

Grupo de Estudos de Visão Computacional

João Pedro de Castro
Bruno Calura
Cleiver Batista

Algoritmos Clássicos vs Modernos para Detecção e Reconhecimento

Goiânia
21/11/2024

Apoio

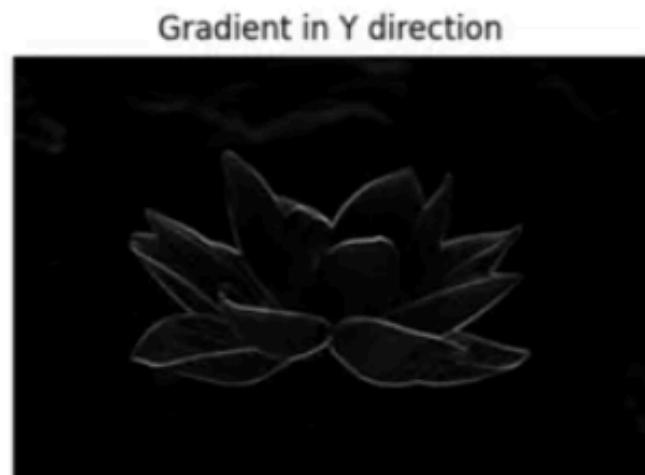
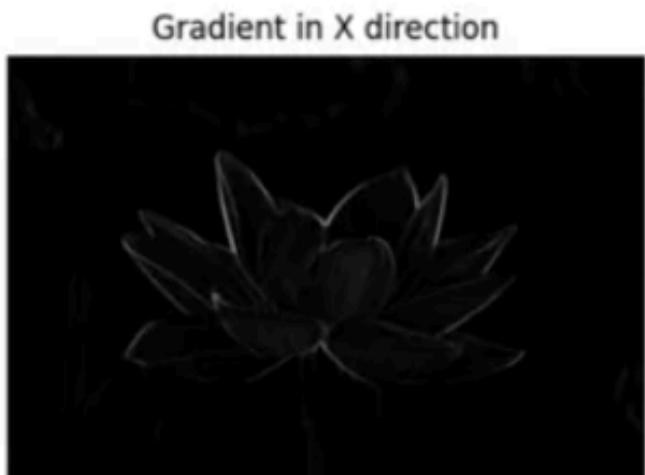


Sumário

1. Introdução Histórica
2. Aprendizado de Máquina
3. Modelos Clássicos (KNN,
Haar Cascade, SVM)
4. Métodos modernos
5. Apresentação prática

Introdução Histórica

- 1950-1960: Pesquisadores começaram a explorar a ideia de ensinar o computador a compreender dados visuais
- 1970-1980: Pesquisadores estavam desenvolvendo algoritmos de detecção de borda e extração de features.

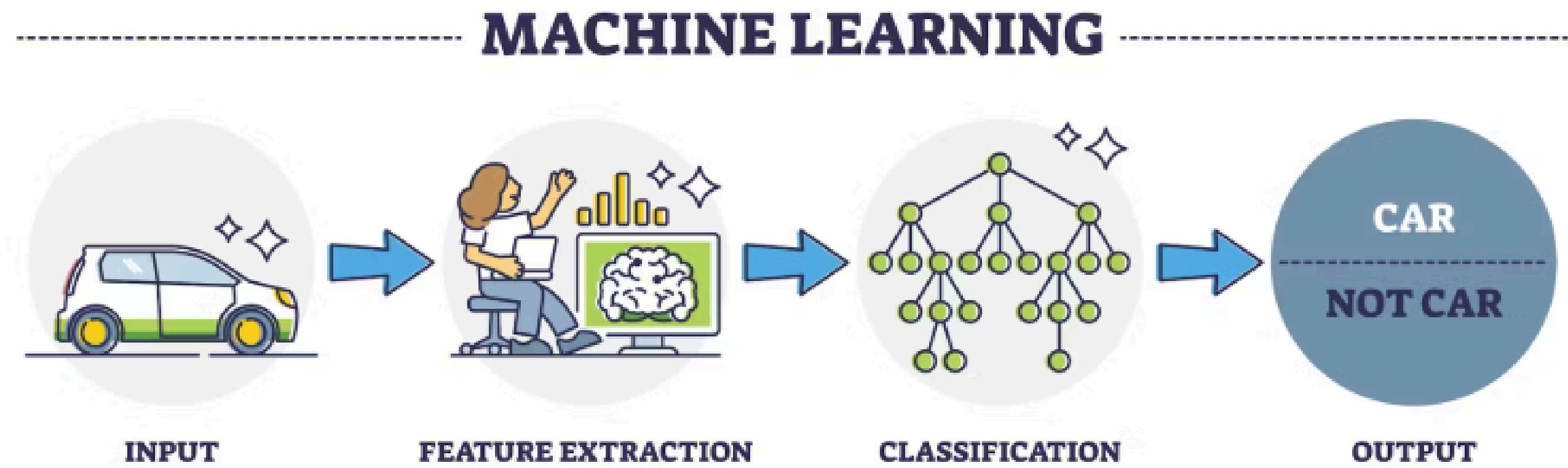


- Essas técnicas forneceram a base para tarefas mais complexas na área de visão computacional.

Introdução Histórica

- 1980-1990: Pesquisadores começaram a explorar o reconhecimento de objetos e cenas por meio de técnicas do aprendizado de máquina.
- Desenvolvimento do algoritmo Scale-Invariant Feature Transform (SIFT)
- 2000: Popularização do Support Vector Machines (SVM) para reconhecimento de objetos e do algoritmo de Viola-Jones para detecção de rostos em tempo real.

Aprendizado de Máquina



Aprendizado de Máquina

Tipo de aprendizado

1. Aprendizado supervisionado:

- Regressão;
- Classificação.

2. Aprendizado não supervisionado:

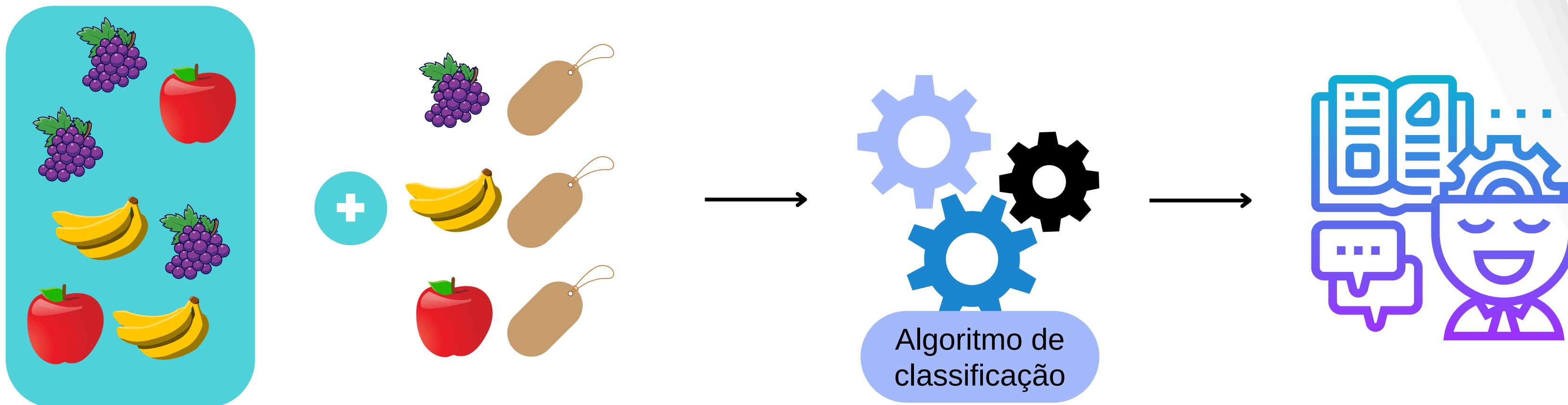
- Agrupamento;
- Redução de dimensionalidade.

