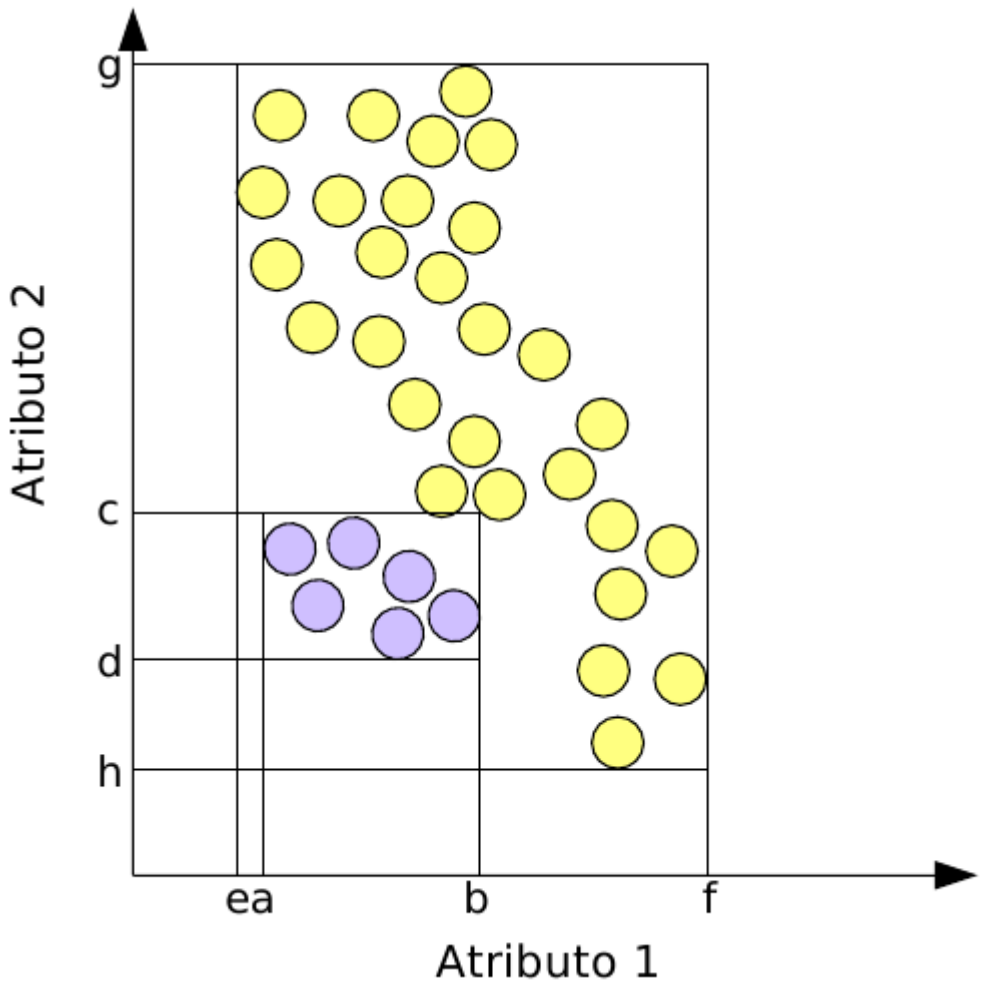


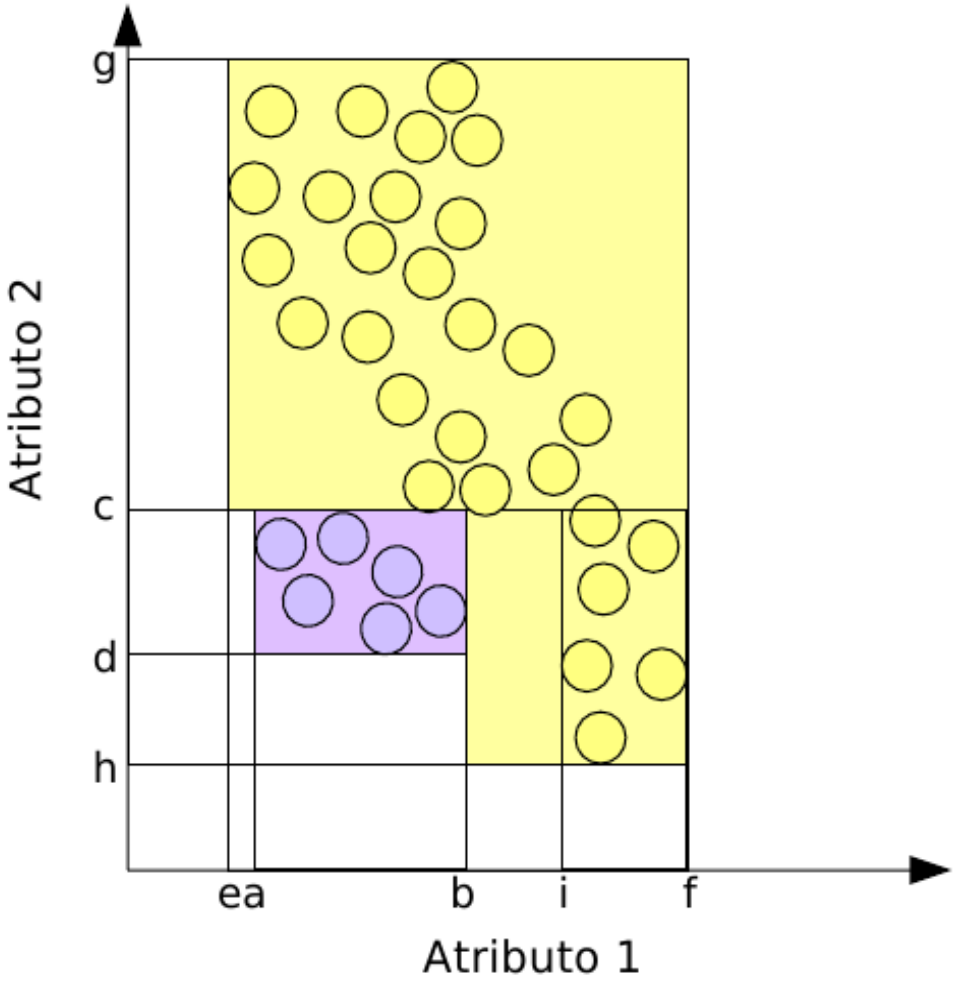
Se atributo 1 está entre *a* e *b* **e** atributo 2 está entre *c* e *d* **então** classe é lilás.

Se atributo 1 está entre *e* e *f* **e** atributo 2 está entre *g* e *h* **então** classe é verde.

Classificação: Hiperparalelepípedo regular

- Empate

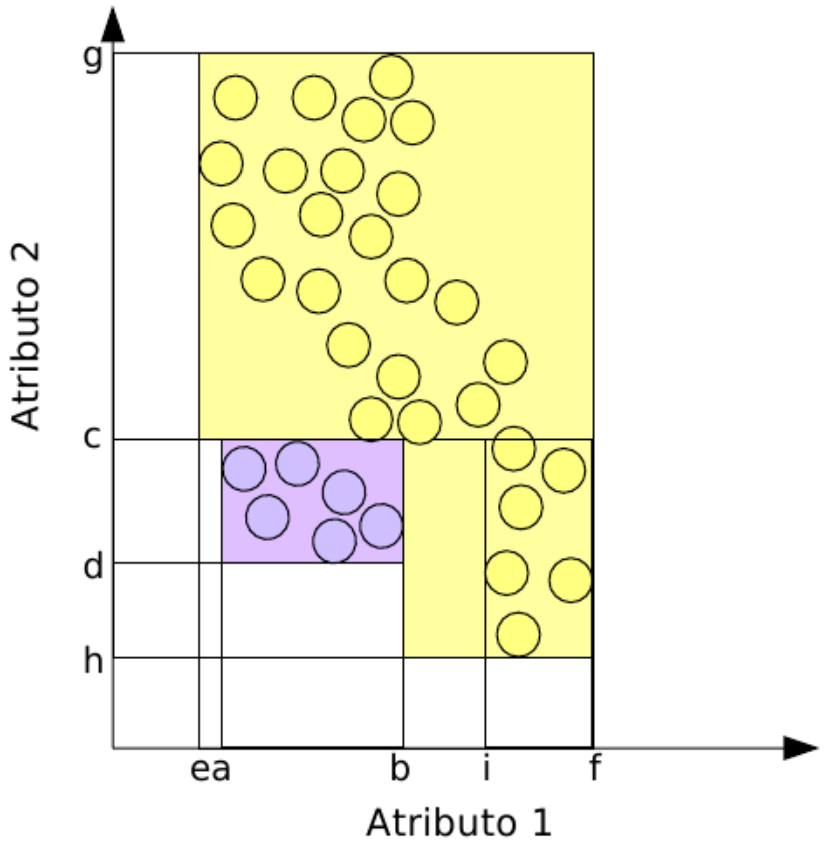


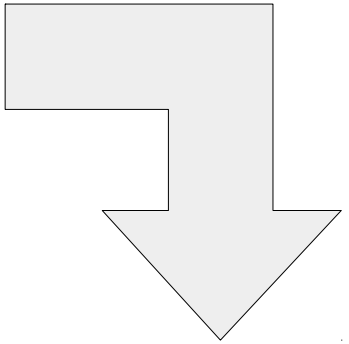


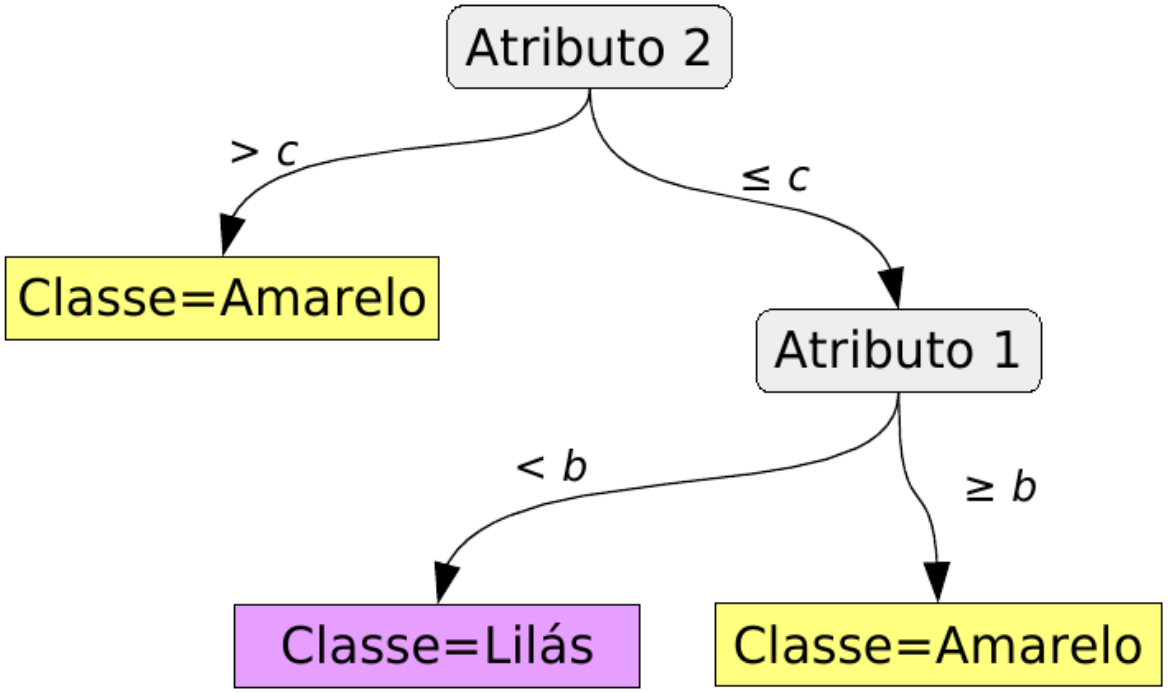
Classificação: Hiperparalelepípedo regular

- Vantagens:
  - Simples de implementar.
  - Modelo de simples interpretação.
- Problemas:
  - Influenciável por casos extremos.
  - Cortes ortogonais nos valores dos atributos.
- Soluções:
  - Filtragem de casos extremos é simples.
  - Cortes não-ortogonais possíveis (PCA, SOM, etc.)...
    - ...mas interpretação complexa!

Classificação: Árvores de decisão







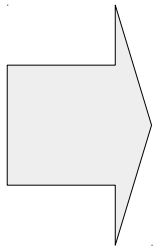
Classificação: Árvores de decisão

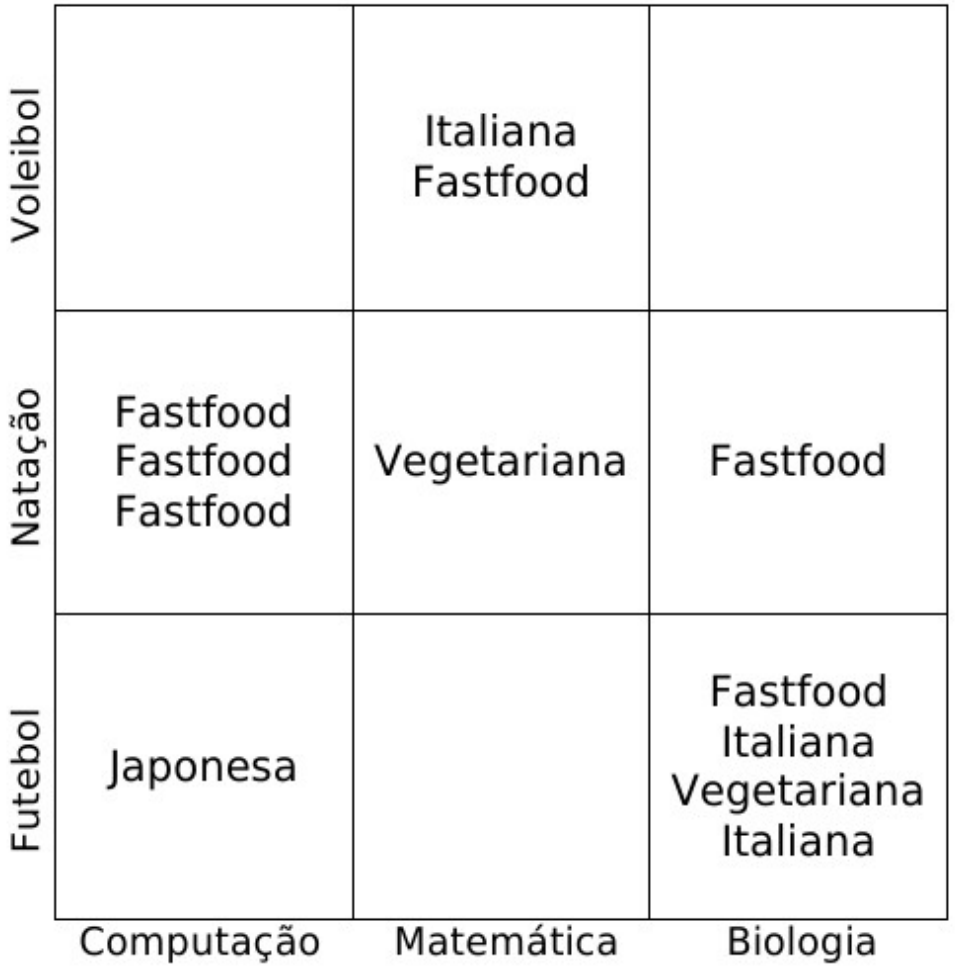
- Para determinar que nós serão criados temos que ter instâncias com classes definidas.
  - Devemos saber também qual é o atributo a ser usado como classe.
- São um conjunto de testes sobre uma base de dados que indica a classe a partir dos valores dos atributos de entrada.
  - Nós em uma árvore de decisão: testes sobre os atributos.
  - Folhas: determinação das classes.
- Muito utilizada por ser facilmente interpretável.
- Semelhança com sistemas especialistas.

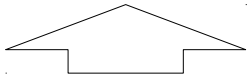
Classificação: Árvores de decisão

- Criação:
  - Exemplo usando força bruta.

