Machine Learning com Python

Hugo Padovani
Data Scientist
Cetax Consultoria

O que é Machine Learning?

É o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados para tal - Arthur Samuel, 1959.

O aprendizado automático explora o estudo e a construção de algoritmos que podem aprender de seus erros e fazer previsões sobre dados.

Porque aprender Machine Learning

- Área de conhecimento interdisciplinar que vem crescendo exponencialmente nos últimos anos.
- Possui uma infinidade de aplicações práticas conforme veremos ao longo do curso;
- Abordagem Hands-On: pode-se aprender ML sem ser necessário conhecer profundamente toda a matemática por trás (se você souber, mais rápido será o entendimento).
- Campo no mercado e em pesquisa acadêmica.

Tipos de Algoritmos de Machine Learning

- Aprendizado Supervisionado;

- Aprendizado Não-Supervisionado;

- Aprendizado por Reforço

Aprendizado Supervisionado

Ocorre quando existem labels (rótulos) para seus dados. Esses rótulos são considerados as saídas corretas, e o algoritmo deve calcular o erro entre as saídas esperadas e as saídas calculadas.

Exemplos: Predição de preços de casas com base na área (contínuo, regressão), classificação de classes (discreto, classificação).

Aprendizado Não Supervisionado

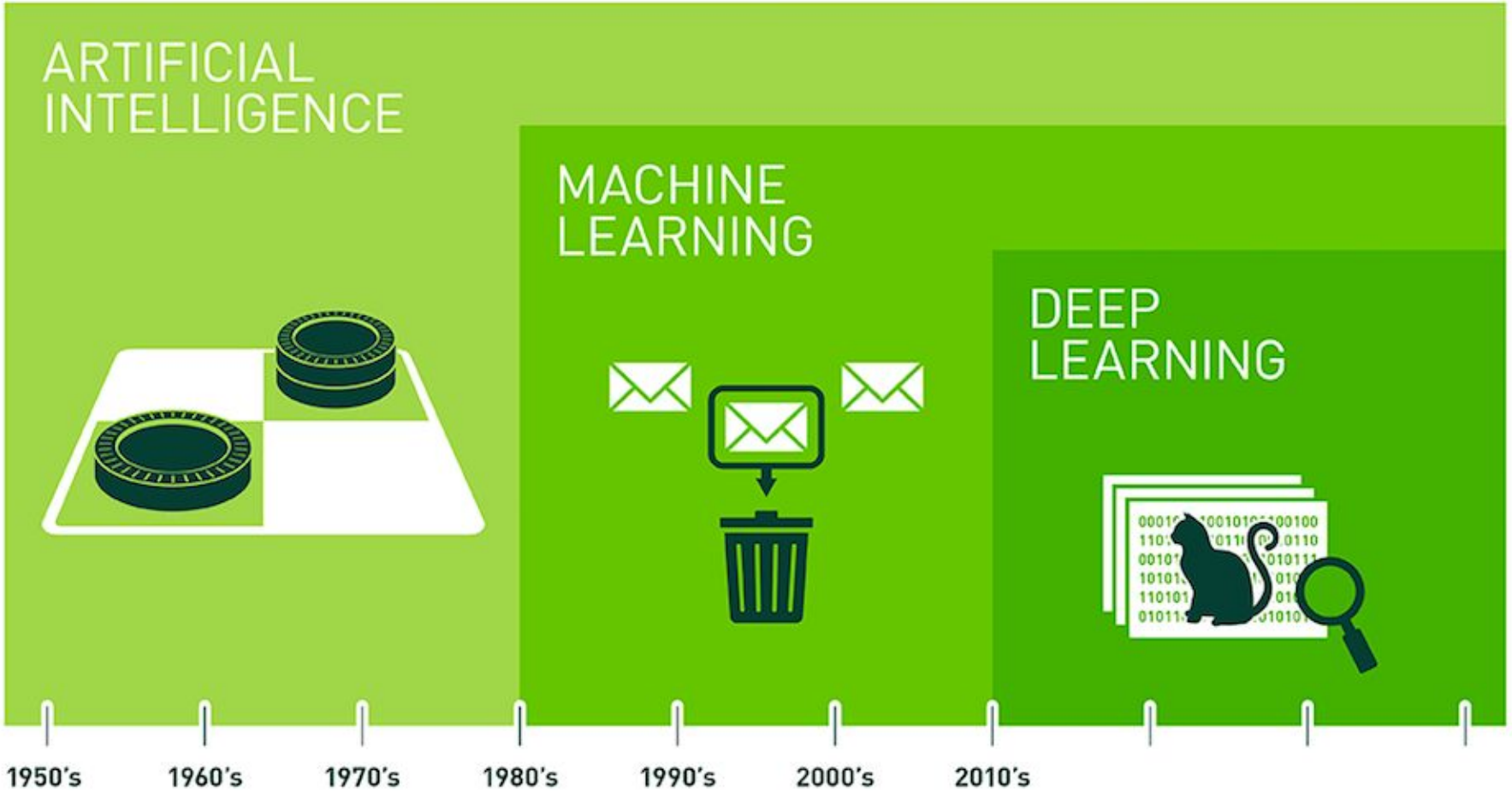
Diferente de aprendizado supervisionado, neste caso não existem rótulos, e é responsabilidade do algoritmo encontrar padrões e estruturas entre os dados.

Exemplos: Clusterização (formação de conjuntos de dados, agrupamentos), e PCA (Principal Component Analysis, redução de dimensionalidade).

Aprendizado por Reforço

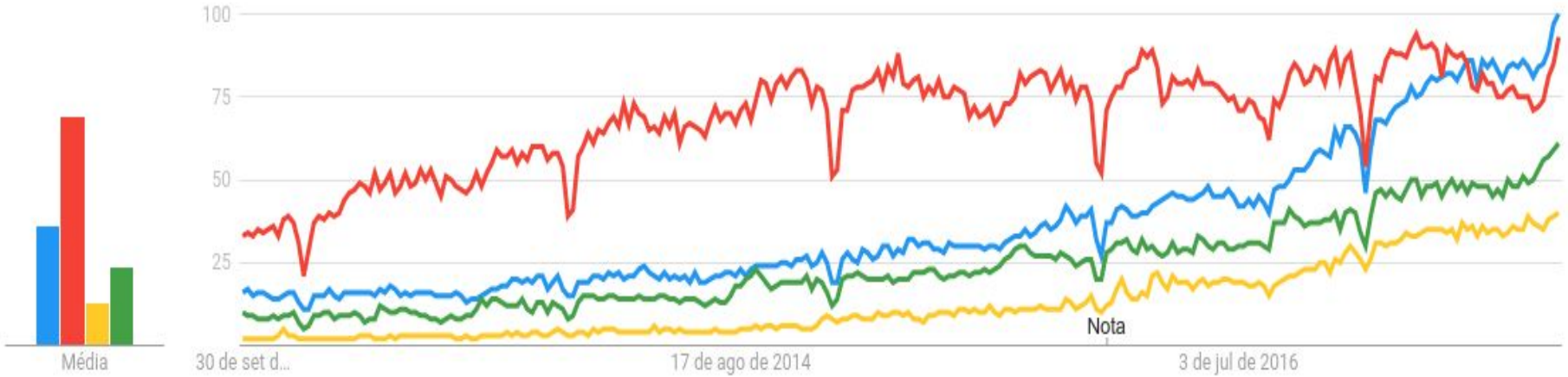
Situação em que o algoritmo interage com o ambiente dinâmico, onde deve desempenhar um determinado objetivo específico. É baseado em premiação e punição, ocorrendo quando o algoritmo desempenha um resultado esperado, ou inesperado.

Exemplos: Veículos autônomos, inteligência artificial em games.


ARTIFICIAL INTELLIGENCE
MACHINE LEARNING
DEEP LEARNING
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Google Trends - 30/09/2012 - 30/09/2017

Big Data, Machine Learning, Data Science, Deep Learning



O que veremos neste curso

0 - Revisão básica de Python

1 - Pré-processamento de dados

- Importar bibliotecas e datasets;
- Tratar missing data, categorical data, feature scaling;
- Divisão entre conjunto de treinamento e testes.

2 - Aprendizado Supervisionado

- Regressões: Linear Simples, Múltipla, Polinomial, Support Vectors, Decision Trees, Random Forest.

O que veremos neste curso

- Classificação: Regressão Logística, KNN, SVM, Kernel SVM, Decision Trees, Random Forest.
- Deep Learning: Redes Neurais Artificiais, Redes Neurais Convolucionais.

3 - Aprendizado Não Supervisionado

- Clustering: K-Means, Hierarchical Clustering.
- Redução de Dimensionalidade: PCA e LDA.

4 - Natural Language Processing;

5 - Grid Search, K-fold Cross Validation e Gradient Boosting

6 - Desafio

Setup

Os códigos serão feitos em Python 3.x, no Jupyter Notebook e Spyder (IDEs Python).

Instalações do pacote Anaconda com as bibliotecas sklearn, keras, tensorflow, statsmodel, matplotlib, numpy, pandas.

Os materiais deste curso serão disponibilizados em:

www.github.com/hgpadovani/Curso-ML

Hora de começarmos, aproveitem :D

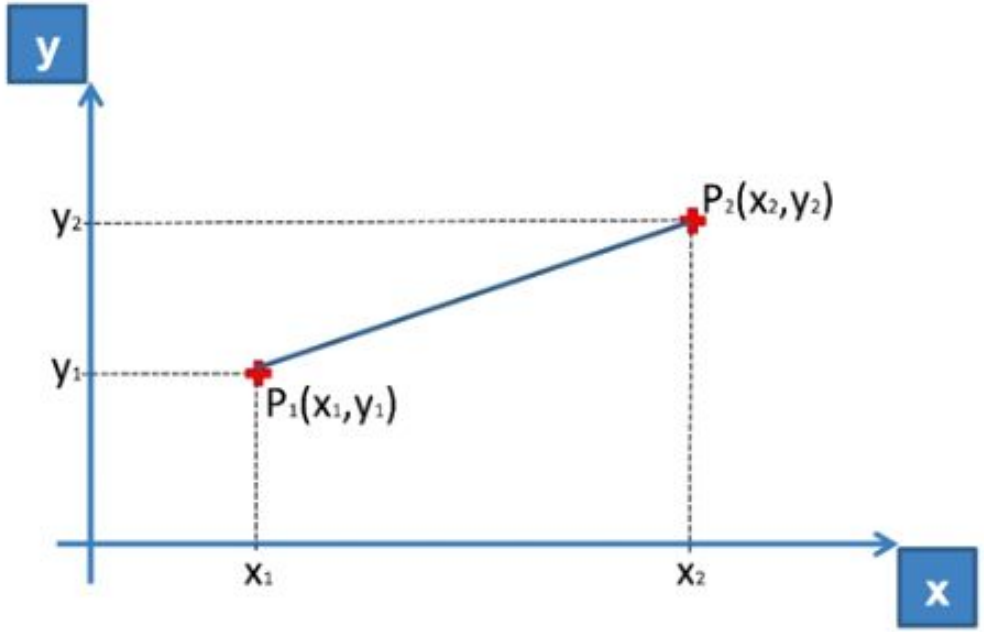
Data Preprocessing

Data Preprocessing - Dados Categóricos

Country
France
Spain
Germany
Spain
Germany
France
Spain
France
Germany
France

France	Germany	Spain
1	0	0
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0
0	0	1
1	0	0
0	1	0
1	0	0

Data Preprocessing - Feature Scaling



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

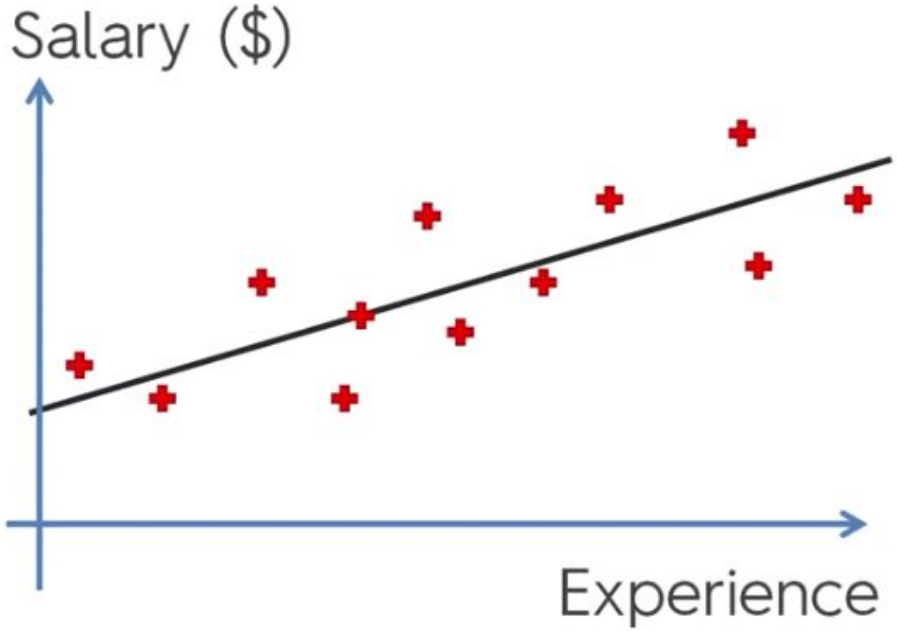
Data Preprocessing - Feature Scaling

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

Aprendizado Supervisionado
Regressão

Aprendizado Supervisionado - Regressão Linear

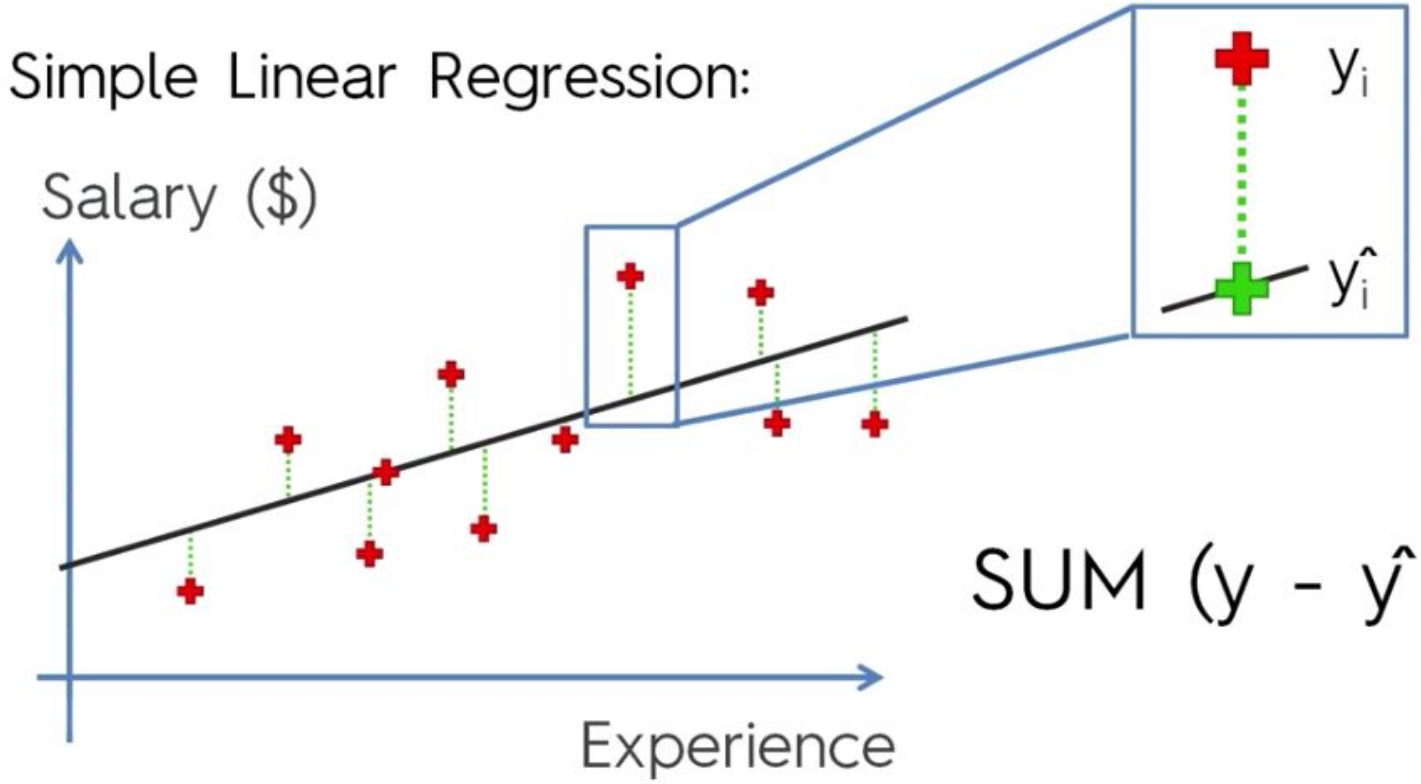
Simple Linear Regression:



$$y = b_0 + b_1 * x$$

$$\text{Salary} = b_0 + b_1 * \text{Experience}$$

Aprendizado Supervisionado - Regressão Linear



$$SUM\ (y - \hat{y})^2 \rightarrow min$$

Aprendizado Supervisionado - RL Múltipla

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State	New York	California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70	151,377.59	443,898.53	California	0	1
191,050.39	153,441.51	101,145.55	407,934.54	California	0	1
182,901.99	144,372.41	118,671.85	383,199.62	New York	1	0
166,187.94	142,107.34	91,391.77	366,168.42	California	0	1

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D_1 + b_5 * D_2$$

Aprendizado Supervisionado - Dummy Variable Trap

Profit	R&D Spend	Admin	Marketing	State	Dummy Variables: New York	Dummy Variables: California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70			ia	0	1
191,050.39	153,441.51			ia	0	1
182,901.99	144,372.41			rk	1	0
166,187.94	142,107.34			ia	0	1

$$D_2 = 1 - D_1$$

$$y = b_0 + b_1*x_1 + b_2*x_2 + b_3*x_3 + b_4*D_1 + b_5*D_2$$

Aprendizado Supervisionado - Dummy Variable Trap

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State	New York	California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70	151,377.59	443,898.53	California	0	1
191,050.39	153,441.51	101,145.55	407,934.54	California	0	1
182,901.99	144,372.41	118,671.85	383,199.62	New York	1	0
166,187.94	142,107.34	91,391.77	366,168.42	California	0	1

$$y = b_0 + b_1*x_1 + b_2*x_2 + b_3*x_3 + b_4*D_1 + b_5*D_2$$

Always omit one dummy variable

Construindo seu Modelo - Selecionando Features

5 métodos mais comuns:

- All-in;
- Backward Elimination;
- Forward Selection;
- Bidirectional Elimination;
- Score Comparison.

Construindo seu Modelo - Backward Elimination

Passo a Passo:

1 - Selecionar um nível de significância (significance level, SL) para permanecer no modelo (normalmente 0.05);

2 - All-in;

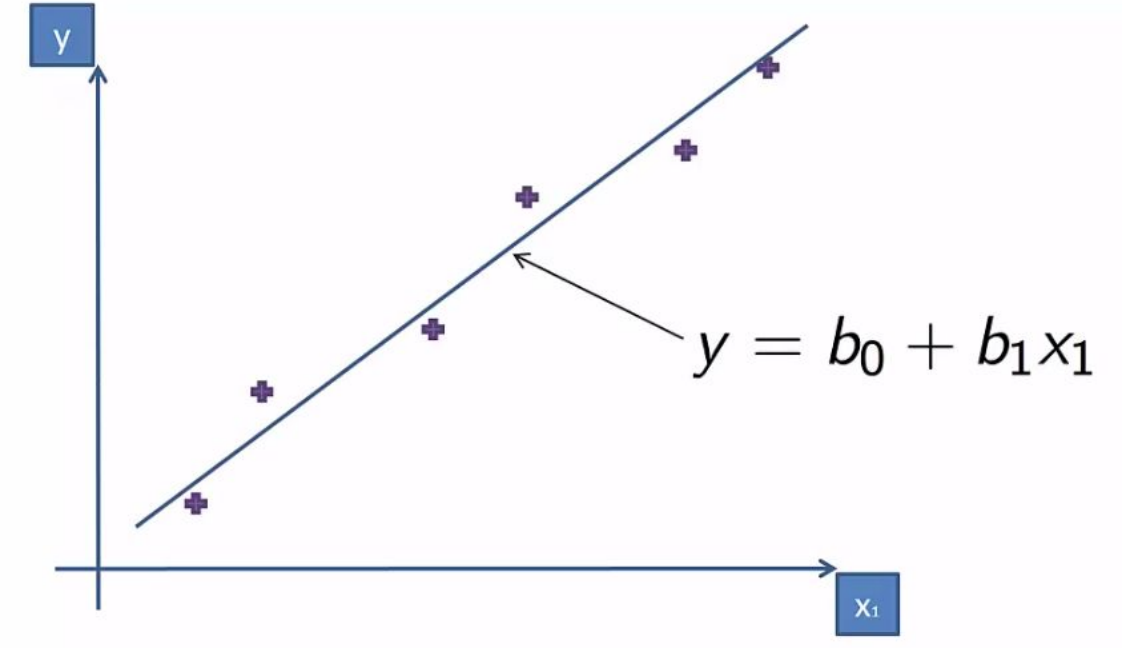
3 - Considere a feature com maior valor P. Se P > SL, remova a feature, cc FIM;

4 - Treine seu modelo sem a feature, volte para 3;

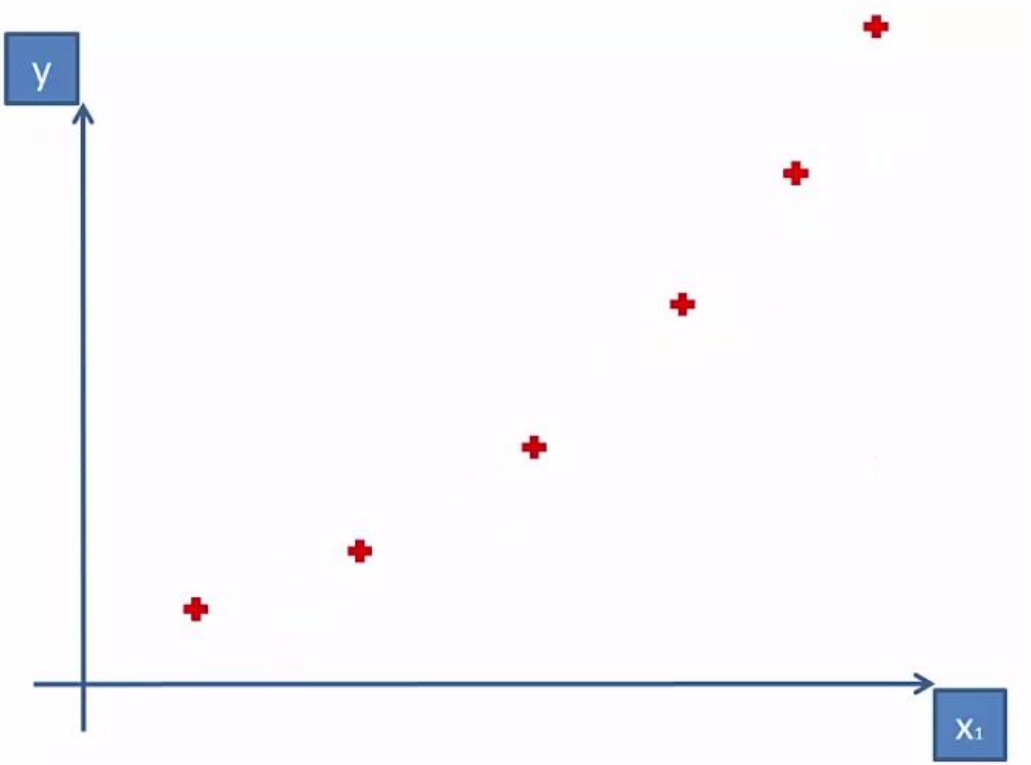
FIM - Seu modelo está pronto.