3. Aprendizado por reforço:

- Métodos heurísticos;
- IA em jogos;
- Decisões em tempo real

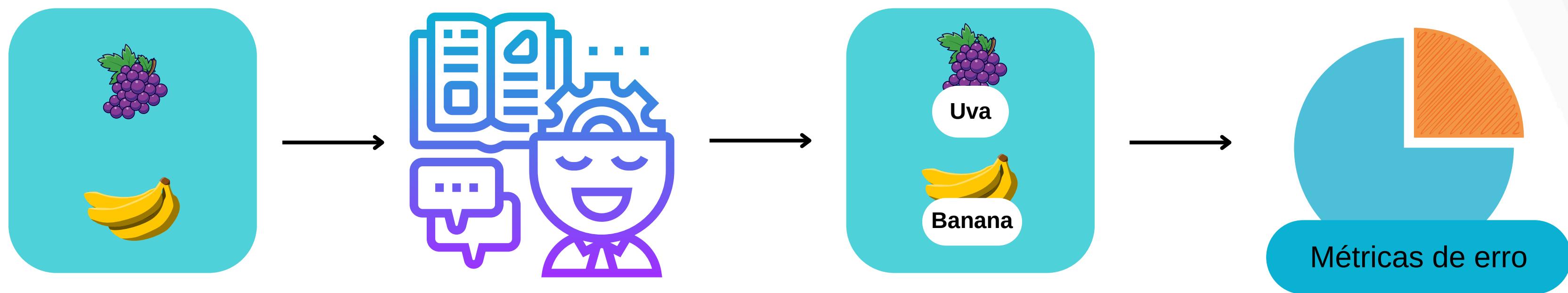
Aprendizado de Máquina

Aprendizado supervisionado



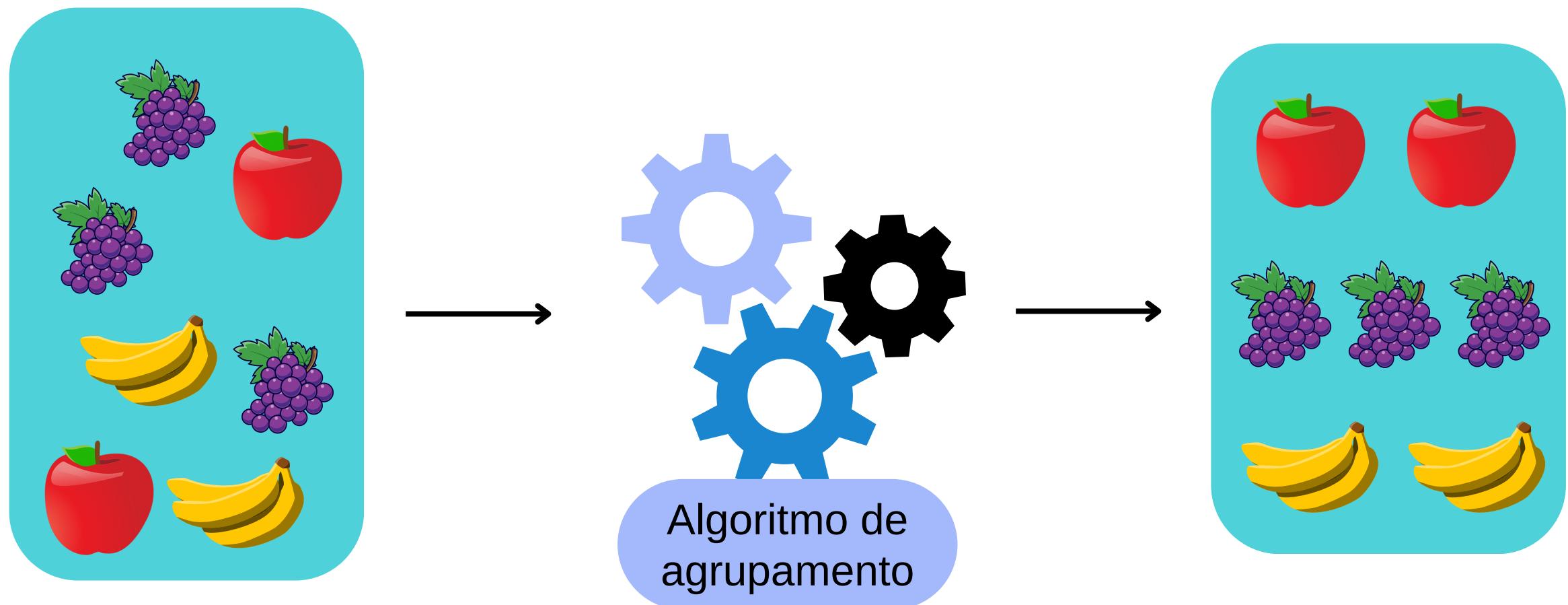
Aprendizado de Máquina

Aprendizado supervisionado



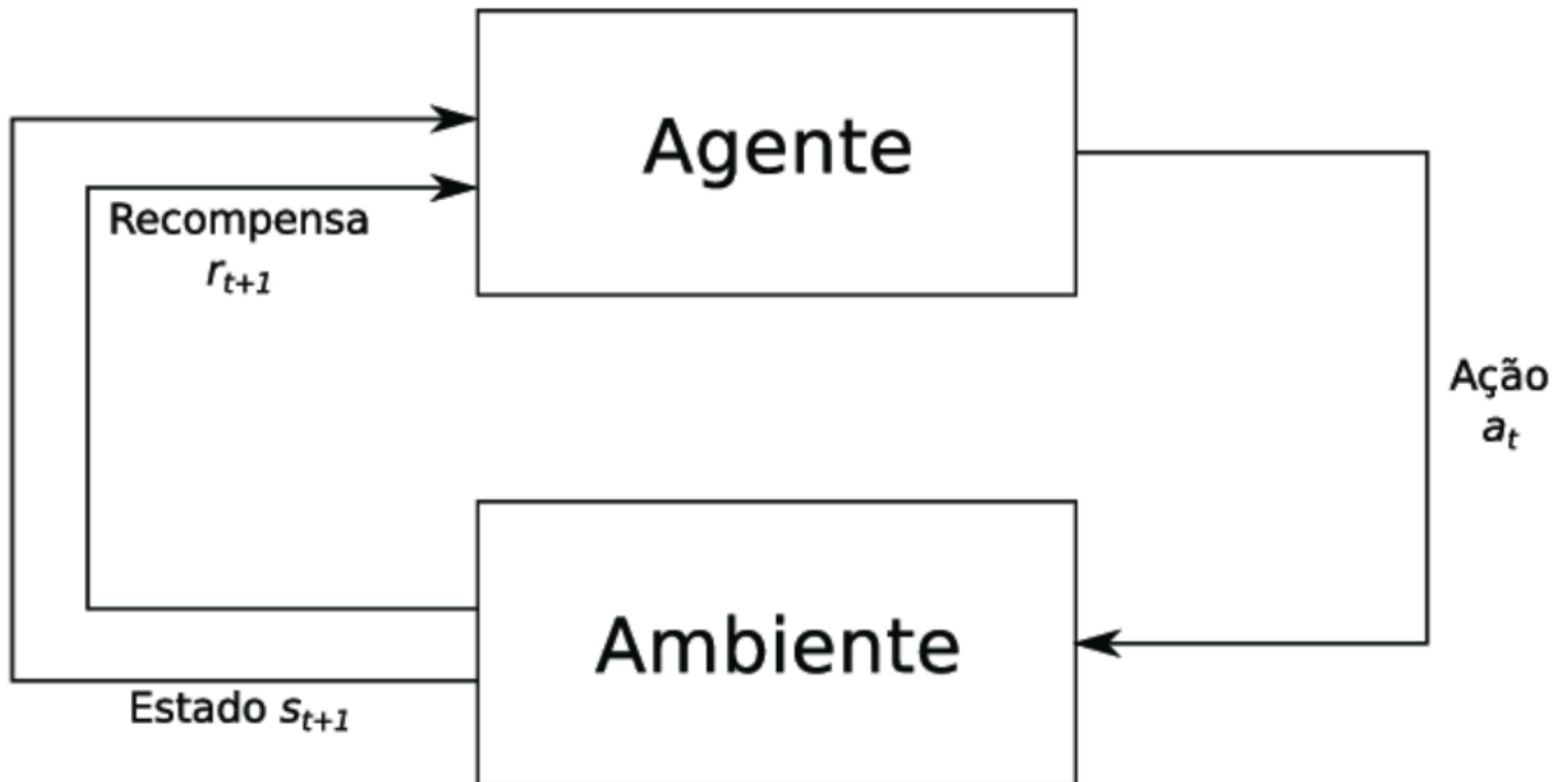
Aprendizado de Máquina

Aprendizado não supervisionado



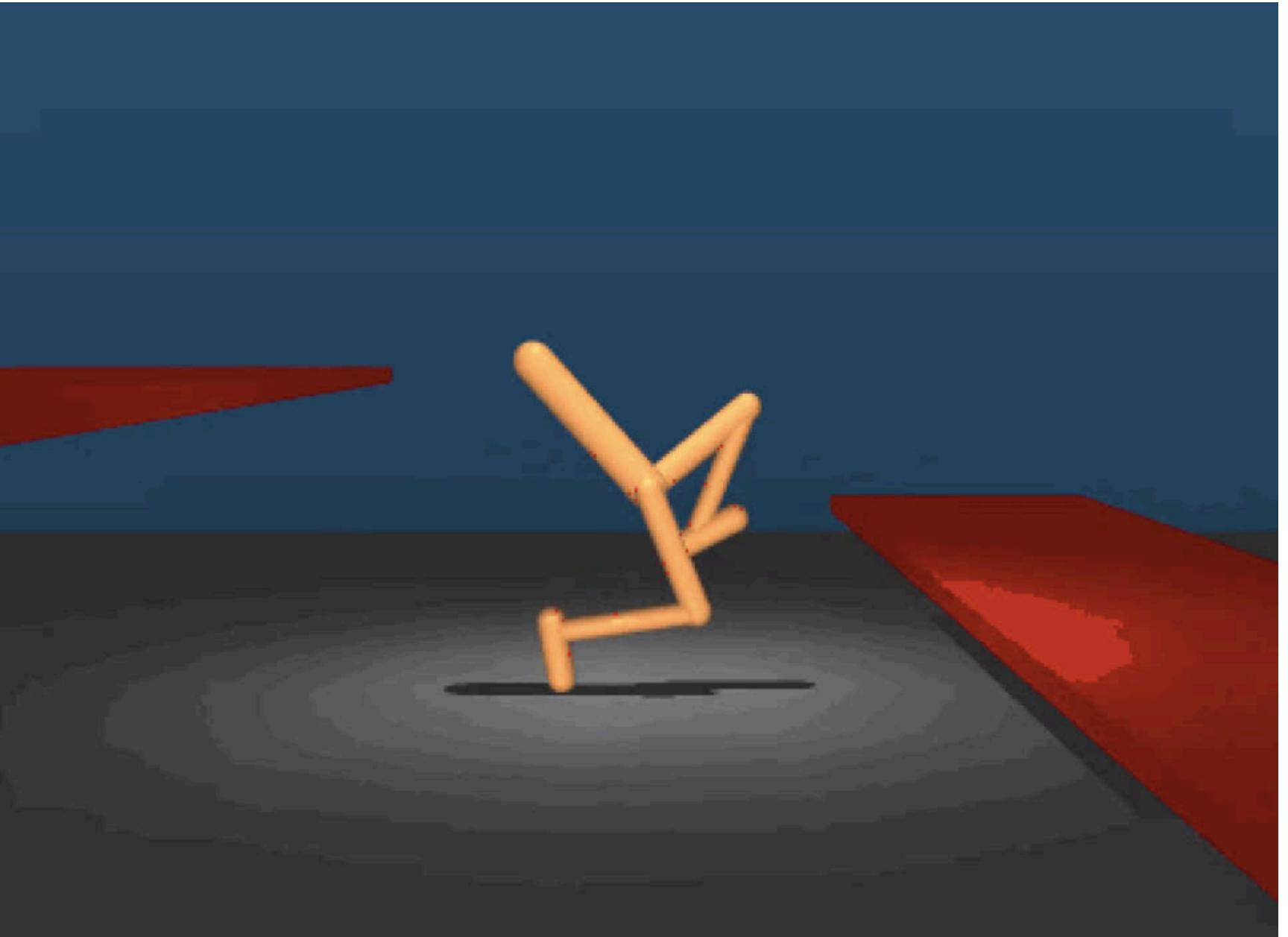
Aprendizado de Máquina

Aprendizado por reforço



Aprendizado de Máquina

Aprendizado por reforço



KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

História:

- Surgiu na estatística, por volta da década de 50 com Evelyn Fix e Joseph Hodges
- Criado como conceito de classificação por proximidade
- Em 1967, Thomas Cover provou que o erro de classificação do KNN é o dobro do erro do classificador ideal, demonstrando ser um conceito muito sólido

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Entendendo como funciona:

Usando o exemplo de flores

- Objetivo: identificar a espécie de uma flor
- Cria-se um espaço multidimensional, sendo uma dimensão para cada característica analisada. Exemplo: tamanho da flor e tamanho da pétala
- É necessário extrair essas características dos dados e colocar no espaço multidimensional

KNN (K-Nearest Neighbors)

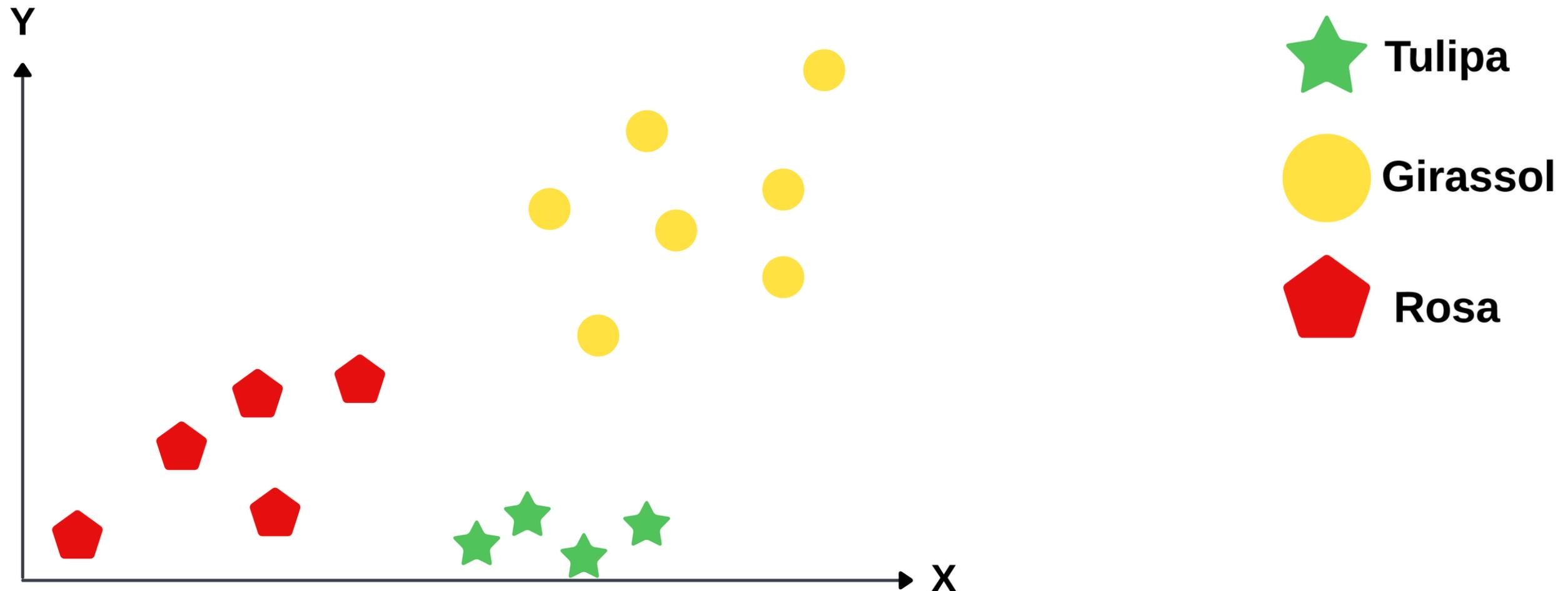
K-Vizinhos mais próximos

Entendendo como funciona:

Usando o exemplo de flores

X = Tamanho da flor (fase adulta)

Y = Tamanho da pétala (fase adulta)



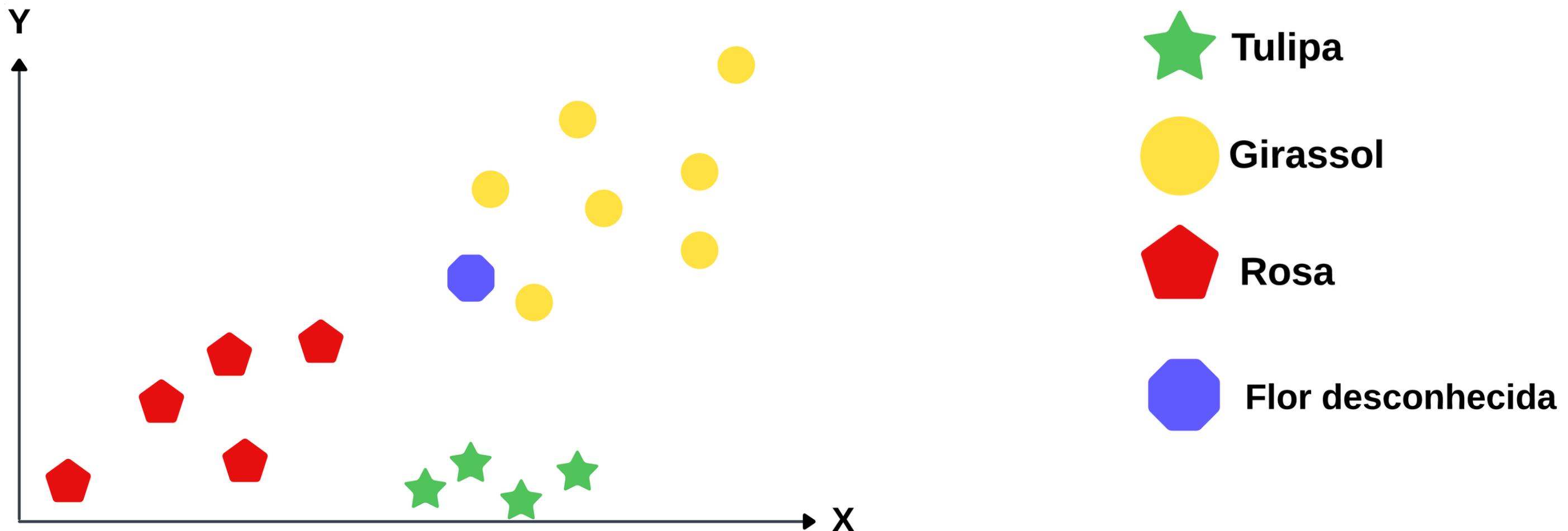
KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Entendendo como funciona:

Usando o exemplo de flores

- Analisa-se as características da flor que quero descobrir a espécie, e depois joga essa flor no espaço multidimensional



KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Entendendo como funciona:

Usando o exemplo de flores

- O K, do KNN, é uma constante.
- O algoritmo realiza o reconhecimento analisando as classificações dos K pontos mais próximos do que quero descobrir
- Exemplo: K = 6. Analisa-se os 6 pontos mais próximos do ponto não classificado
- O KNN reconhece que o ponto desconhecido pertence a moda dos K pontos mais próximos

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Entendendo como funciona:

Calculando a distância

- O cálculo da distância pode ser feito usando diversas formas, como:
Distância Euclidiana, Manhattan, Minkowski, etc.
- Irei utilizar a distância euclidiana: $d(x,y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
 - x e y são vetores que representam os pontos no espaço
 - x_i e y_i são as coordenadas dos pontos em cada dimensão i
 - n é o número de dimensões do espaço.