Curso	Esporte	Tipo de comida
computação	futebol	japonesa
computação	natação	fastfood
computação	natação	fastfood
computação	natação	fastfood
matemática	voleibol	italiana
matemática	natação	vegetariana
matemática	voleibol	fastfood
biologia	futebol	fastfood
biologia	futebol	italiana
biologia	futebol	vegetariana
biologia	futebol	italiana
biologia	natação	fastfood







Consequente

Classificação: Árvores de decisão

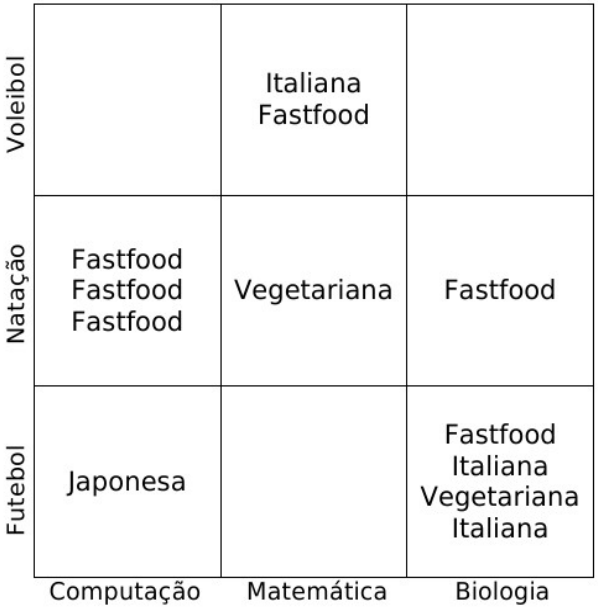
- Criação com força bruta:
  - Tabela de Decisão com cada combinação e consequentes.
  - 3 cursos, 3 esportes: 9 células.
  - 8 cursos, 6 esportes, 4 preferências musicais, 3 preferências por filmes: 576 células em quatro dimensões.
  - Cada célula contém mistura de consequentes: como generalizar?

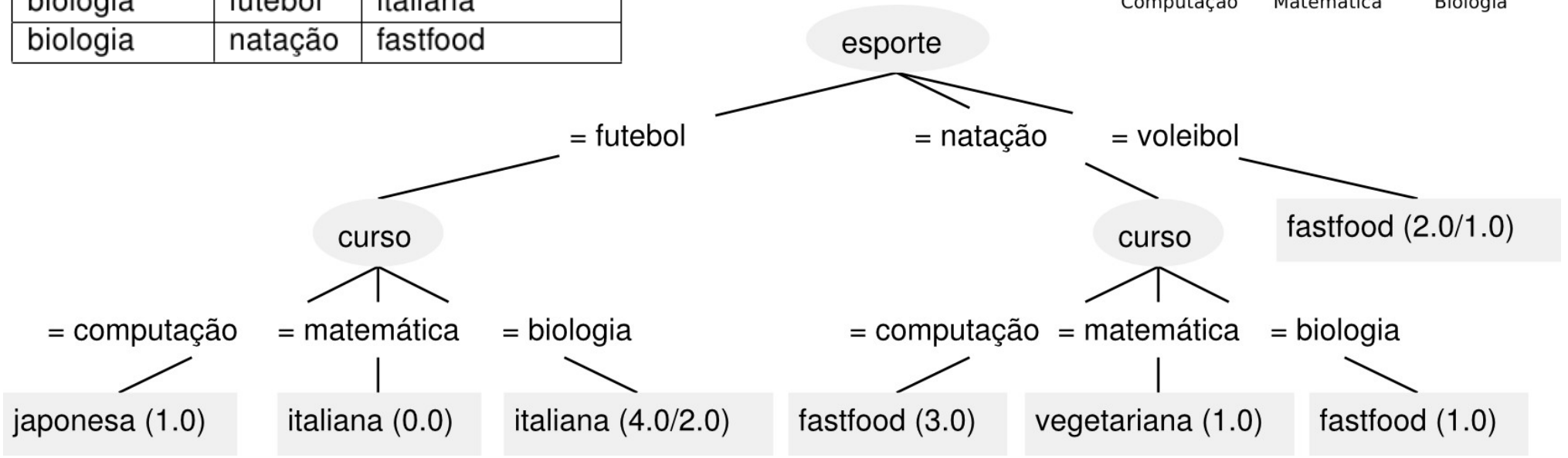
Classificação: Árvores de decisão

- Criação mais inteligente:
- Tenta minimizar erros de classificação/agrupamento **e** número de nós e folhas através do *ganho de informação*.
  - Existe maior ganho de informação → correlação/dependência entre **esporte** e **comida** do que entre **curso** e **comida**.
  - Conseguiremos classificar com menos perda de informação → de forma mais compacta comida a partir de **esporte** do que comida a partir de **curso**.
- Implementada nos algoritmos ID3, C4.5 (J4.8 Weka), C5.0

Classificação: Árvores de decisão

Curso	Esporte	Tipo de comida
computação	futebol	japonesa
computação	natação	fastfood
computação	natação	fastfood
computação	natação	fastfood
matemática	voleibol	italiana
matemática	natação	vegetariana
matemática	voleibol	fastfood
biologia	futebol	fastfood
biologia	futebol	italiana
biologia	futebol	vegetariana
biologia	futebol	italiana
biologia	natação	fastfood

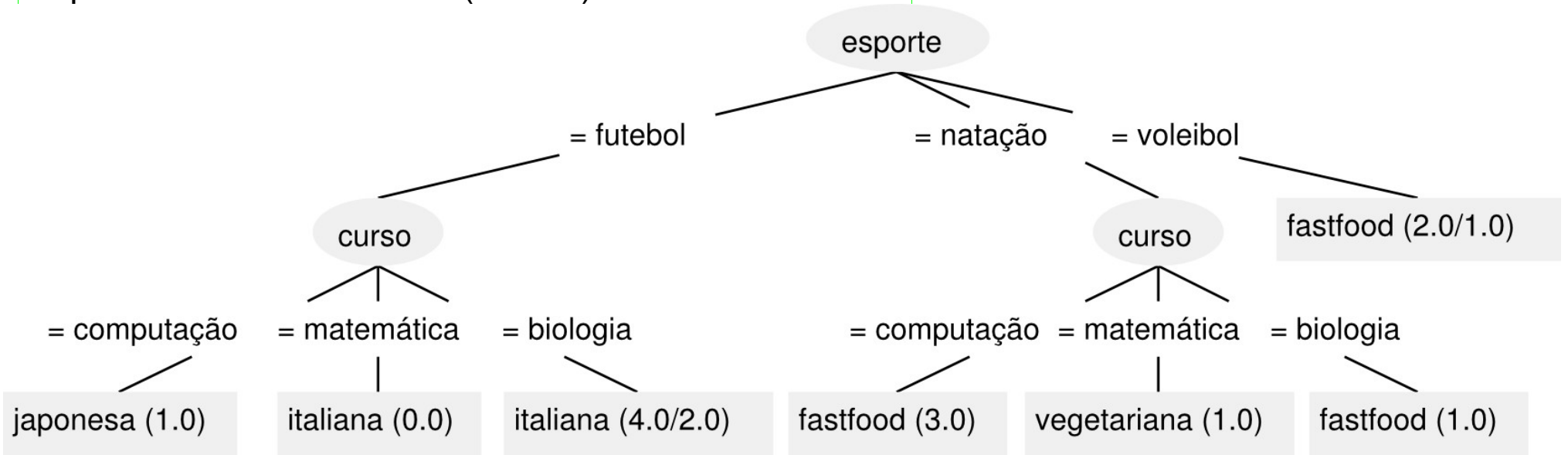




Classificação: Árvores de decisão

```
esporte = futebol
|   curso = computação: japonesa (1.0)
|   curso = matemática: italiana (0.0)
|   curso = biologia: italiana (4.0/2.0)
esporte = natação
|   curso = computação: fastfood (3.0)
|   curso = matemática: vegetariana (1.0)
|   curso = biologia: fastfood (1.0)
esporte = voleibol: fastfood (2.0/1.0)
```

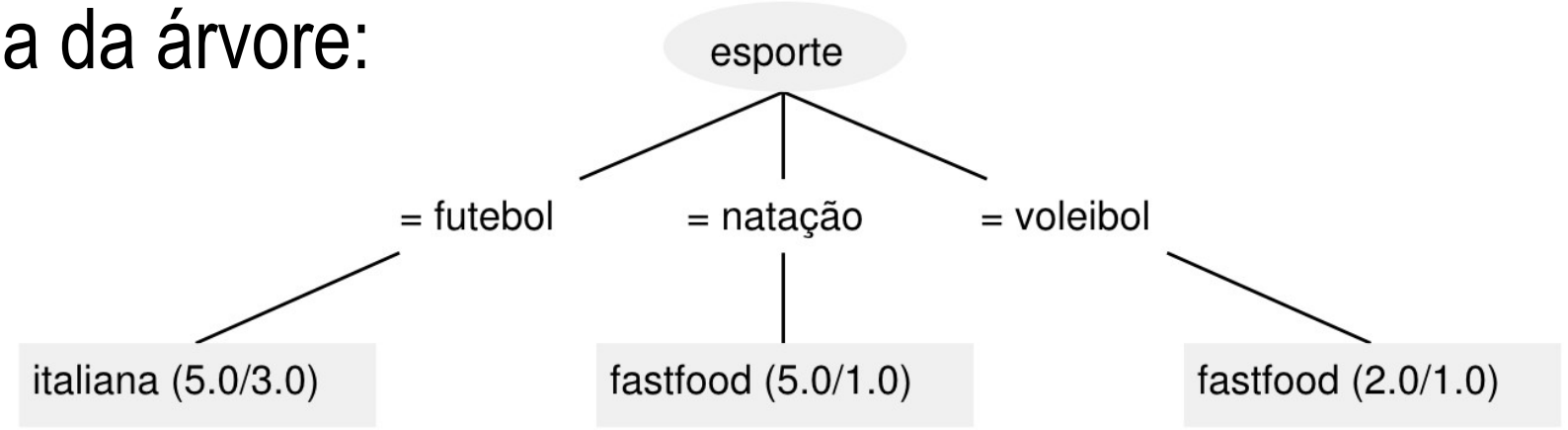
```
=== Confusion Matrix ===
 a b c d   <-- classified as
 1 0 0 0 | a = japonesa
 0 5 1 0 | b = fastfood
 0 1 2 0 | c = italiana
 0 0 1 1 | d = vegetariana
```



weka.classifiers.trees.J48 -U -M 1

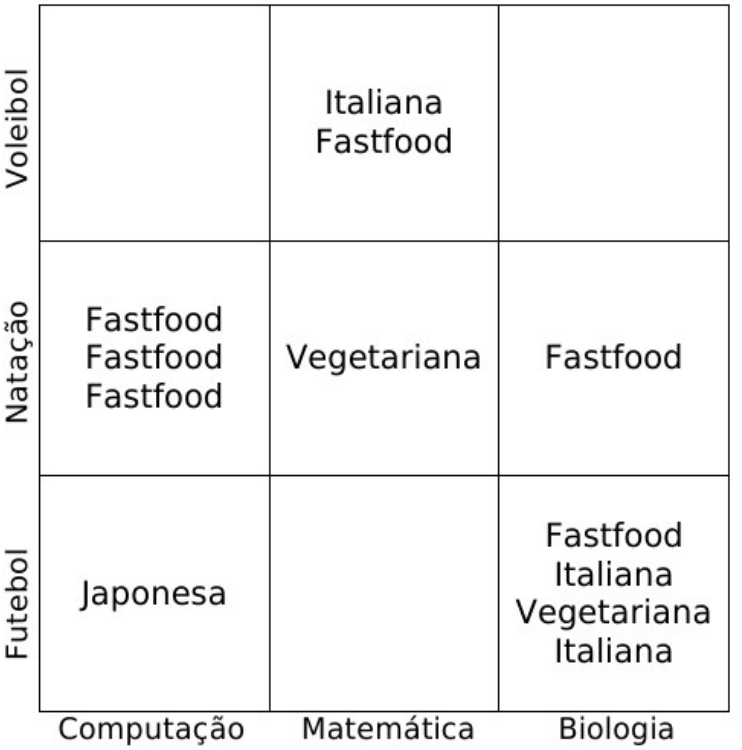
Classificação: Árvores de decisão

- Poda da árvore:




```
esporte = futebol: italiana (5.0/3.0)
esporte = natação: fastfood (5.0/1.0)
esporte = voleibol: fastfood (2.0/1.0)
```

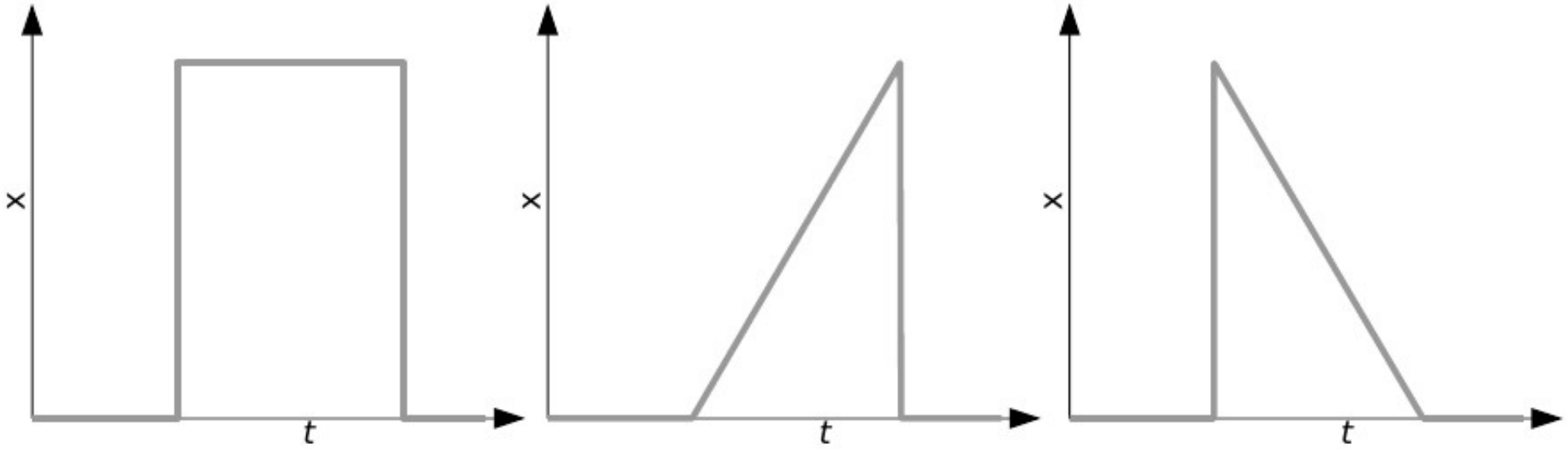
```
=== Confusion Matrix ===
 a b c d   <-- classified as
 0 0 1 0 | a = japonesa
 0 5 1 0 | b = fastfood
 0 1 2 0 | c = italiana
 0 1 1 0 | d = vegetariana
```