Regressão Linear Polinomial

Relembrando: Regressão Linear



Regressão Linear Polinomial

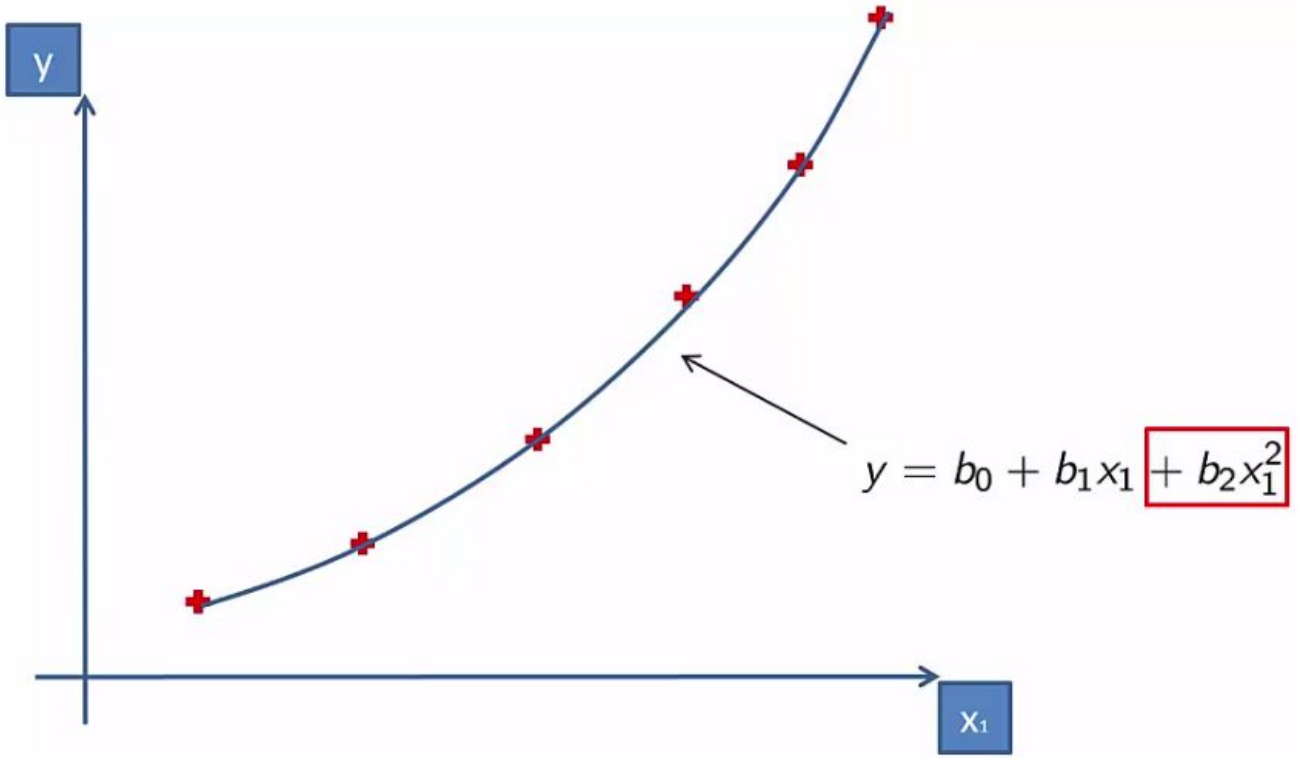


O que fazer para este caso?

Será que uma Regressão Linear Simples é capaz de representar estes pontos?

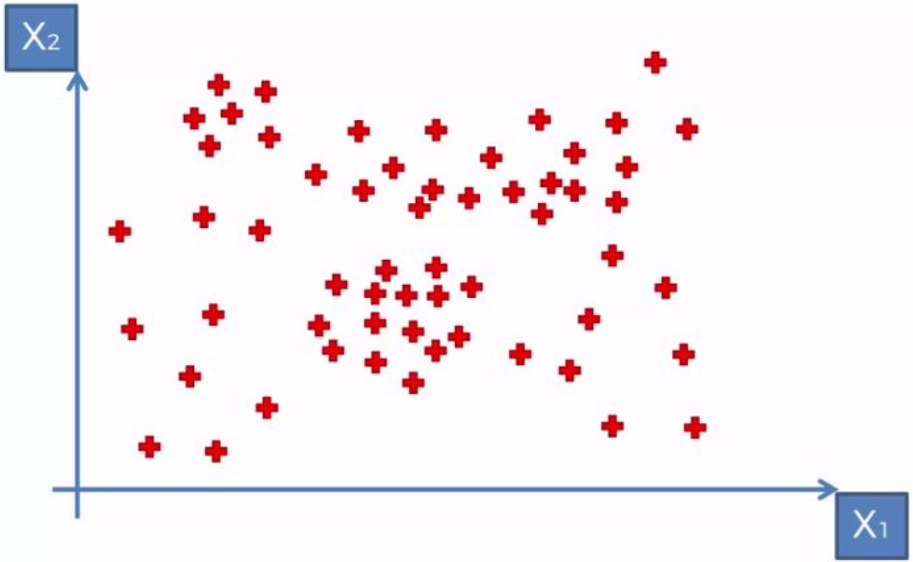
Regressão Linear Polinomial

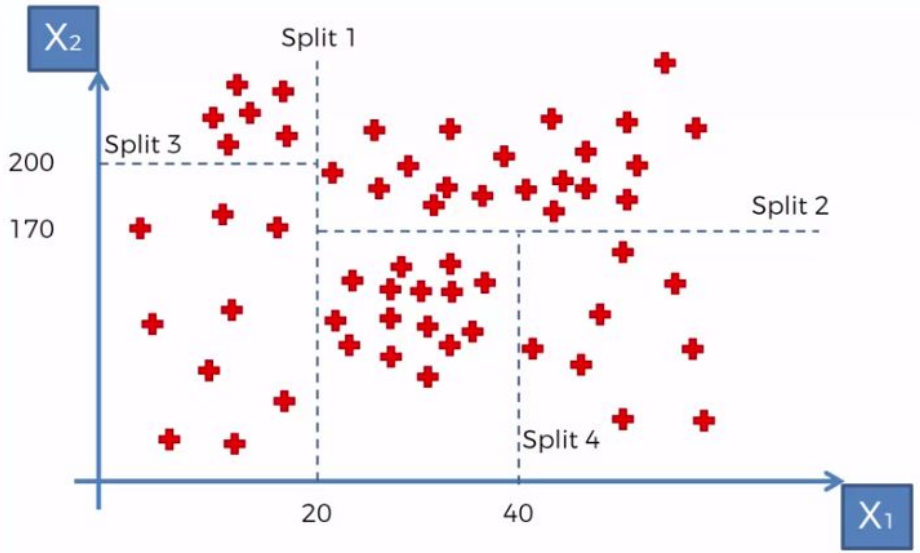
$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$



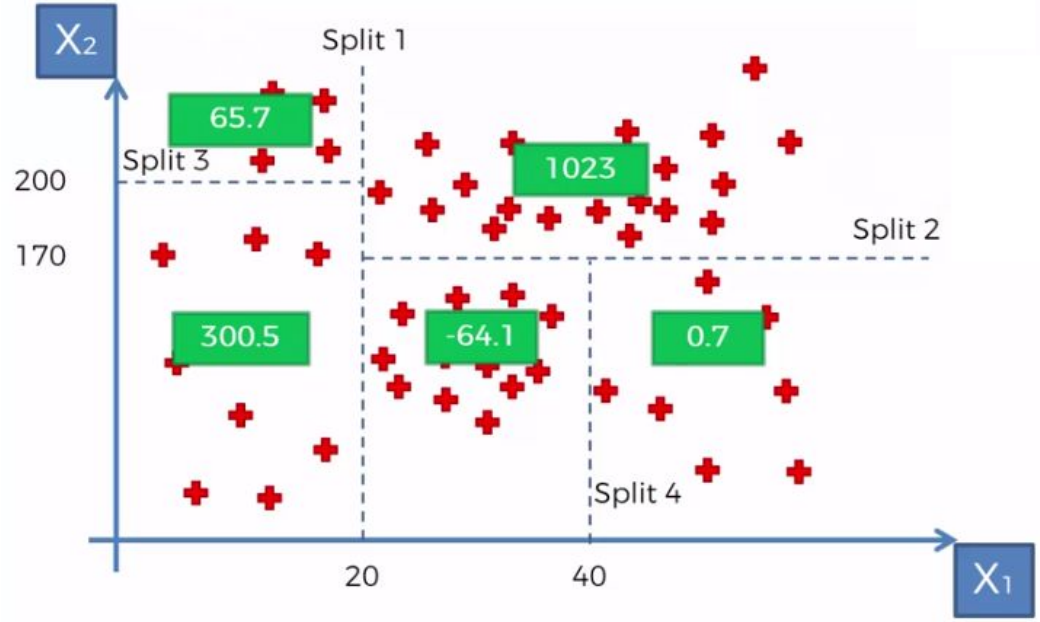
Adicionar o termo quadrático

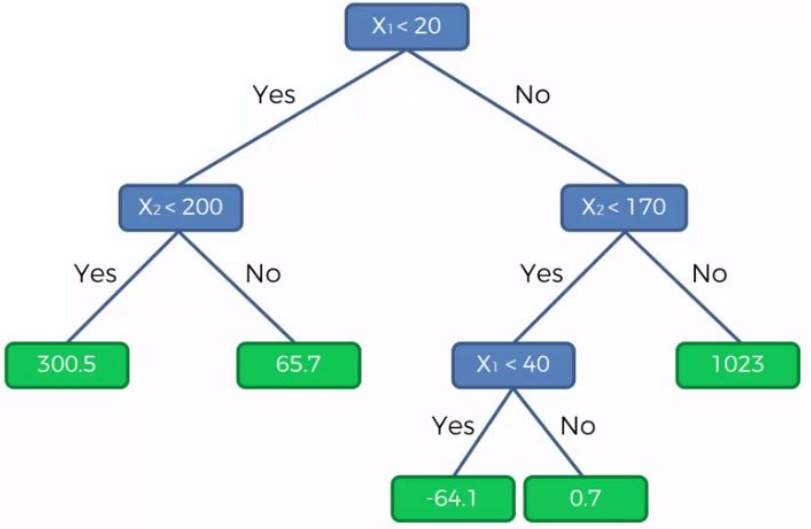
Decision Trees Regressor





Decision Trees Regressor





Random Forest - Ensemble Learning

Ensemble Learning: método que utiliza múltiplos algoritmos de aprendizado para obter um modelo melhorado. Para Random Forest, usa-se K Decision Trees.

1 - Selecionar K pontos do training set;

2 - Construir uma Decision Tree associada com esses K pontos;

3 - Escolha o número N de árvores que deseja construir, repita 1 e 2.

4 - Para um novo ponto, faça as N árvores preverem Y, depois faça uma média desses N valores, obtendo o valor final.

Random Forest - Exemplo prático

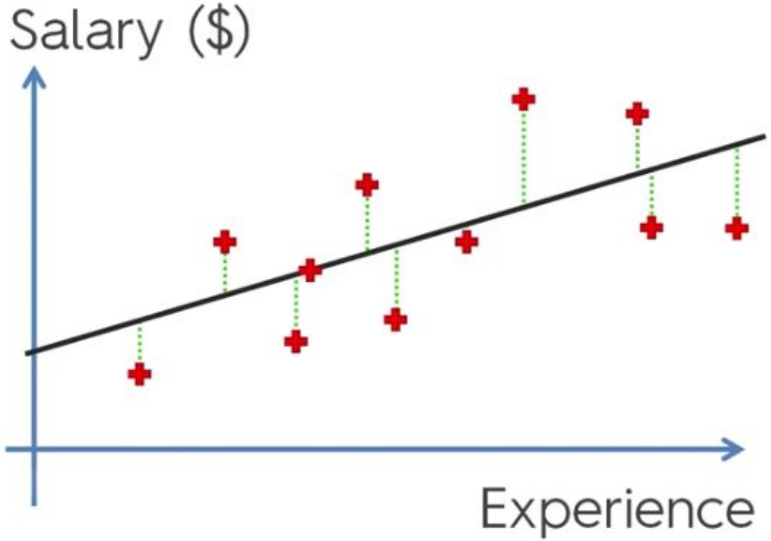
Jarro de jujubas. Adivinhar a quantidade de jujubas no jarro.

- Criar N modelos,
- Fazer a média dos N resultados.

R^2 - Coeficiente de Determinação

- Métrica de ajustamento de um modelo estatístico.

Simple Linear Regression:



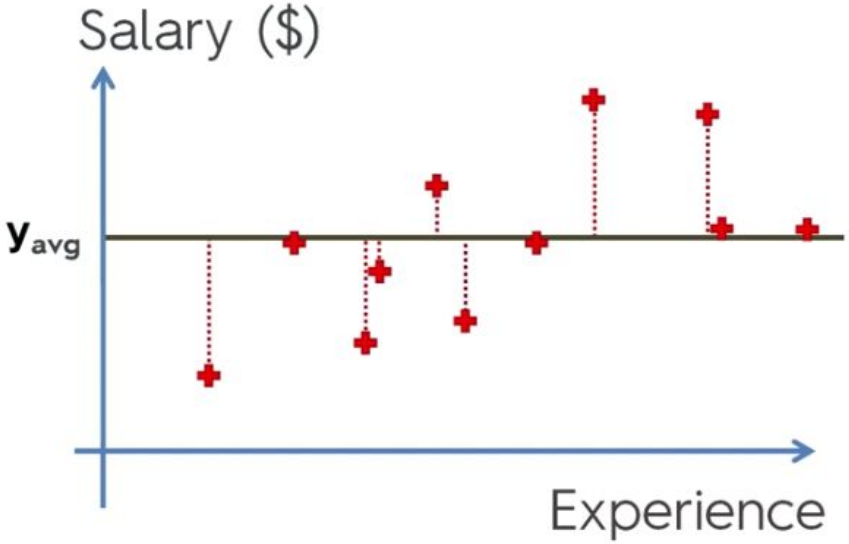
$$SS_{res} = SUM\ (y_i - \hat{y_i})^2$$

SSres = Sum of Squares of Residuals - Soma Quadrática de Resíduos

R^2 - Coeficiente de Determinação

- Métrica de ajustamento de um modelo estatístico.

Simple Linear Regression:



$$SS_{res} = SUM\ (y_i - \hat{y_i})^2$$

$$SS_{tot} = SUM\ (y_i - y_{avg})^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

R^2 Ajustado

- Adicionando variáveis (features), mesmo que tenham pouco valor explicativo sobre a variável dependente, sempre aumentarão o valor de R^2. Por isso o R^2 Ajustado tem mais significância.

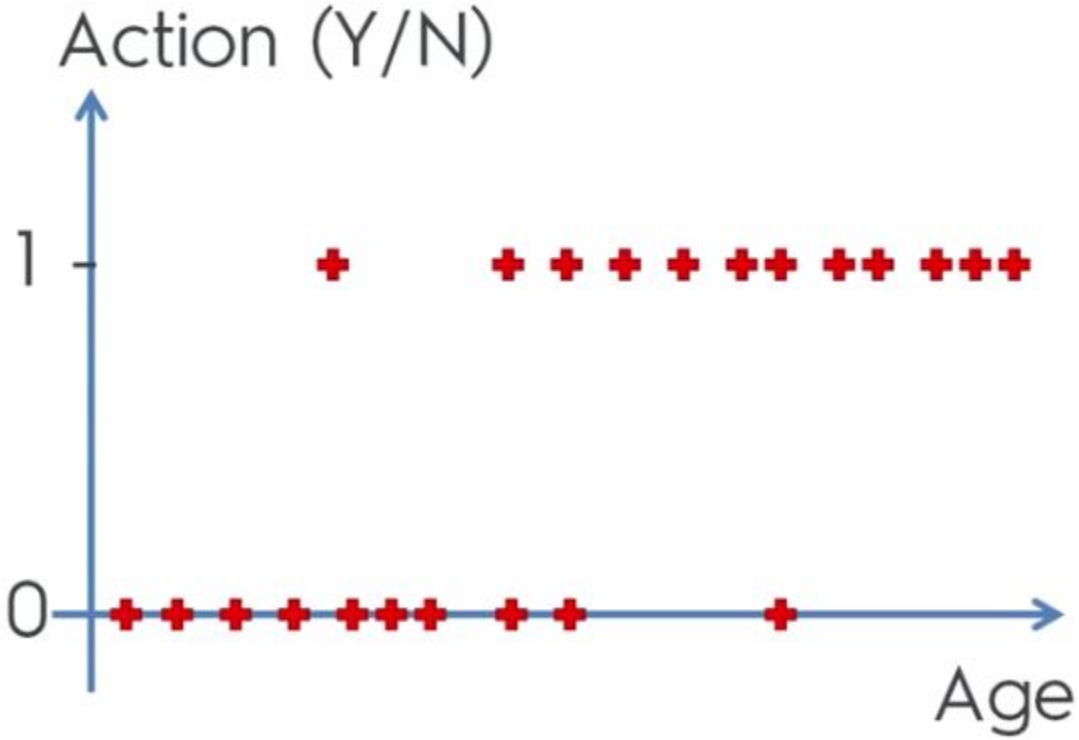
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$\text{Adj } R^2 = 1 - (1 - R^2)\frac{n - 1}{n - p - 1}$$

p - number of regressors
n - sample size

Aprendizado Supervisionado
Classificação

Regressão Logística



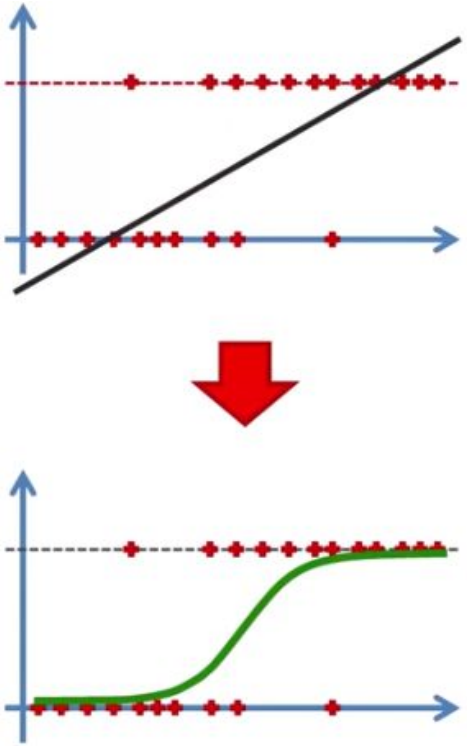
Regressão Logística

$$y = b_0 + b_1 * x$$

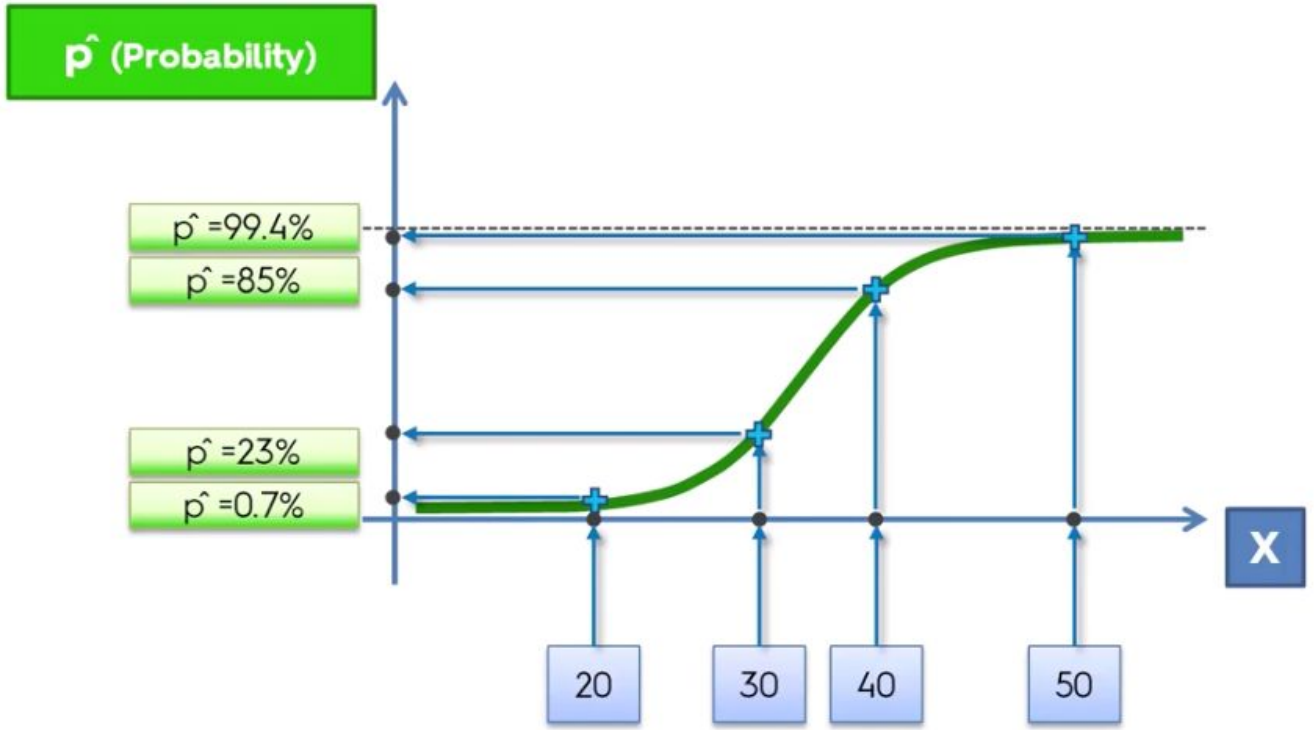
Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

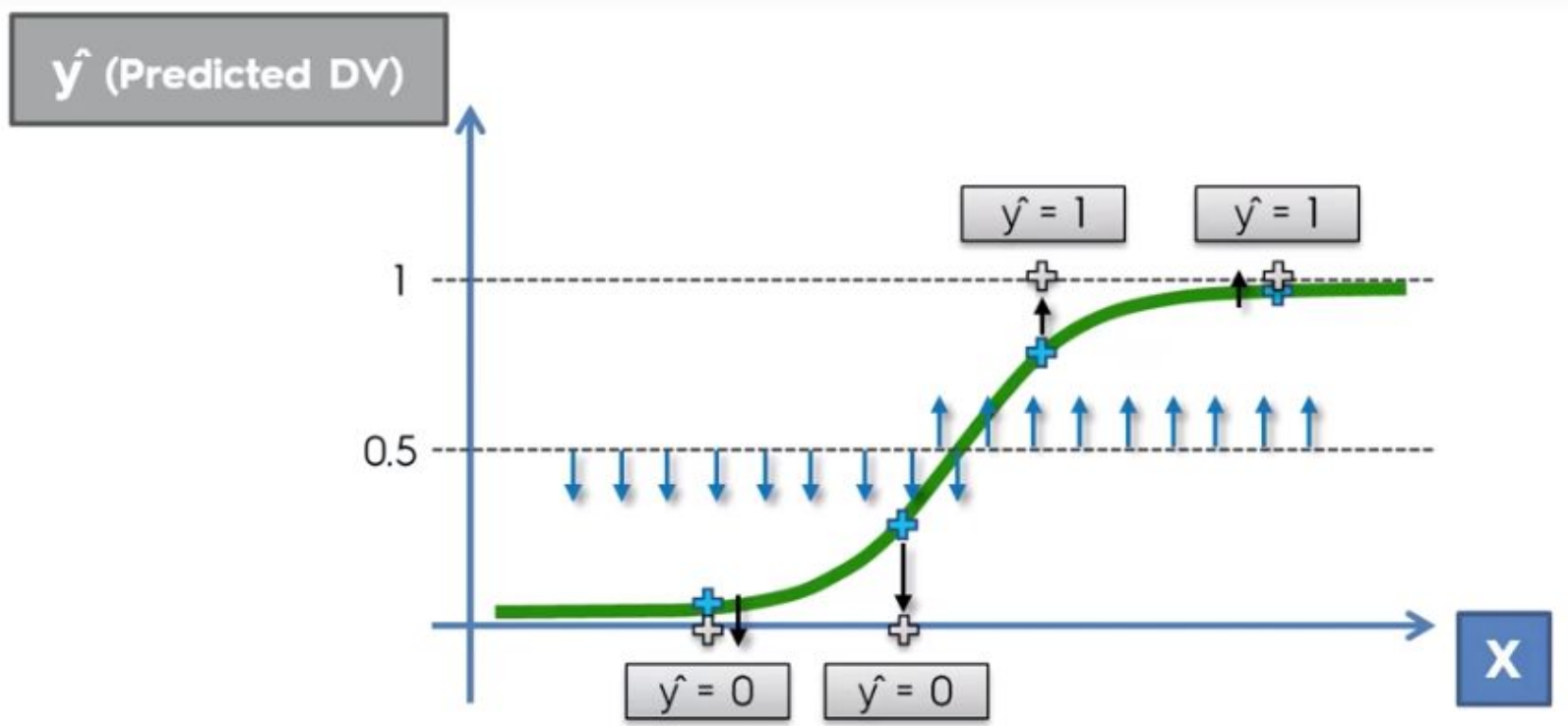
$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$