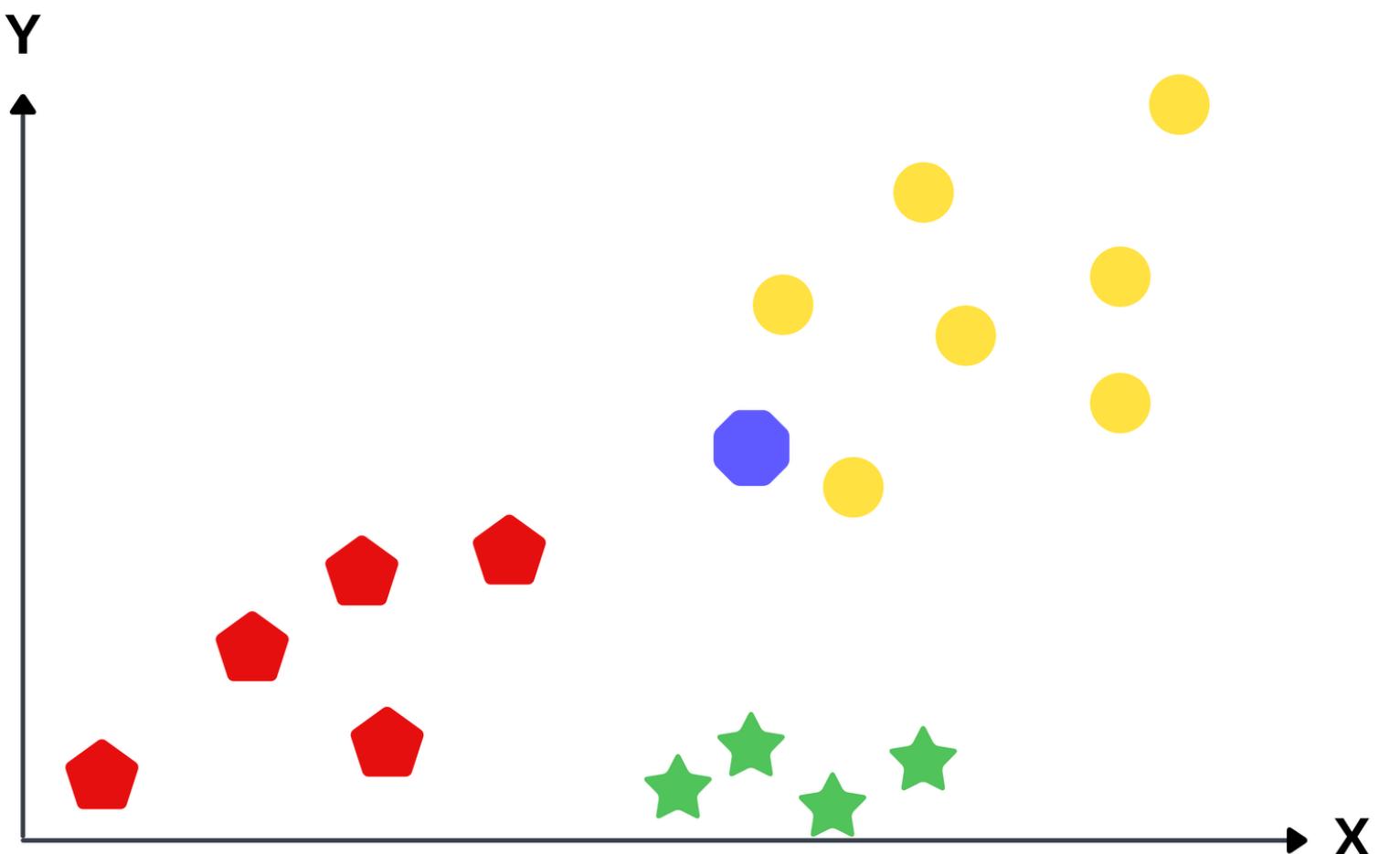
KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Entendendo como funciona:

Usando o exemplo de flores

- Usando $K = 6$



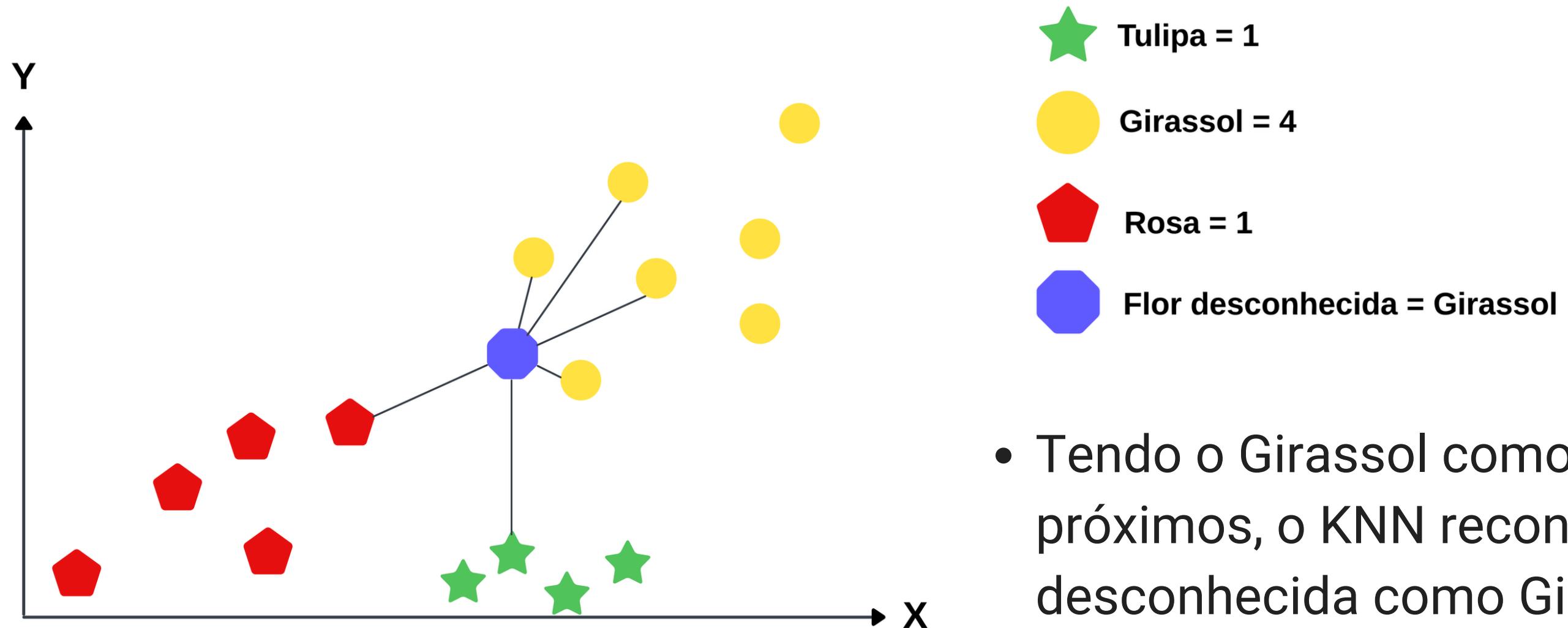
KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Entendendo como funciona:

Usando o exemplo de flores

- Usando K = 6



KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

- Utilizamos a fórmula seguinte para dar maior importância aos pontos mais próximos do ponto desconhecido (X_q)

$$f'(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \cdot \delta(v, f(x))$$

- O w_i é o peso que vai ser atribuído. Uma forma, de diversas, que podemos calcular é usando métodos de ponderação como: $\frac{1}{D}$
- D : é a distância do ponto desconhecido até um vizinho próximo

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

$$f'(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \cdot \delta(v, f(x_i))$$

- $f'(x_q)$: é a classe final atribuída à amostra x_q (ponto desconhecido)
- $\operatorname{argmax}_{v \in V}$: indica que a classe v será aquela que maximiza o valor da soma calculada
- V : Conjunto de todas classes possíveis
- $\sum_{i=1}^k w_i \cdot \delta(v, f(x_i))$: Soma os pesos (w_i - peso atribuído ao i -ésimo vizinho) dos k vizinhos mais próximos, considerando suas contribuições para cada classe v .
 $\delta(v, f(x_i))$ é a função que verifica se o i -ésimo vizinho pertence à classe v

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

$$f'(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \cdot \delta(v, f(x_i))$$

De forma simplificada:

- Vou pegar um vizinho próximo
- Faço a cálculo do peso (Inversamente proporcional a distância) w_i ,
- Verifico a qual classe esse vizinho pertence. Se ele pertence a classe v , ele recebe o valor de 1, caso contrario recebe 0 $\delta(v, f(x_i))$
- Depois soma os pesos que são da mesma classe. A classe que tiver o maior resultado dessa soma será a classe atribuída a amostra

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

- Utilizamos a fórmula seguinte para aplicar pesos nas características, sendo pesos maiores para as features mais importantes

$$D = \sqrt{\sum_{i=1}^n w_i (x_i - x'_i)^2}$$

- Existem features menos relevantes, e para elas não impactarem o resultado do algoritmo de KNN, utiliza-se diversos métodos, sendo esse apenas 1 exemplo

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

Exemplificando

- Vamos treinar um algoritmo para identificar raças de cachorros
- Analisaremos 2 características: o tamanho em fase adulta e a cor dos pelos

**Se for falado que o cachorro que queremos descobrir
a raça tem pelagem preta e marrom
É possível ter grandes chances de acertar sabendo
apenas essa característica?**

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

Exemplificando

- Vamos treinar um algoritmo para identificar raças de cachorros
- Analisaremos 2 características: o tamanho em fase adulta e a cor dos pelos

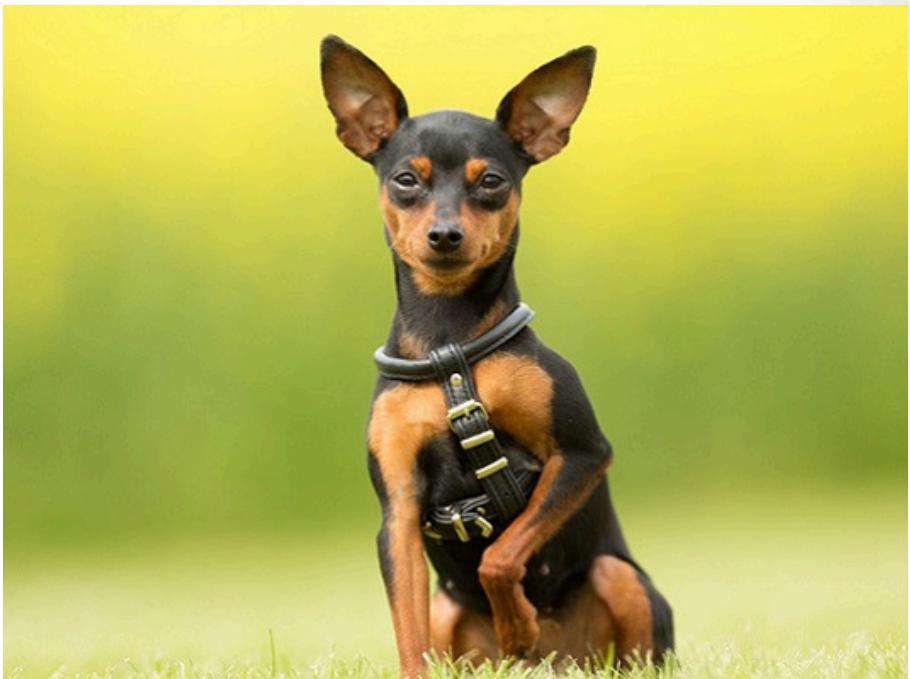
Se for falado que o cachorro que queremos descobrir a raça é de pequeno porte e, normalmente, não ultrapassa os 30 cm

Concordam que essa característica é muito mais relevante do que a cor do pelo?