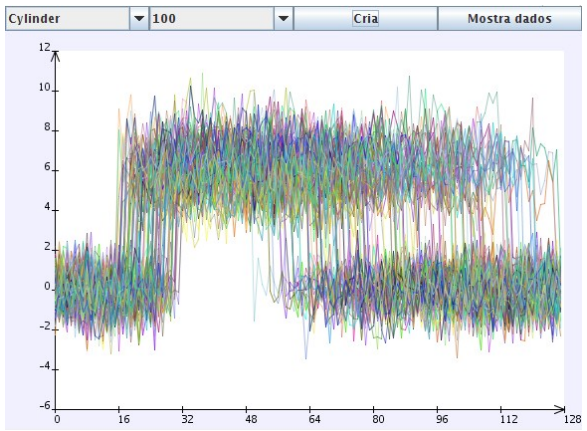


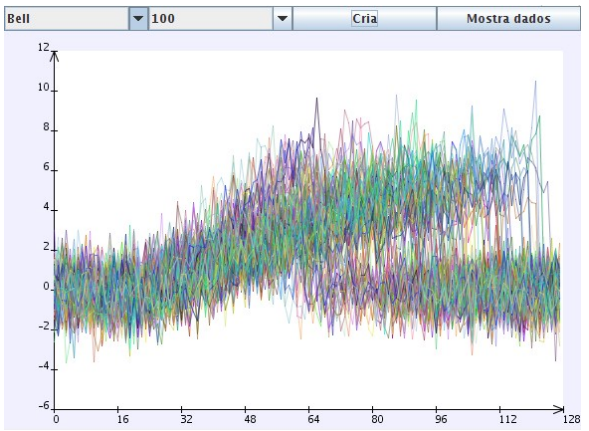
weka.classifiers.trees.J48 -C 0.7 -M 2

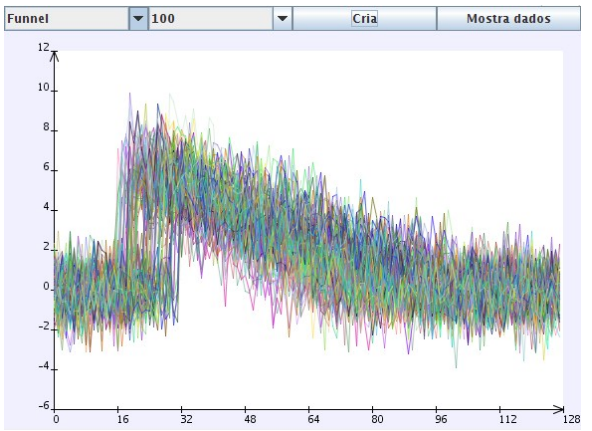
Classificação: Árvores de decisão

- Outro exemplo: *Cylinder, Bell, Funnel*









Classificação: Árvores de decisão

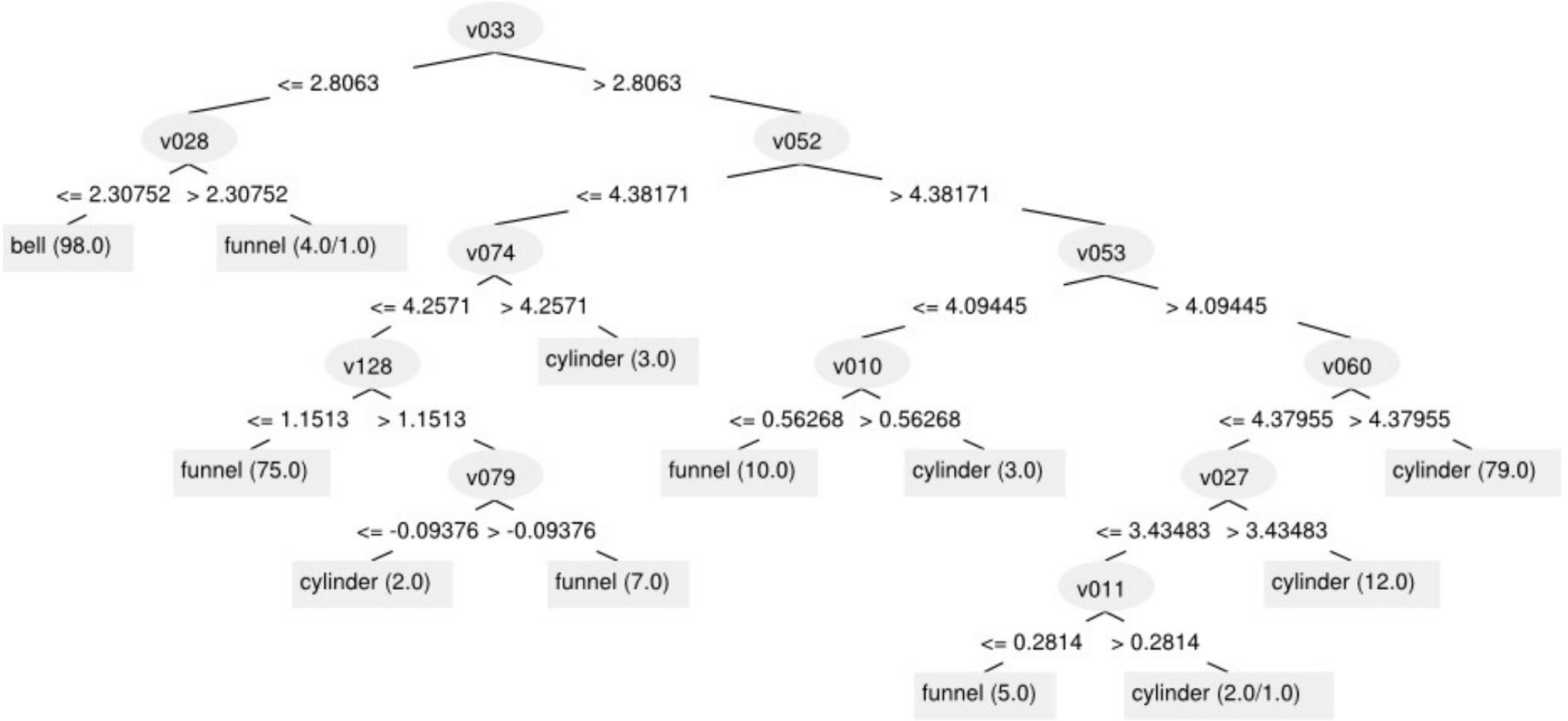
- Outro exemplo: *Cylinder, Bell, Funnel*

```
v033 <= 2.8063
|   v028 <= 2.30752: bell (98.0)
|   v028 > 2.30752: funnel (4.0/1.0)
v033 > 2.8063
|   v052 <= 4.38171
|   |   v074 <= 4.2571
|   |   |   v128 <= 1.1513: funnel (75.0)
|   |   |   v128 > 1.1513
|   |   |   |   v079 <= -0.09376: cylinder (2.0)
|   |   |   |   v079 > -0.09376: funnel (7.0)
|   |   v074 > 4.2571: cylinder (3.0)
|   v052 > 4.38171
|   |   v053 <= 4.09445
|   |   |   v010 <= 0.56268: funnel (10.0)
|   |   |   v010 > 0.56268: cylinder (3.0)
|   |   v053 > 4.09445
|   |   |   v060 <= 4.37955
|   |   |   |   v027 <= 3.43483
|   |   |   |   |   v011 <= 0.2814: funnel (5.0)
|   |   |   |   |   v011 > 0.2814: cylinder (2.0/1.0)
|   |   |   |   v027 > 3.43483: cylinder (12.0)
|   |   |   v060 > 4.37955: cylinder (79.0)
```

```
=== Confusion Matrix ===
  a  b  c   <-- classified as
 87  1 12 |  a = cylinder
  1 97  2 |  b = bell
 14  2 84 |  c = funnel
```

Classificação: Árvores de decisão

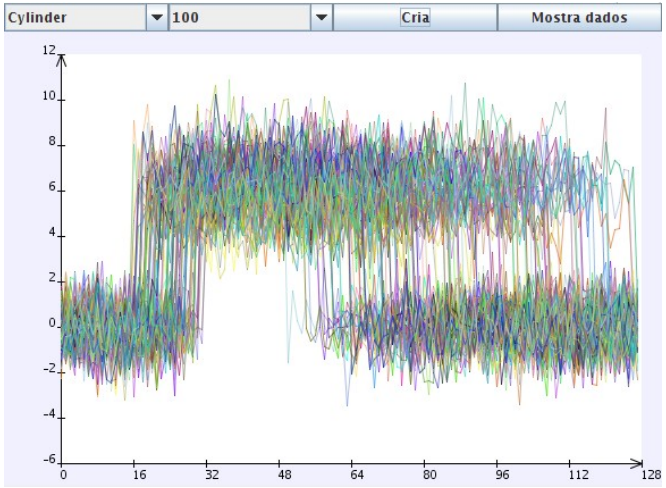
- Outro exemplo: *Cylinder, Bell, Funnel*

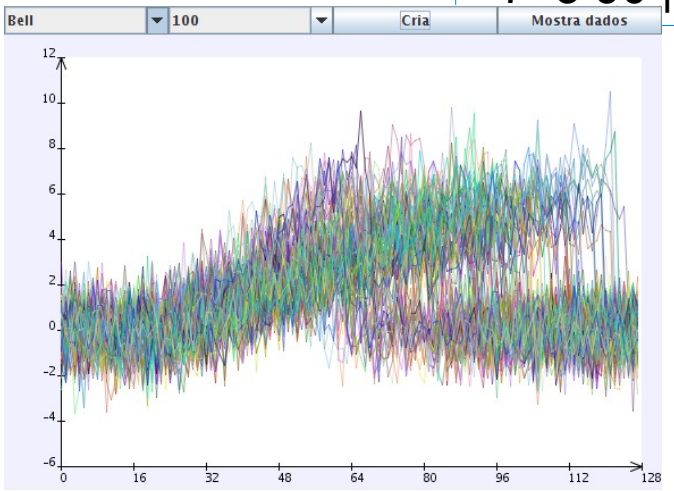


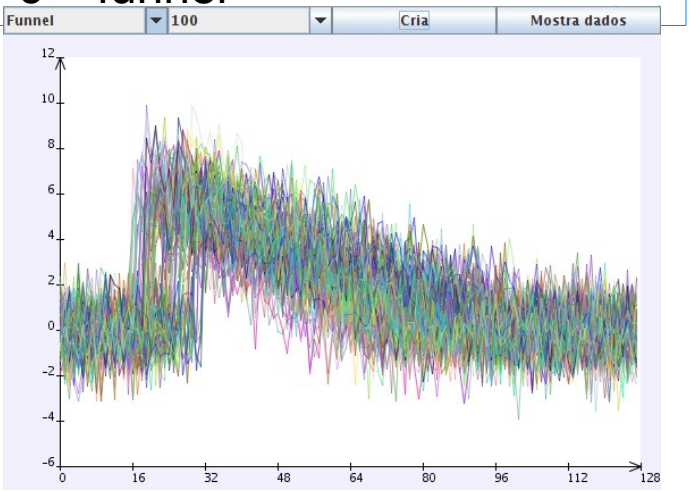
Classificação: Árvores de decisão

```
v033 <= 2.8063: bell (102.0/3.0)
v033 > 2.8063
|   v052 <= 4.38171: funnel (87.0/5.0)
|   v052 > 4.38171
|   |   v053 <= 4.09445: funnel (13.0/3.0)
|   |   v053 > 4.09445: cylinder (98.0/6.0)
```

```
=== Confusion Matrix ===
  a  b  c   <-- classified as
 86  0 14 |  a = cylinder
  0 98  2 |  b = bell
  7  3 90 |  c = funnel
```