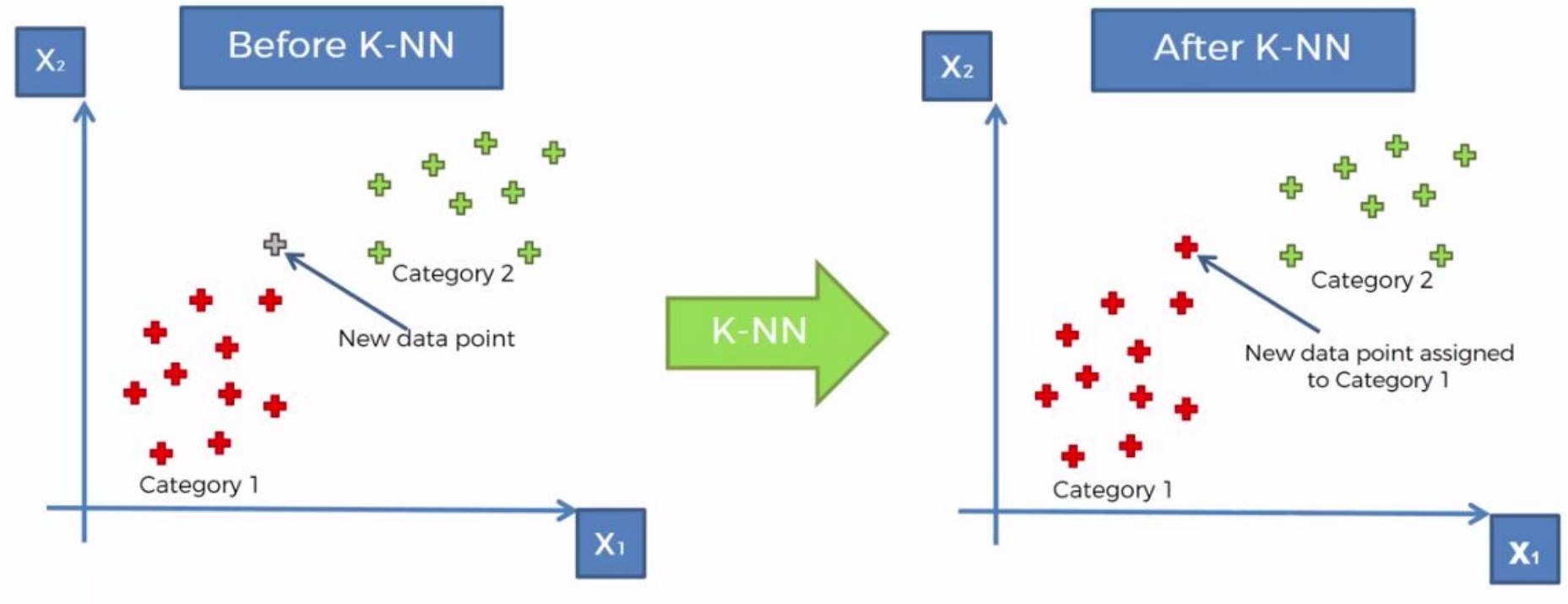
Regressão Logística



Regressão Logística



K-Nearest Neighbours



K-Nearest Neighbours

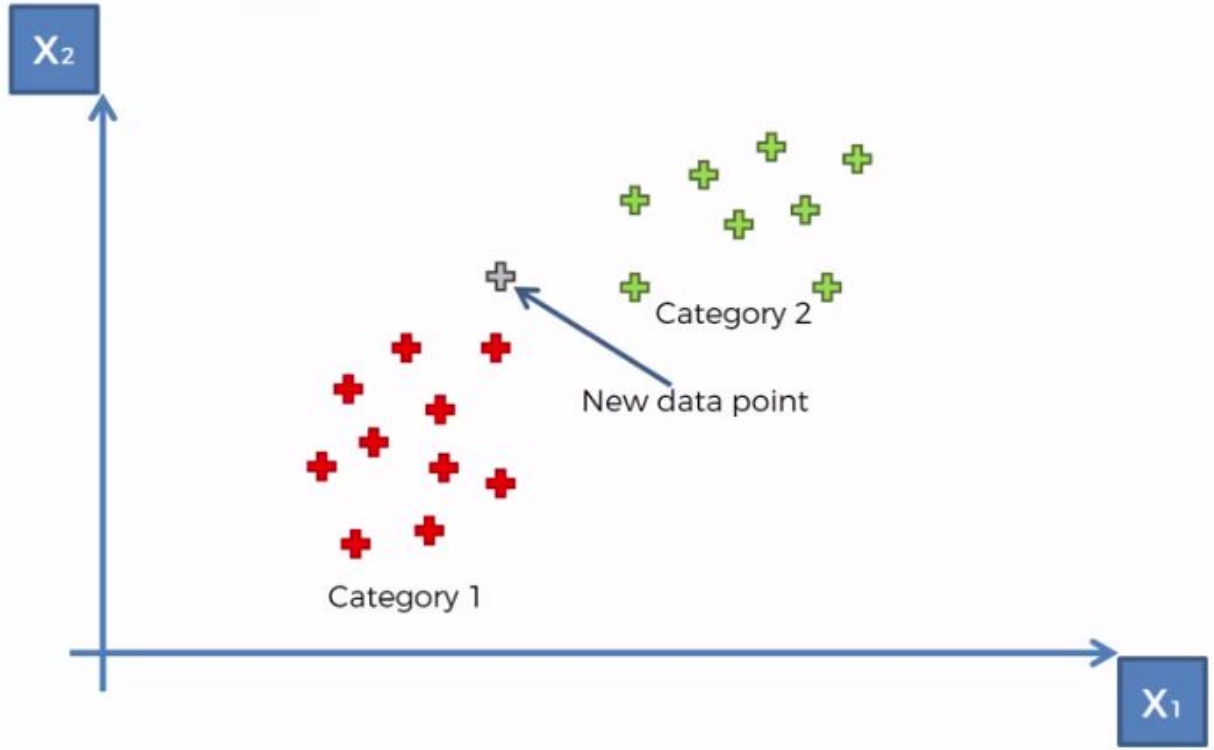
1 - Escolha o número K de vizinhos;

2 - Pegue os K vizinhos mais próximos do novo ponto, de acordo com distância Euclidiana;

3 - Entre esses K vizinhos, conte o número de pontos em cada categoria;

4 - Dê ao novo ponto o rótulo da classe onde você contou o maior número de vizinhos.