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:
Exemplificando

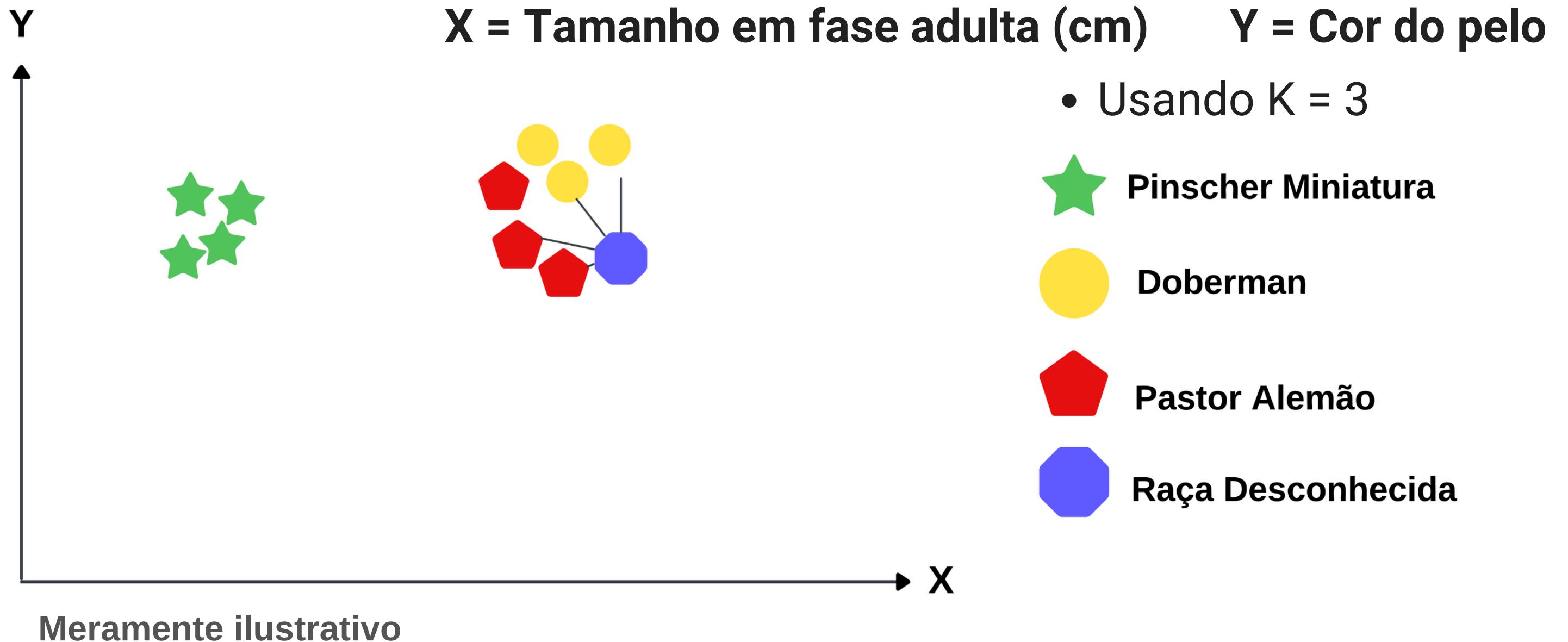


- Entre os dados, terão 3 raças de cachorros: Pastor Alemão, Doberman e Pinscher Miniatura

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:
Exemplificando



KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais: Exemplificando

- Quando $K = 3$, a amostra está mais próximo de 2 pastores alemão e de 1 doberman. Portanto, ela deveria ser classificada como pastor alemão, mas como a característica de tamanho é mais importante do que a de cor, vamos fazer o cálculo para descobrir a que raça pertence esse cachorro
- Irei atribuir peso 0.9 para a feature de tamanho
- Irei atribuir peso 0.1 para a feature de cor de pelo

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:

Exemplificando

- Irei atribuir coordenadas para os pontos (não serão pontos exatos, será apenas para demonstrar a fórmula). A coordenada da amostra é (73, 66)

Pastor Alemão 1 (68, 64)

$$\begin{aligned} D_1 &= \sqrt{0.9 \cdot (73 - 68)^2 + 0.1 \cdot (66 - 64)^2} \\ D_1 &= \sqrt{0.9 \cdot 5^2 + 0.1 \cdot 2^2} = \sqrt{0.9 \cdot 25 + 0.1 \cdot 4} \\ D_1 &= \sqrt{22.5 + 0.4} = \sqrt{22.9} \approx 4.79 \end{aligned}$$

Pastor Alemão 2 (63, 70)

$$\begin{aligned} D_2 &= \sqrt{0.9 \cdot (73 - 63)^2 + 0.1 \cdot (66 - 70)^2} \\ D_2 &= \sqrt{0.9 \cdot 10^2 + 0.1 \cdot (-4)^2} = \sqrt{0.9 \cdot 100 + 0.1 \cdot 16} \\ D_2 &= \sqrt{90 + 1.6} = \sqrt{91.6} \approx 9.57 \end{aligned}$$

Pastor Alemão 3 (61, 85)

$$\begin{aligned} D_3 &= \sqrt{0.9 \cdot (73 - 61)^2 + 0.1 \cdot (66 - 85)^2} \\ D_3 &= \sqrt{0.9 \cdot 12^2 + 0.1 \cdot (-19)^2} = \sqrt{0.9 \cdot 144 + 0.1 \cdot 361} \\ D_3 &= \sqrt{129.6 + 36.1} = \sqrt{165.7} \approx 12.87 \end{aligned}$$

Doberman 1 (68, 64)

$$\begin{aligned} D_4 &= \sqrt{0.9 \cdot (73 - 70)^2 + 0.1 \cdot (66 - 80)^2} \\ D_4 &= \sqrt{0.9 \cdot 3^2 + 0.1 \cdot (-14)^2} = \sqrt{0.9 \cdot 9 + 0.1 \cdot 196} \\ D_4 &= \sqrt{8.1 + 19.6} = \sqrt{27.7} \approx 5.26 \end{aligned}$$

Doberman 2 (72, 90)

$$\begin{aligned} D_5 &= \sqrt{0.9 \cdot (73 - 72)^2 + 0.1 \cdot (66 - 90)^2} \\ D_5 &= \sqrt{0.9 \cdot 1^2 + 0.1 \cdot (-24)^2} = \sqrt{0.9 \cdot 1 + 0.1 \cdot 576} \\ D_5 &= \sqrt{0.9 + 57.6} = \sqrt{58.5} \approx 7.65 \end{aligned}$$

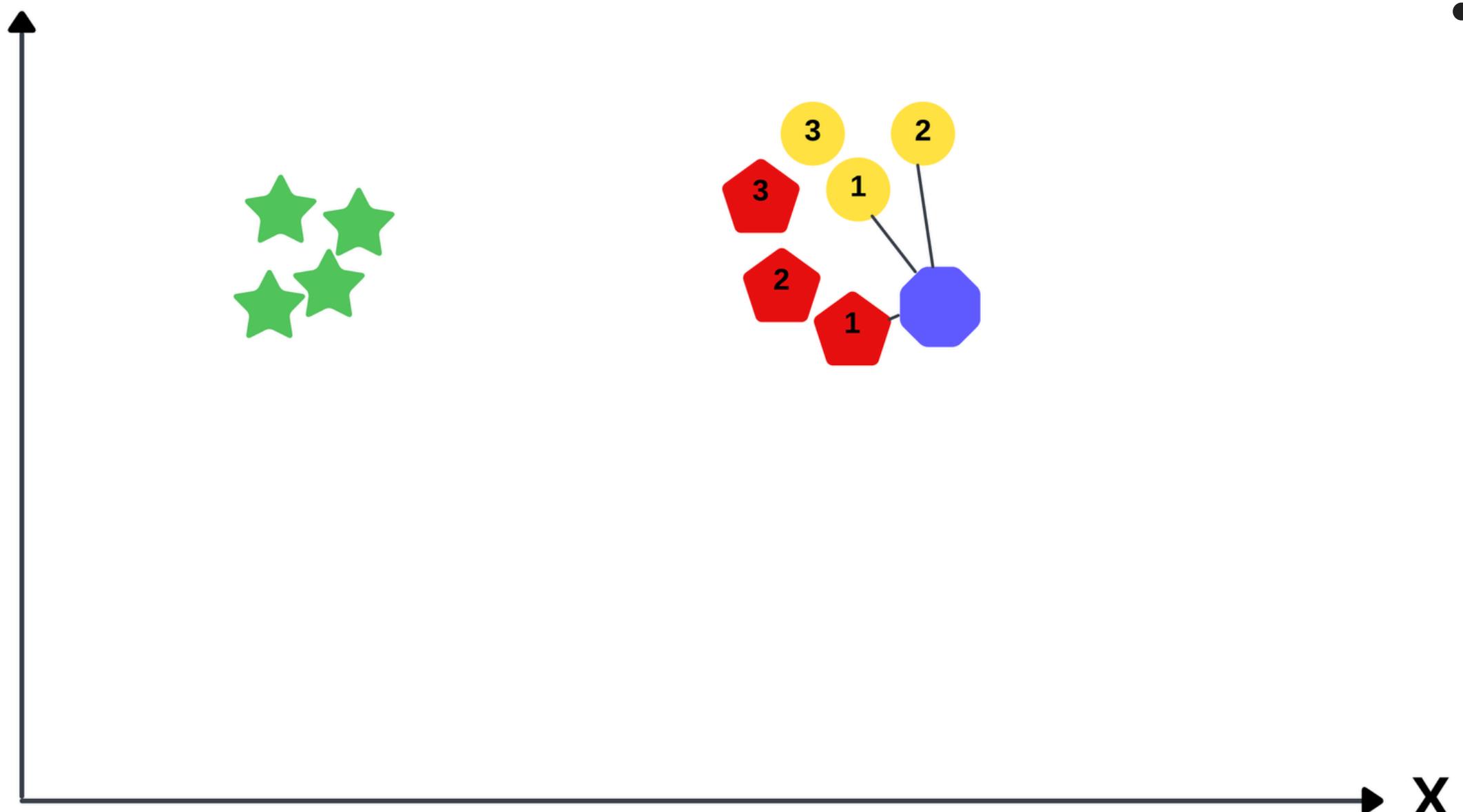
Doberman 3 (68, 88)

$$\begin{aligned} D_6 &= \sqrt{0.9 \cdot (73 - 68)^2 + 0.1 \cdot (66 - 88)^2} \\ D_6 &= \sqrt{0.9 \cdot 5^2 + 0.1 \cdot (-22)^2} = \sqrt{0.9 \cdot 25 + 0.1 \cdot 484} \\ D_6 &= \sqrt{22.5 + 48.4} = \sqrt{70.9} \approx 8.42 \end{aligned}$$

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:
Exemplificando



- Usando pesos, assim fica os 3 vizinhos mais próximos

★ Pinscher Miniatura

○ Doberman

△ Pastor Alemão

□ Raça Desconhecida

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais: Exemplificando

Classificação com $k = 3$:

Os 3 vizinhos mais próximos são:

1. Pastor Alemão 1 (4.79)
2. Doberman 1 (5.26)
3. Doberman 2 (7.65)

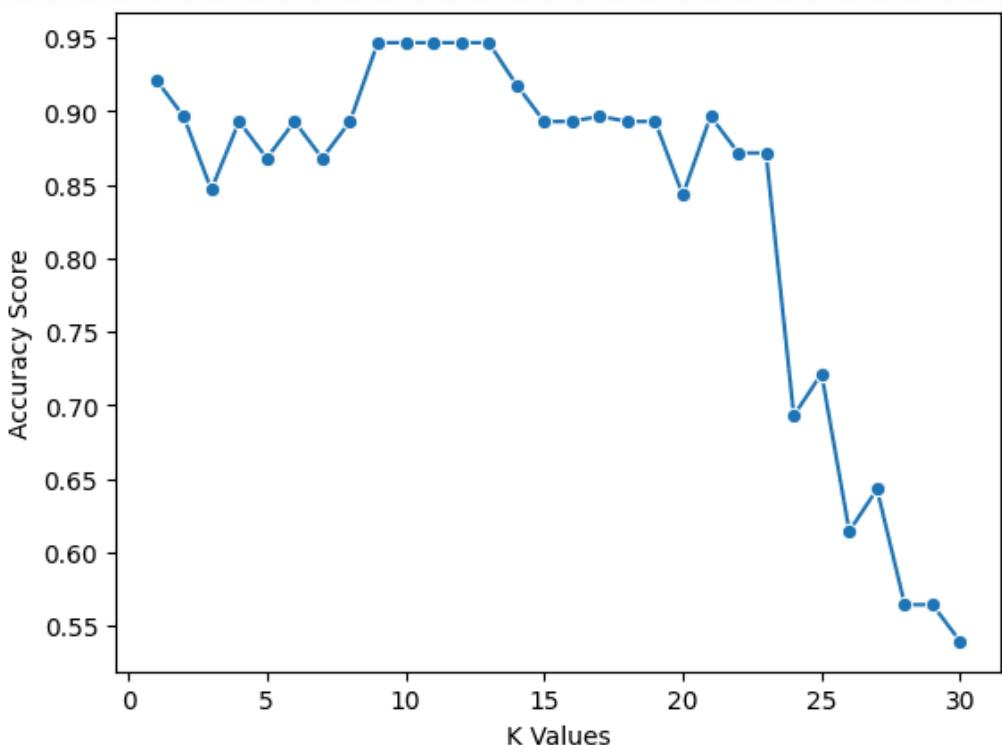
- Assim fica os K vizinhos mais próximos, demonstrando como a resposta do algoritmo pode mudar de acordo com a importância de cada característica

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais:
Como descobrir o K ideal?

- O valor usado no K é muito importante para o resultado do algoritmo
- Valores baixos podem resultar em overfitting
- Valores altos podem resultar em underfitting



KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Aprofundando um pouco mais: Como descobrir o K ideal?

- Existem diversos métodos para achar o K ideal, como:
- **Validação Cruzada:** Divide os dados em partes e, para cada valor de K , treina o modelo numa parte e faz o teste em outra
- **Conjunto de Validação:** Divide os dados em três conjuntos - treinamento, validação e teste. Testa o desempenho em diferentes valores de K usando a parte de validação.
- **Heurísticas e métodos comuns**
- **Escolha baseada na Métrica de Desempenho:** MAE, MSE, entre outras métricas.

KNN (K-Nearest Neighbors)

K-Vizinhos mais próximos

Características:

- Algoritmo preguiçoso: o modelo armazena os dados de treinamento e calcula as respostas durante a inferência, o que pode ter um custo computacional elevado
- Funcionamento simples e fácil de interpretar
- Algoritmo muito versátil: pode ser utilizado para regressão e classificação
- Sensibilidade aos dados: ruídos, características irrelevantes ou outliers podem impactar negativamente o desempenho. Dessa forma, é necessário normalizar os dados para evitar que uma feature domine as outras

Haar Cascade

Cascade de Haar

História:

- Método proposto, por volta de 2001, pelos pesquisadores Paul Viola e Michael Jones.
- Foi apresentado no paper “ **Rapid Object Detection using a Boosted Cascade of Simple Features** ”

Haar Cascade

Cascade de Haar

Entendendo como funciona:

- Dataset com imagens positivas e negativas;
- Haar features;
- Criação da imagem integral;
- Adaboost (variante);
- Implementação dos classificadores em cascata.