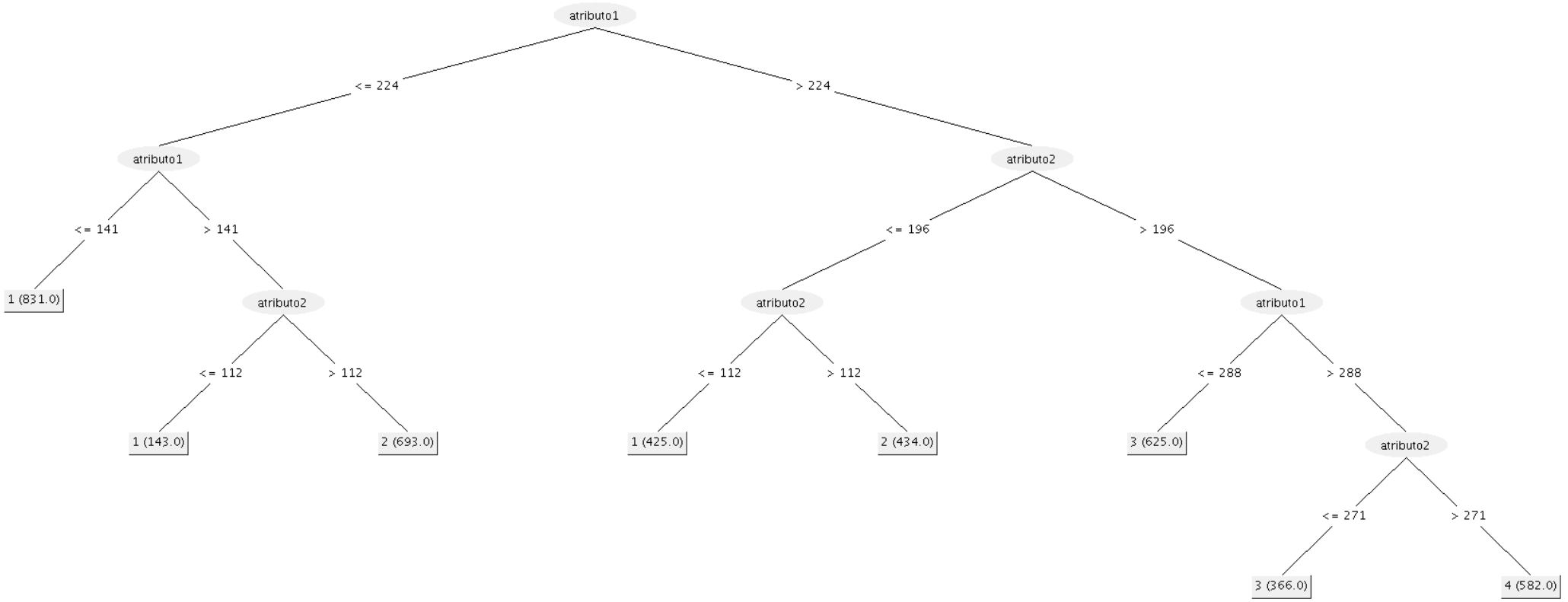




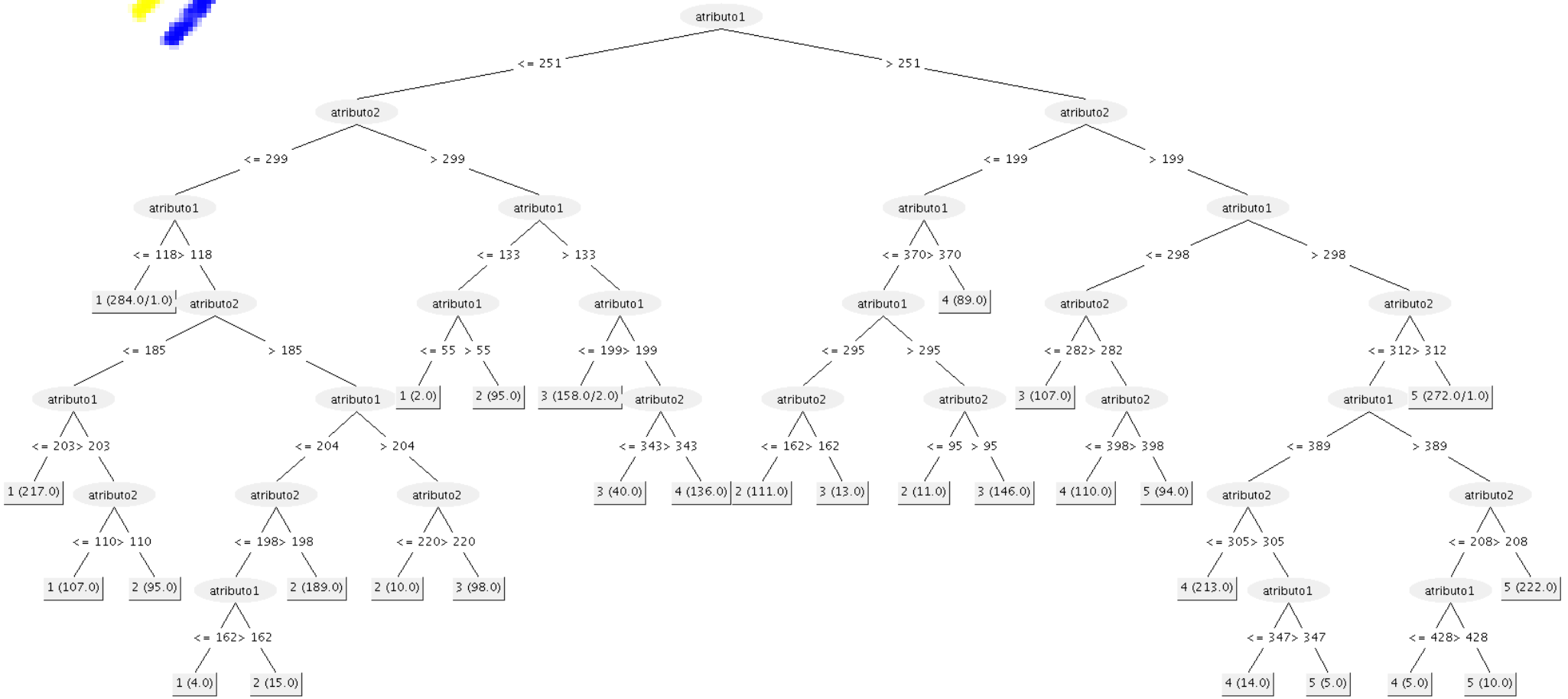


Classificação: Árvores de decisão

Classificação: Árvores de decisão



Classificação: Árvores de decisão



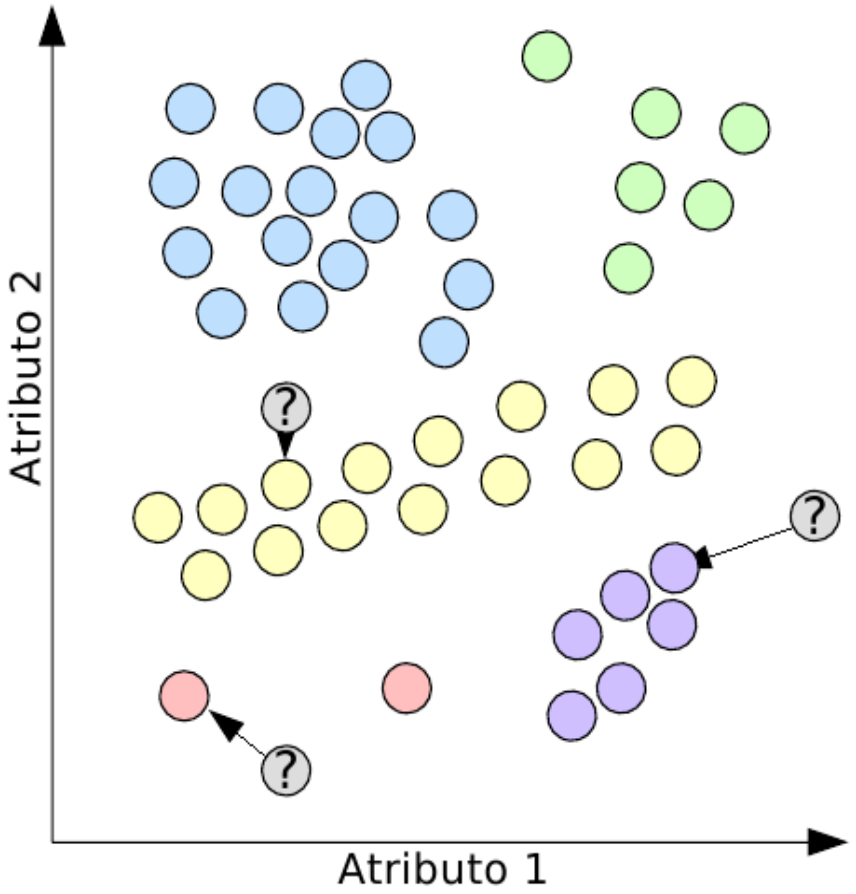
Classificação: Árvores de decisão

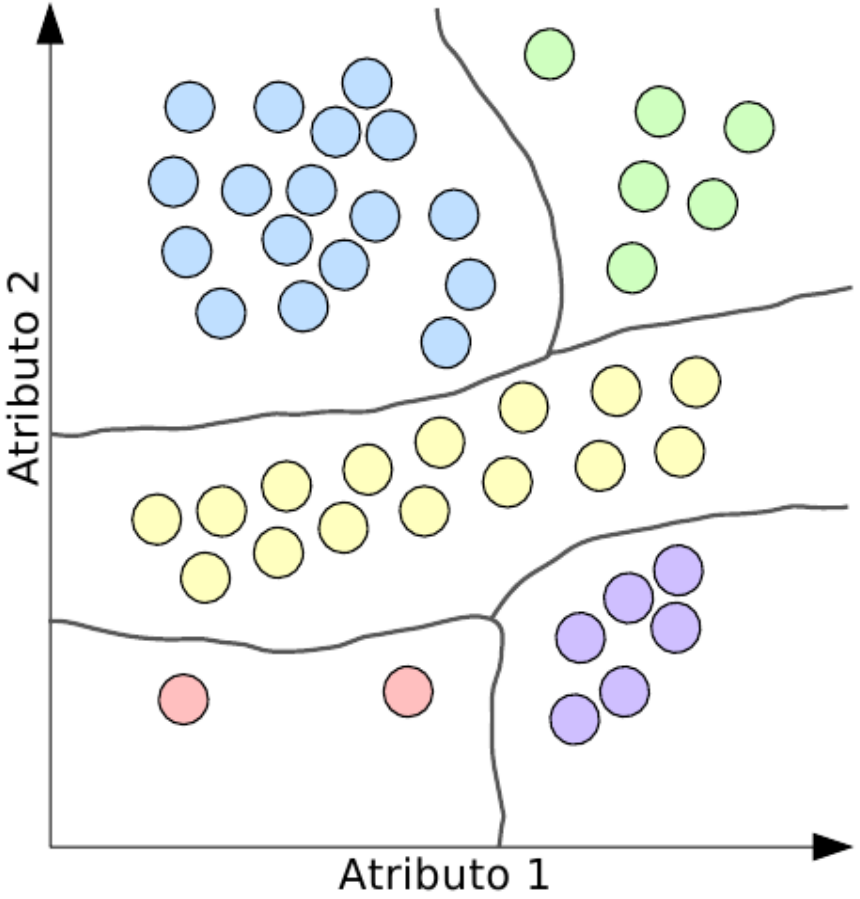
- Vantagens:
  - Modelo de simples interpretação.
  - Possível variar precisão x concisão.
- Problemas:
  - Estrutura (*ordem*) da árvore depende dos dados.
  - Em muitos casos a árvore pode ser extensa!
- Soluções:
  - Análise da árvore é passo de mineração de dados (avaliação do modelo).
    - Como/quando/onde podar?

Classificação: Vizinhos mais Próximos

- Bastante intuitivo: se uma instância de classe desconhecida estiver bem próxima de uma de classe conhecida, as classes devem ser as mesmas.
  - Proximidade **sempre** no espaço de atributos!
- Não criamos protótipos ou assinaturas: usamos as próprias instâncias.
- Cria hipersuperfícies de separação (conjunto de hiperplanos).

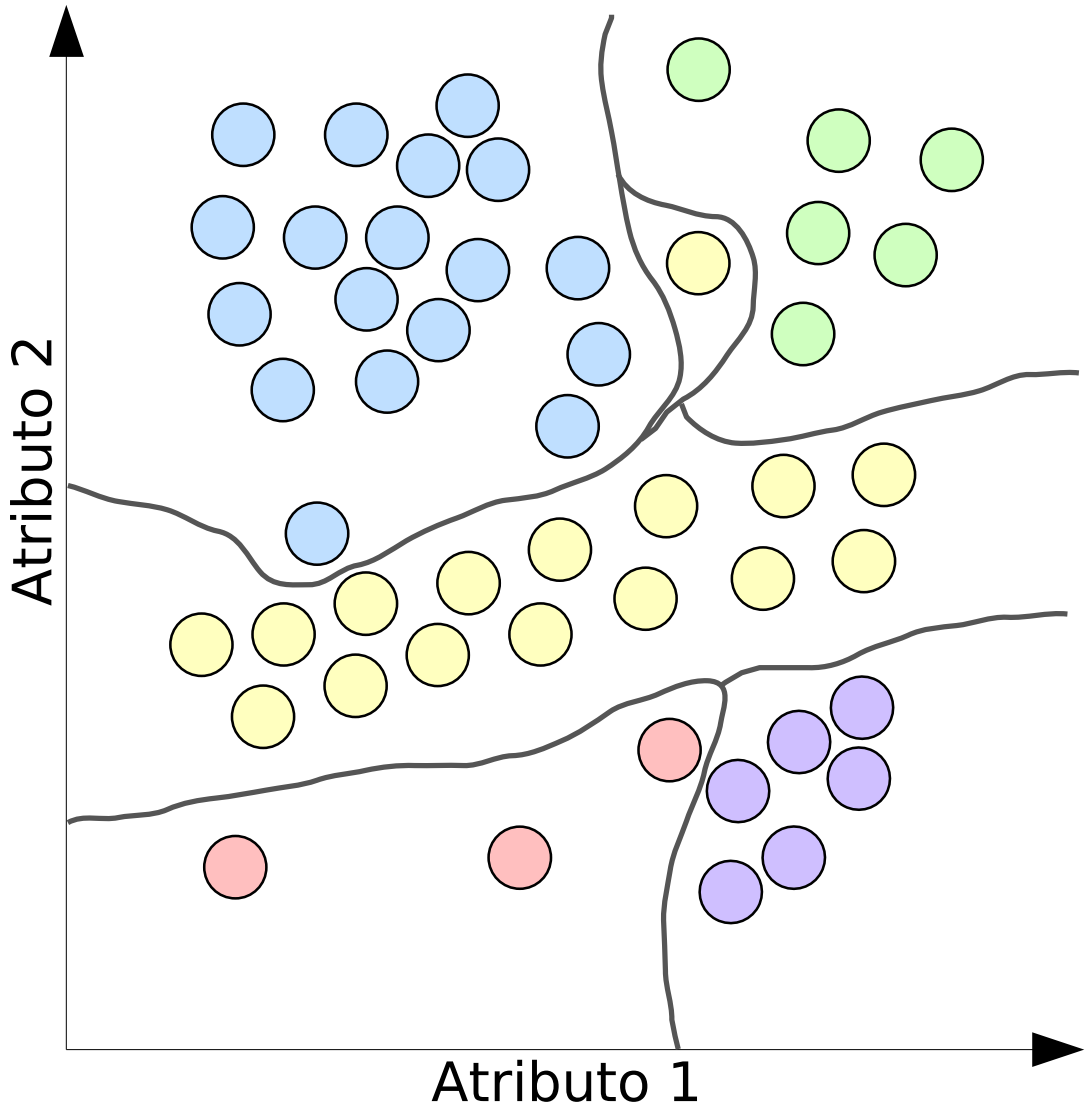
Classificação: Vizinhos mais Próximos





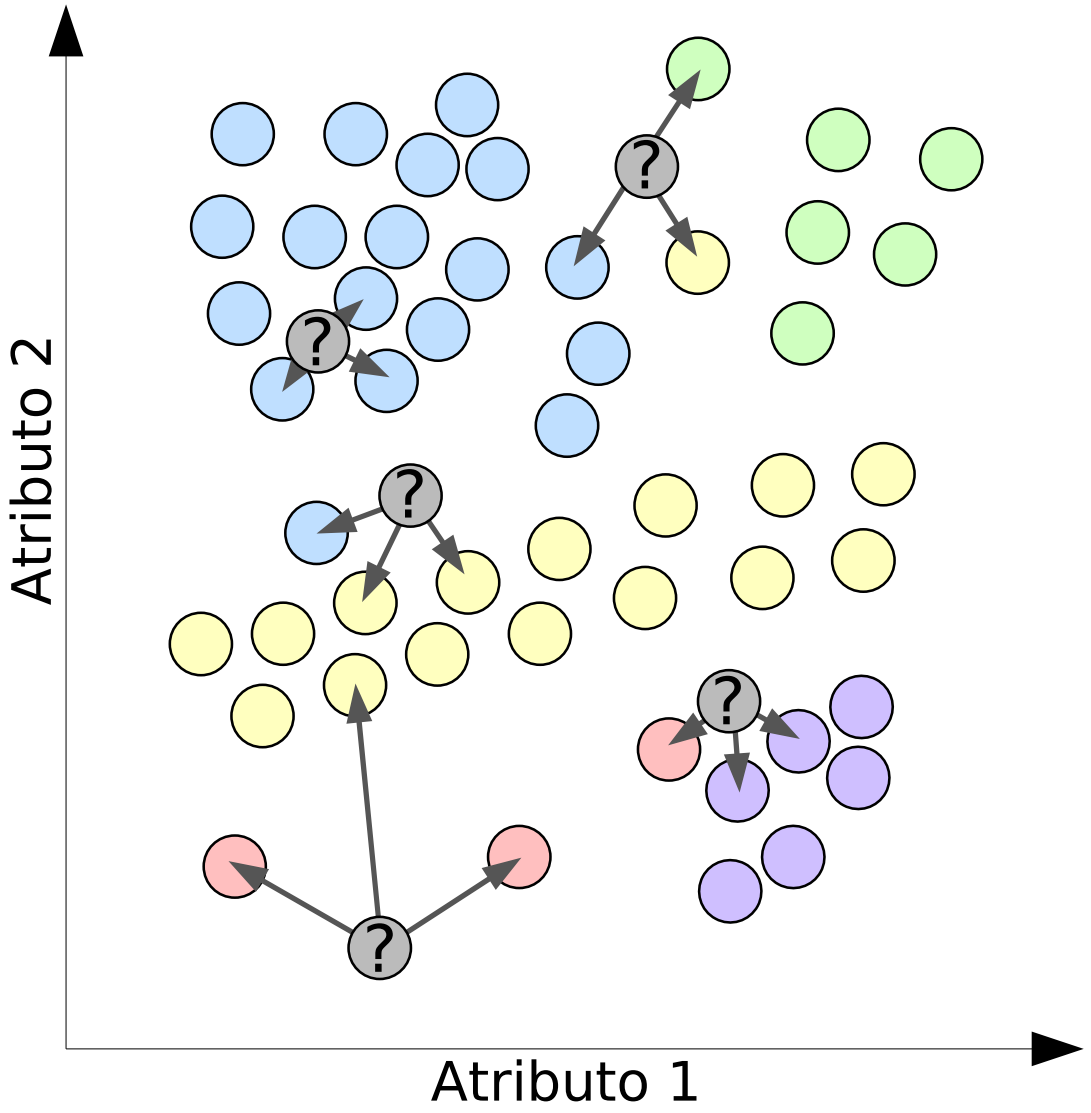
Classificação: Vizinhos mais Próximos

- Problema (?) potencial: *outliers*.