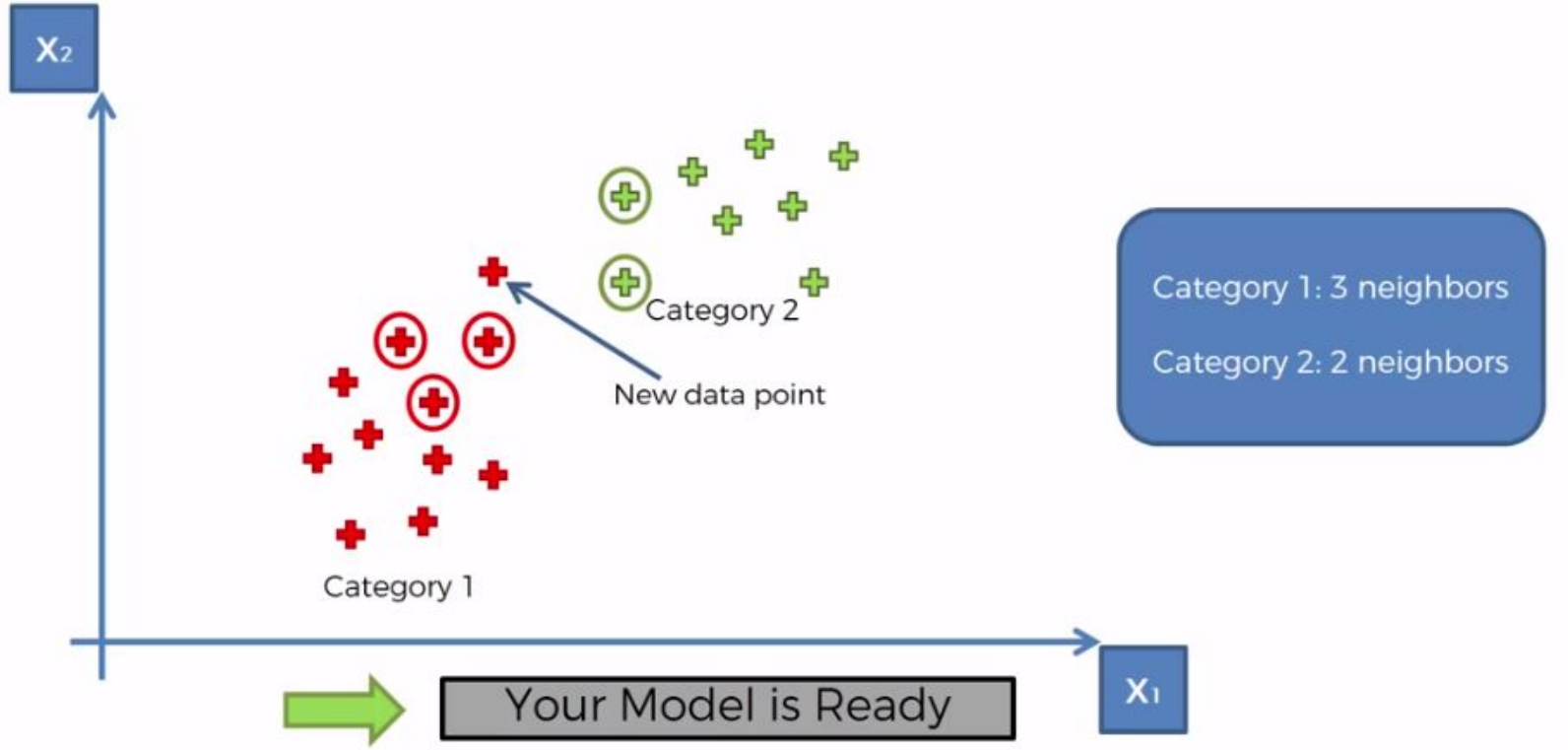
K-Nearest Neighbours - Passo 1



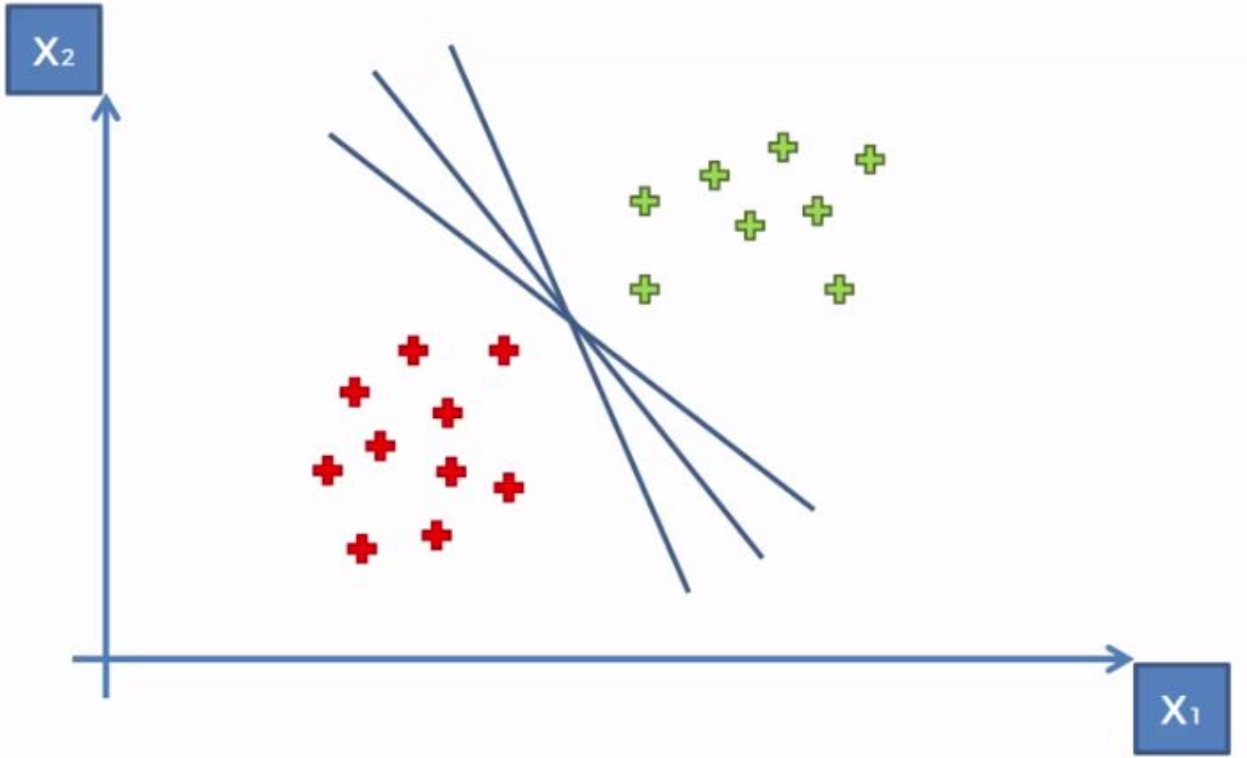
K-Nearest Neighbours - Passo 2

K-Nearest Neighbours - Passo 3

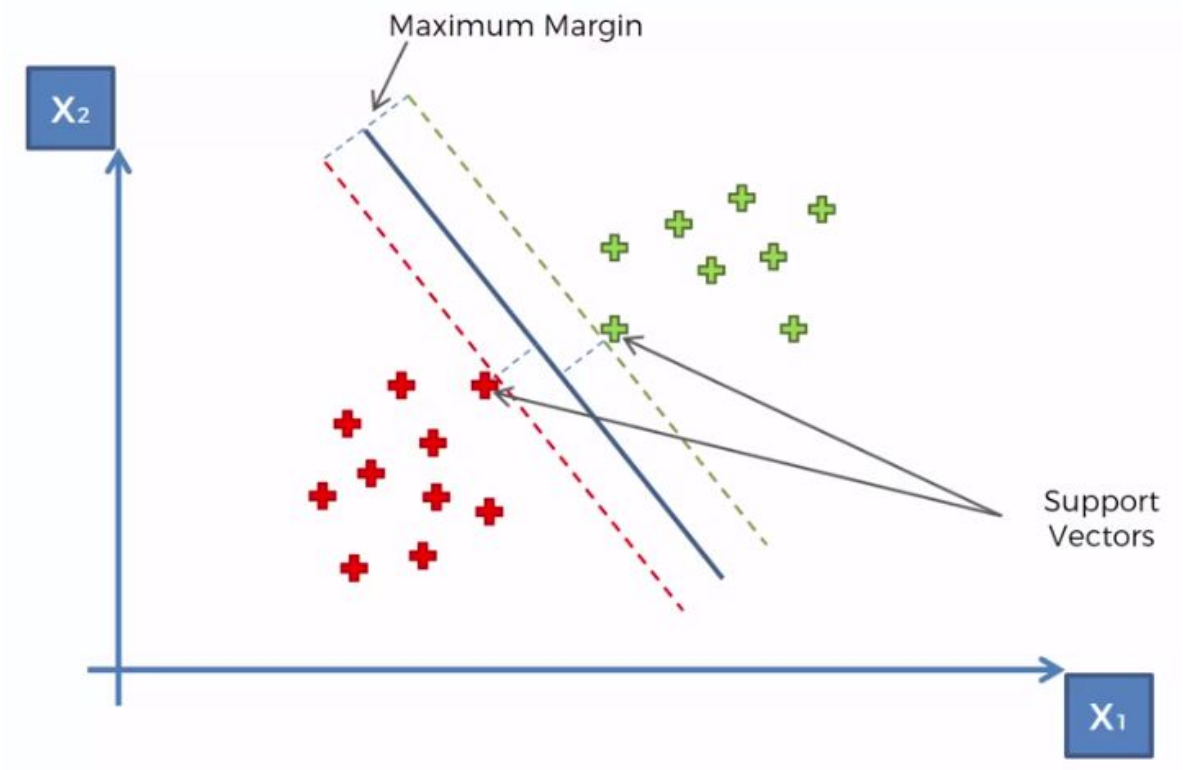
K-Nearest Neighbours - Passo 4



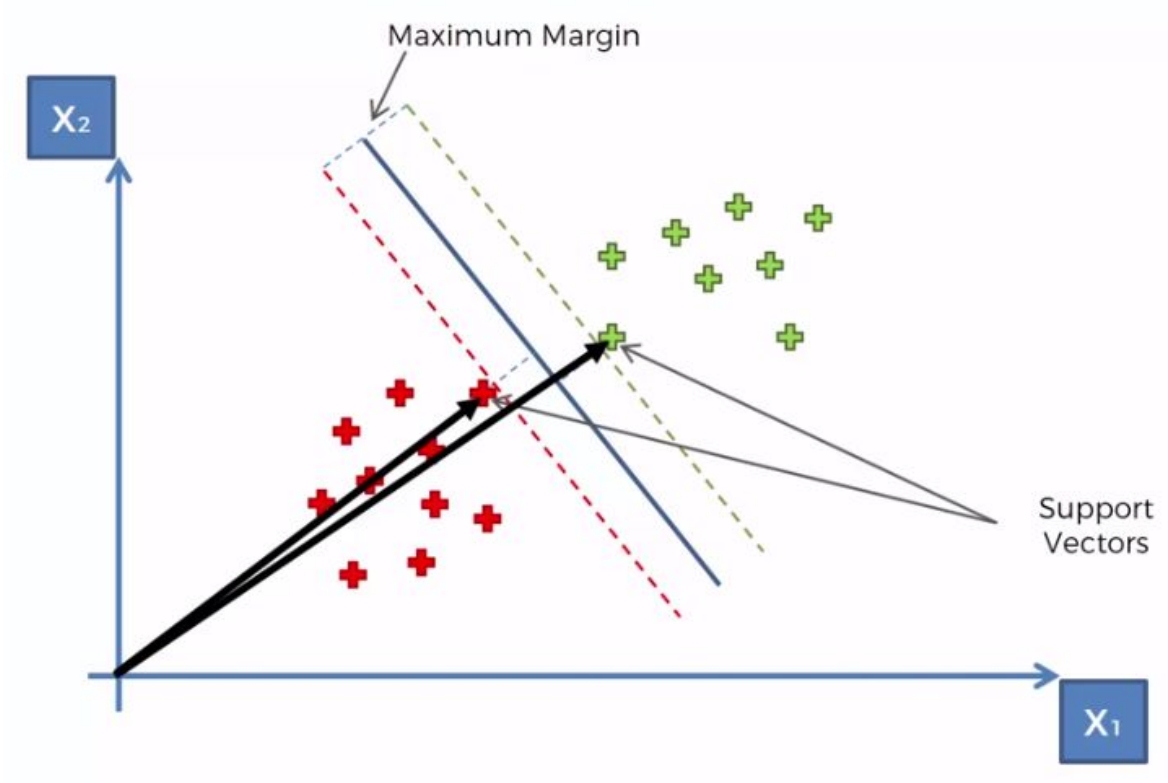
SVM - Support Vector Machine



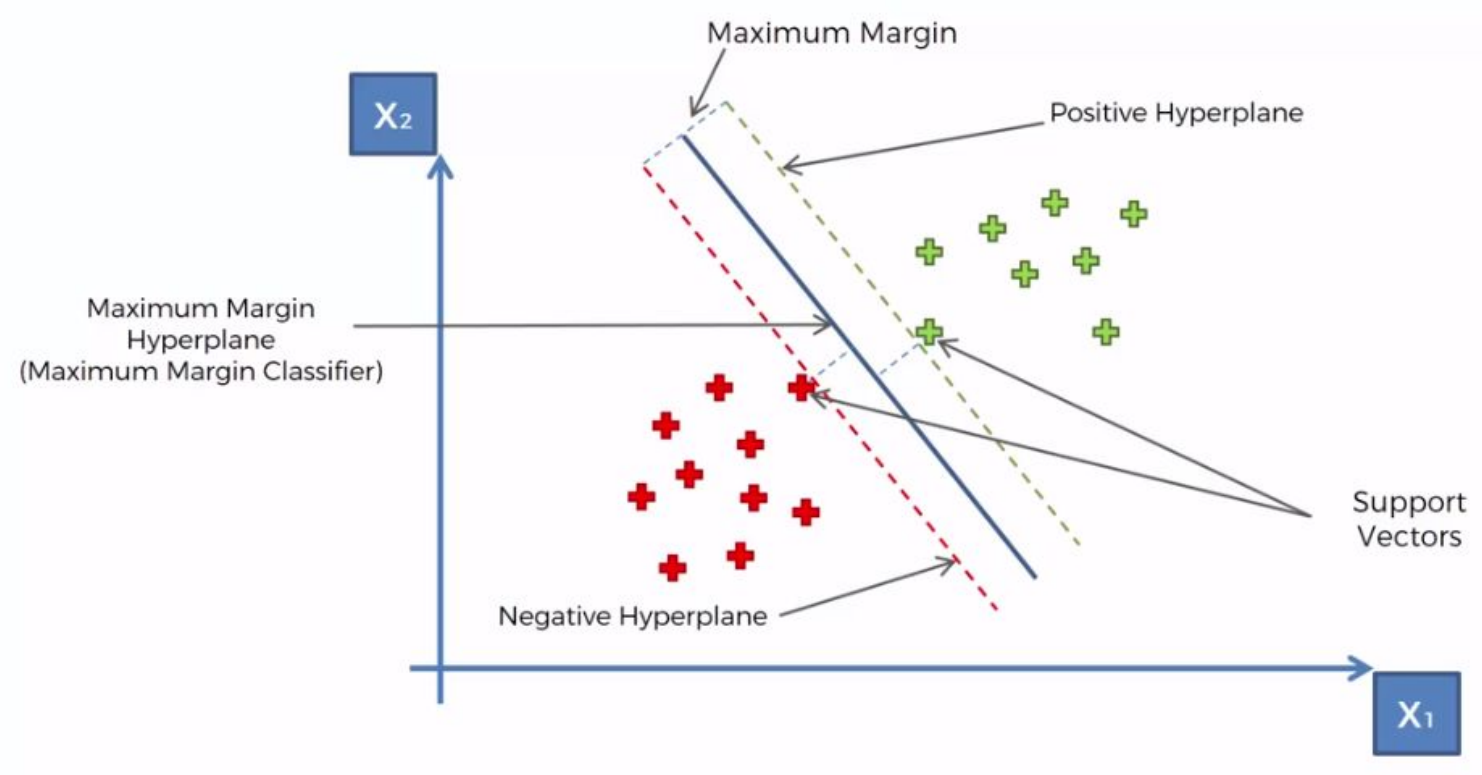
SVM - Support Vector Machine



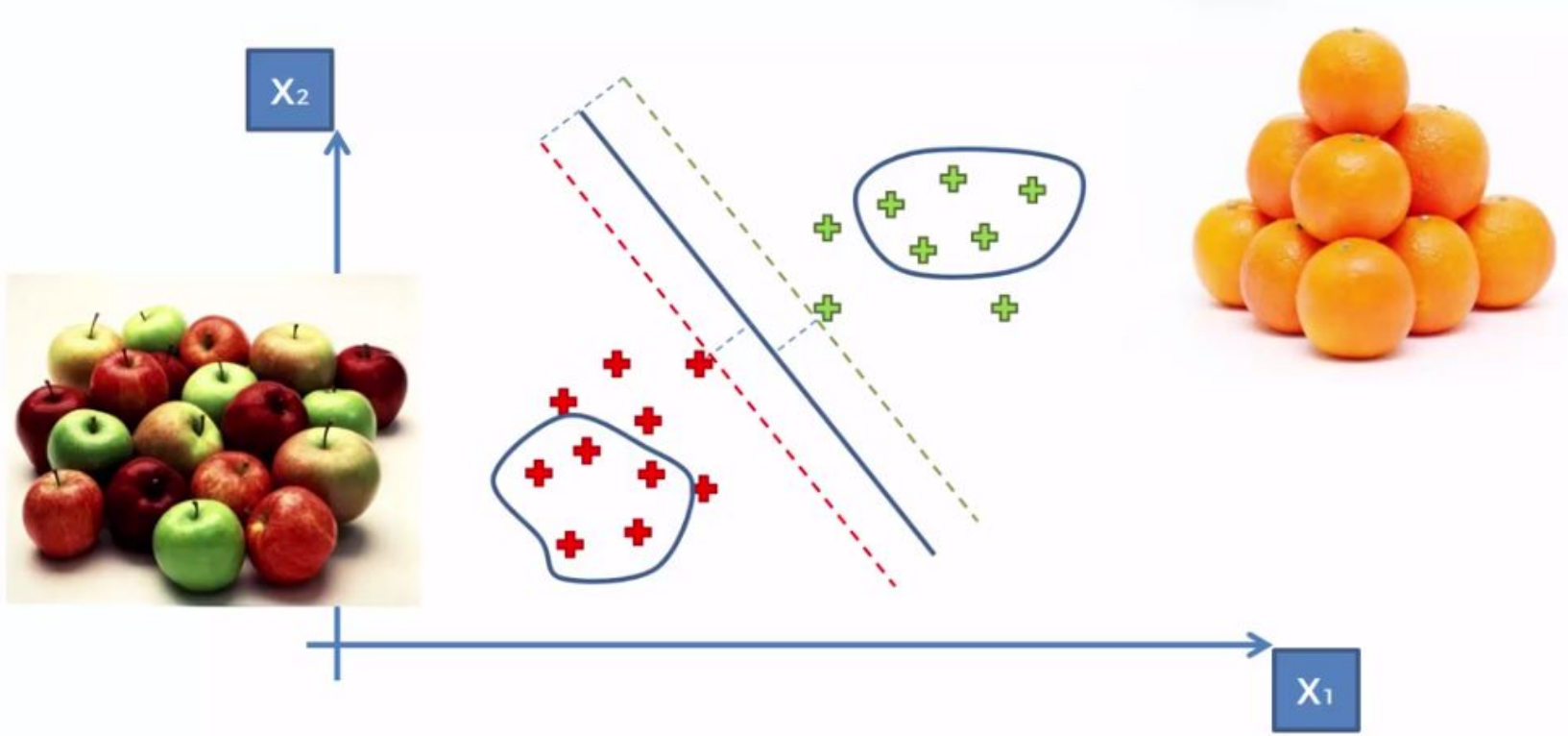
SVM - Support Vector Machine



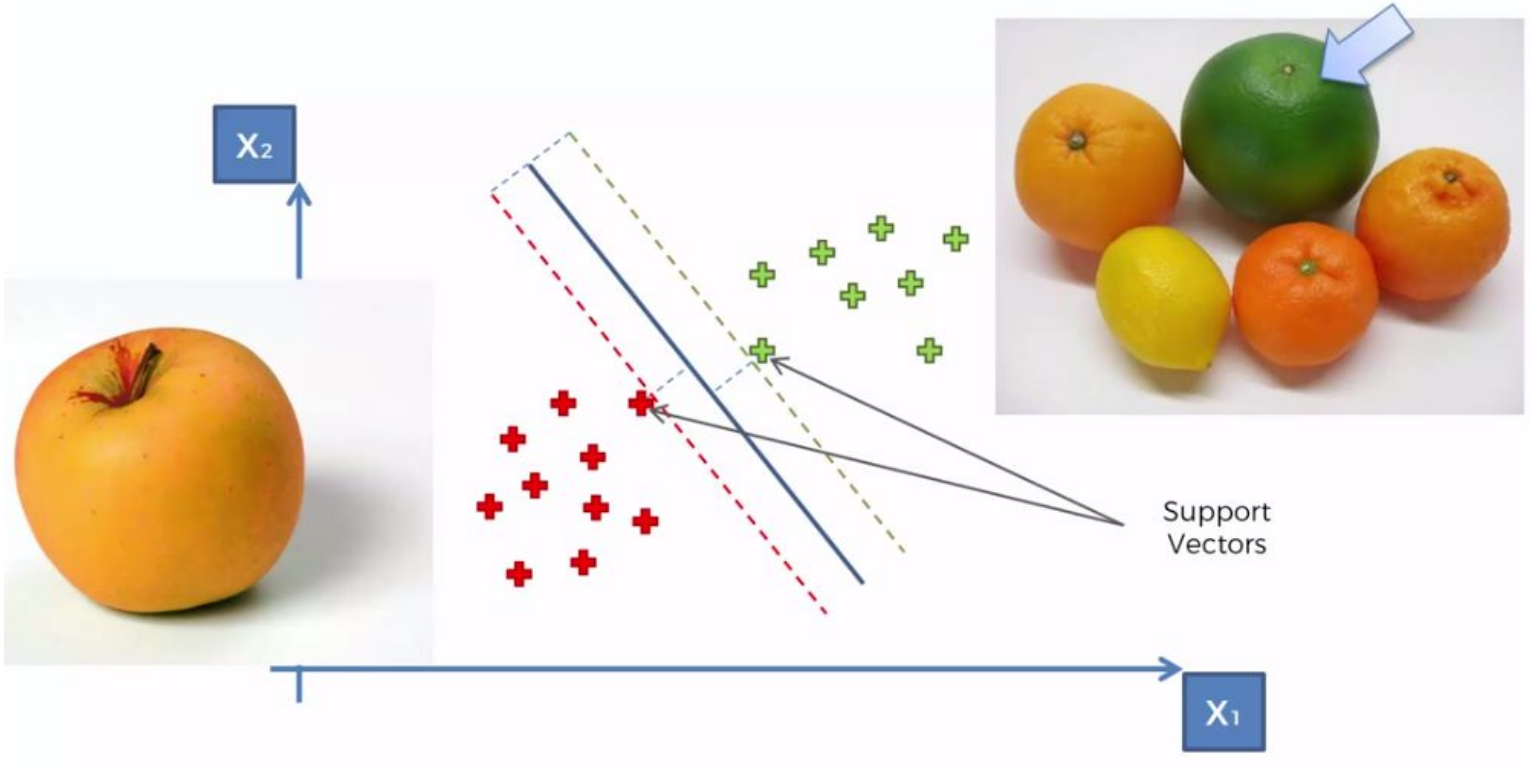
SVM - Support Vector Machine



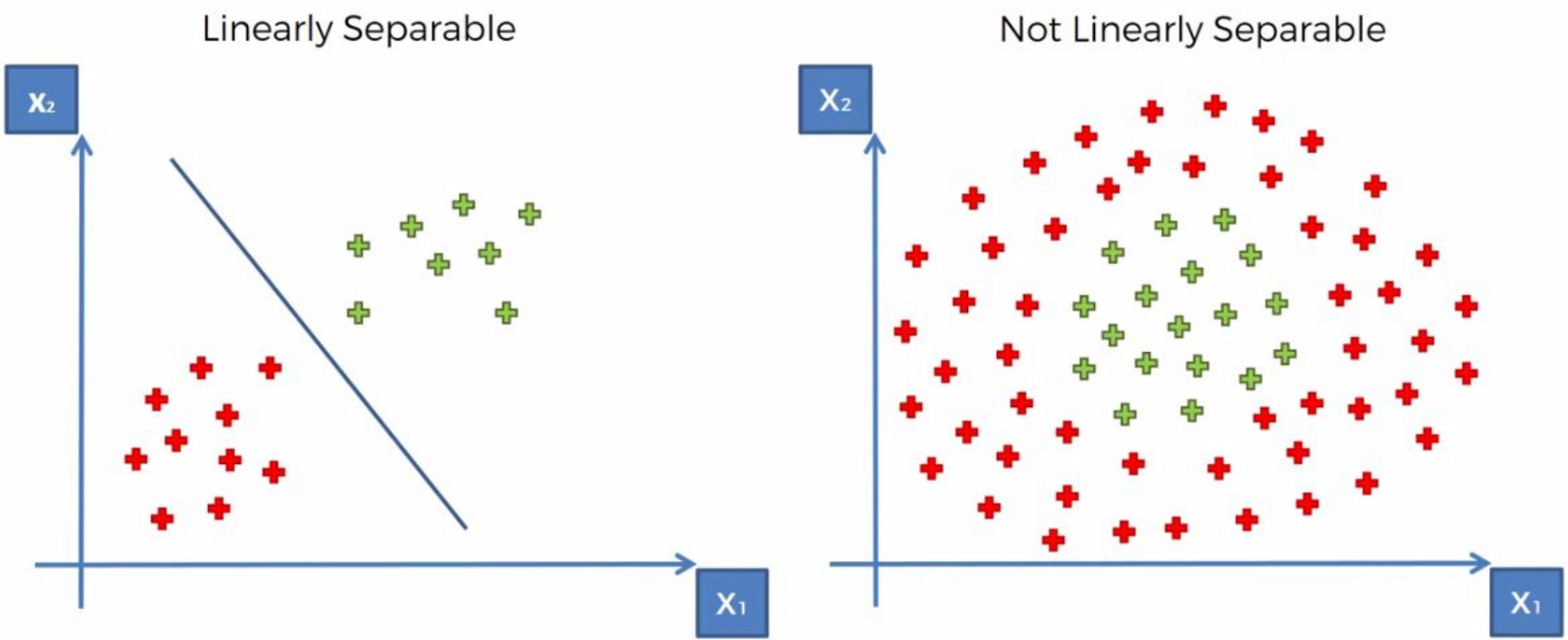
SVM - Support Vector Machine - Vetores?



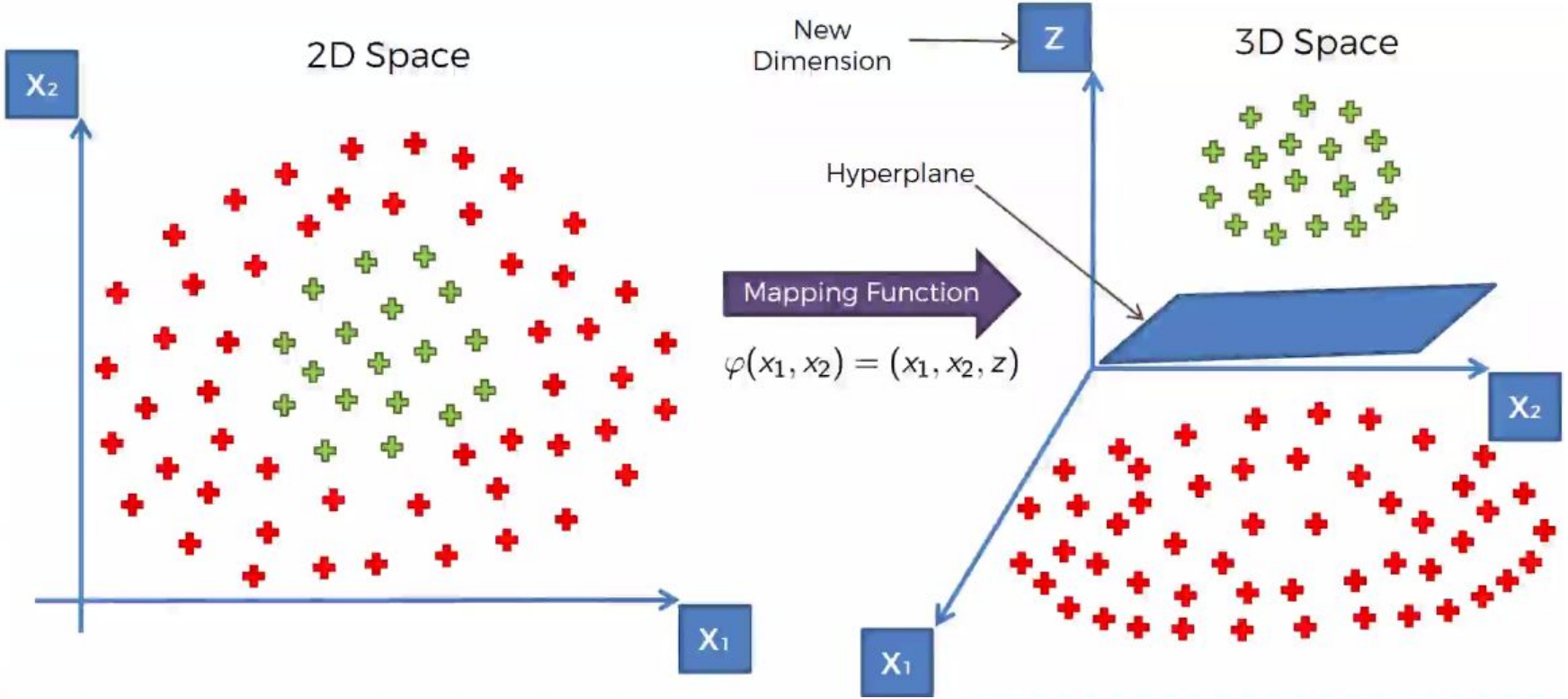
SVM - Support Vector Machine



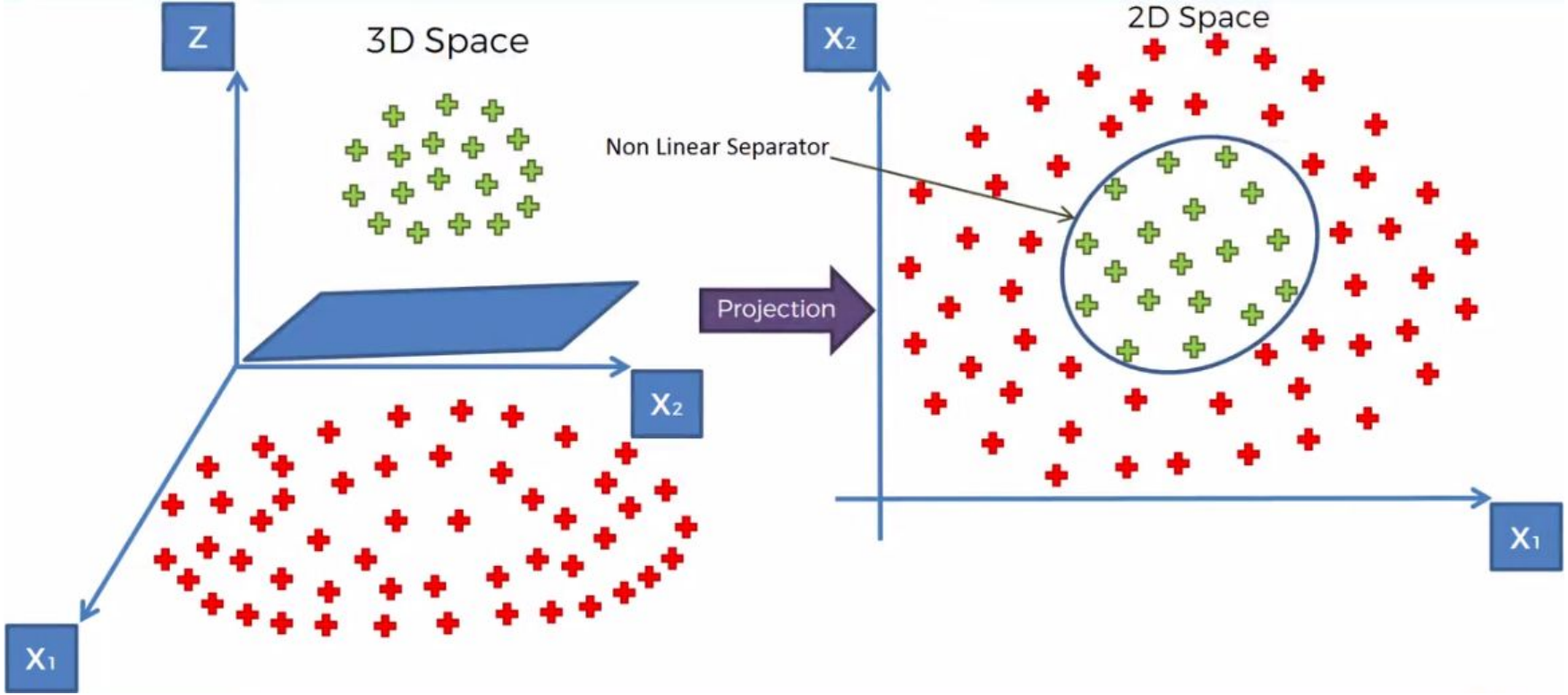
Kernel SVM



Kernel SVM


2D Space
X2
X1
New Dimension
Z
3D Space
Hyperplane
Mapping Function
$\varphi(x_1, x_2) = (x_1, x_2, z)$
X2
X1

Kernel SVM

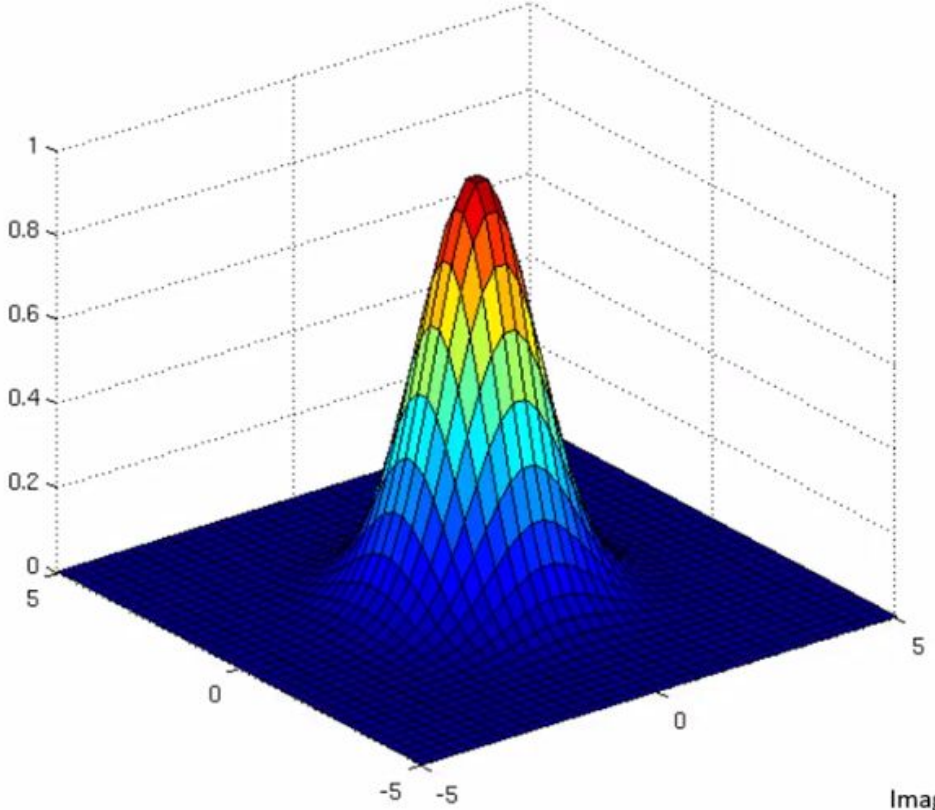

Z
3D Space
X2
X1
Non Linear Separator
Projection
2D Space
X2
X1

Kernel SVM

Porém, mapear os dados para uma dimensão superior, e retornar para a dimensão original é um processo altamente custoso do ponto de vista computacional.

Para evitar esse tipo de abordagem, usa-se o Truque do Kernel, ou Kernel Trick.

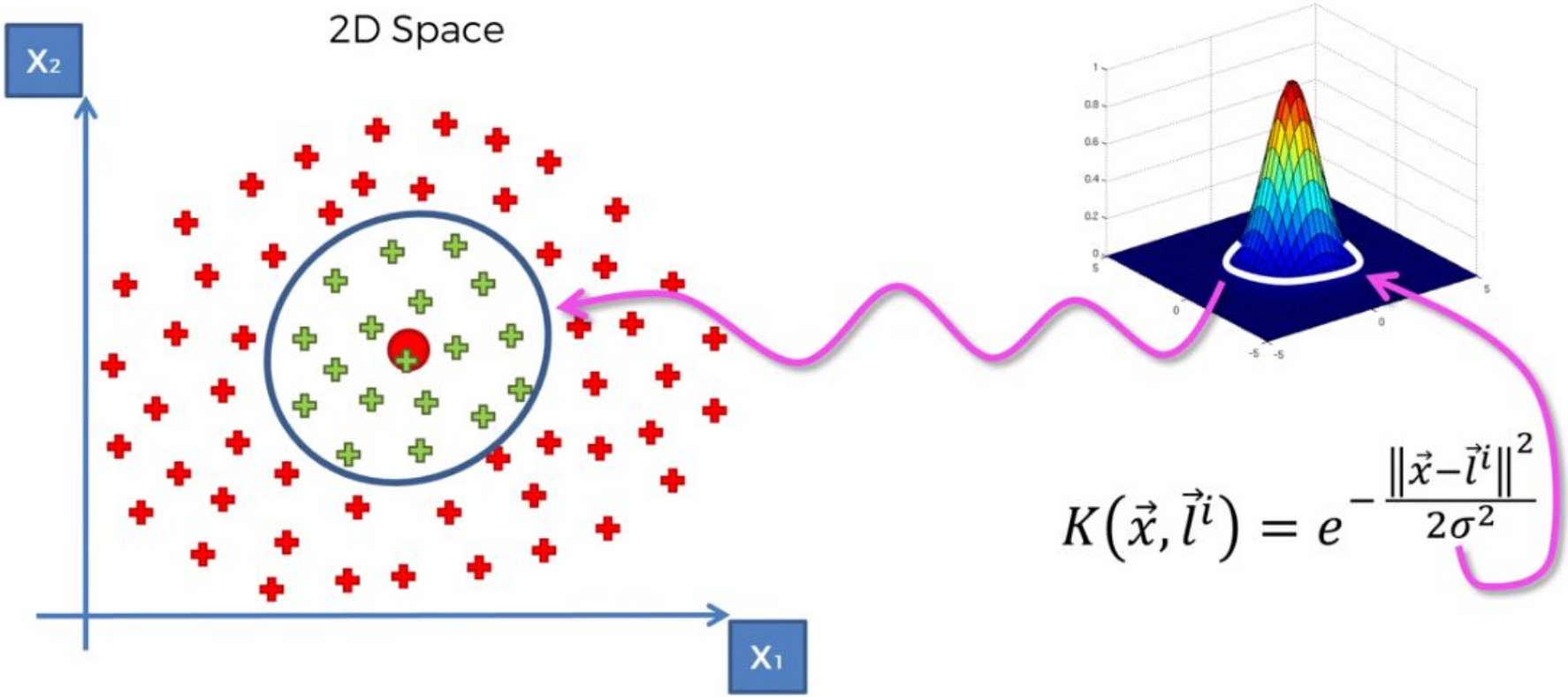
Kernel Trick - Kernel Gaussiano



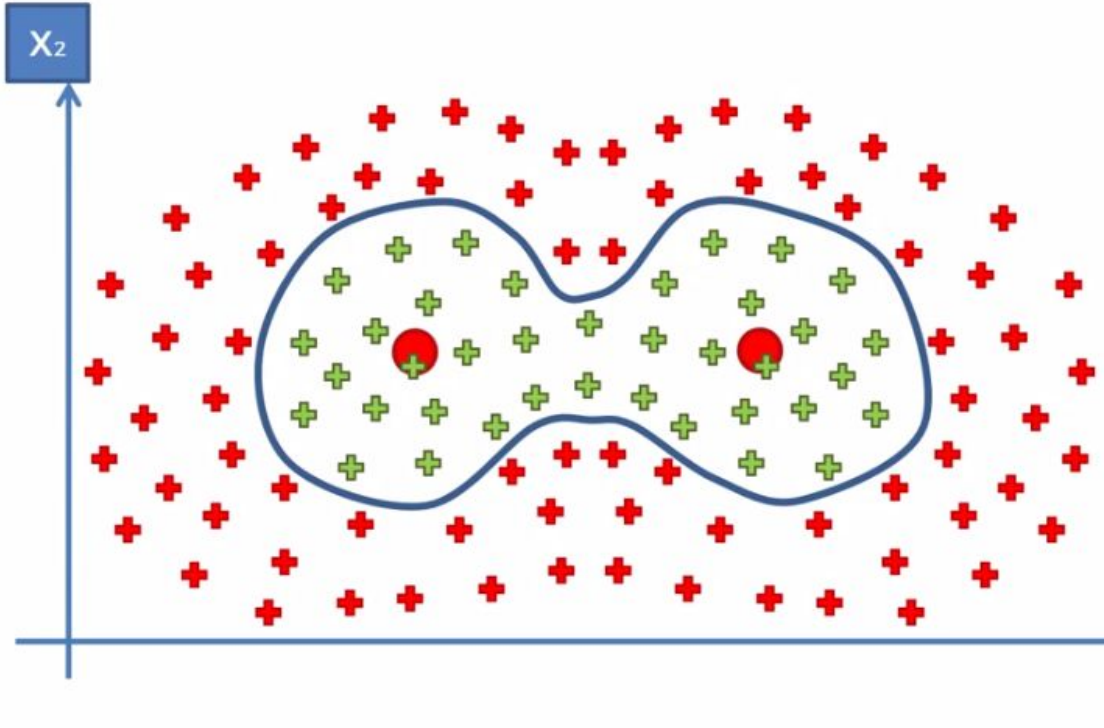
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

Kernel Trick - Kernel Gaussiano



Kernel Trick - Kernel Gaussiano



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

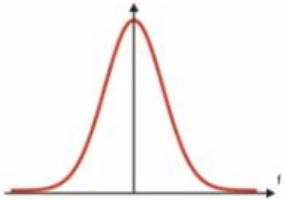
Green when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$

Red when:

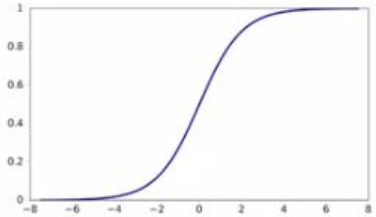
$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) = 0$$

Kernel Trick - Tipos de Kernel



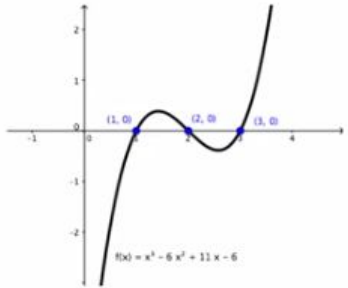
Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$



Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$



Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

Teorema de Bayes

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Exemplo:
Máquina 1: 30 alicates/hora
Máquina 2: 20 alicates/hora

De todos os alicates produzidos,
pode-se ver que 1% são defeituosos.