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Haar features:

- Características usadas para identificar padrões em imagens;
- Analisam as diferenças de intensidade entre regiões claras e escuras da imagem;



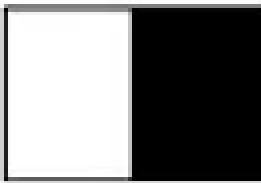
Haar Cascade

Cascade de Haar

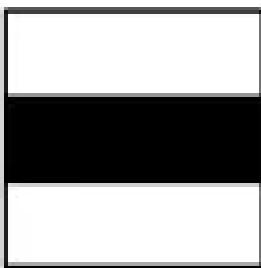
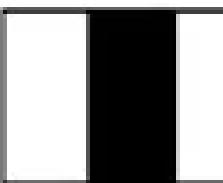
Entendendo como funciona:

Haar features:

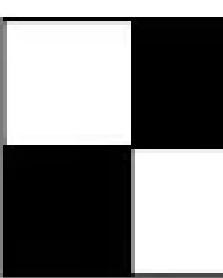
- Para extrair as Haar features, utiliza-se uma janela de 24x24 pixels que deslizará sobre a imagem e, dentro dela, serão geradas todas as combinações possíveis das features.



(a) Edge Features



(b) Line Features



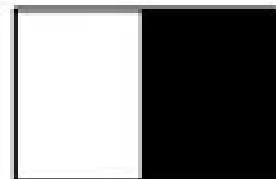
(c) Four-rectangle features

Haar Cascade

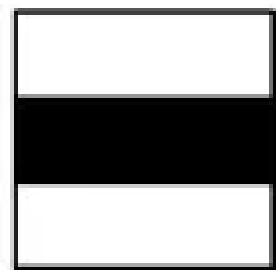
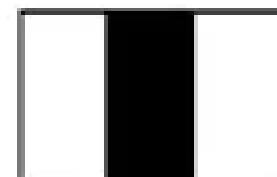
Cascade de Haar

Entendendo como funciona:

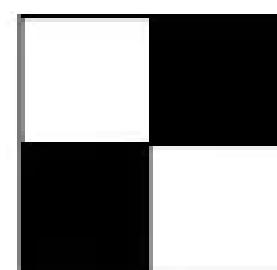
Haar features:



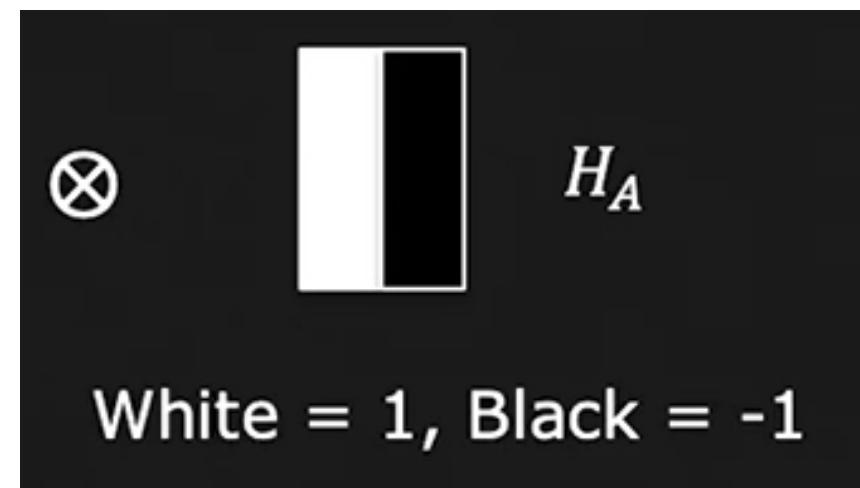
(a) Edge Features



(b) Line Features



(c) Four-rectangle features



Aplicação do filtro Ha na posição (i,j):

$$V_A[i, j] = \sum_m \sum_n I[m - i, n - j] H_A[m, n]$$

Haar Cascade

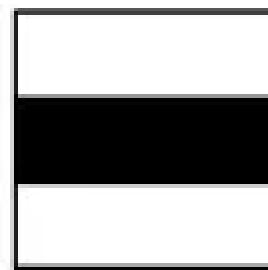
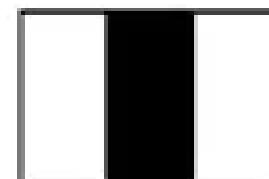
Cascade de Haar

Entendendo como funciona:

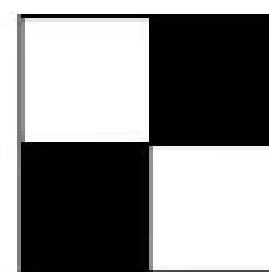
Haar features:



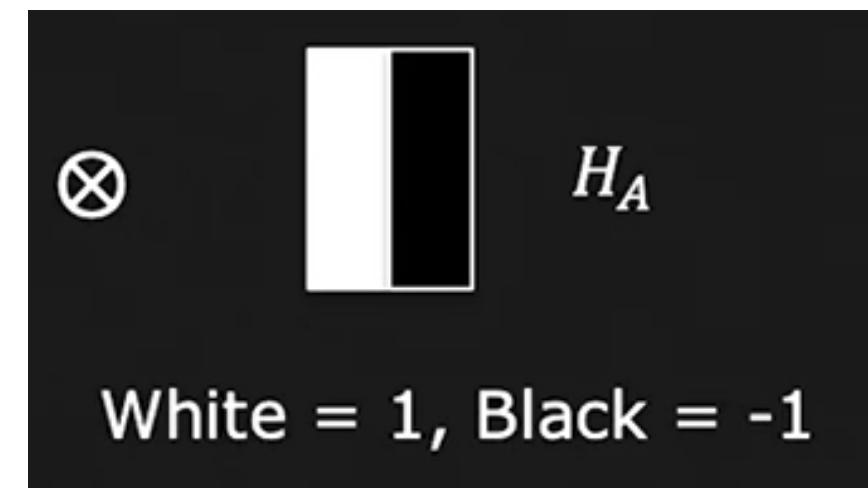
(a) Edge Features



(b) Line Features



(c) Four-rectangle features



$$V_A[i,j] = \sum (\text{pixel intensities in white area}) - \sum (\text{pixels intensities in black area})$$

Haar Cascade

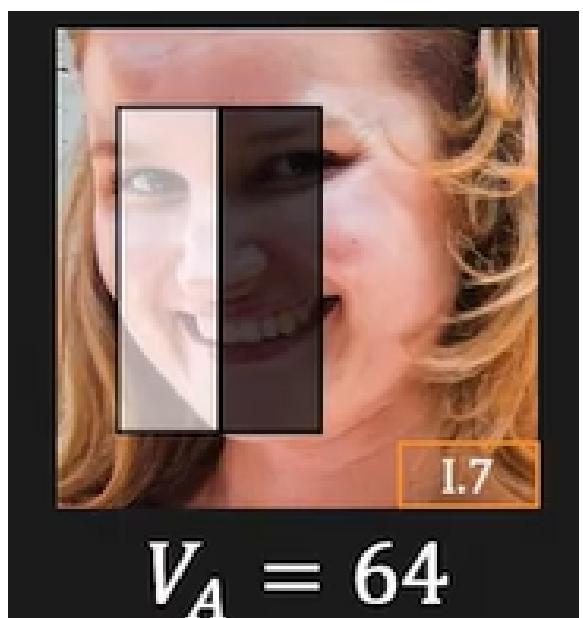
Cascade de Haar

Entendendo como funciona:

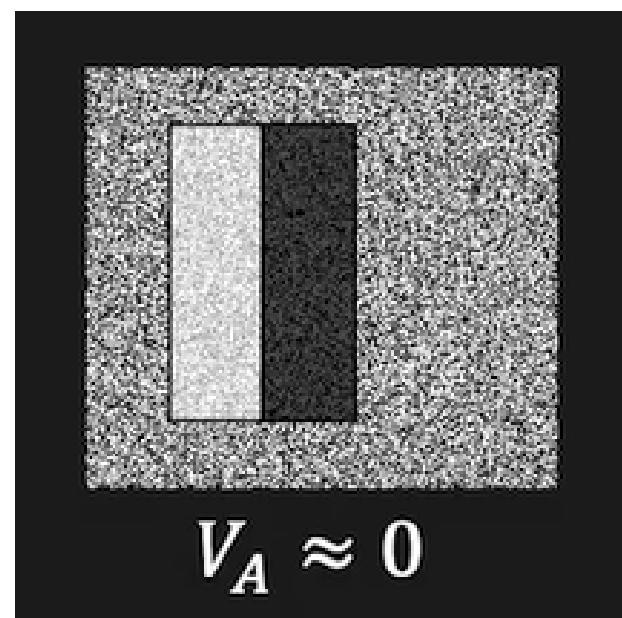
Haar features:

- Qual o objetivo?

Capturar padrões simples como bordas e os valores obtidos ajudam o treinamento dos classificadores usados nesse algoritmo.



Muitas bordas = valor do filtro alto
(positivo/negativo)



Poucas bordas = valor do filtro próximo de zero

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagen integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Imagen Integral

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagen integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Imagen Integral

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagen integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Imagen Integral

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagen integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

98	208	329	454	576	705	
197	417	658	899	1137	1395	
294	623	988	1340	1701	2093	
392	833	1330	1790	2274	2799	
489	1043	1687	2255	2864	3531	
584	1249	2061	2751	3490	4294	
680	1449	2433	3253	4118	5052	

Imagen Integral

Haar Cascade

Cascade de Haar

Entendendo como funciona:
Imagen integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Imagen Integral

$$\begin{aligned}
 \text{Sum} &= II_P - II_Q - II_S + II_R \\
 &= 3490 - 1137 - 1249 + 417 = 1521
 \end{aligned}$$

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagen integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

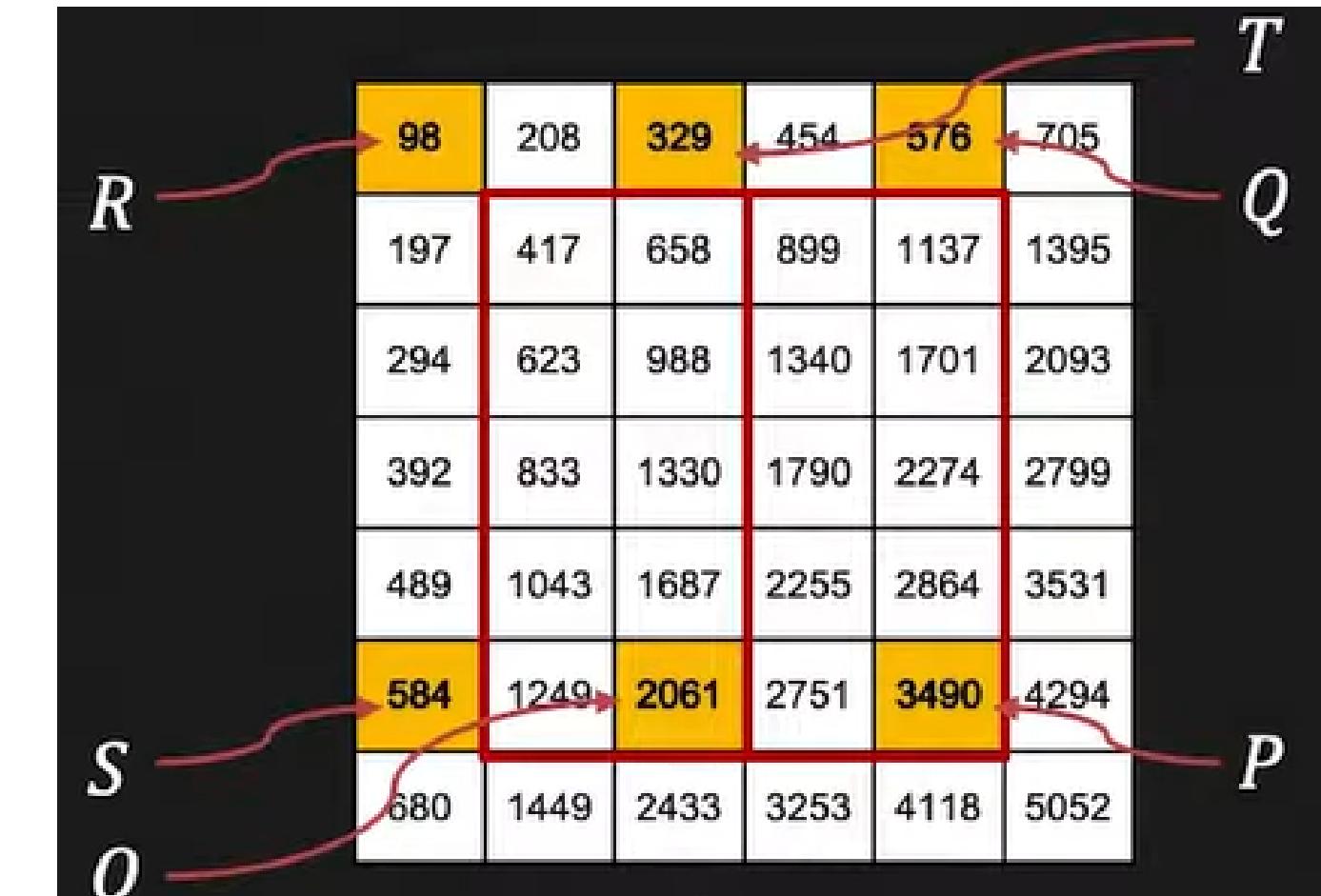


Imagen Integral

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagem integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