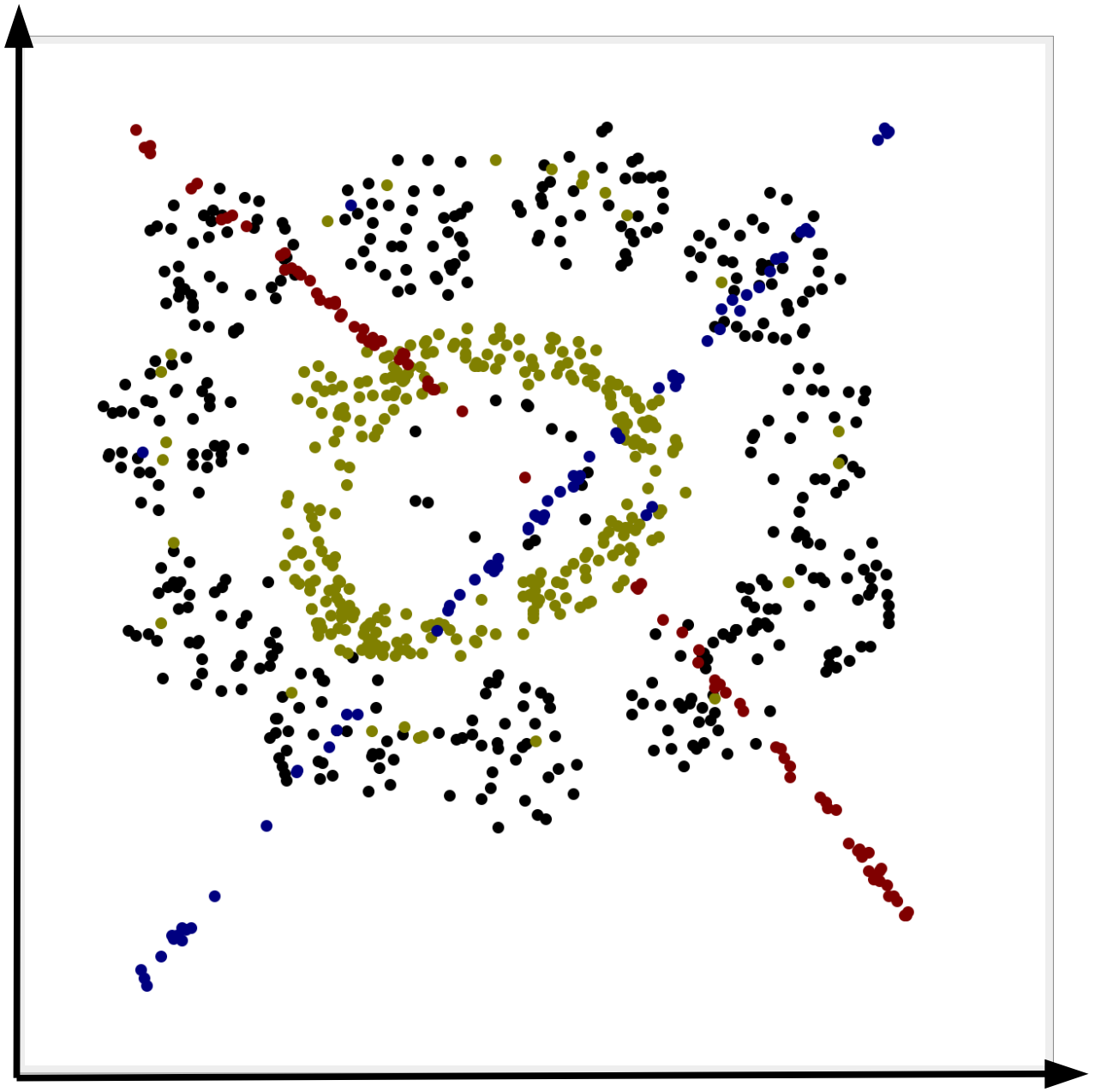


Classificação: Vizinhos mais Próximos

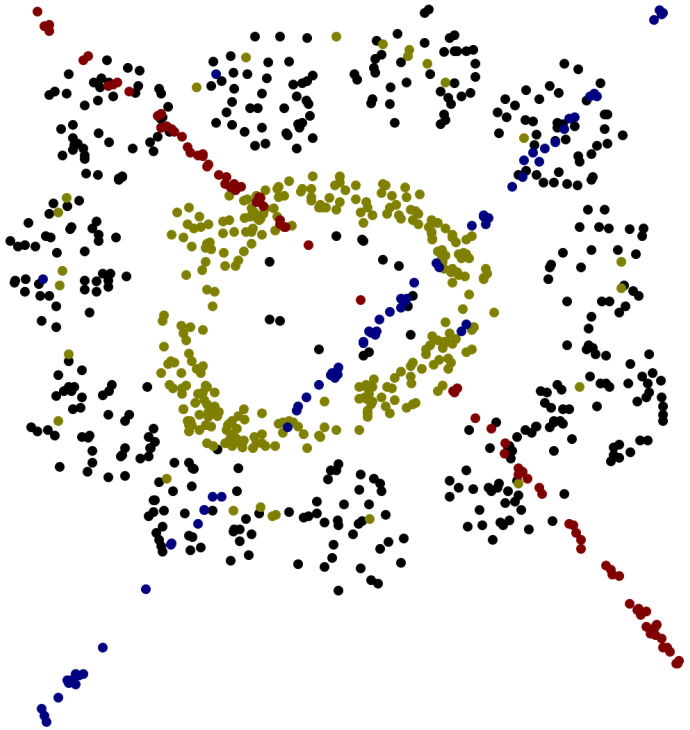
- Solução: usar ***K*** vizinhos mais próximos.

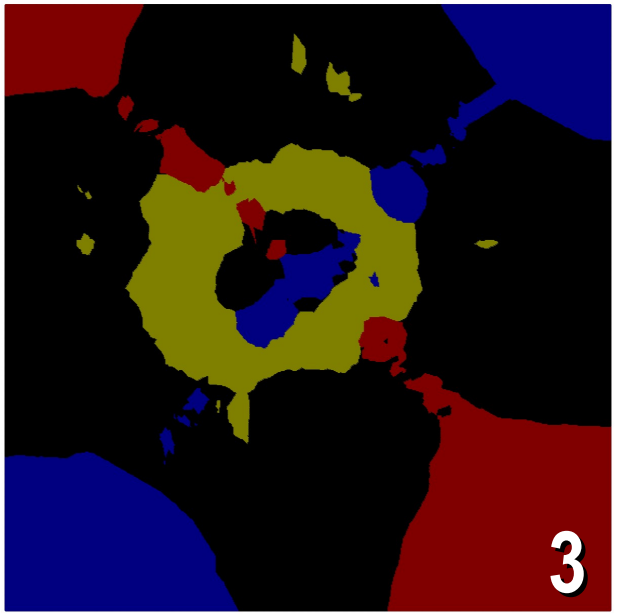


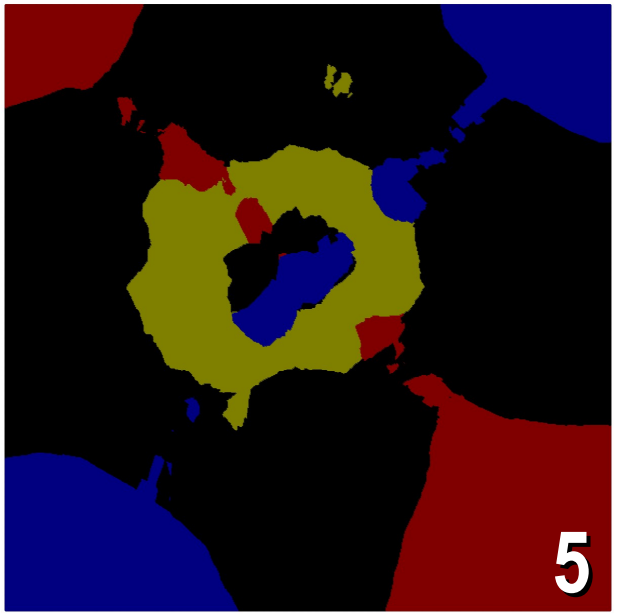
Classificação: Vizinhos mais Próximos

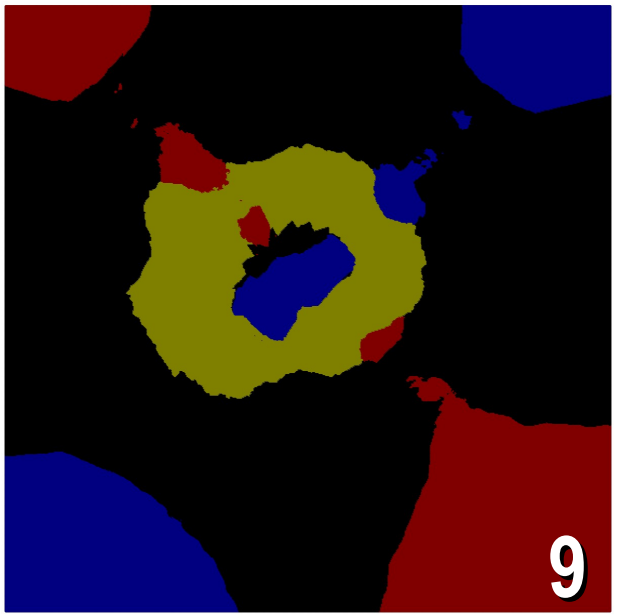


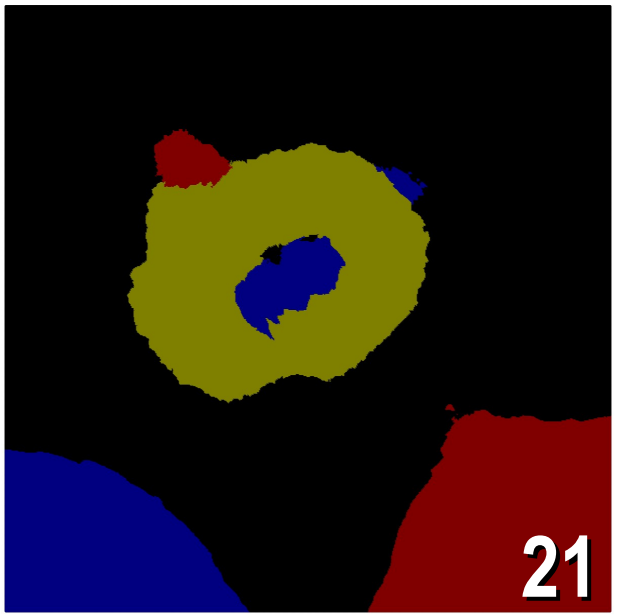
Classificação: Vizinhos mais Próximos









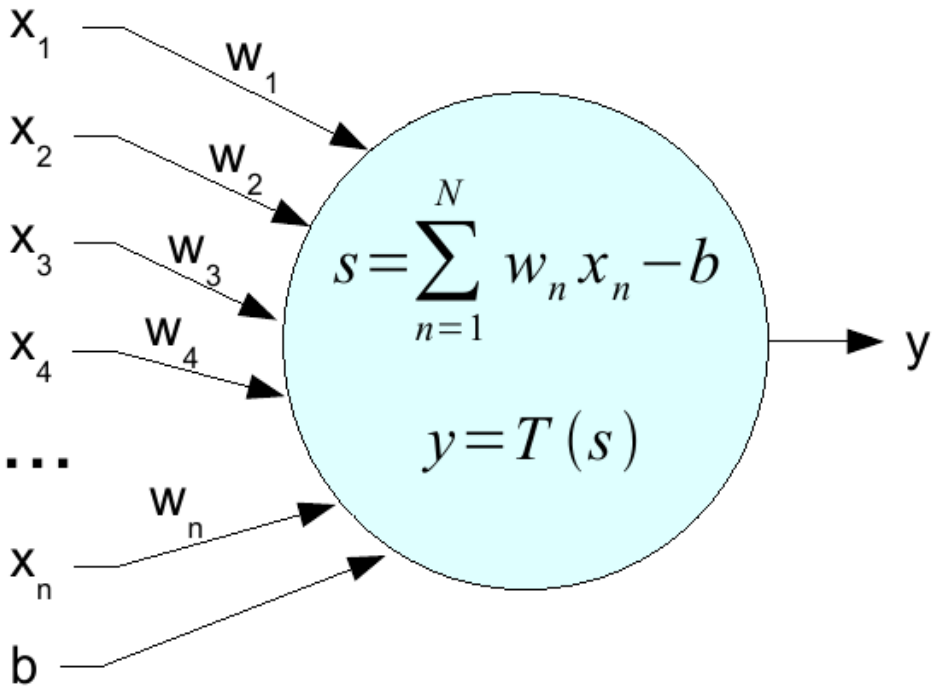


Classificação: Vizinhos mais Próximos

- Vantagens:
  - Dispensa fase de treinamento.
  - Aplicável para classes com qualquer tipo de distribuição (até disjuntas!)
- Problemas:
  - Difícil explicar/interpretar o “modelo”.
  - Complexidade computacional.
  - Influência de *outliers* e *clumps*.
- Soluções:
  - Algoritmos híbridos: redução do número de instâncias para comparação.

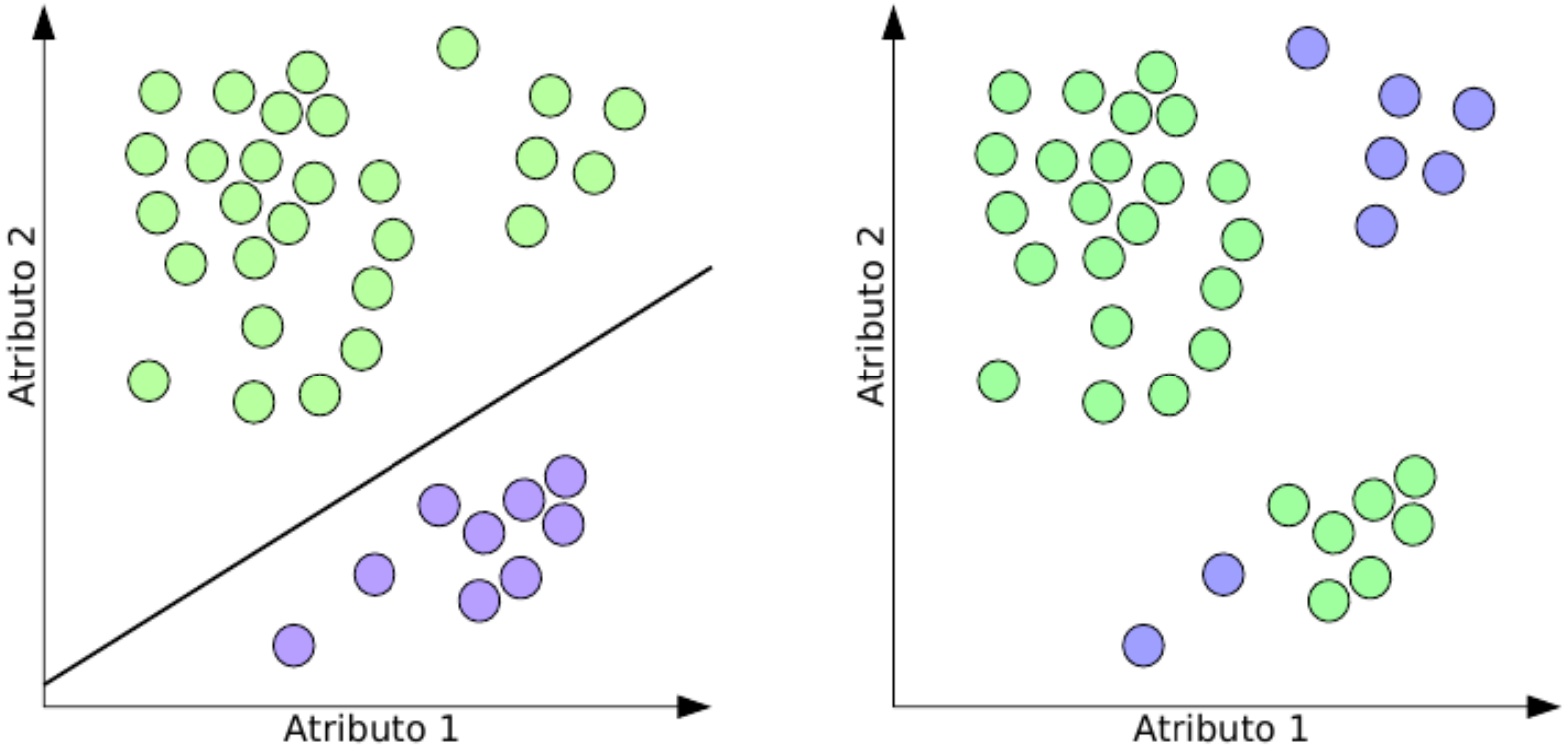
Classificação: Redes Neurais

- Este exemplo: Perceptrons em múltiplas camadas.
  - Existem outros modelos e variantes.
- Redes Neurais Artificiais (RNAs ou NNs): algoritmos baseados em simulações simplificadas de neurônios reais.
  - Neurônios processam valores de entrada e apresentam um de saída.
  - Vários neurônios artificiais conectados ⟶ rede neural.