De todos os alicates defeituosos:
Pode-se ver que 50% veio da M1,
e 50% veio da M2.

Questão:
Qual a probabilidade de que um alicate
produzido pela M2 seja defeituoso?

P(M1) = 30/50 = 0.6
P(M2) = 20/50 = 0.4

P(defeito) = 1% = 0.01

P(M1 | defeito) = 50%
P(M2 | defeito) = 50%

P(defeito | M2) = ?

$$P(\text{defeito} \mid M2) = \frac{P(M2 \mid \text{defeito}) * P(\text{defeito})}{P(M2)}$$

$$P(\text{defeito} \mid M2) = \frac{0.5 * 0.01}{0.4} = 0.0125 = \mathbf{1.25\%}$$

Teorema de Bayes

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Exemplo:
Máquina 1: 30 alicates/hora
Máquina 2: 20 alicates/hora

De todos os alicates produzidos,
pode-se ver que 1% são defeituosos.

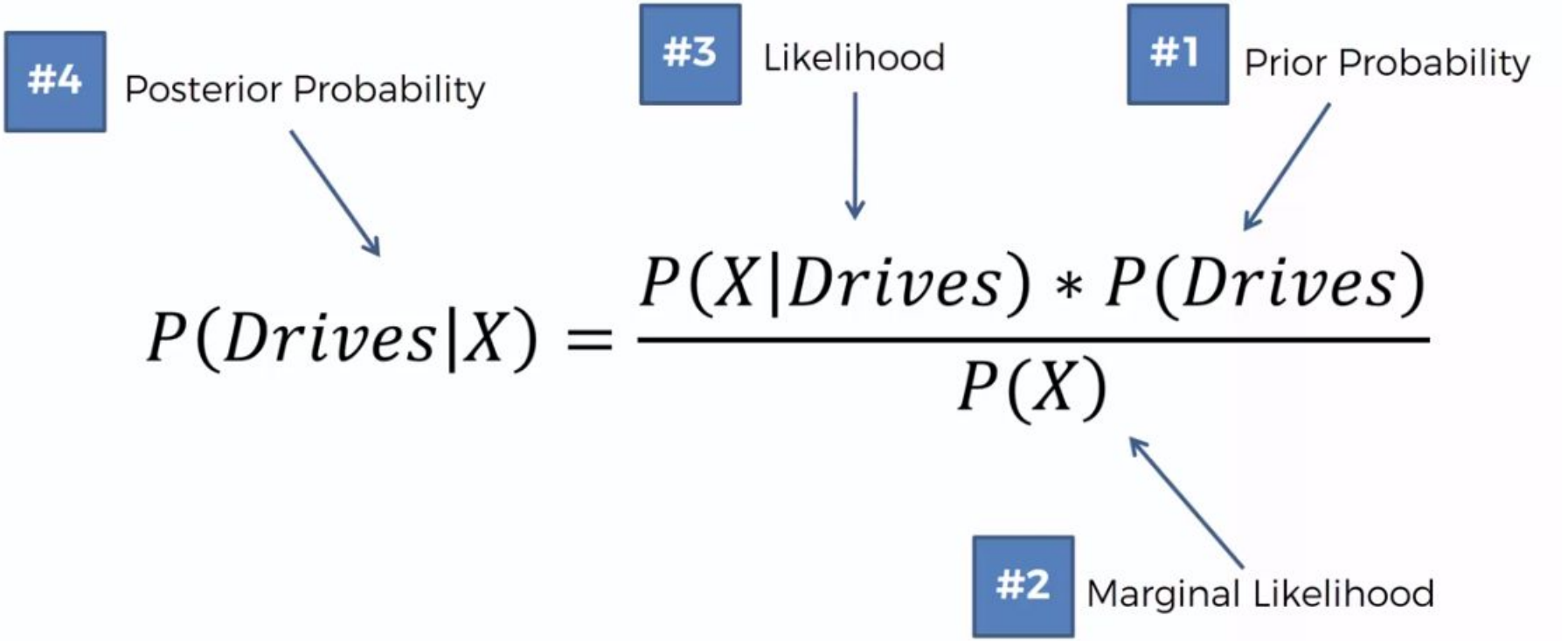
De todos os alicates defeituosos:
Pode-se ver que 50% veio da M1,
e 50% veio da M2.

Questão:
Qual a probabilidade de que um alicate
produzido pela M2 seja defeituoso?

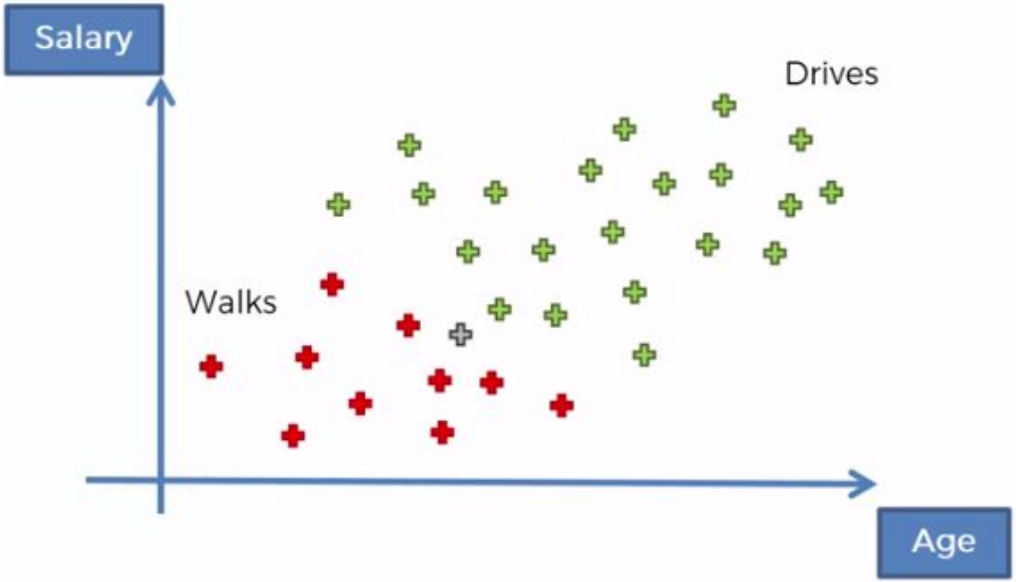
Exemplo:

- 1000 alicates
- 400 vieram da M2
- 1% possuem defeito = 10
- Desses 1%, 50% vieram da M2 = 5
- % defeituosos da M2 = 5/400 = **1.25%**

Naive Bayes


#4
Posterior Probability
#3
Likelihood
#1
Prior Probability
$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$
#2
Marginal Likelihood

Naive Bayes

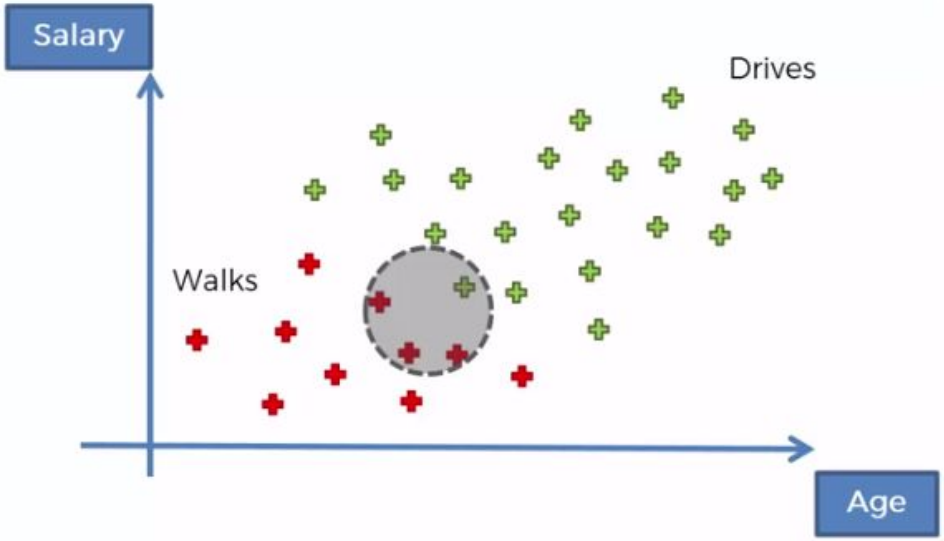


#1. P(Walks)

$$P(Walks) = \frac{Number\ of\ Walkers}{Total\ Observations}$$

$$P(Walks) = \frac{10}{30}$$

Naive Bayes

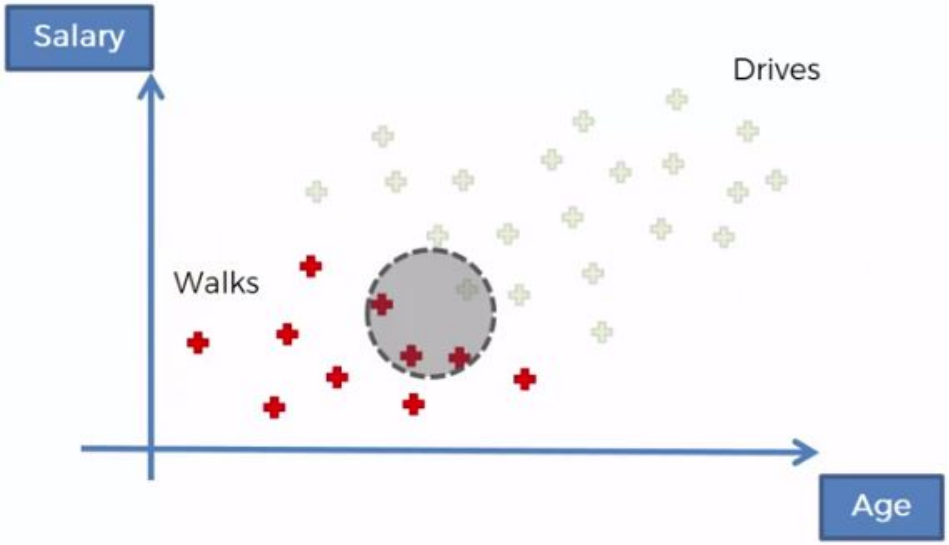


#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

Naive Bayes

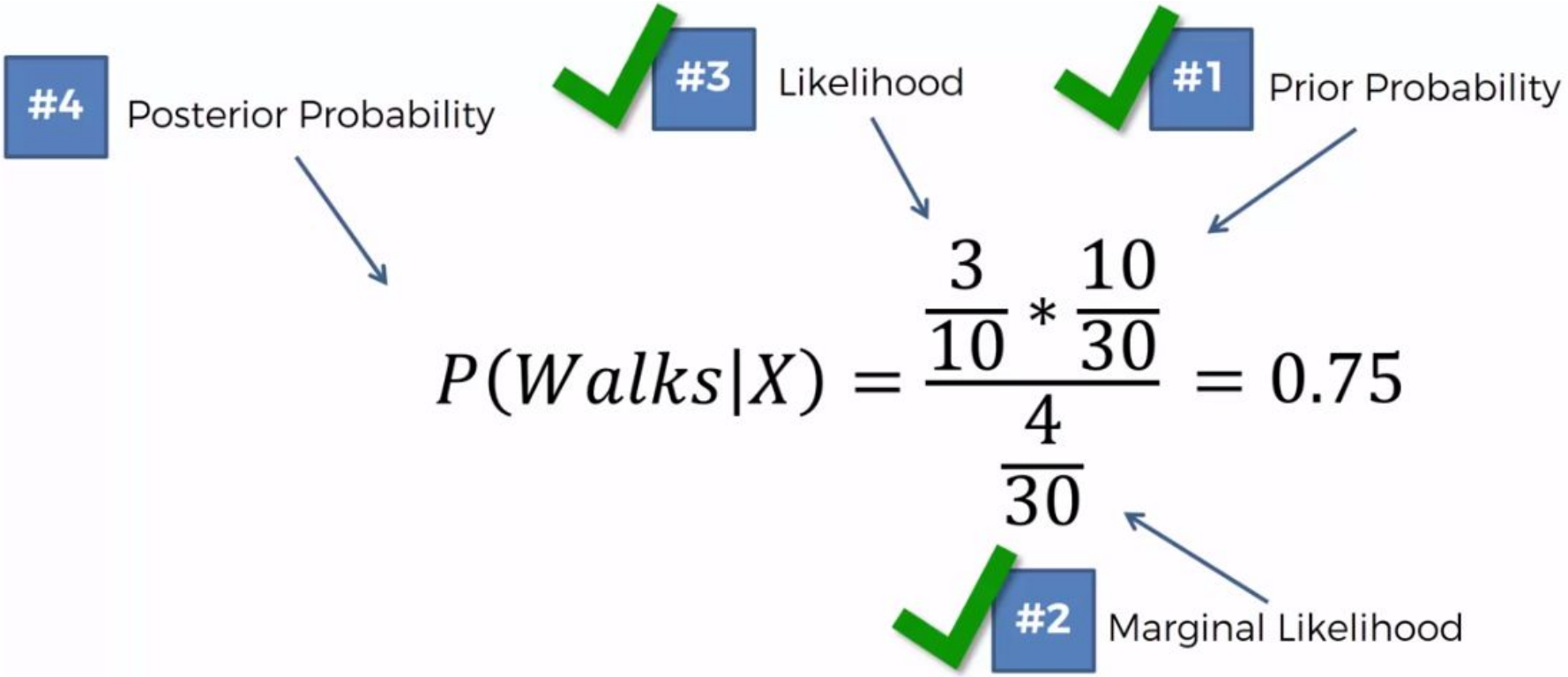


#3. P(X|Walks)

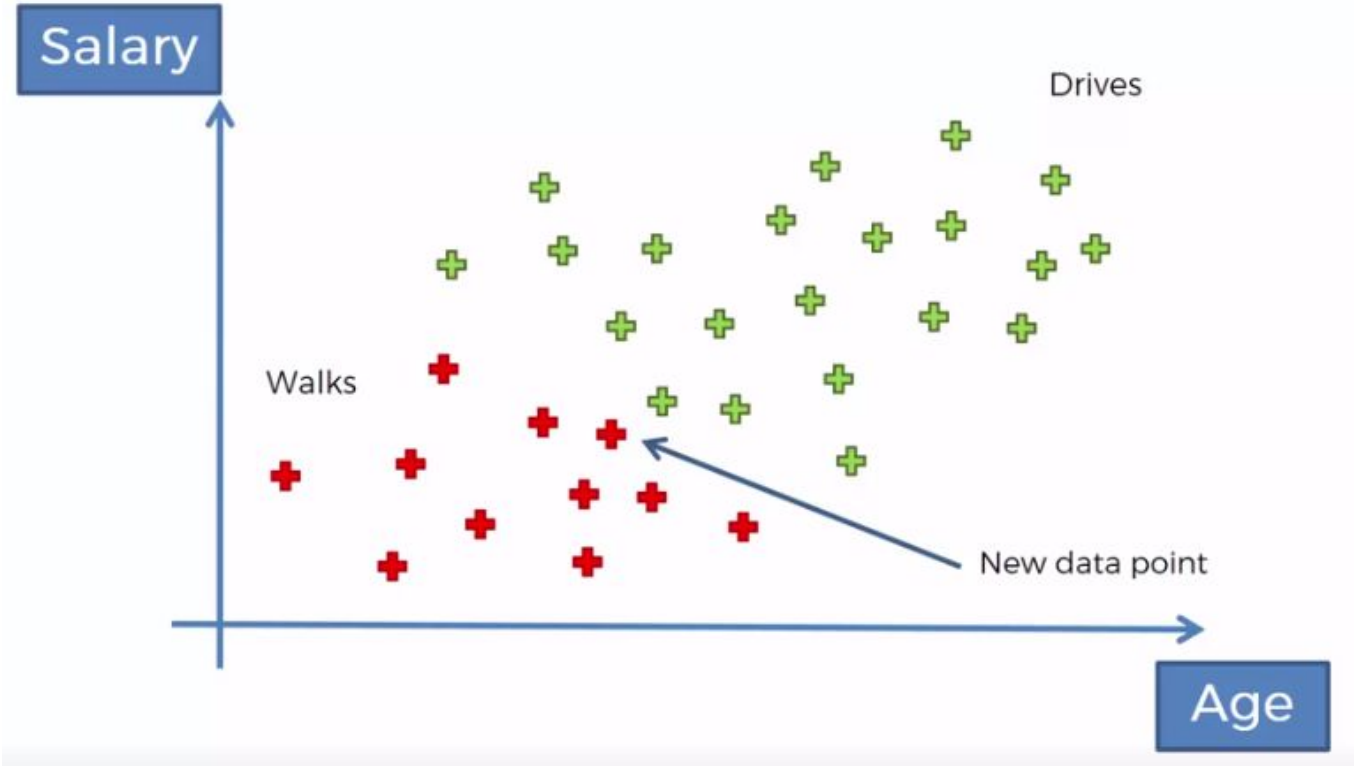
$$P(X|Walks) = \frac{Number\ of\ Similar\ Observations\ Among\ those\ who\ Walk}{Total\ number\ of\ Walkers}$$

$$P(X|Walks) = \frac{3}{10}$$

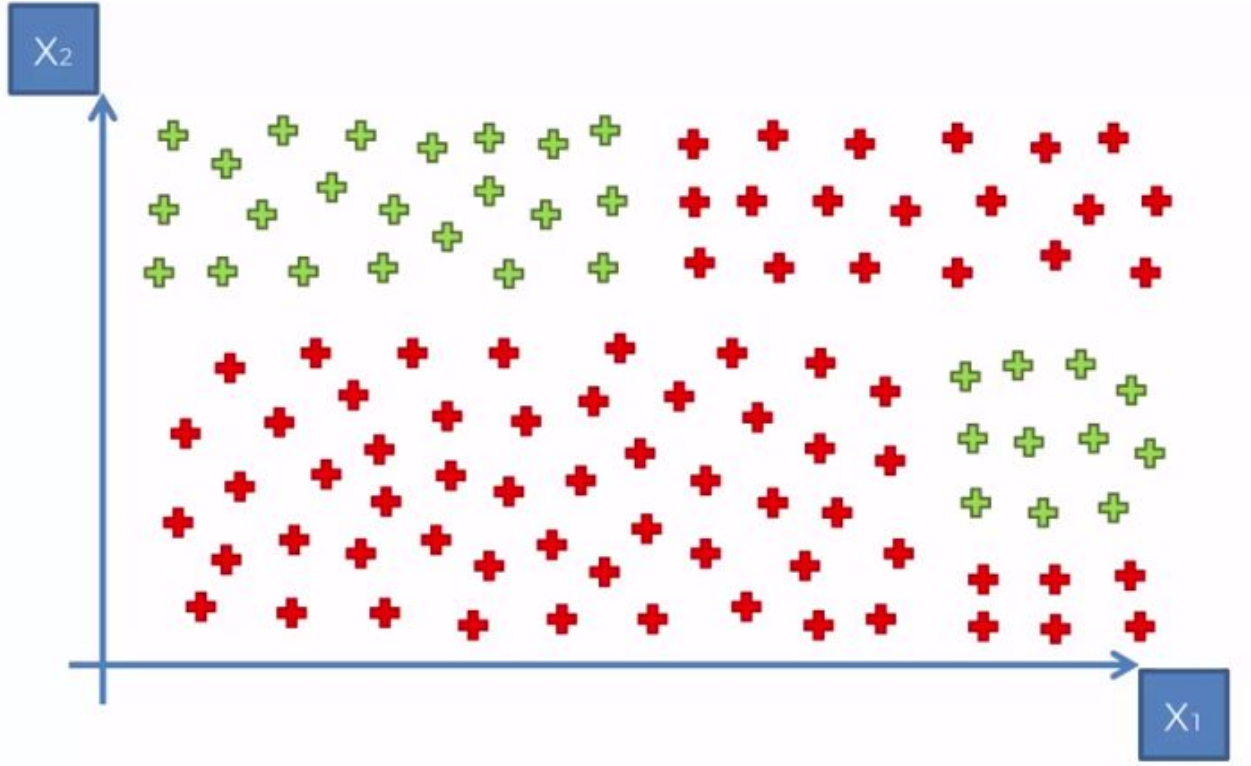
Naive Bayes


#4
Posterior Probability
#3
Likelihood
#1
Prior Probability
$P(Walks|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$
#2
Marginal Likelihood