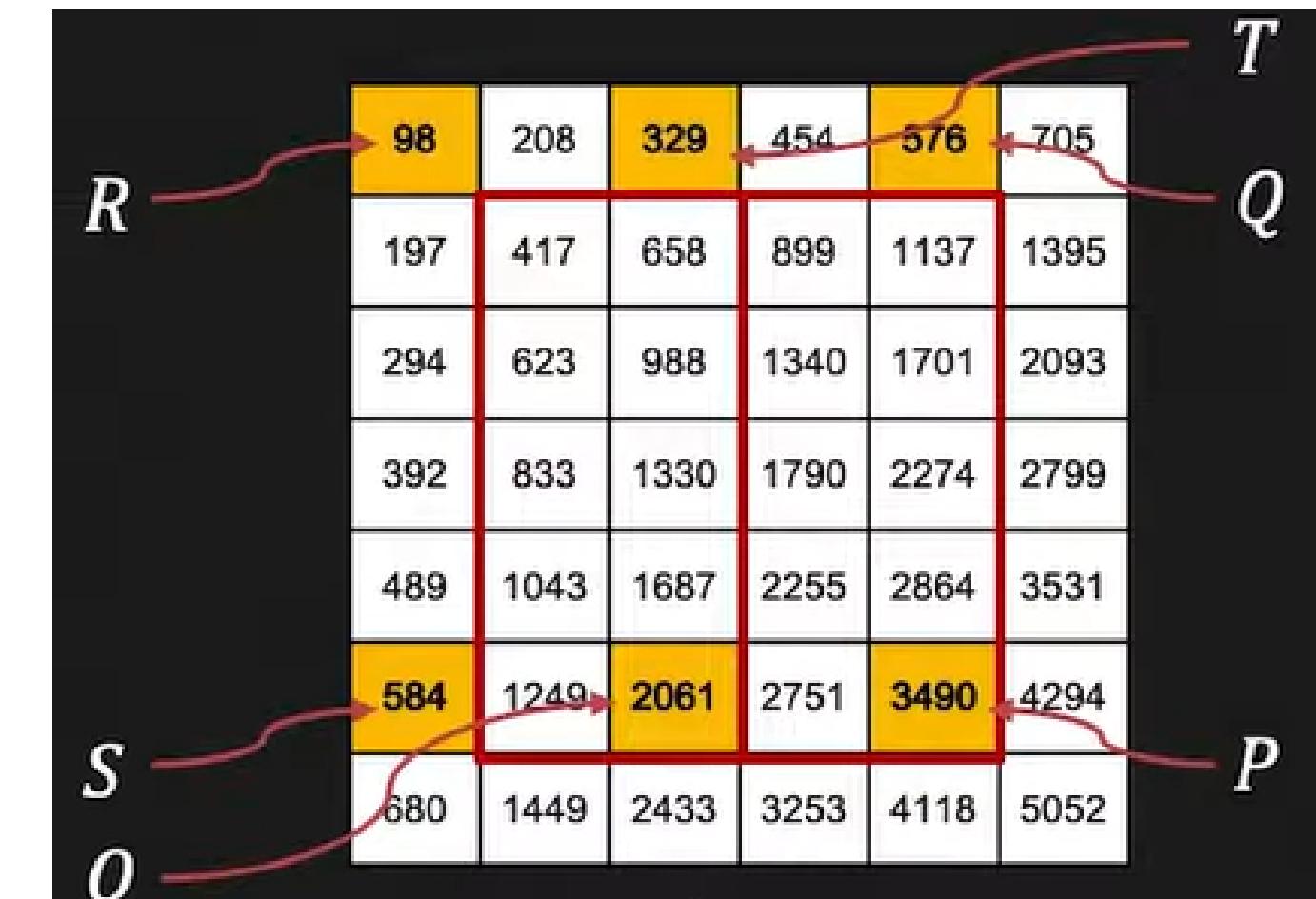


Imagen Integral

$$\begin{aligned}
 V_A &= \sum(\text{pixel intensities in white}) - \sum(\text{pixel intensities in black}) \\
 &= (II_O - II_T + II_R - II_S) - (II_P - II_Q + II_T - II_O) \\
 &= (2061 - 329 + 98 - 584) - (3490 - 576 + 329 - 2061) = 64
 \end{aligned}$$

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Imagem integral:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Imagen I

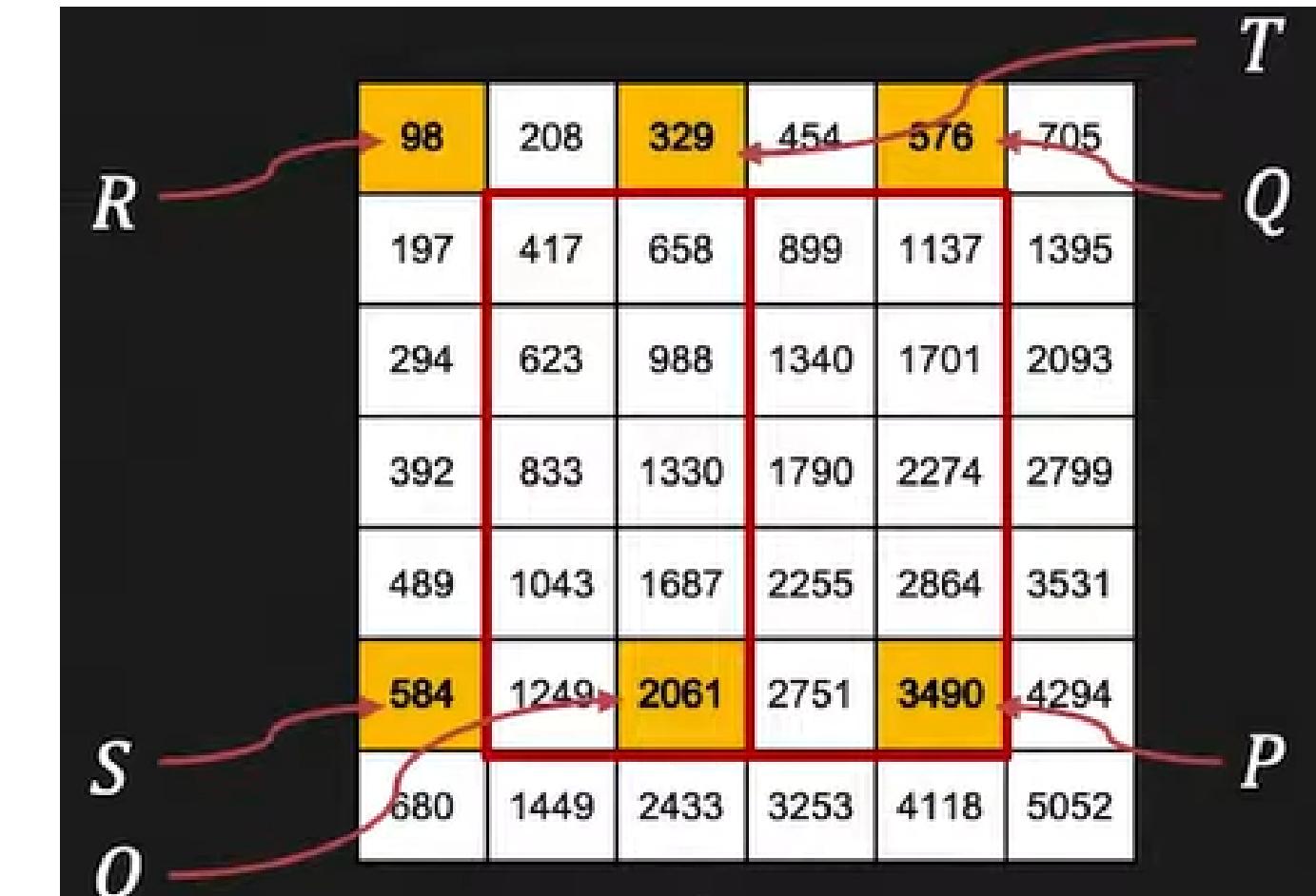


Imagen Integral

$$\begin{aligned}
 V_A &= \sum(\text{pixel intensities in white}) - \sum(\text{pixel intensities in black}) \\
 &= (II_O - II_T + II_R - II_S) - (II_P - II_Q + II_T - II_O) \\
 &= (2061 - 329 + 98 - 584) - (3490 - 576 + 329 - 2061) = 64
 \end{aligned}$$

Custo computacional
diminui bastante

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Adaboost:

As etapas de funcionamento do Adaboost são:

- Exemplos do conjunto de treinamento são selecionados e todos tem o mesmo peso.
- Após cada iteração o Adaboost altera os pesos desses exemplos, diminuindo o peso dos bem classificados e aumentando o peso dos mal classificados.
- Em cada iteração o Adaboost treina um classificador fraco com base nesses pesos.

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Adaboost:

- Ele escolhe um classificador fraco que minimiza o erro ponderado.

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right)$$

- O peso dos exemplos mal classificados aumentam, enquanto o dos bem classificados diminuem. Isso força os classificadores a focarem nos exemplos com maior peso (mais difíceis).

Haar Cascade

Cascade de Haar

Entendendo como funciona:

Adaboost:

- O Adaboost escolhe as melhores features e treina os classificadores para que eles sejam usados.
- O objetivo do Adaboost é combinar classificadores fracos em um classificador forte, que seja capaz de distinguir entre objetos e não objetos com alta precisão

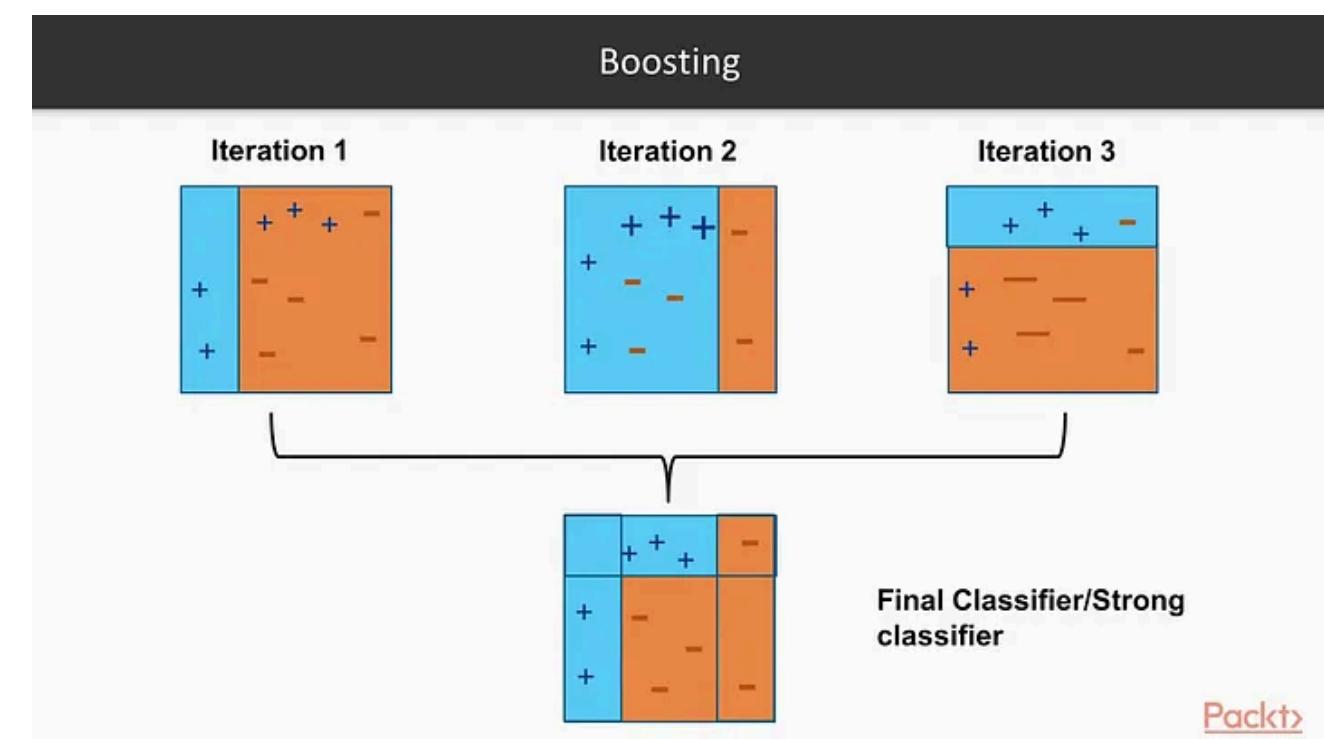
Haar Cascade

Cascade de Haar

Entendendo como funciona:

Adaboost:

- Em uma janela 24x24, normalmente usada pelo Adaboost, existem mais de 180 mil Haar features. Com isso, o Adaboost seleciona as features mais relevantes, ou seja, durante o treinamento ele escolhe as features que mais contribuem com a classificação.



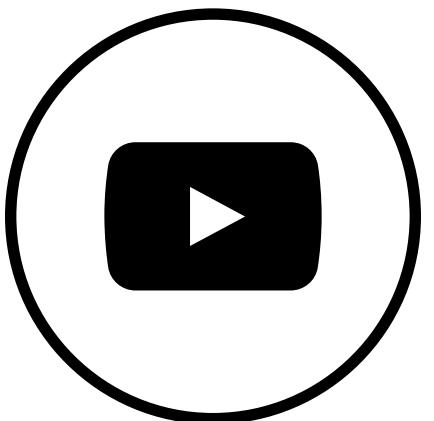
Haar Cascade

Cascade de Haar

Entendendo como funciona:

Classificadores em cascata:

- Os classificadores desenvolvidos pelo Adaboost agora são usados nas imagens com o objetivo de detectar o objeto.
- Os classificadores fracos são treinados de modo que, ao passar pela imagem, descarte locais onde não há nenhum indício da imagem. Aumentando a velocidade do processamento.