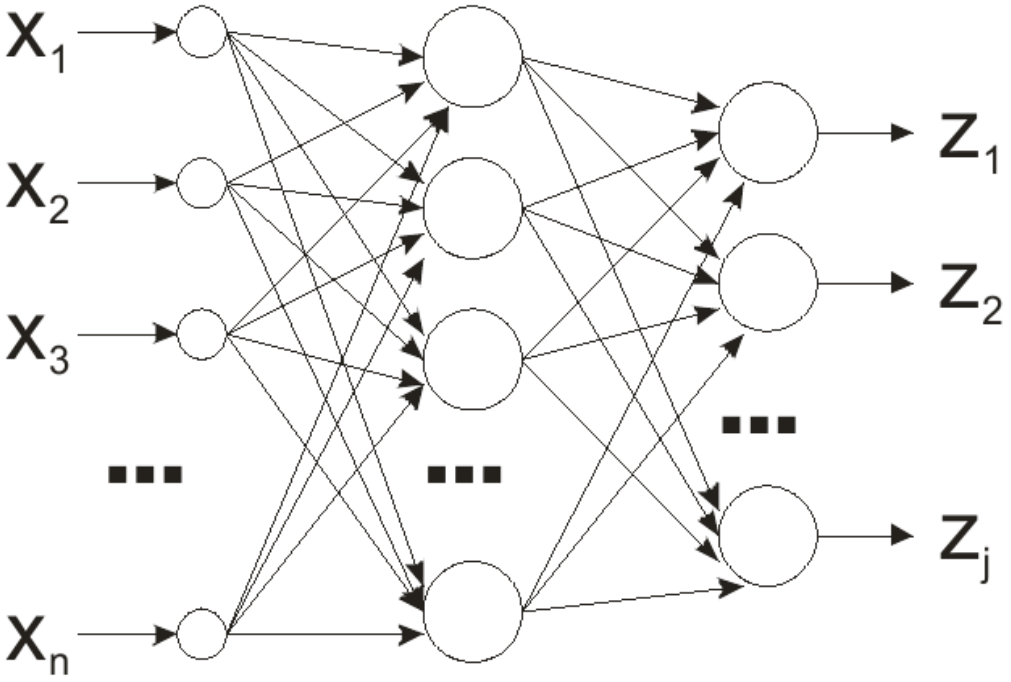
Classificação: Redes Neurais

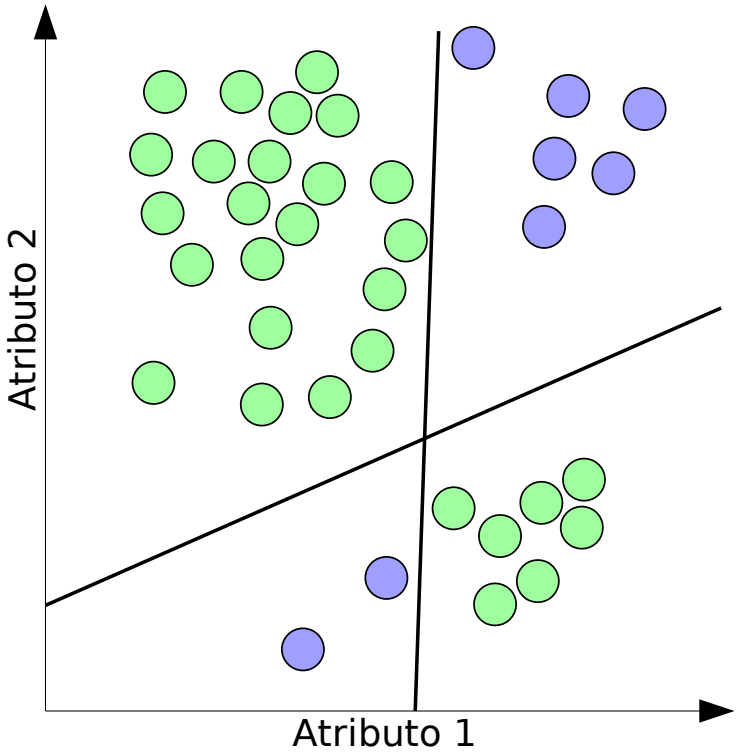
- Um perceptron corresponde a um hiperplano no espaço de atributos:
  - Separa duas classes linearmente separáveis,
  - Não separa mais que duas classes ou faz separações não-lineares.



Classificação: Redes Neurais

- Solução: perceptrons podem ser combinados em camadas.
  - Entrada: distribui valores para perceptrons na próxima camada.
  - Camada(s) escondida(s) (*hidden layer*): criam hiperplanos e combinações.
  - Saída: apresenta resultados.

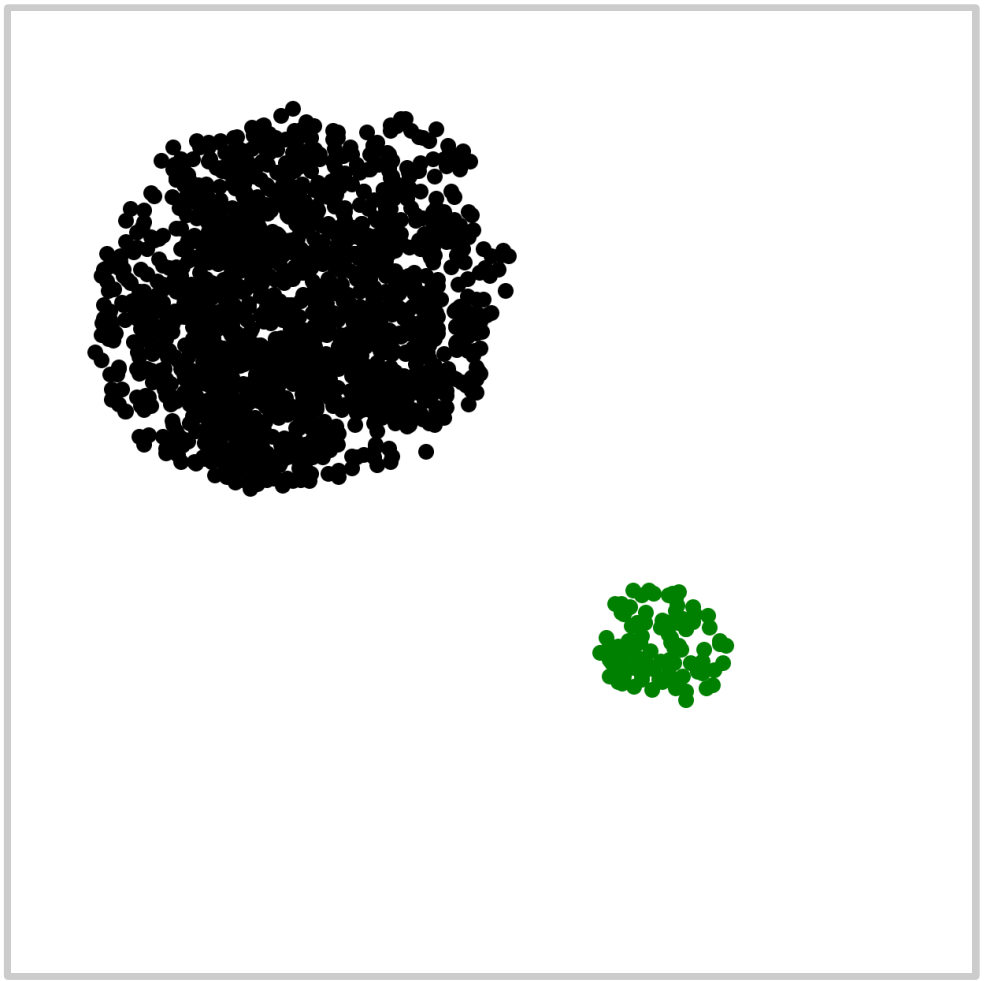




Classificação: Redes Neurais

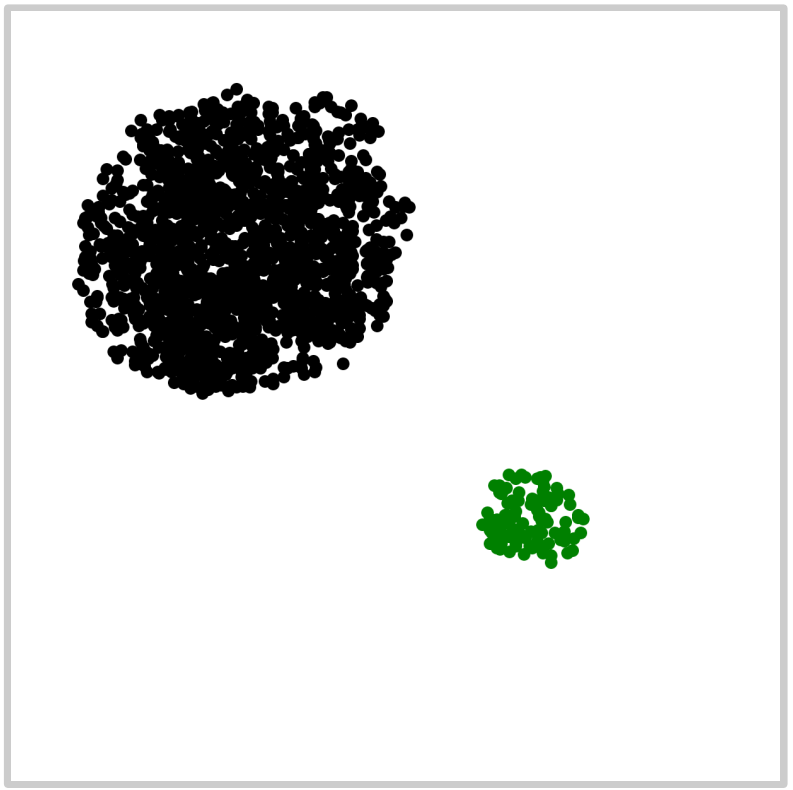
- Vantagens:
  - Capacidade de separar bem classes não linearmente separáveis (com múltiplas camadas).
- Problemas:
  - Difícil explicar/interpretar o “modelo” (caixa preta).
  - Treinamento pode ser complexo (computacionalmente caro), definição da arquitetura também.
- Soluções:
  - Múltiplas arquiteturas e avaliação da qualidade de classificação.

Classificação: Redes Neurais



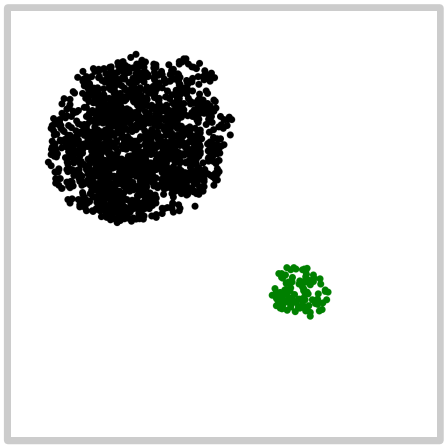
- Primeiro exemplo simples:
  - Duas classes que podem facilmente ser linearmente separadas.
  - Arquitetura 2x2x2:
    - 2 neurônios na camada de entrada (2 atributos).
    - 1 camada escondida com 2 neurônios.
    - 2 neurônios na camada de saída (2 classes).

Classificação: Redes Neurais

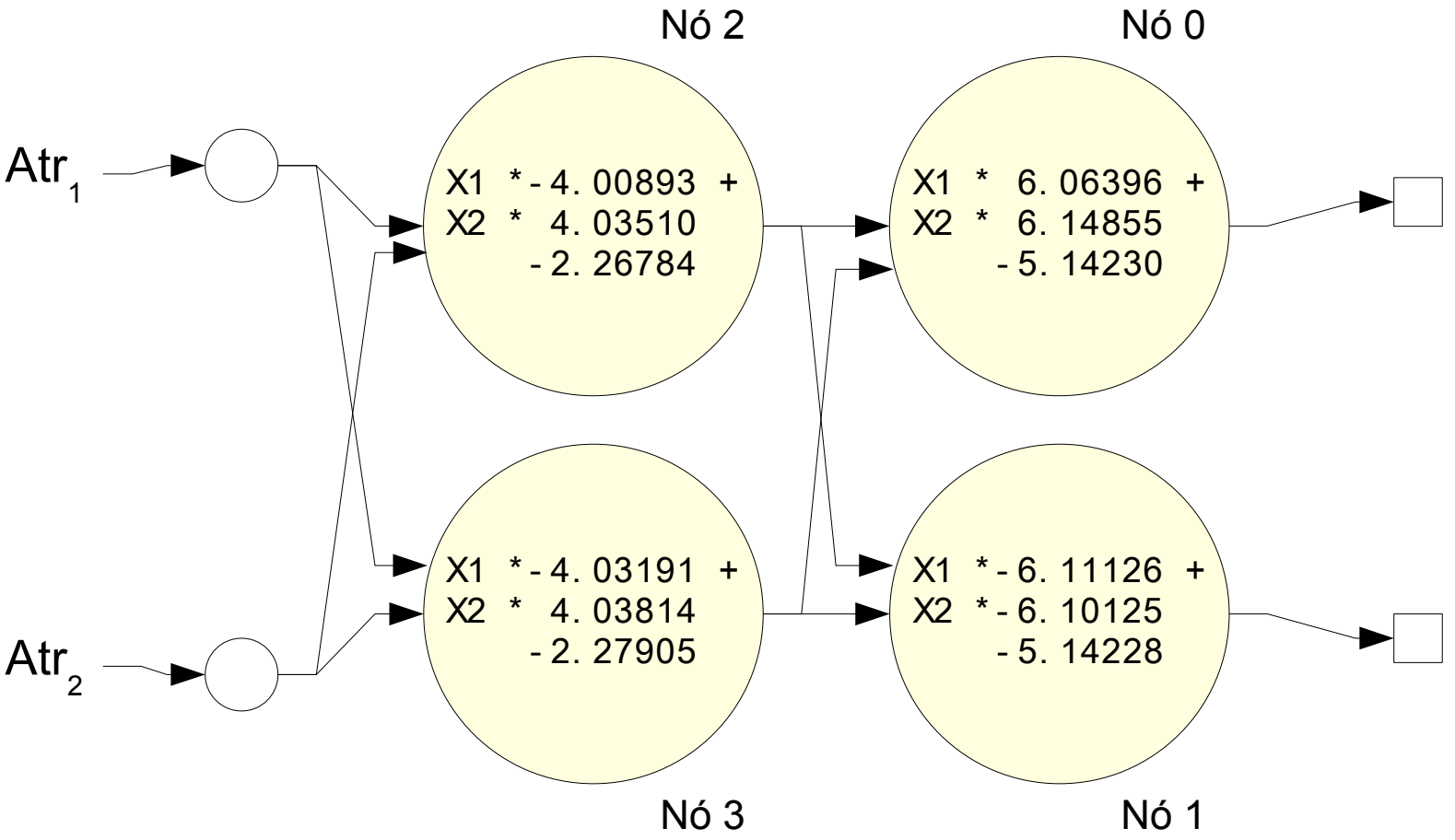


```
Sigmoid Node 0
   Inputs    Weights
   Threshold    -5.142297502584935
   Node 2    6.063964228629336
   Node 3    6.148552185386907
Sigmoid Node 1
   Inputs    Weights
   Threshold    5.1422827635337285
   Node 2    -6.111259386964719
   Node 3    -6.101248971633012
Sigmoid Node 2
   Inputs    Weights
   Threshold    2.267842125362997
   Attrib atributo1    -4.008925758147538
   Attrib atributo2    4.035102089969922
Sigmoid Node 3
   Inputs    Weights
   Threshold    2.2790468422497985
   Attrib atributo1    -4.031913175764998
   Attrib atributo2    4.038136743308941
Class 0
   Input
   Node 0
Class 8
   Input
   Node 1
```