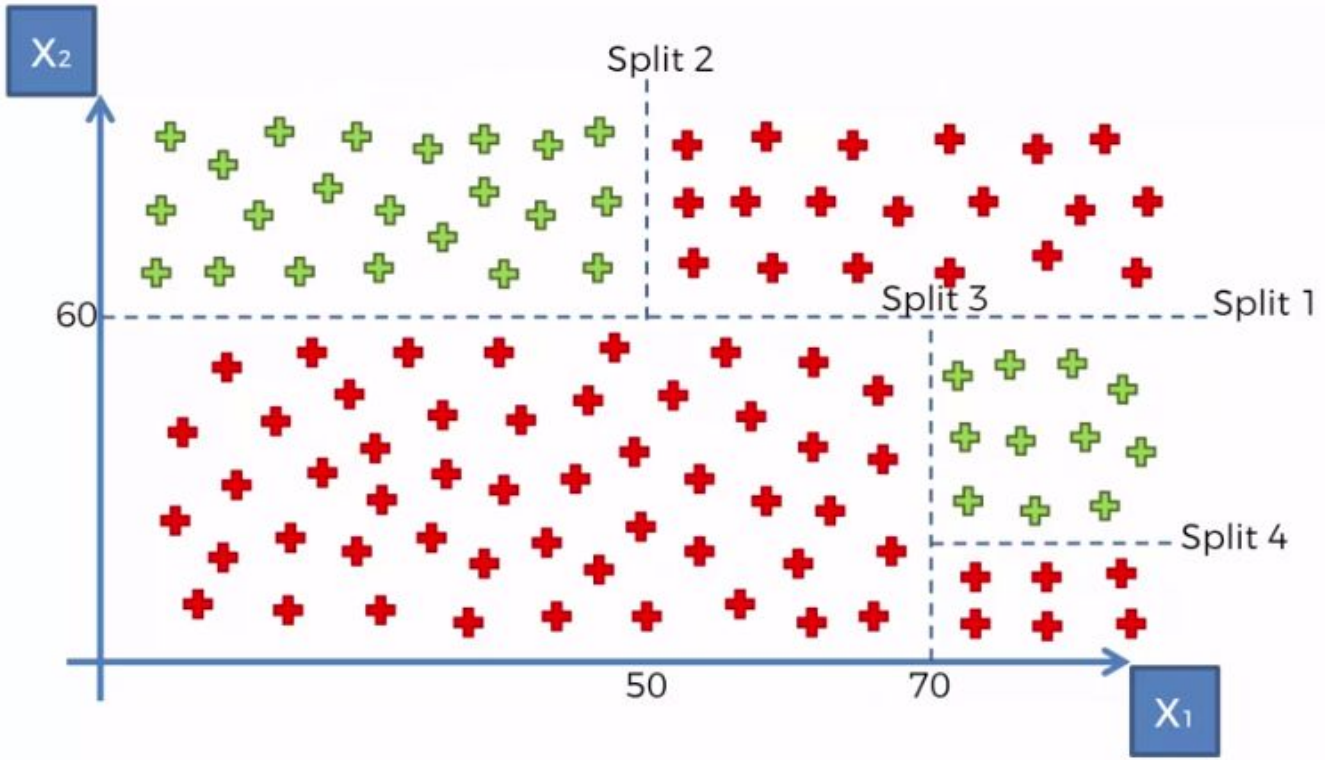
Naive Bayes



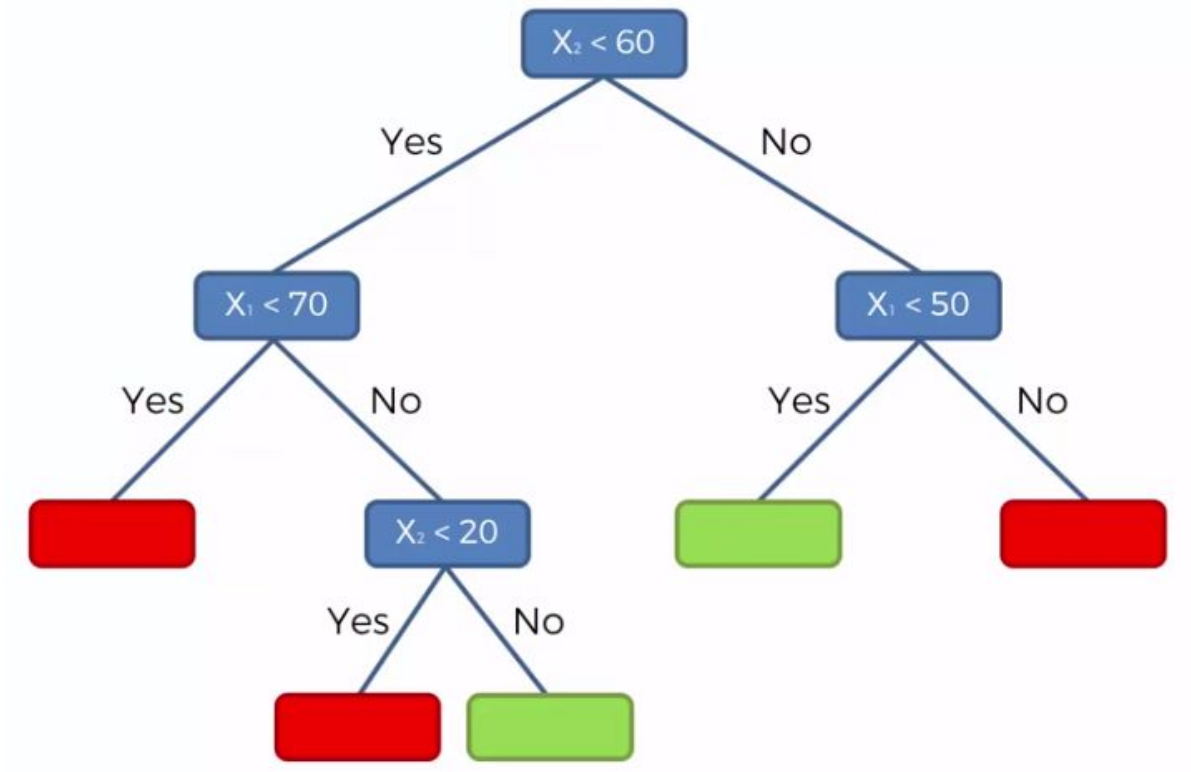
Decision Trees Classifier



Decision Trees Classifier



Decision Trees Classifier



Random Forest Classifier

Ensemble Learning: método que utiliza múltiplos algoritmos de aprendizado para obter um modelo melhorado. Para Random Forest, usa-se K Decision Trees.

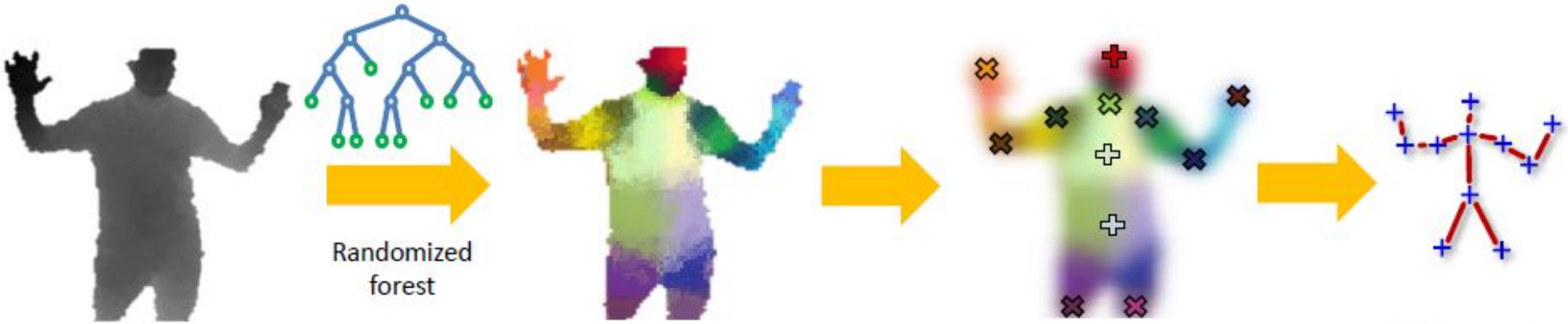
1 - Selecionar K pontos do training set;

2 - Construir uma Decision Tree associada com esses K pontos;

3 - Escolha o número N de árvores que deseja construir, repita 1 e 2.

4 - Para um novo ponto, faça as N árvores preverem a classe, depois associe o ponto à classe com maior número de previsões

Random Forest Classifier - Kinect



Capture depth image and remove background

Assign body part labels to each pixel

cluster pixels to hypothesize body joint positions

fit model & track skeleton


Source: http://www.youtube.com/watch?v=YGO2lwAgnq

Métricas para Classificação - Matriz de Confusão

		"Golden Standard" (Real Truth Values)		
		Positive	Negative	
Observed	Predicted positive	True Positive	False Positive (Type 1 error)	Precision
	Predicted Negative	False Negative (Type 2 error)	True Negative	
		Recall/ Sensitivity	(Specificity)	

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

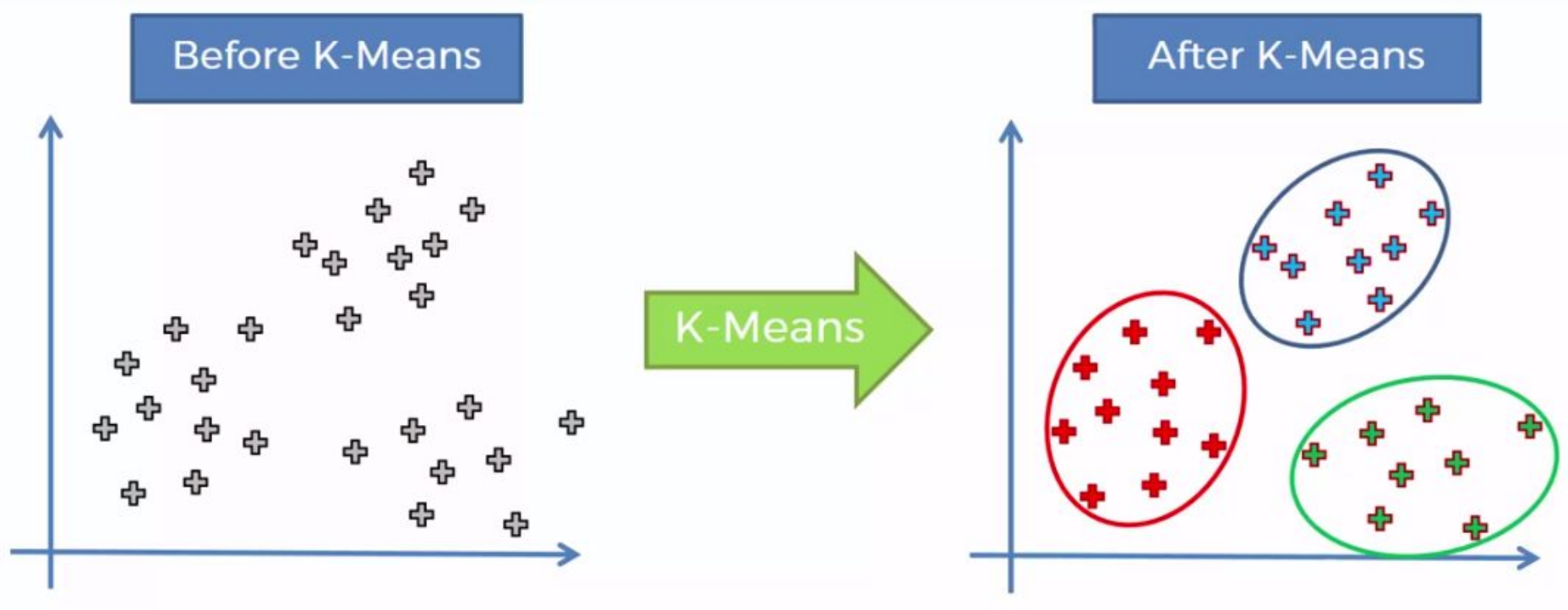
$$\text{Precision} = \frac{TP}{FP + TP}$$

$$\text{Recall} = \frac{TP}{FN + TP}$$

$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Aprendizado Não-Supervisionado - Clustering

K-means



K-means - Algoritmo

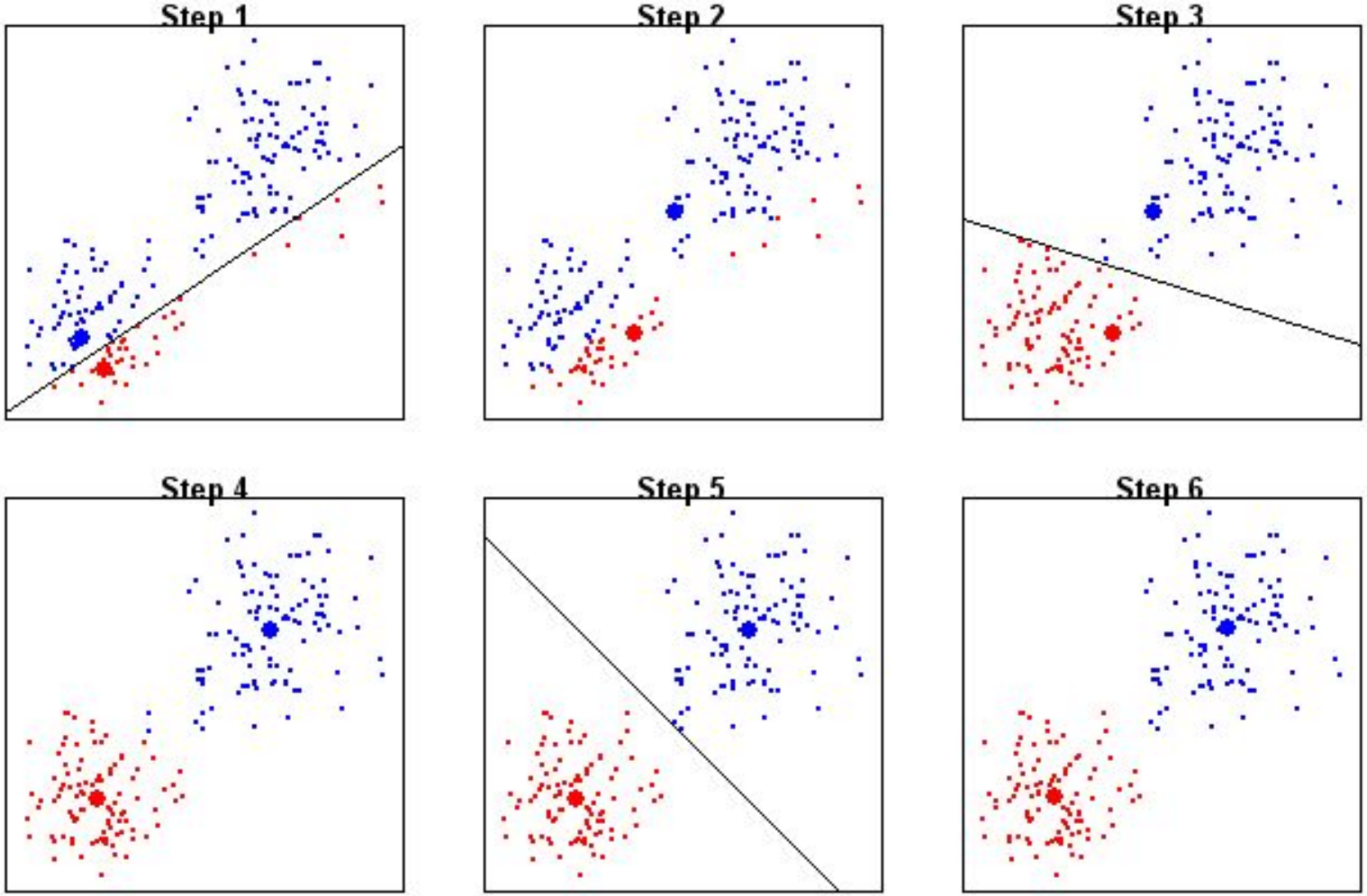
1 - Escolha o número K de clusters;

2 - Selecione K pontos aleatoriamente -> Centróides;

3 - Para cada ponto, relacione-o com o centróide mais próximo;

4 - Atualize os novos centróides (média entre os pontos relacionados);

5 - Relacione novamente os pontos ao centróide mais próximo. Se não for possível atualizar os centróides, FIM, senão vá para o passo 4.


Step 1
Step 2
Step 3
Step 4
Step 5
Step 6

K-means - Visualização

http://shabal.in/visuals/kmeans/3.html

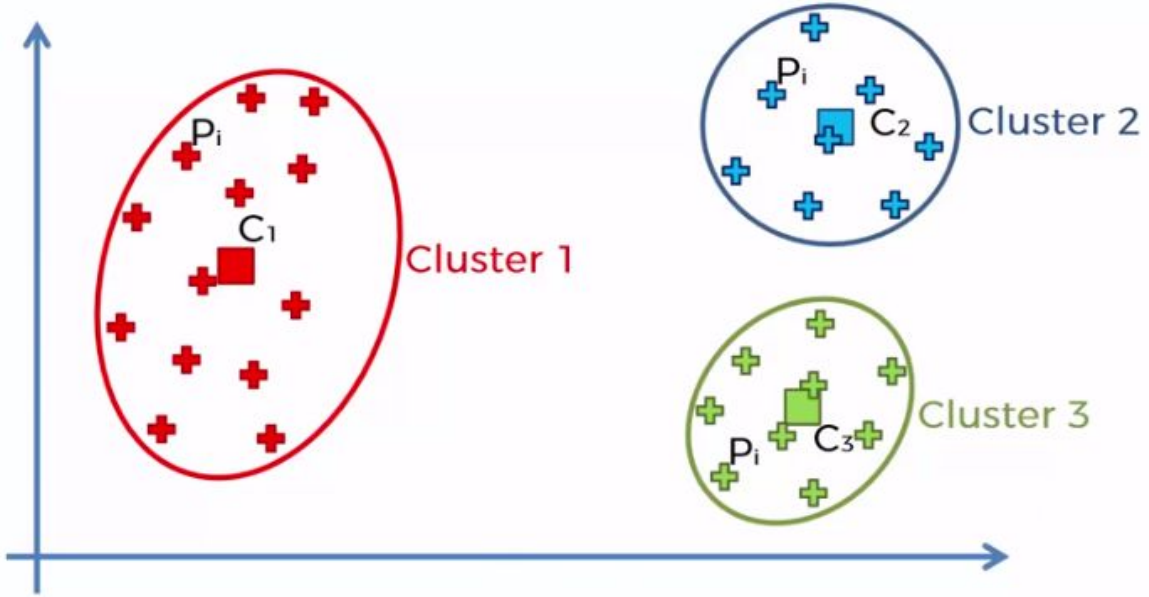
- Atenção: Armadilha da Inicialização - dependendo do ponto onde os centróides forem inicializados, os clusters podem ser diferentes. Para isso, usa-se heurísticas de inicialização - Kmeans++.

K-means - Escolha do K

- Métrica utilizada: WCSS (Within-Cluster Sum of Squares) deve ser mínima

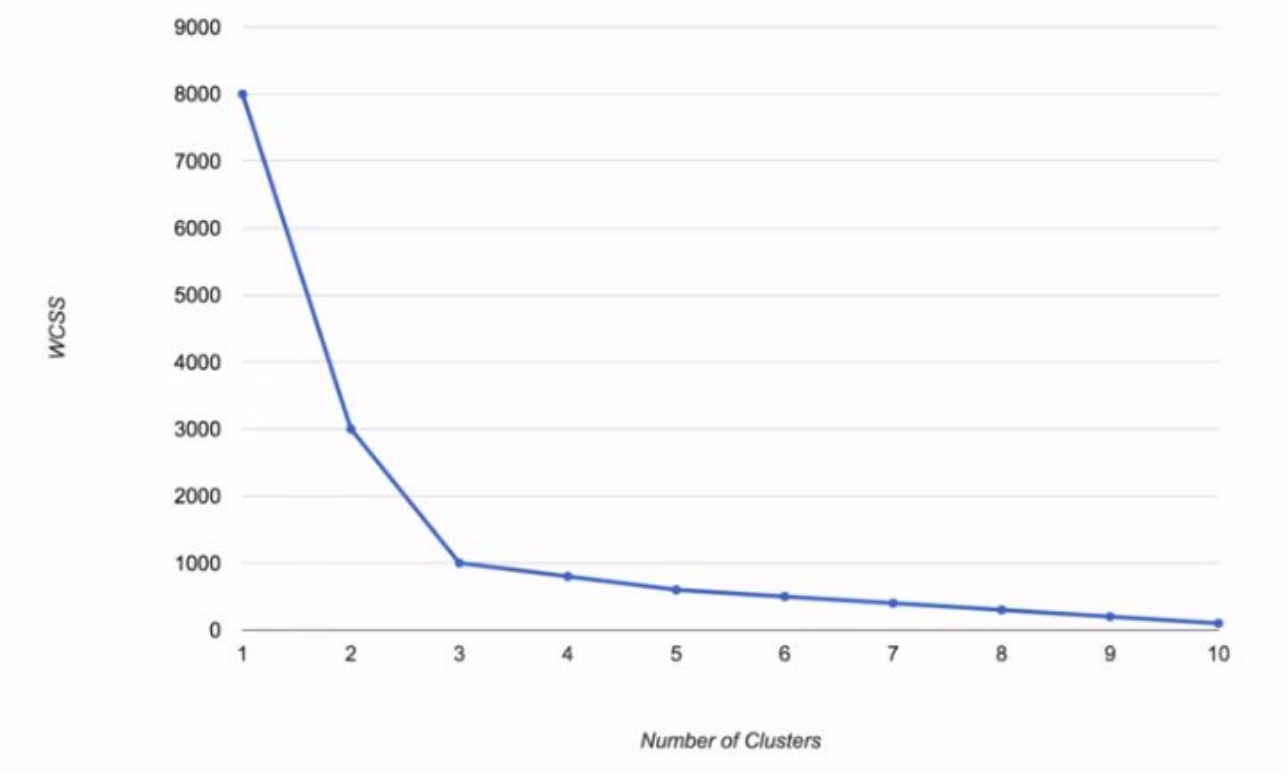
$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