Haar Cascade

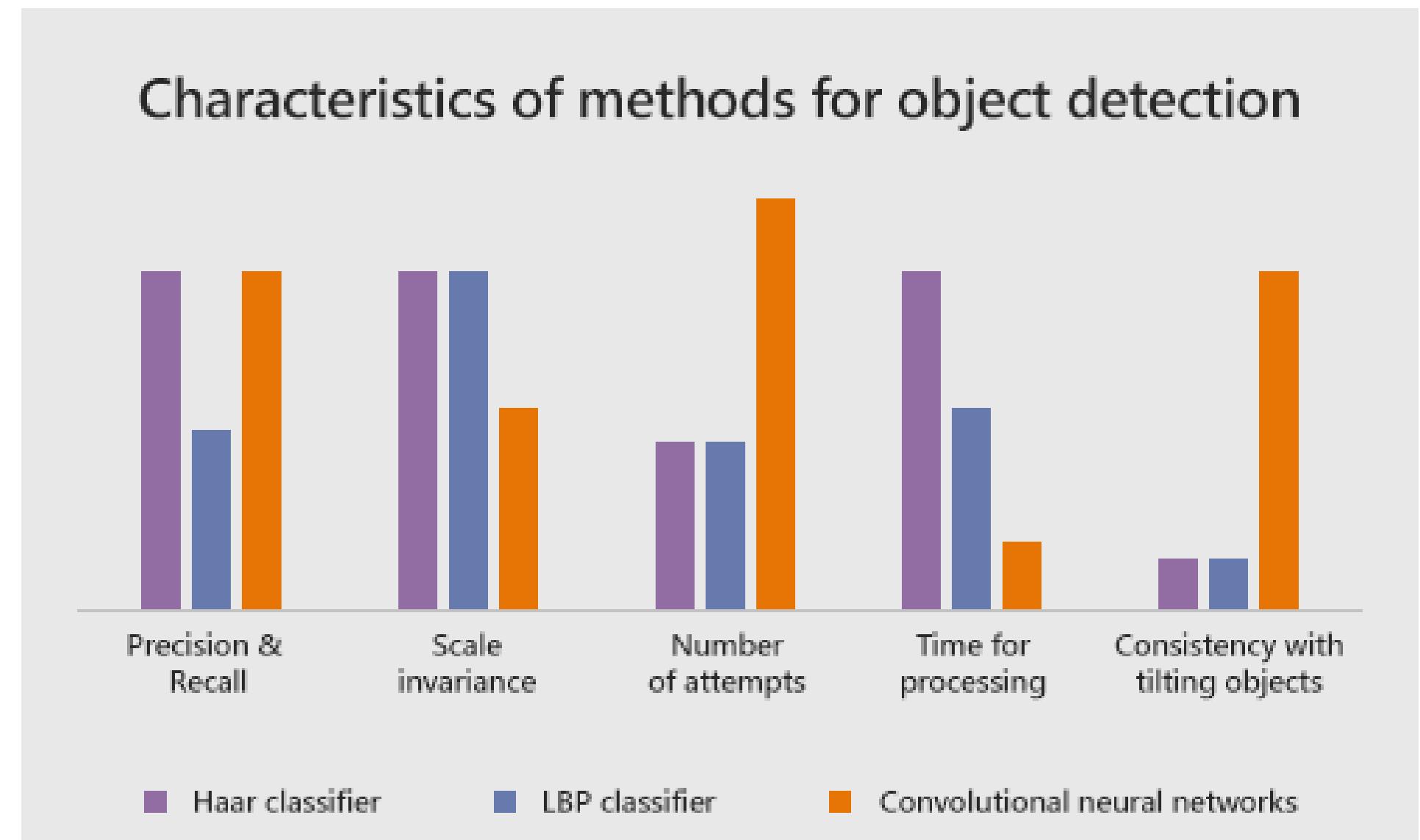
Cascade de Haar

Entendendo como funciona:

Vantagens:

- Eficiência computacional;
- Simplicidade (amplamente documentado);
- Baixo consumo de recursos;
- Robustez em cenários simples (padrões simples e contrastes previsíveis);
- Integração simples (diferentes tipos de dados).

Haar Cascade X CNN



SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

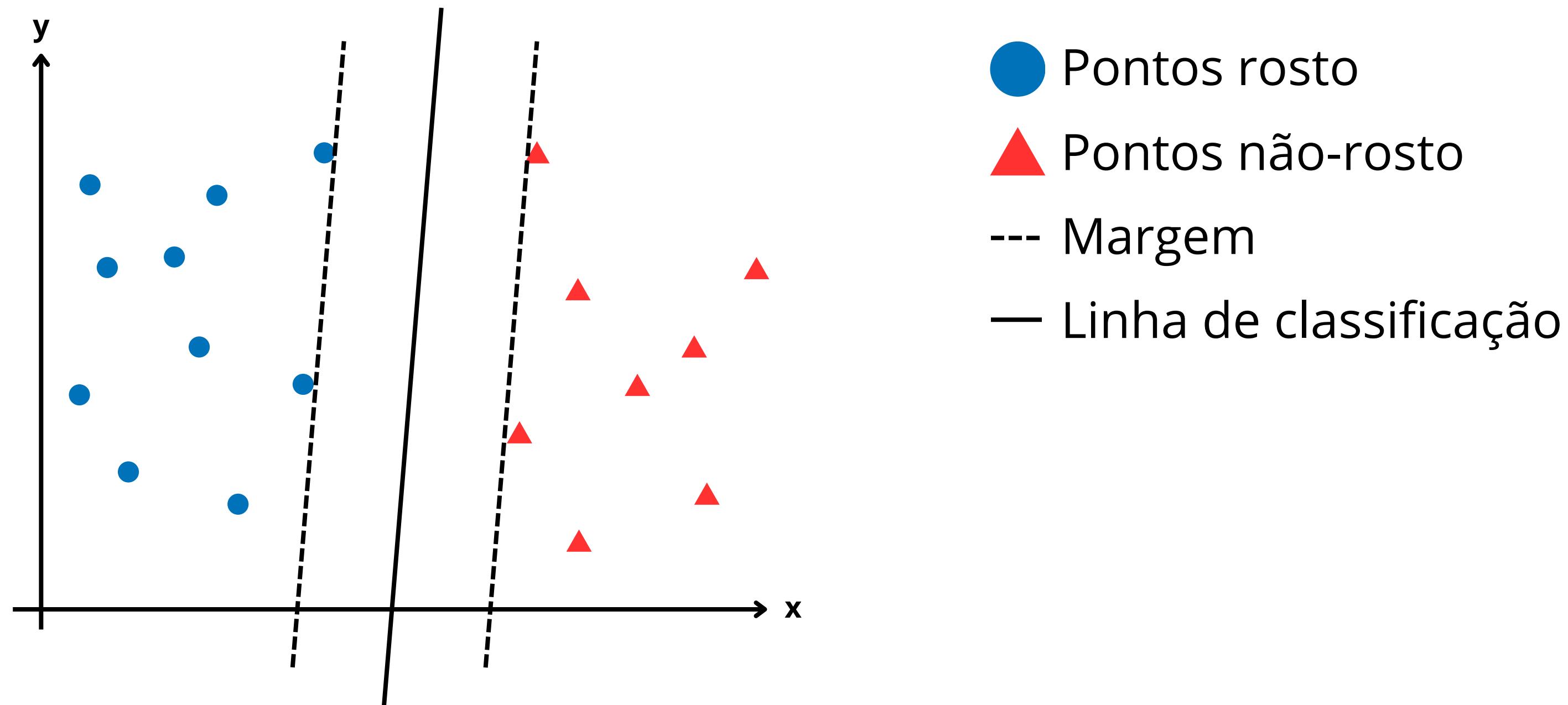
História:

- Foi desenvolvida em 1992 por um grupo de pesquisa da AT&T Bell Laboratories
- Conhecida inicialmente como “algoritmo de treinamento para classificadores de margens ótimas”
- Em 1993 o mesmo grupo demonstrou que os classificadores podem ser treinados com poucos dados, os vetores de suporte.

SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

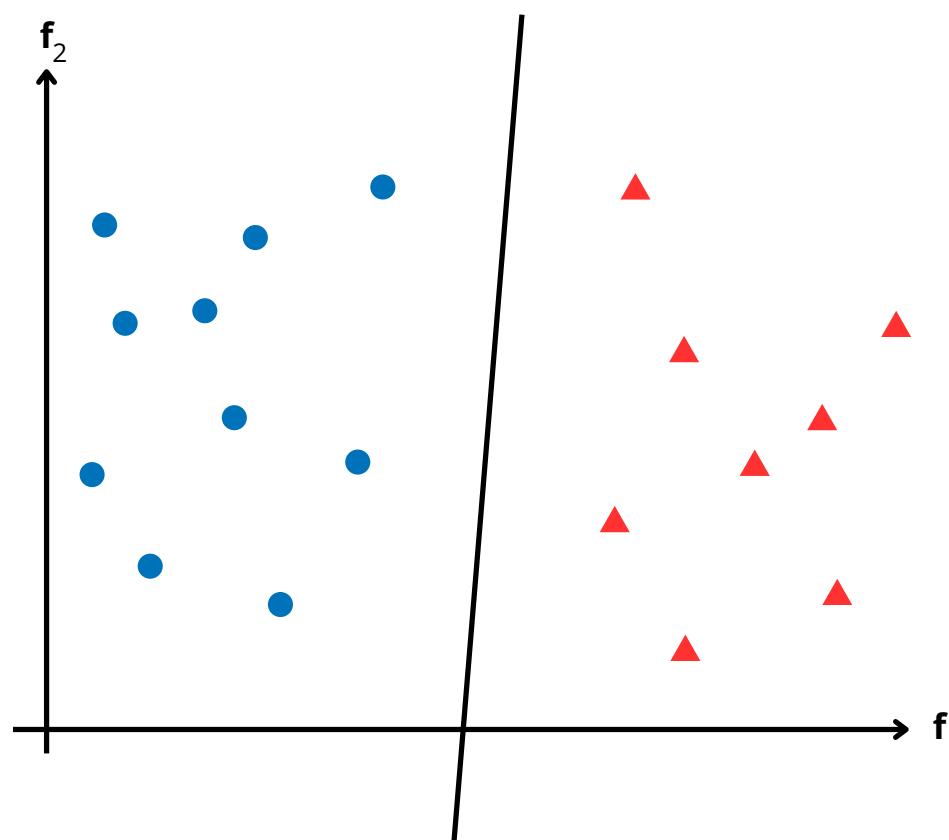
Entendendo como funciona:



SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

Entendendo como funciona:
Equação da reta



$$w_1 f_1 + w_2 f_2 + b = 0$$

$$[w_1 \ w_2] \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + b = 0$$

Na forma vetorial

$$w^T f + b = 0$$

$$sgn(d(w, x, b)) = \begin{cases} +1 & se w \cdot x + b > 0 \\ -1 & se w \cdot x + b < 0 \end{cases}$$

SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

**Entendendo como funciona:
Equações para planos tridimensionais e hiperplanos**

$$w_1 f_1 + w_2 f_2 + w_3 f_3 + b = 0$$

3 dimensões

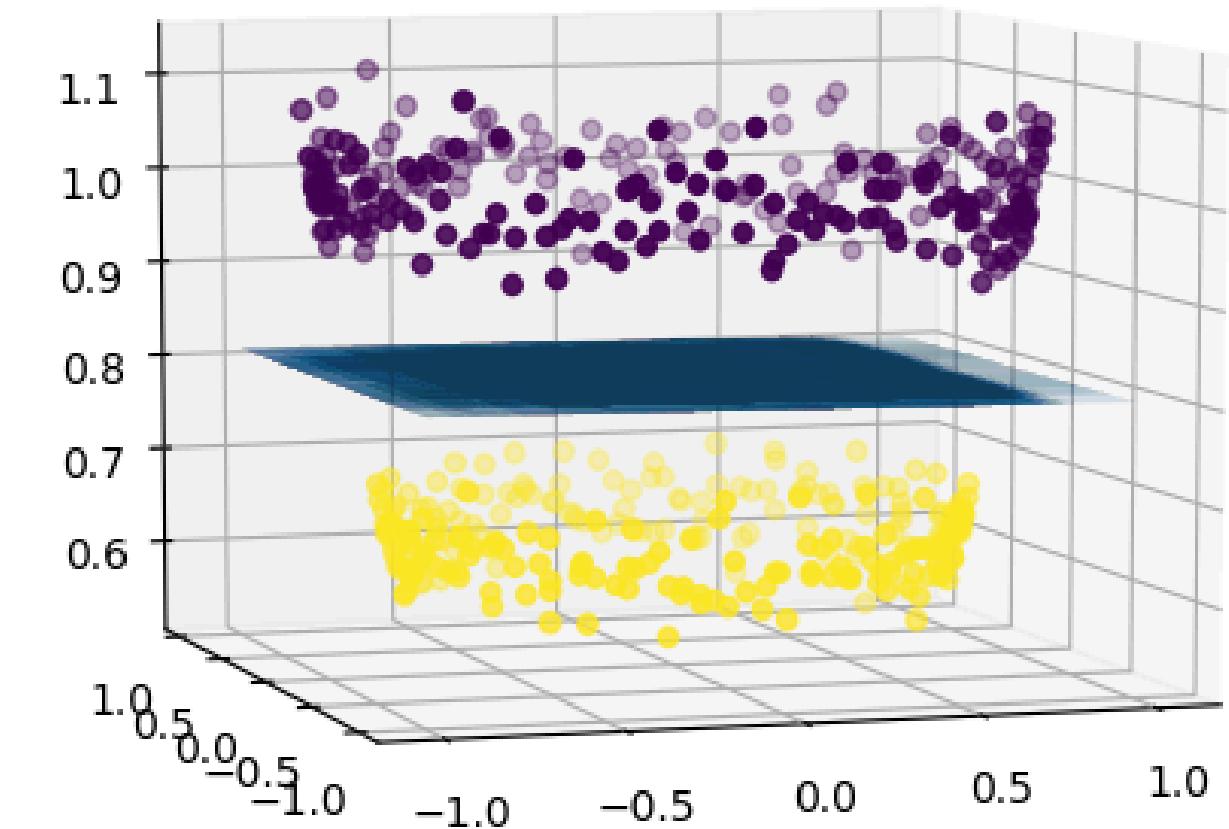
$$w^T f + b = 0$$

$$w_1 f_1 + w_2 f_2 + \dots + w_n f_n + b = 0$$

n dimensões

$$w^T f + b = 0$$

$$sgn(d(w, x, b)) = \begin{cases} +1 & se w \cdot x + b > 0 \\ -1 & se w \cdot x + b < 0 \end{cases}$$