Classificação: Redes Neurais

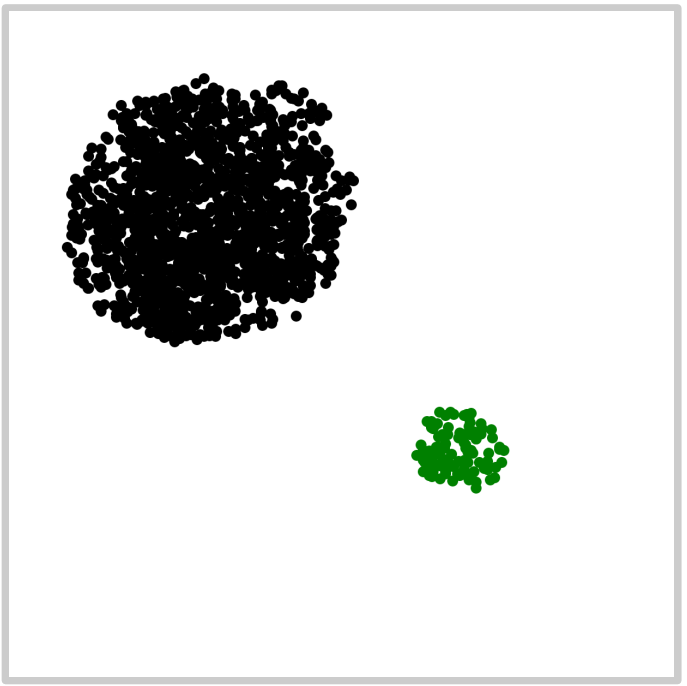


- Interpretação dos pesos



Classificação: Redes Neurais

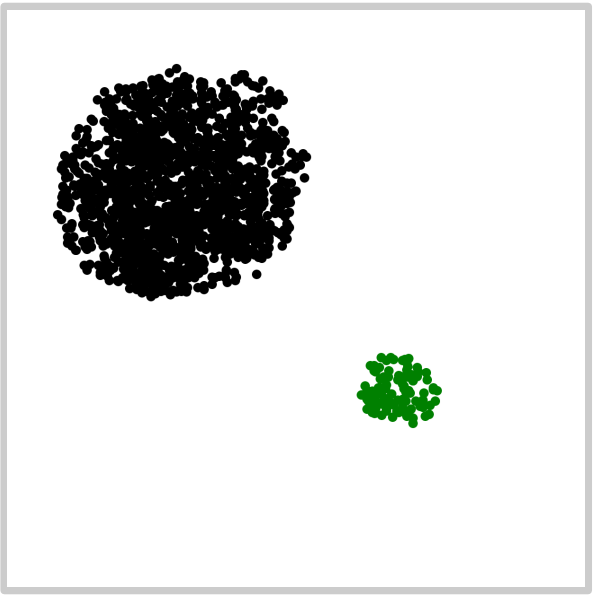
- Classificação com o modelo

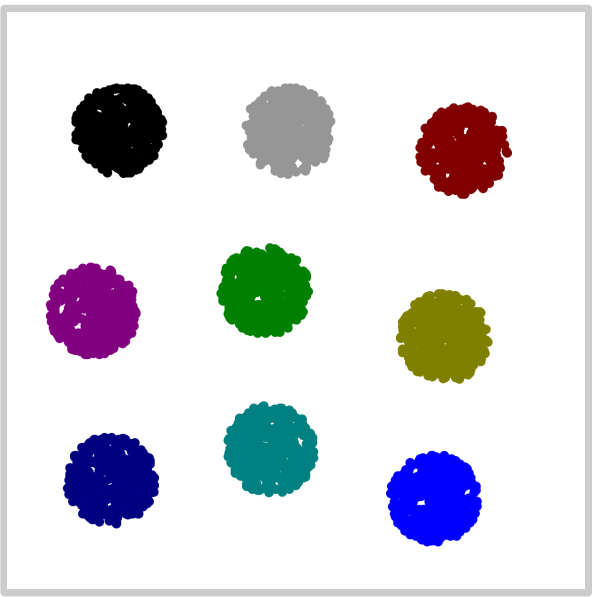


Atr. 1	Atr. 2	Saída Nó 0	Saída Nó 1	Classe	Observações
0	0	-22.62	22.62	8	Canto Inf. Esq.
0	640	31527.72	-31527.61	0	Canto Sup. Esq.
640	0	-31446.89	31446.17	8	Canto Inf. Dir.
640	640	103.46	-104.06	0	Canto Sup. Dir.
320	320	40.42	-40.72	0	Ponto central
191	540	17219.8	-17219.92	0	Ex. Classe 0
458	237	-10827.12	10826.65	8	Ex. Classe 8

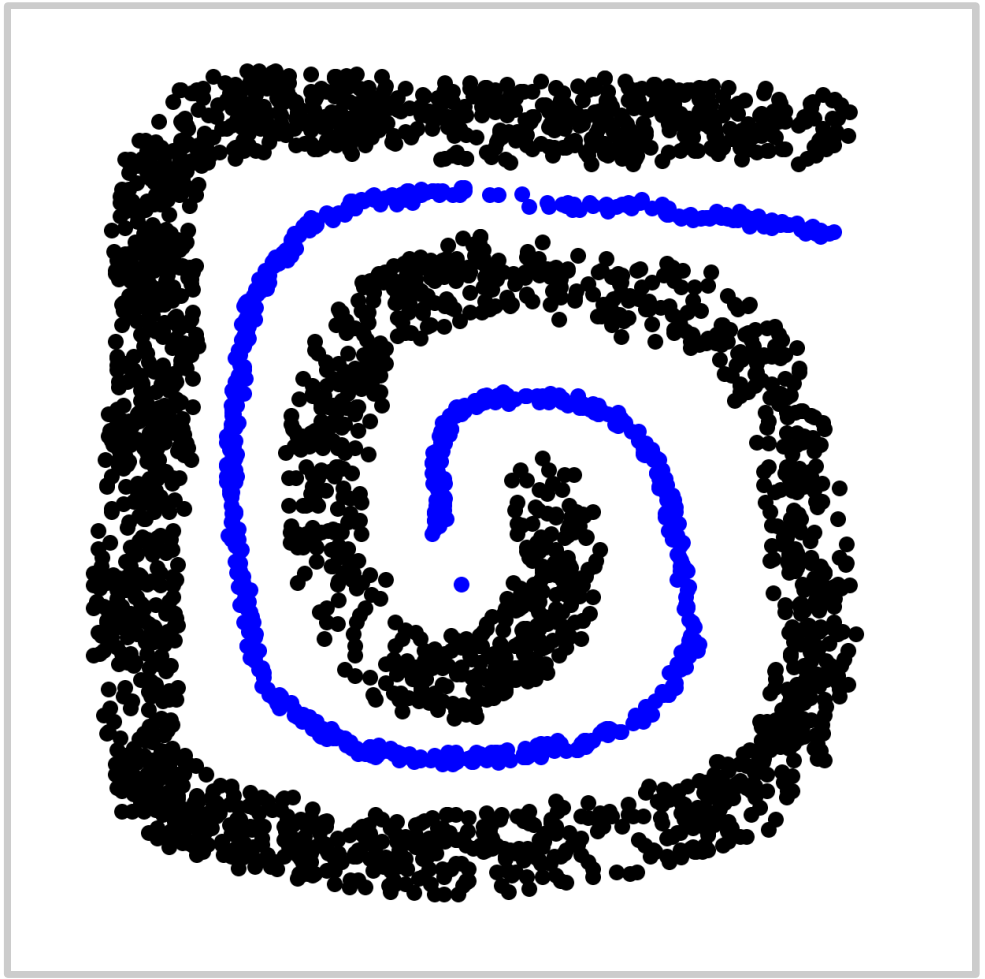
Classificação: Redes Neurais

- Arquitetura 2x1x2 também classifica corretamente 100% das amostras no primeiro exemplo.
- Segundo exemplo:
  - Arquitetura 2x1x9: muitos erros, não adequada para este problema.
  - 2 neurônios na camada escondida são suficientes para classificar com 100% de acerto.





Classificação: Redes Neurais



- Três arquiteturas para classificação:
  - 1 camada escondida, 5 neurônios.
  - 1 camada escondida, 25 neurônios.
  - 1 camada escondida, 250 neurônios.
- Camada de entrada: sempre 2 neurônios (x,y).
- Camada de saída: sempre 2 neurônios (k,b).

Classificação: Redes Neurais

5

```
Correctly Classified Instances   3050 88.4314 %
Incorrectly Classified Instances  399 11.5686 %
=== Confusion Matrix ===
   a    b   <-- classified as
 2689   18 |    a = 0
  381  361 |    b = 13
```

25s

25

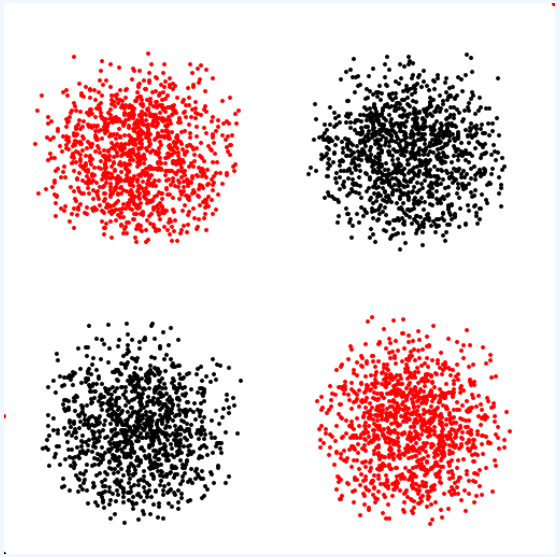
```
Correctly Classified Instances   3446 99.913 %
Incorrectly Classified Instances    3  0.087 %
=== Confusion Matrix ===
   a    b   <-- classified as
 2705    2 |    a = 0
    1  741 |    b = 13
```

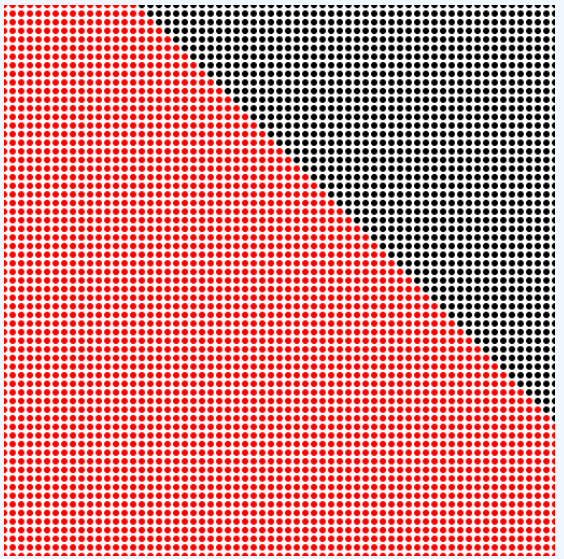
96s

250

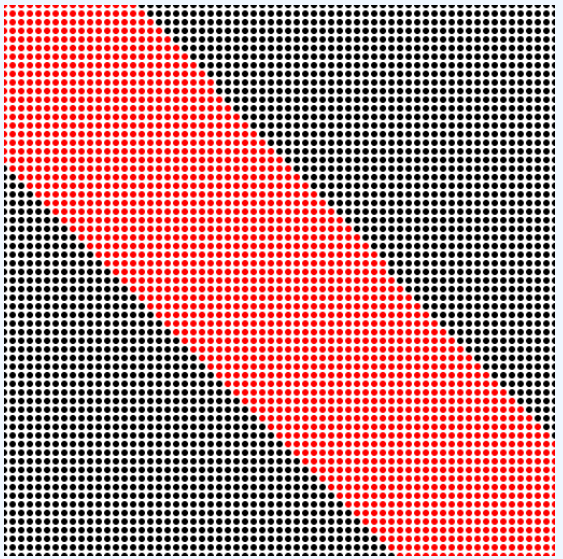
```
Correctly Classified Instances   3448 99.971 %
Incorrectly Classified Instances    1  0.029 %
=== Confusion Matrix ===
   a    b   <-- classified as
 2707    0 |    a = 0
    1  741 |    b = 13
```