K-means - Escolha do K

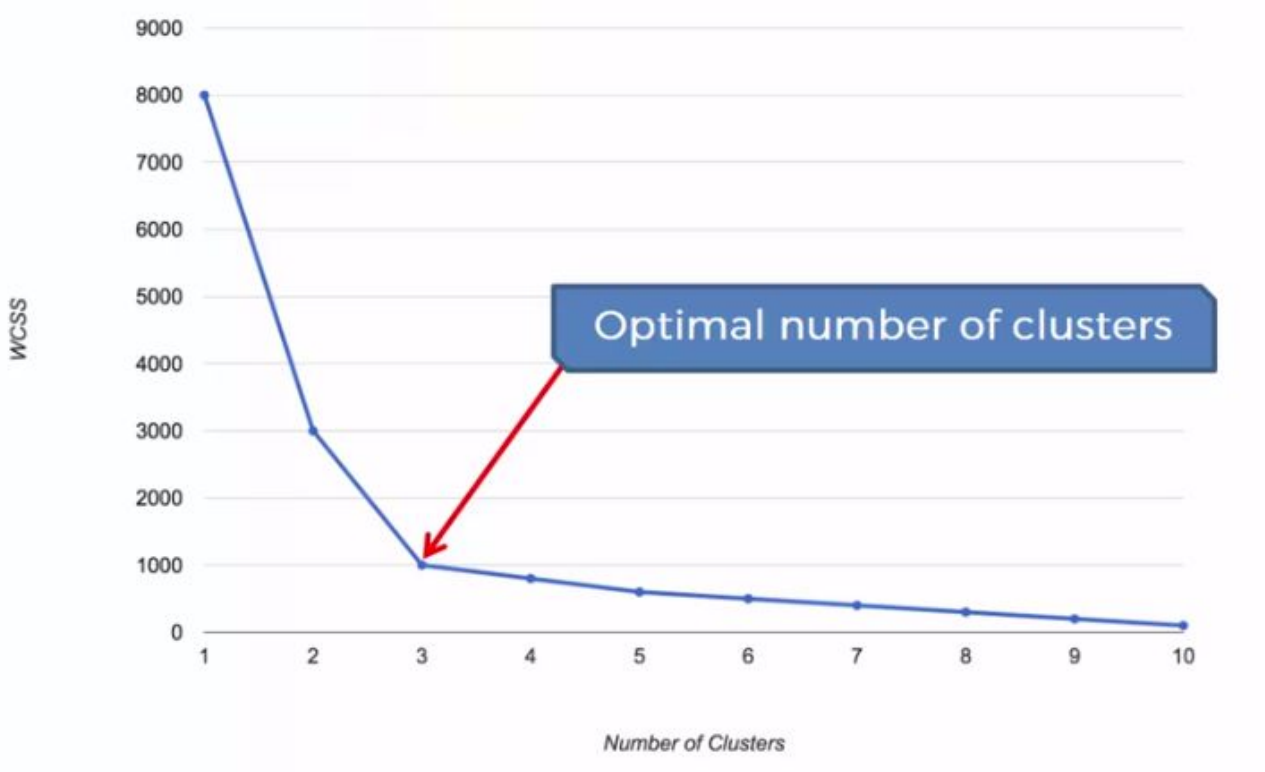


$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

K-means - Escolha do K - Método do Cotovelo



K-means - Escolha do K - Método do Cotovelo



Hierarchical Cluster - Aglomerativo* e Divisivo

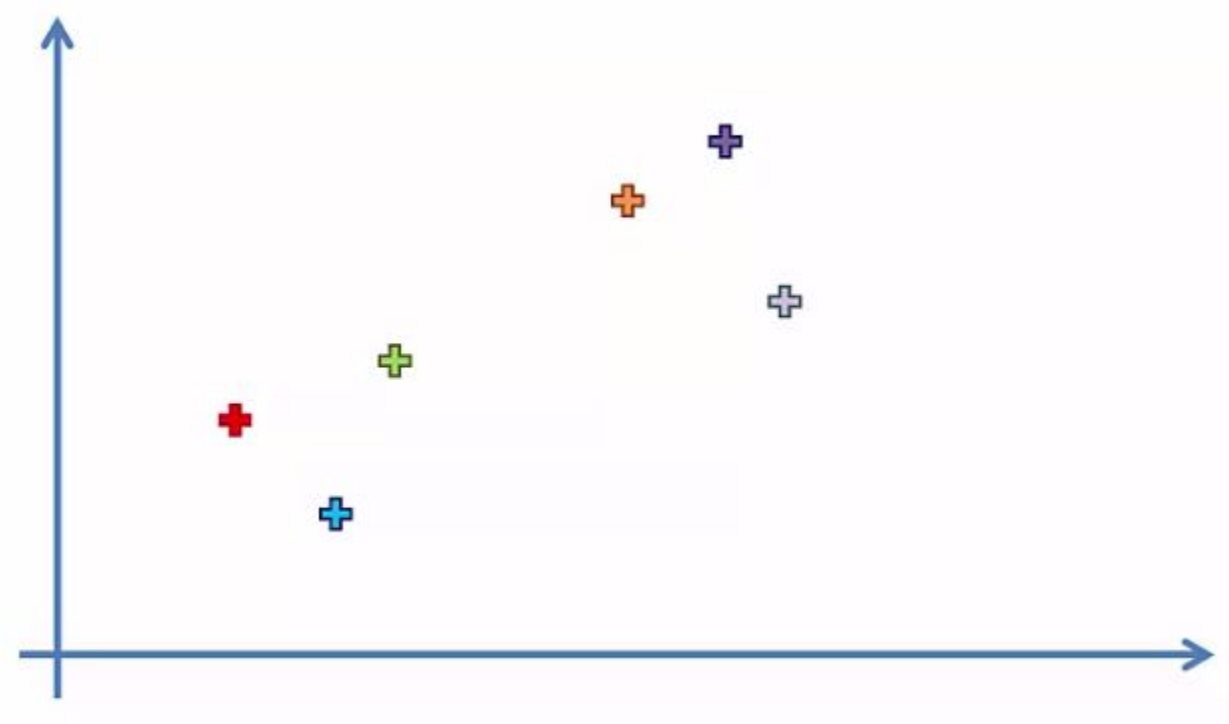
1 - Cada ponto do dataset é um cluster -> formar N clusters;

2 - Pegue os dois pontos mais próximos e forme um cluster -> Isso forma N-1 clusters;

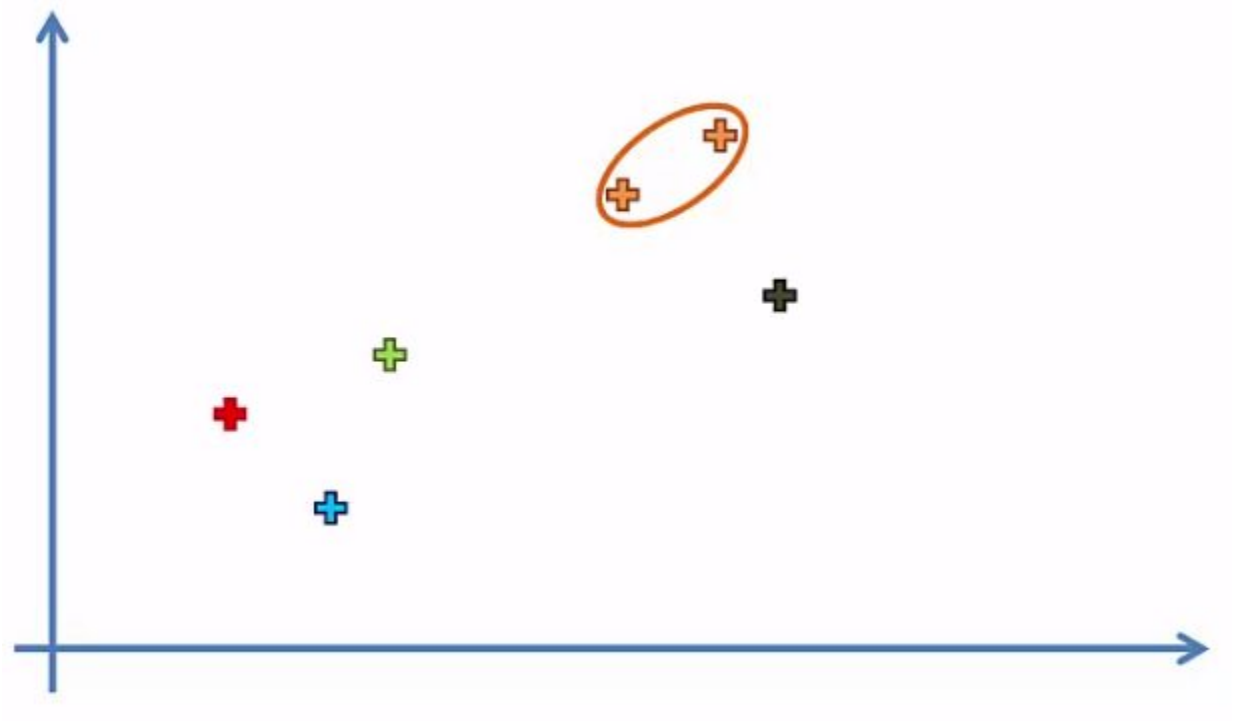
3 - Pegue os dois clusters mais próximos e forme um cluster -> Isso forma N-2 clusters;

4 - Repita 3 até ter apenas um cluster.