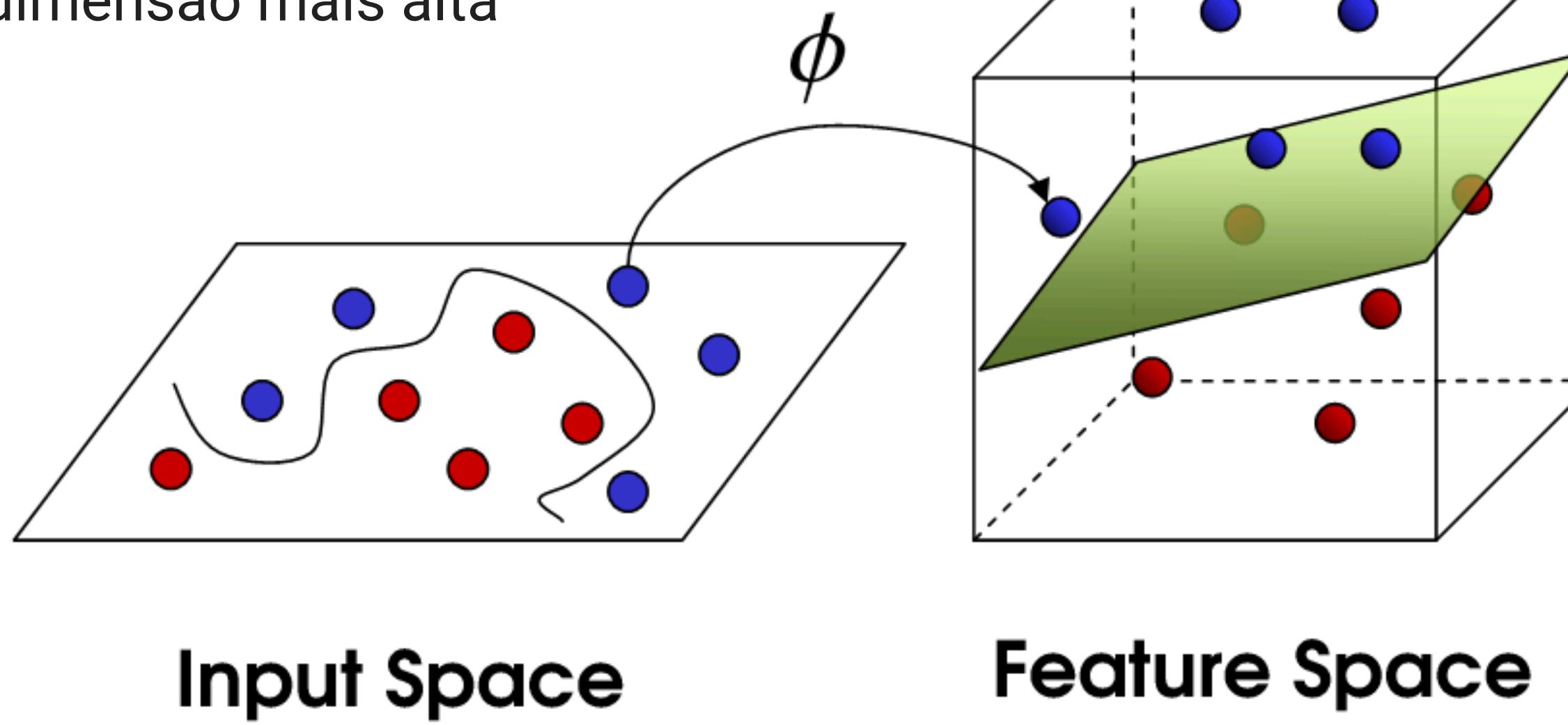


SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

Entendendo como funciona:
Kernel

“Transformar o espaço de entrada em um espaço de dimensão mais alta”



SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

Entendendo como funciona: Kernel

Uma função que recebe dois pontos (vetores de suporte), do espaço de características original e retorna o seu produto escalar no espaço de características de dimensões mais elevadas.

$$k(\vec{w}, \vec{x}) = \Phi(\vec{w}) \cdot \Phi(\vec{x})$$

$\Phi(\vec{x})$ representa a função de mapeamento utilizada

SVM (Support Vector Machine)

Máquinas de Vetores de Suporte

Entendendo como funciona: Kernel

Funções de kernel	
Kernel	Função de mapeamento Φ
Linear	$\vec{w} \cdot \vec{x}$
Polinomial	$(gama * \vec{w} \cdot \vec{x} + b)^{grau}$
Base Radial	$e^{-gama \vec{w} - \vec{x} ^2}$
Sigmoidal	$\tanh(gama * \vec{w} \cdot \vec{x} + b)$

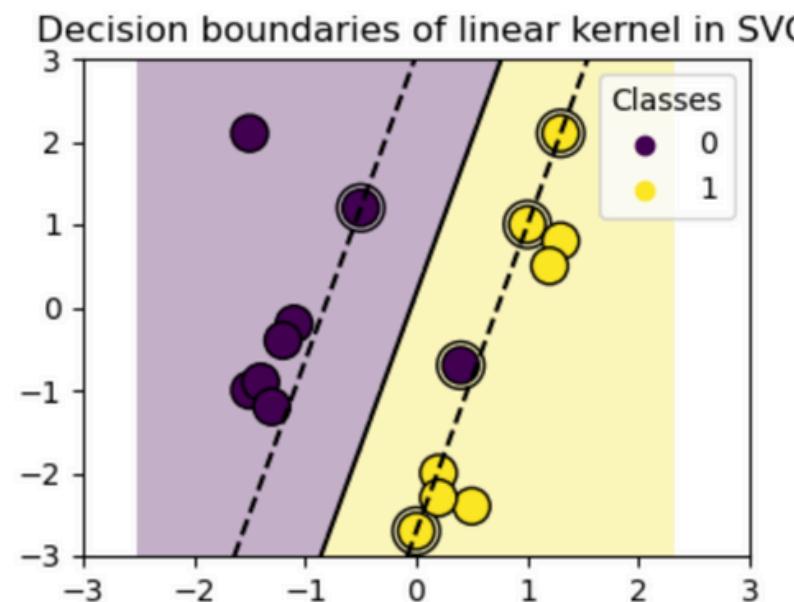
$gamma$ e b , correspondem respectivamente ao coeficiente angular e o deslocamento linear

SVM (Support Vector Machine)

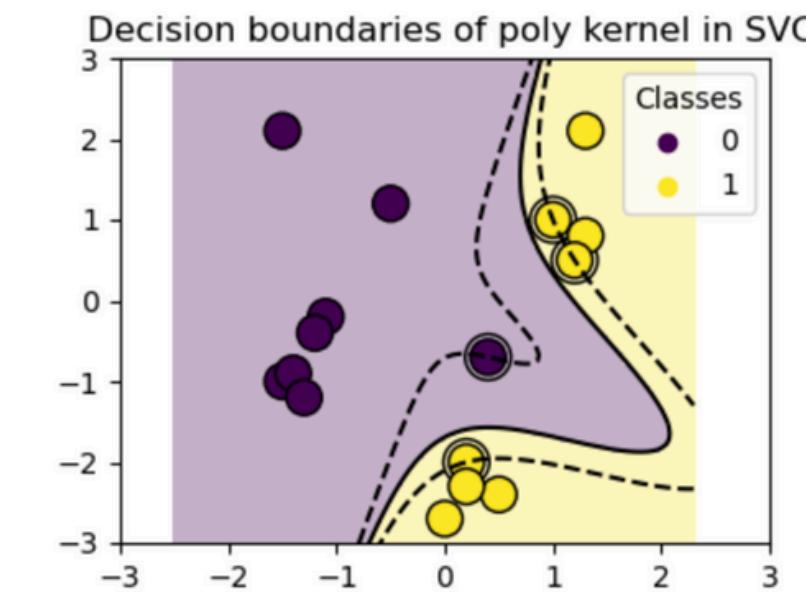
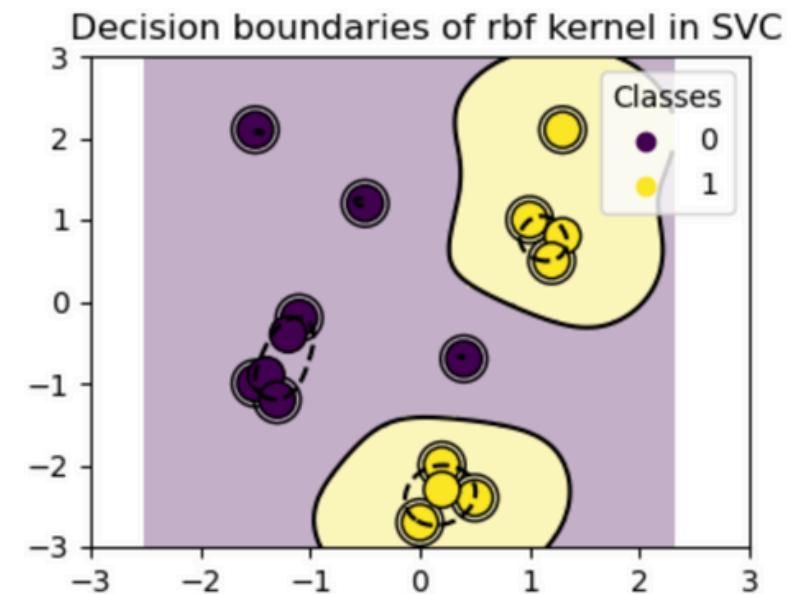
Máquinas de Vetores de Suporte

Entendendo como funciona: Kernel

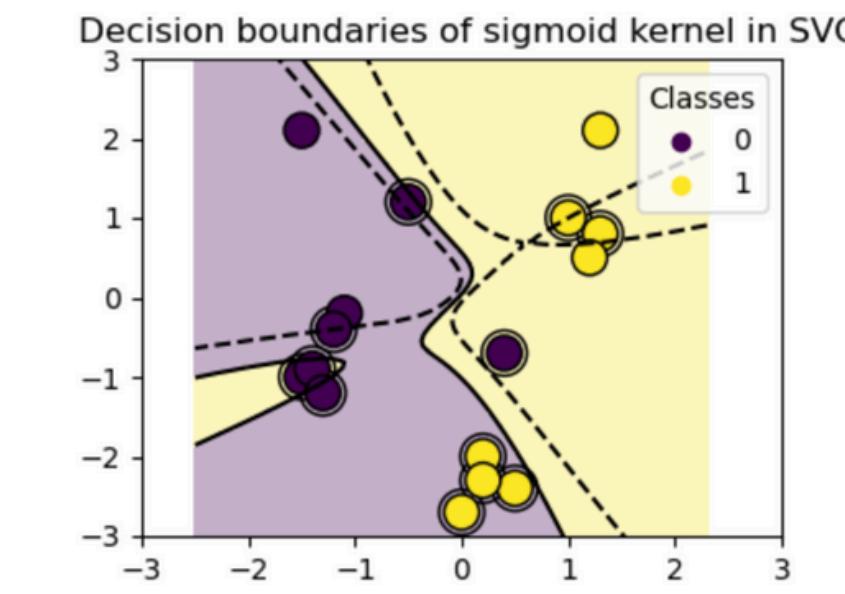
Ex: Identificar se
é parede branca
ou não



Ex: comparar
rostos



Ex: Detecção de
emoções em
rostos



Transição entre os clássicos para os modernos: 2000 - 2012

- Explosão de Dados
- Avanços nos hardwares. Uso de GPUs no treinamento de redes
- Aperfeiçoamento de Deep Learning
- Redes neurais modernas já começaram a se consolidar com acuráciais melhores, comparado à algoritmos clássicos

Algoritmos Modernos

Características e diferenças dos algoritmos clássicos

Algoritmos Clássicos

- Baseado em regras e similaridade
- Baixa adaptabilidade a dados não estruturados
- Desempenho limitado em dados grandes e complexos
- Escalabilidade reduzida para grandes volumes de dados

Algoritmos Modernos

- Automação na extração de features
- Modelo hierárquico de aprendizado
- Especializadas em visão computacional (CNN)
- Reduzem a dimensionalidade mantendo informações importantes (CNN)

Algoritmos Modernos

Características e diferenças dos algoritmos clássicos

Algoritmos Clássicos

- Complexidade computacional baixa a moderada
- Tendência alta a overfitting
- Capacidade limitada com dados complexos
- Desempenho ruim em dados não estruturados

Algoritmos Modernos

- Complexidade computacional alta
- Tendência moderada a overfitting
- Boa capacidade de generalização em dados grandes e complexos
- Desempenho excelente em dados não estruturados

Algoritmos Modernos

- Os dados atuais são complexos e, normalmente, sistemas trabalham com uma grande quantidade deles
- CNNs e MLPs dominam esses tipos de dados, deixando para trás os algoritmos clássicos
- Redes neurais modernas já começaram a se consolidar com acurárias melhores, comparado à algoritmos clássicos

Obrigado!

Dúvidas ou sugestões:

joaopedro234567@discente.ufg.br

brunocalura@discente.ufg.br

cleiver@discente.ufg.br

INF
INSTITUTO DE
INFORMÁTICA