786s

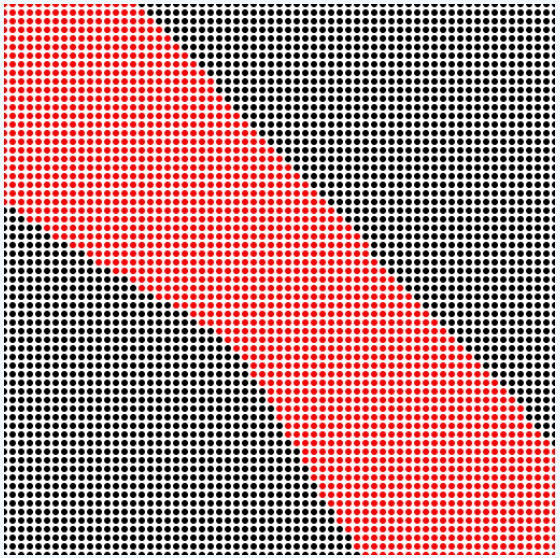




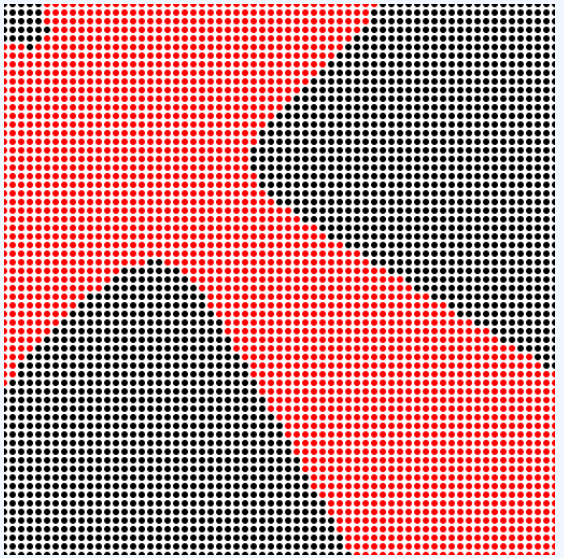
1



2

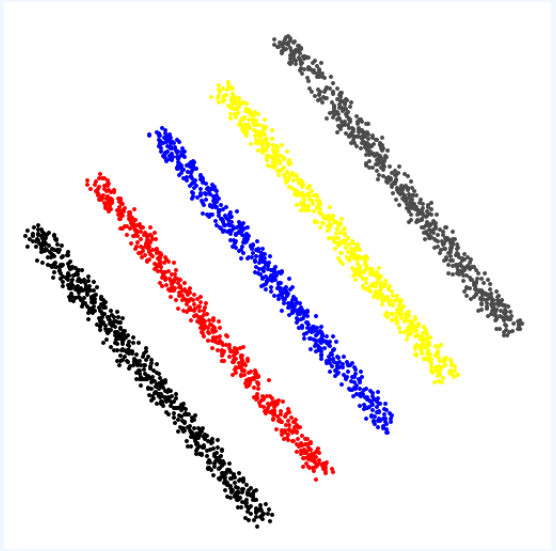


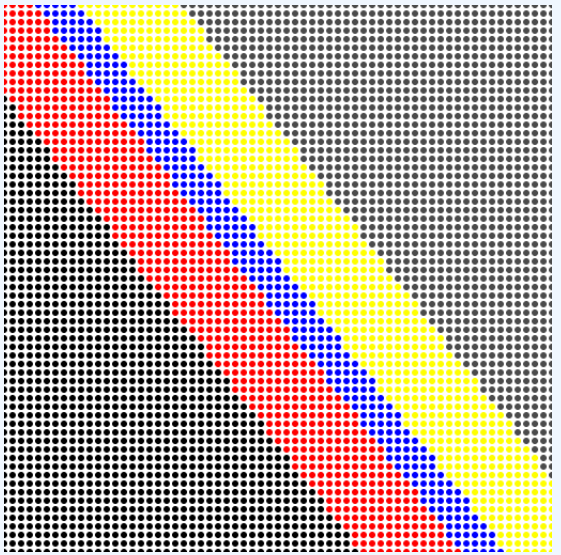
5



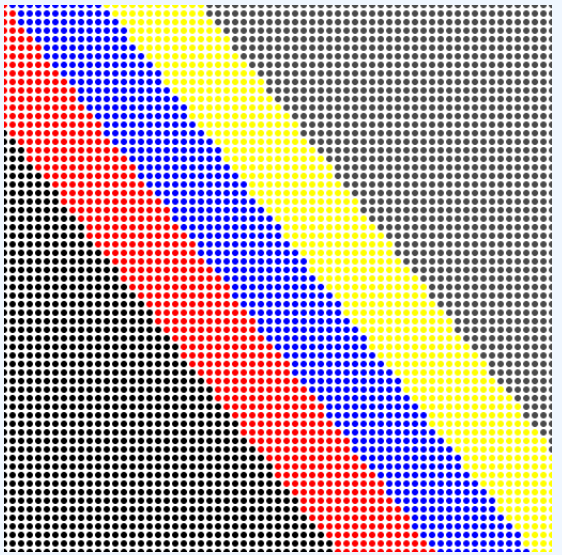
10

25

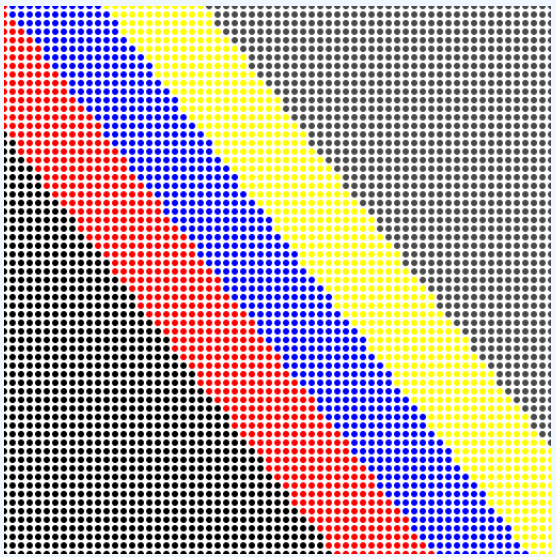




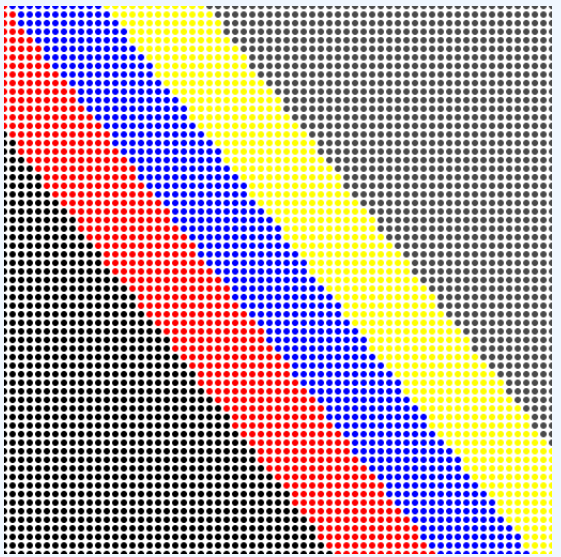
1



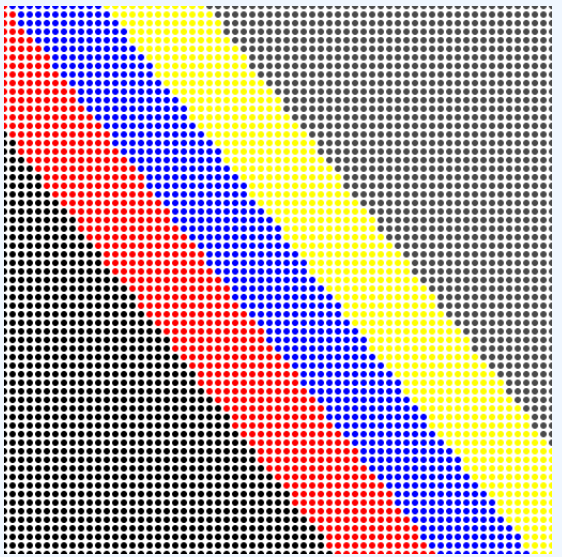
5



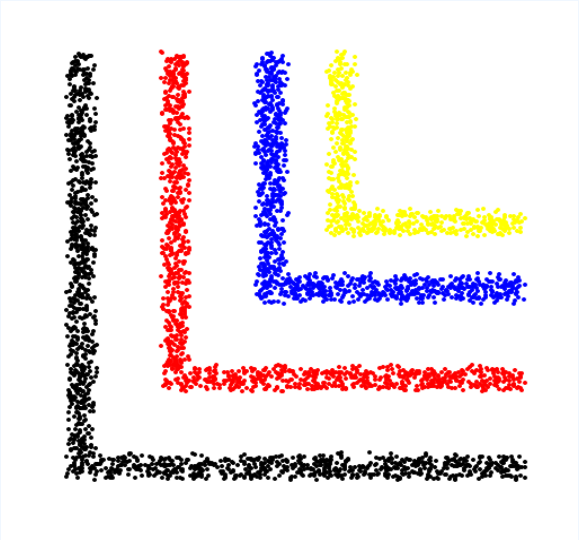
10

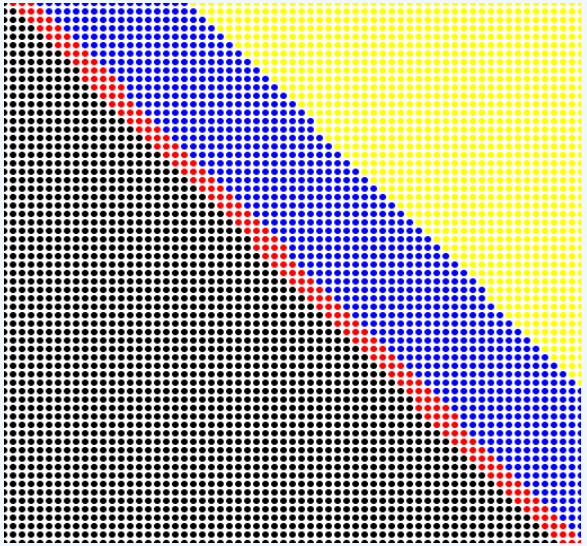


25

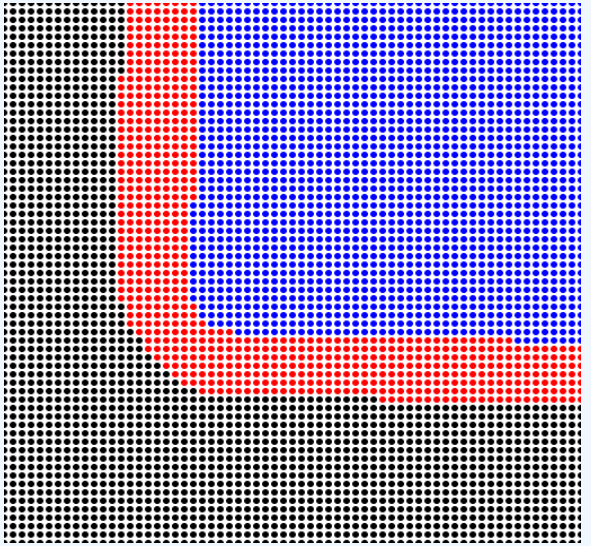


50

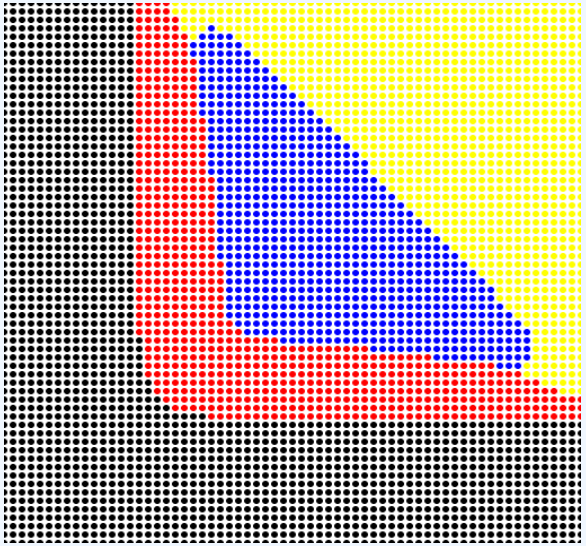




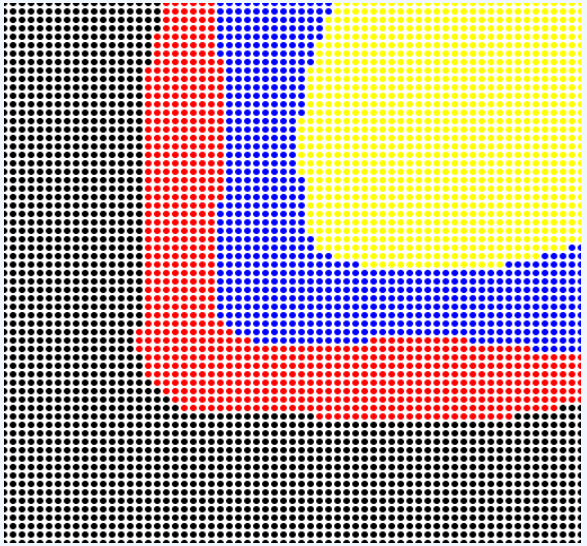
1



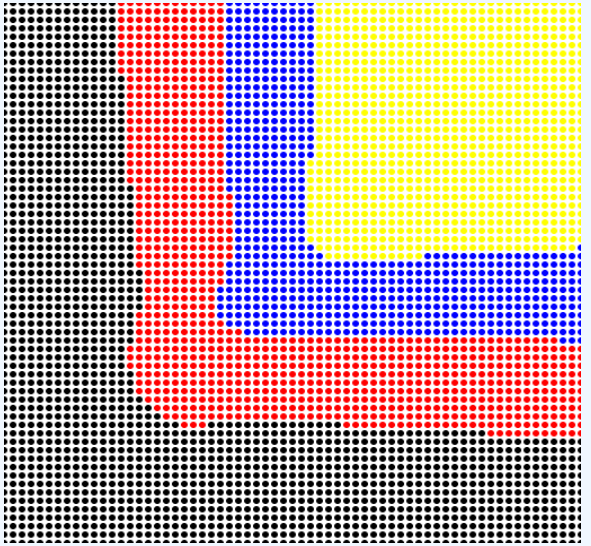
2



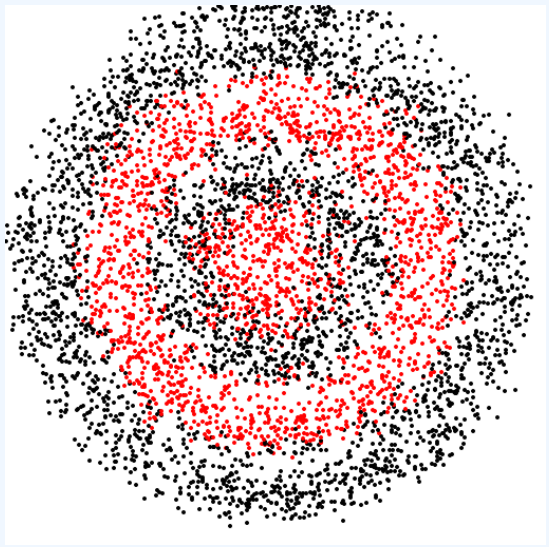
3

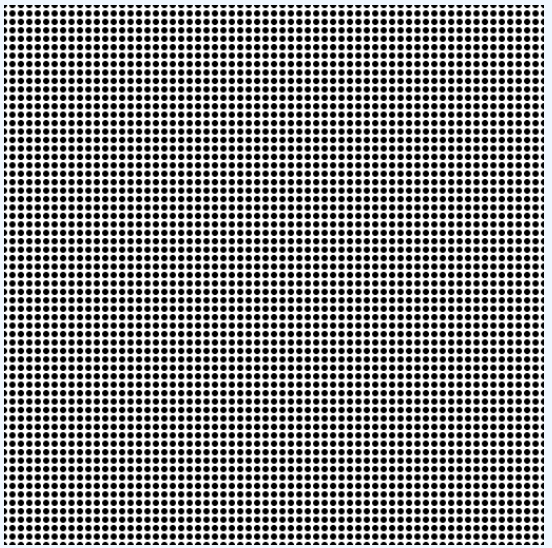


5

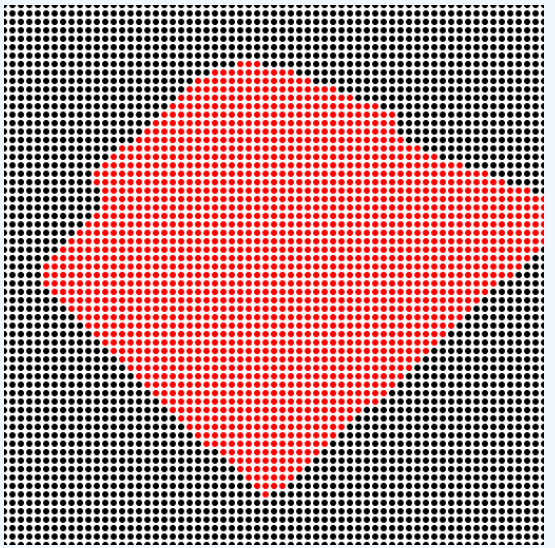


15

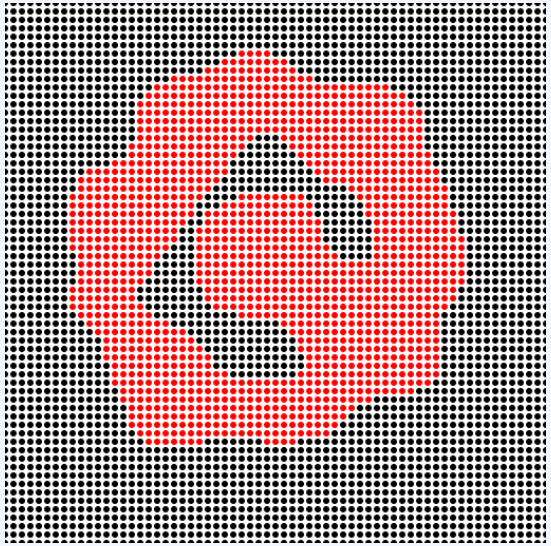




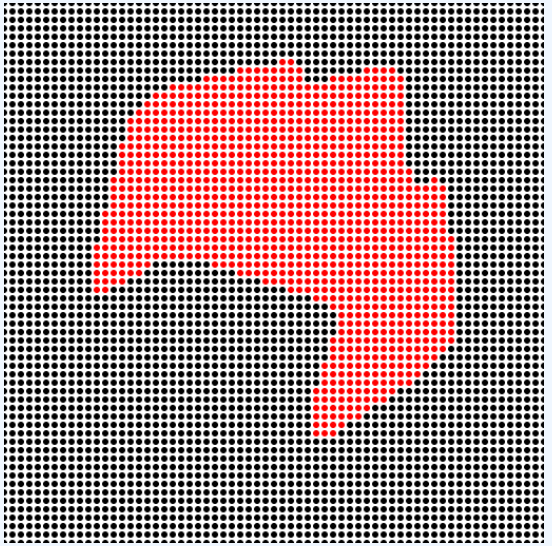
1



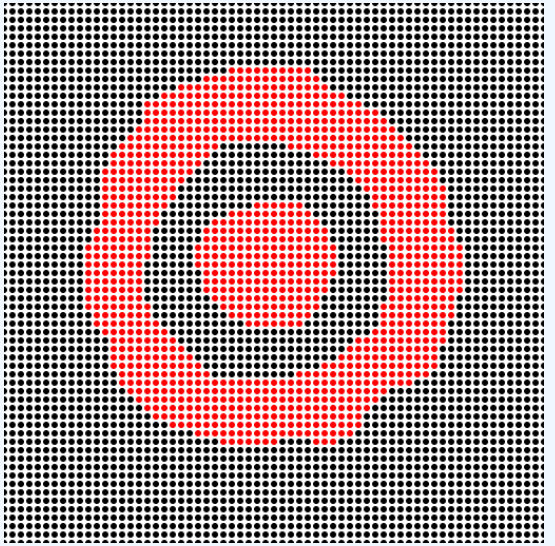
5



10



30



100

Classificação: Redes Neurais

- Vantagens:
  - Capacidade de separar bem classes não linearmente separáveis (com múltiplas camadas).
- Problemas:
  - Difícil explicar/interpretar o “modelo” (caixa preta).
  - Treinamento pode ser complexo (computacionalmente caro), definição da arquitetura também.
- Soluções:
  - Múltiplas arquiteturas e avaliação da qualidade de classificação.
  - Avaliação de treinamento e teste.

Próximos passos...

- ***Dia 1:*** Apresentação dos conceitos de mineração de dados, motivação e alguns exemplos.
- ***Dia 2:*** Algoritmos de classificação supervisionada e aplicações.
- ***Dia 3:*** Algoritmos de classificação não-supervisionada e aplicações. Algoritmos de mineração de associações.
- ***Dia 4:*** Visualização e mineração de dados. Outros algoritmos e idéias. Onde aprender mais.

Mais informações em...

- http://www.lac.inpe.br/~rafael.santos
  - http://www.lac.inpe.br/~rafael.santos/dmapresentacoes.jsp
  - http://www.lac.inpe.br/~rafael.santos/cap359-2010.jsp
- http://www.lac.inpe.br/ELAC/index.jsp