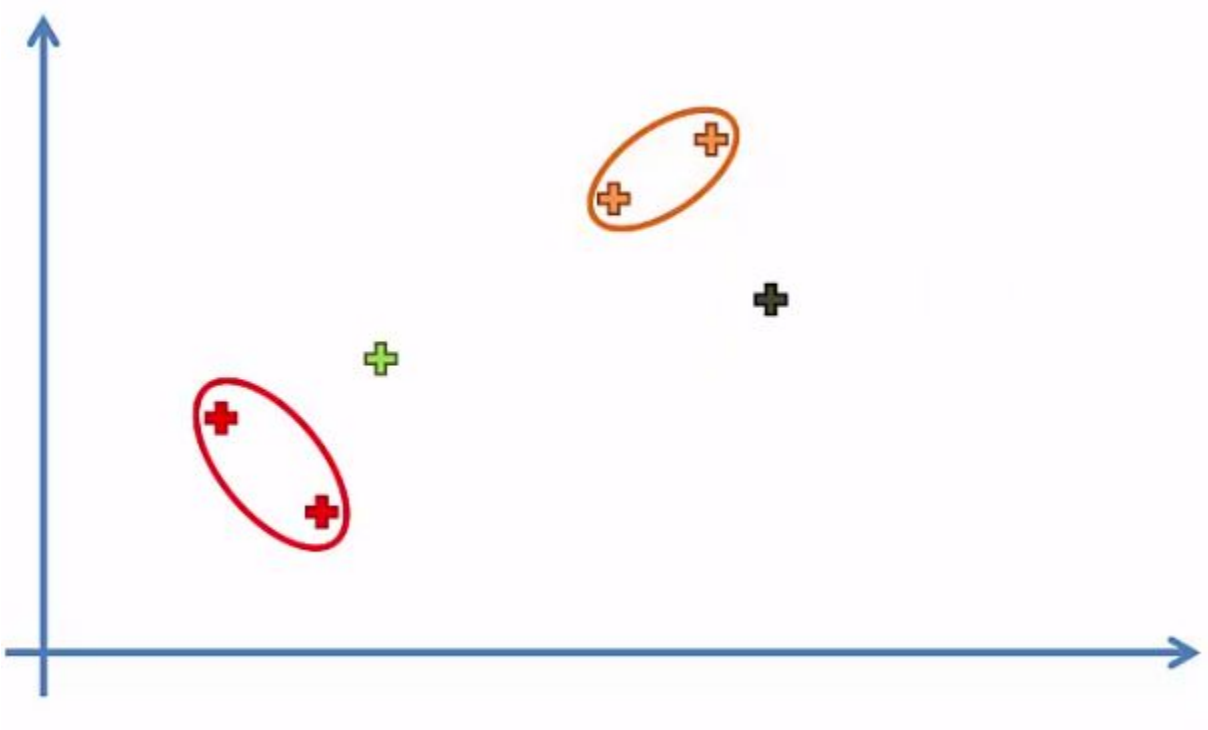
Hierarchical Cluster - Passo 1



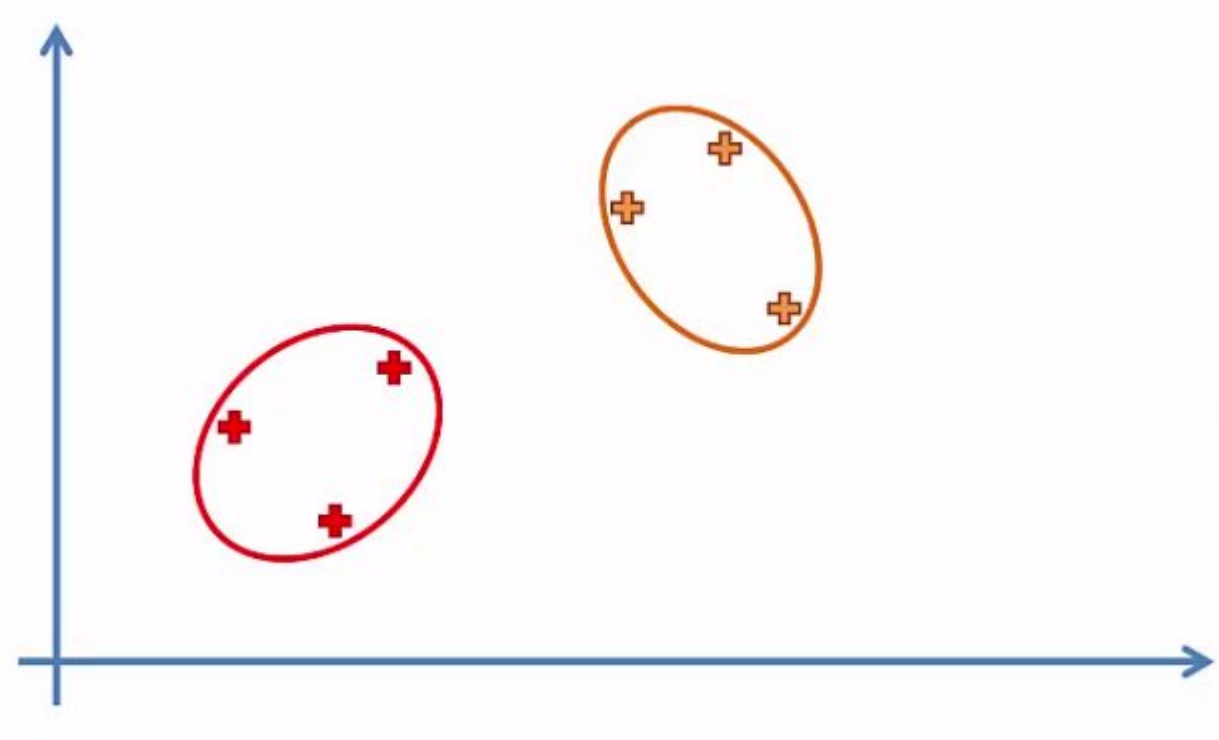
Hierarchical Cluster - Passo 2



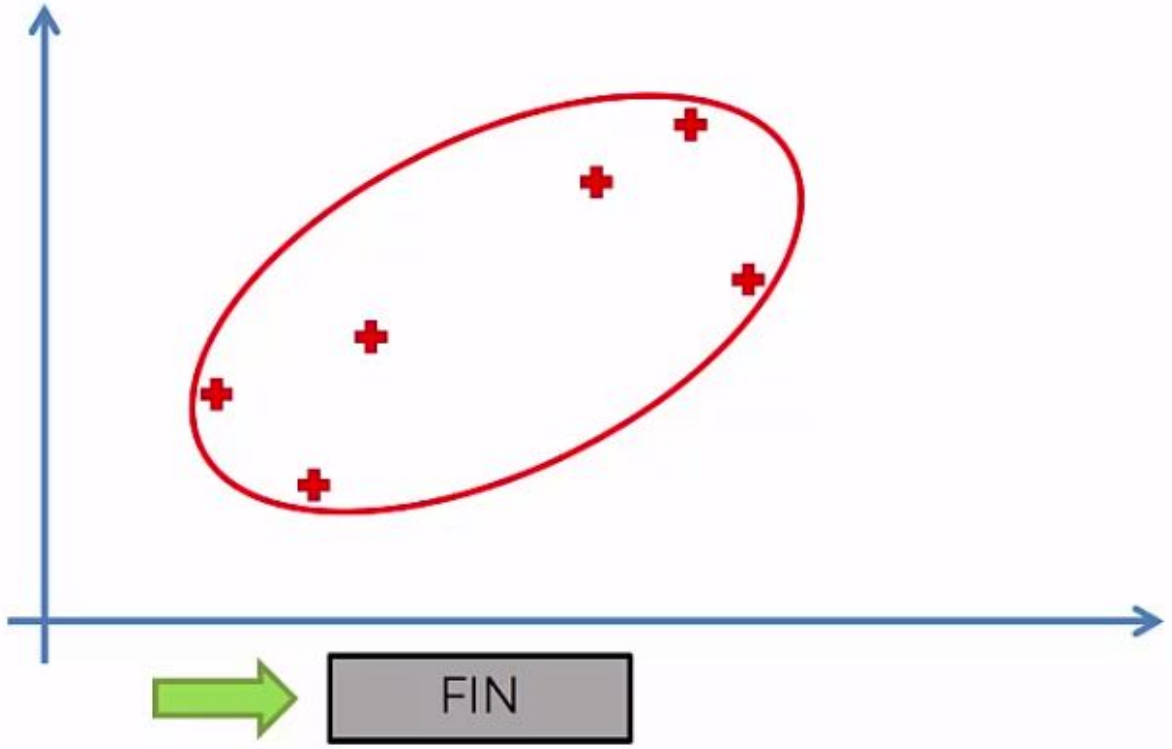
Hierarchical Cluster - Passo 3



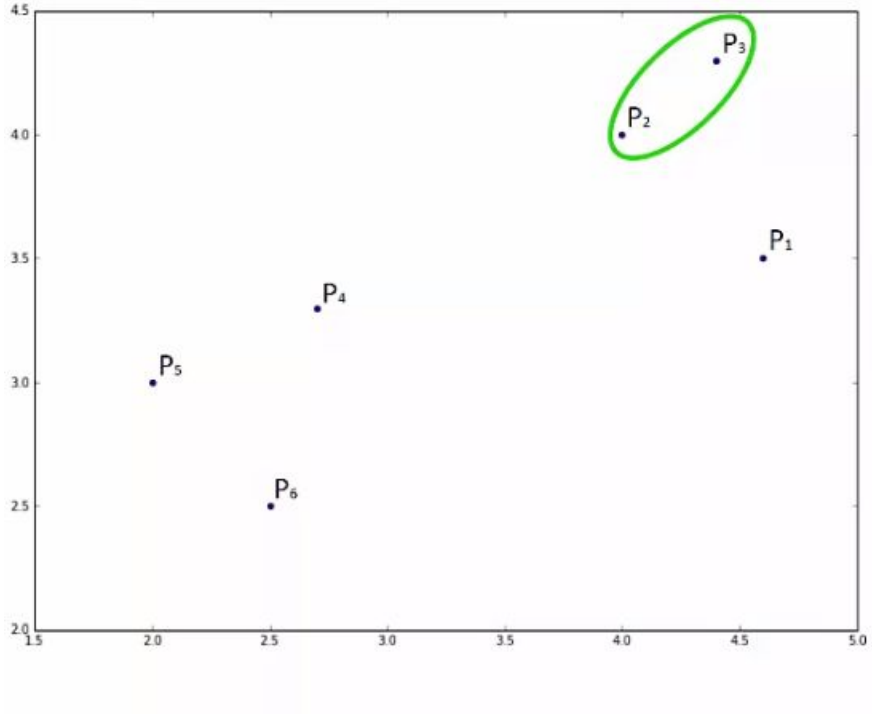
Hierarchical Cluster - Passo 4

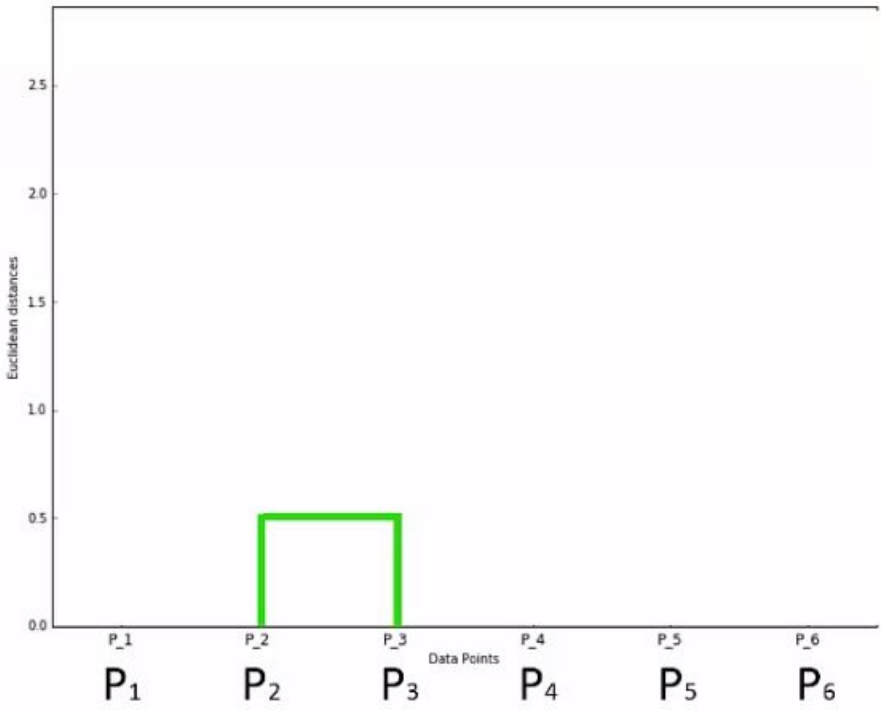


Hierarchical Cluster - Passo 4

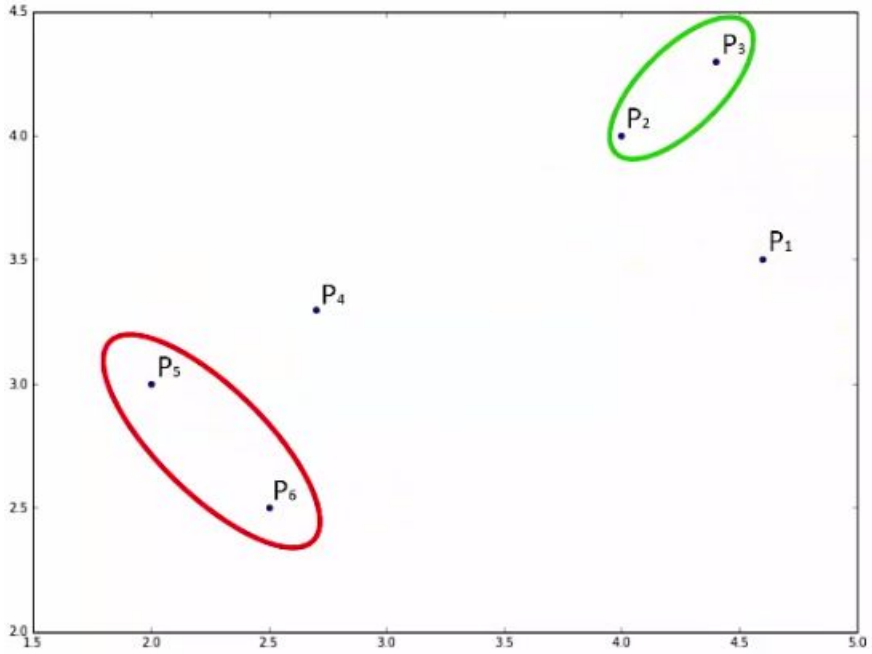


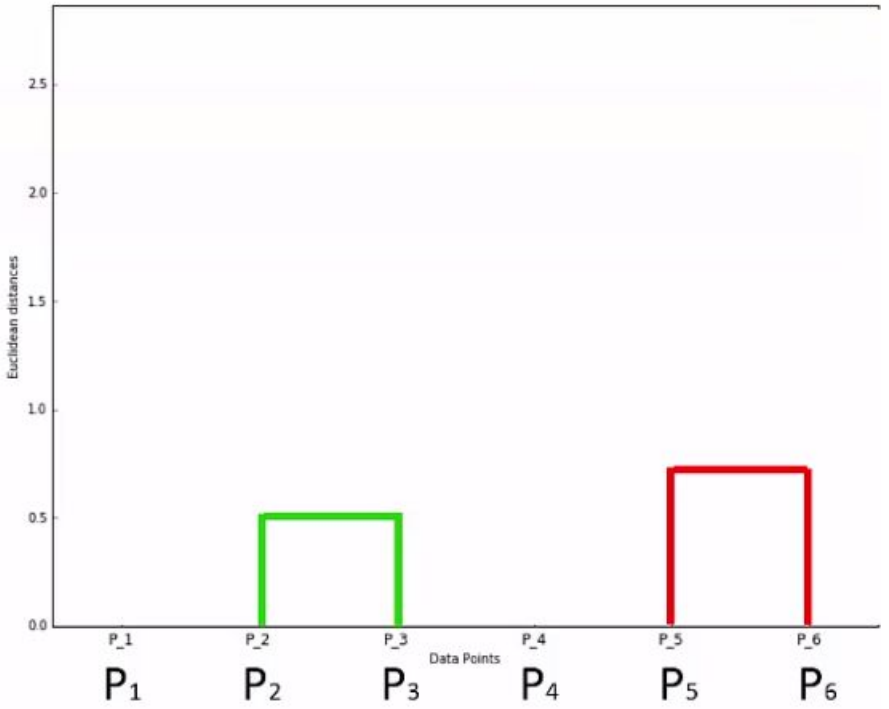
Hierarchical Cluster - Dendogramas



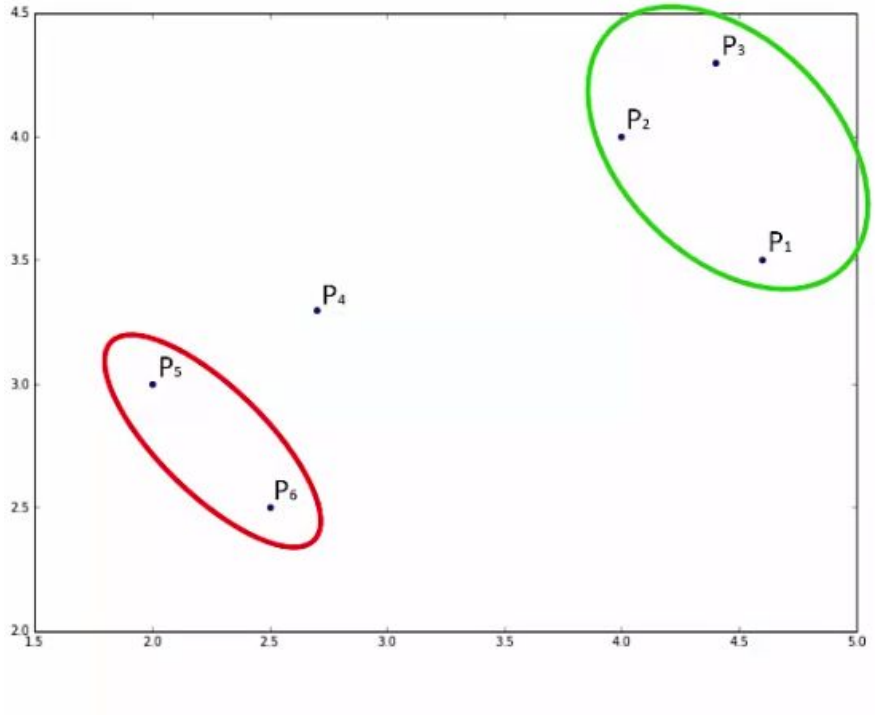


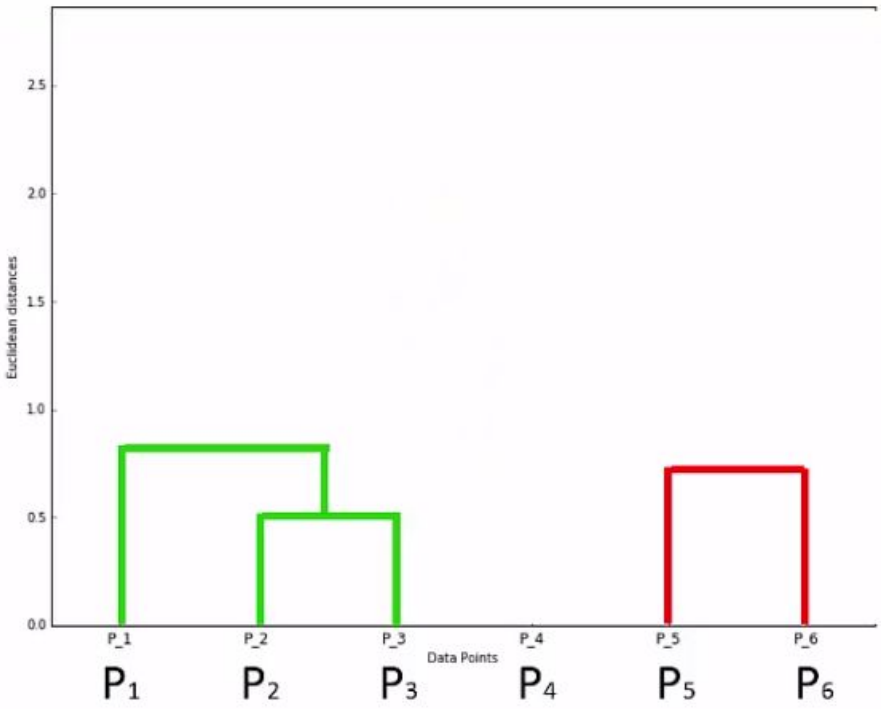
Hierarchical Cluster - Dendogramas



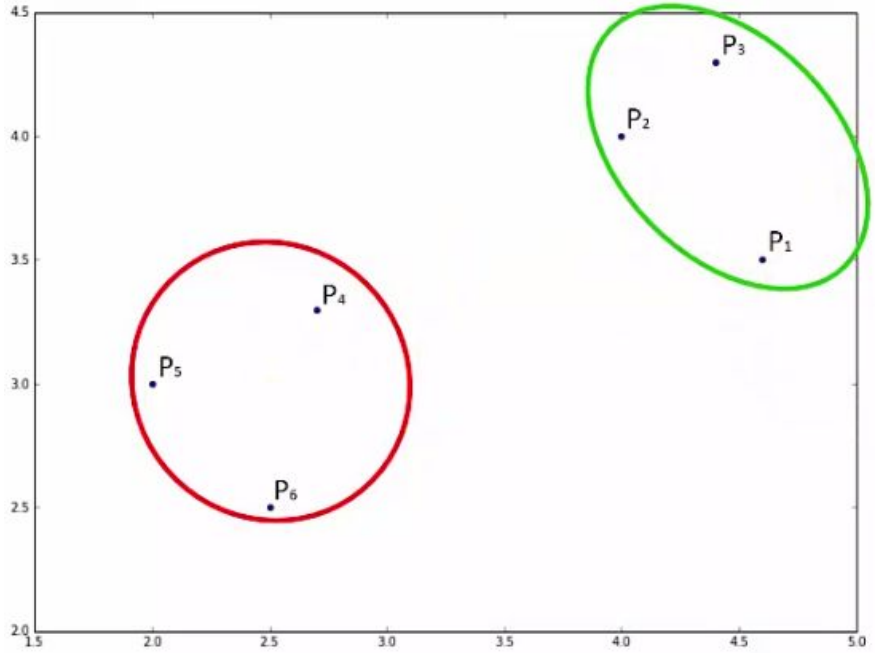


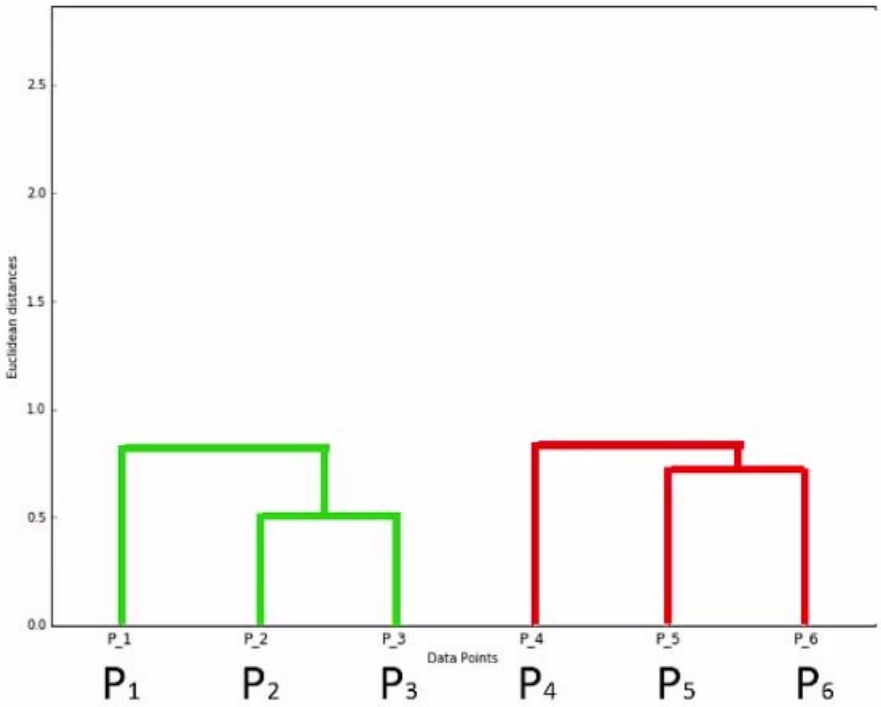
Hierarchical Cluster - Dendogramas



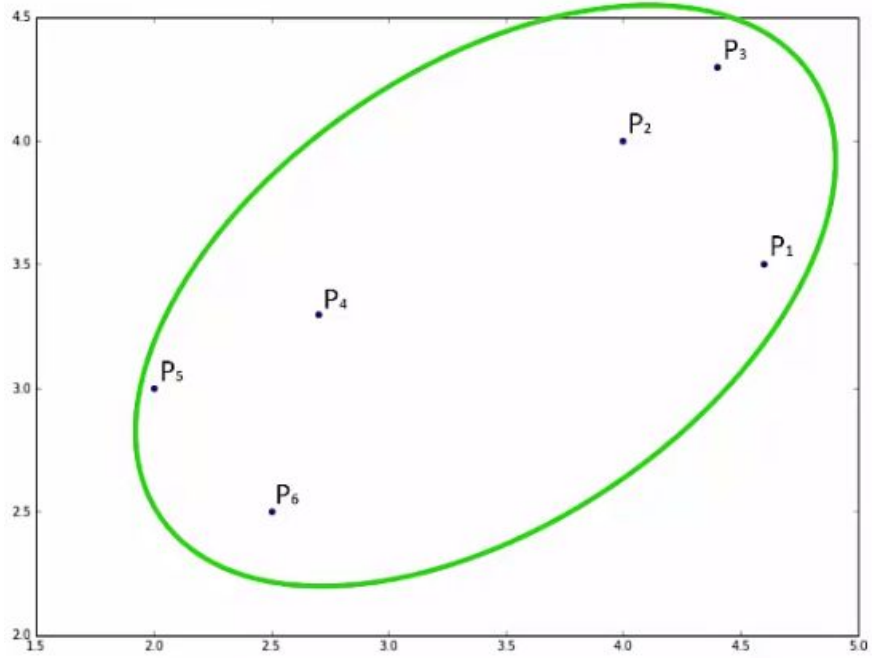


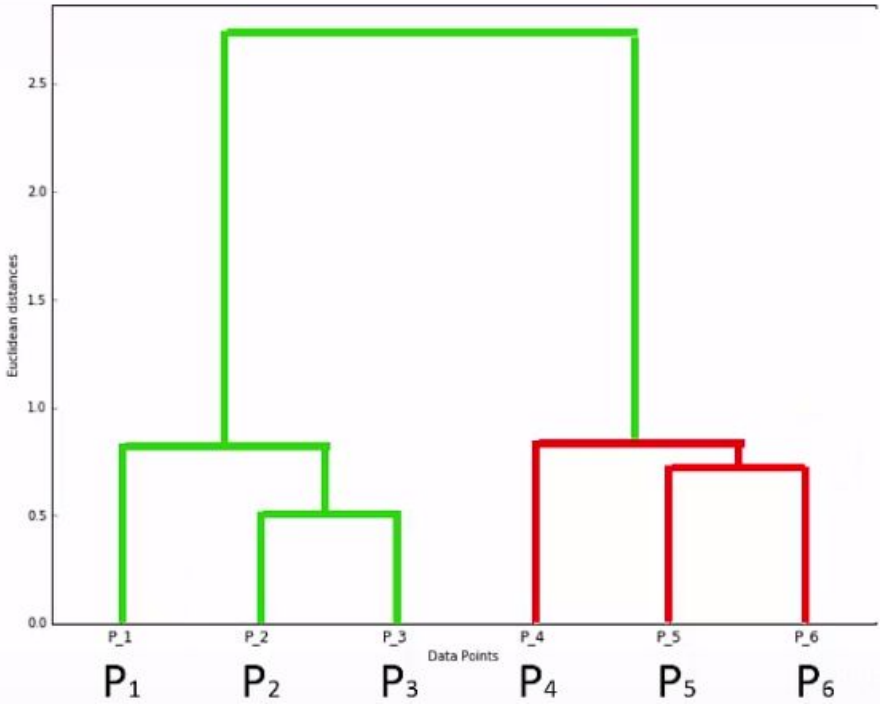
Hierarchical Cluster - Dendogramas





Hierarchical Cluster - Dendogramas



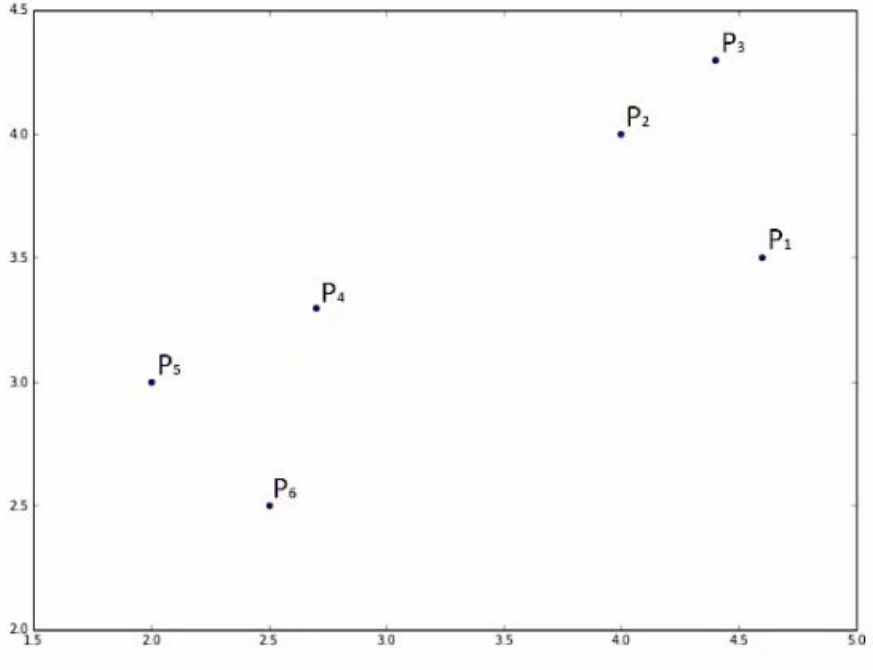


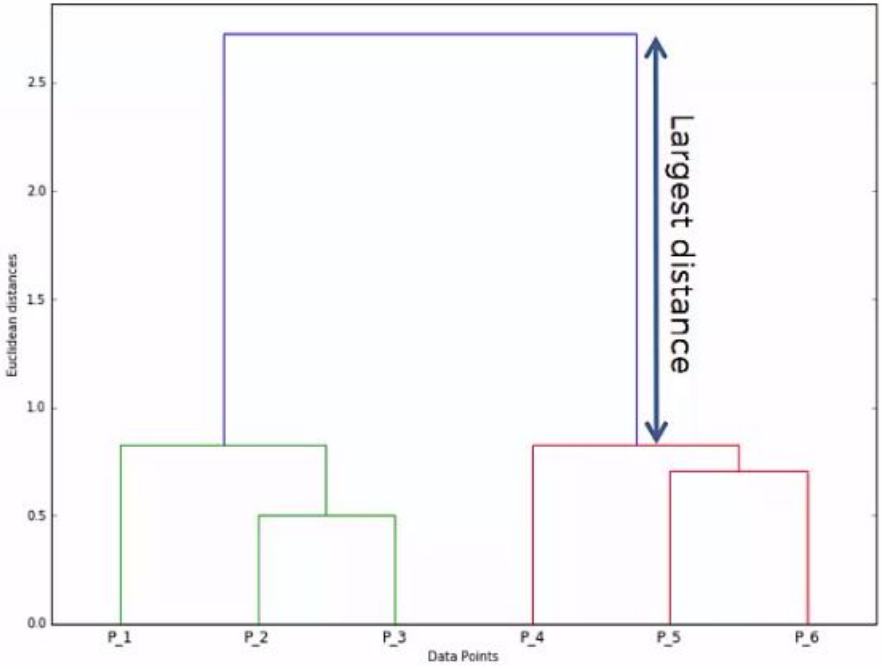
Hierarchical Cluster - Dendogramas

- Mas como decidir o número de clusters?

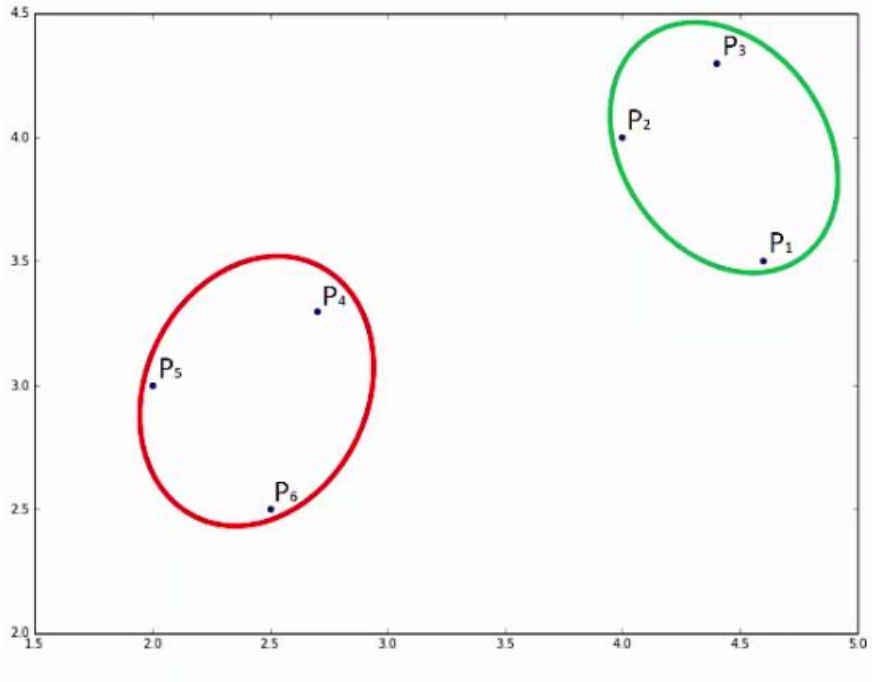
Simples. Basta encontrar a maior linha vertical que não cruza nenhuma linha horizontal (Rule of thumb).

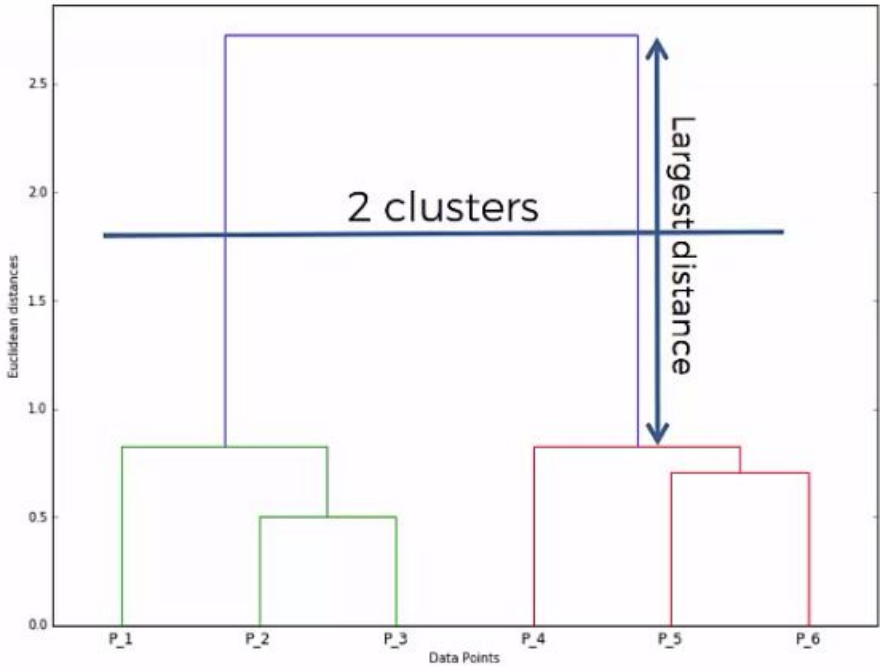
Hierarchical Cluster - Dendogramas





Hierarchical Cluster - Dendogramas

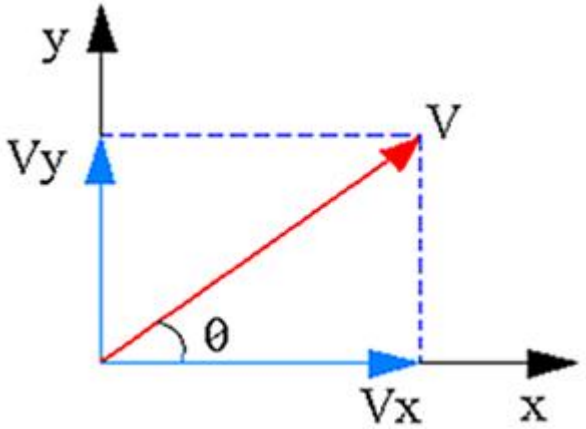




Aprendizado Não-Supervisionado - Redução de Dimensionalidade

PCA - Principal Component Analysis

- PCA é um método de Feature Extraction;
- Das m variáveis independentes, PCA extrai p < m novas variáveis independentes que explicam melhor a variância dos dados através de um método de projeção, independente da variável dependente, o que configura um problema de Aprendizado Não-Supervisionado.

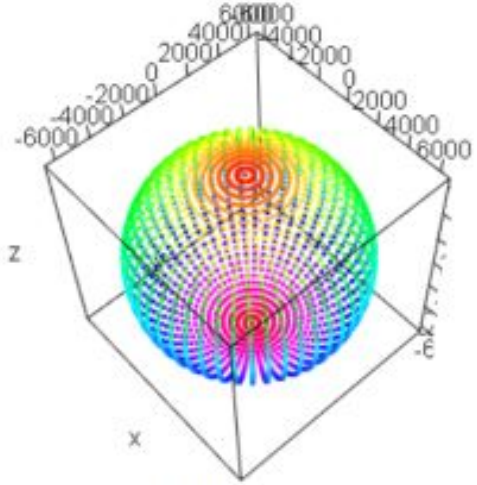


PCA - Principal Component Analysis

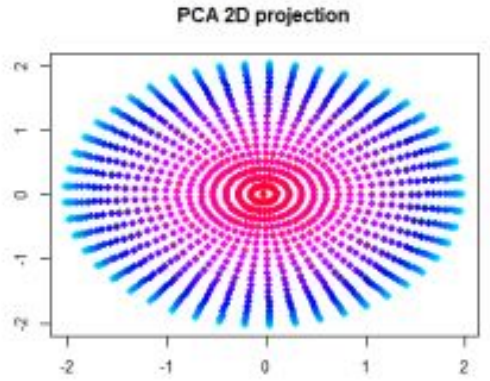
- Desta forma é possível reduzir a dimensão do seu dataset sem comprometer a qualidade das features e suas correlações.

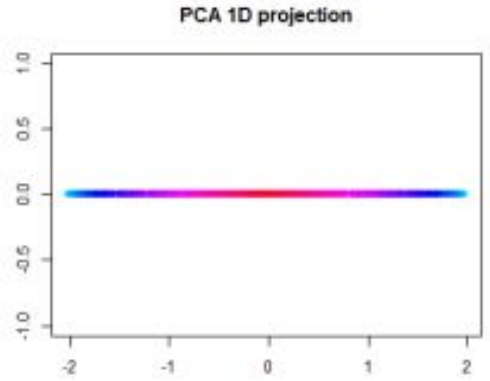
- Pode ser empregado sempre que for necessário trazer visualização para seus dados, reduzindo assim para 1 ou 2 dimensões, sendo possível realizar o plot. Também pode ser empregado para redução do número de features, aliviando o custo computacional de processamento.

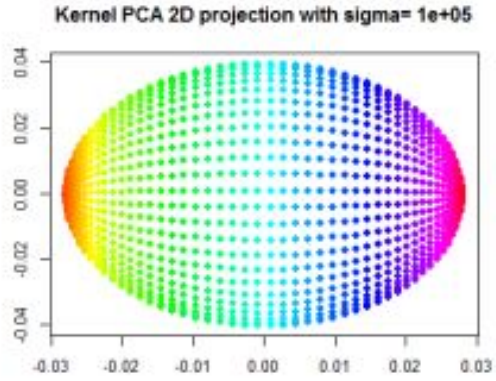
PCA - Principal Component Analysis

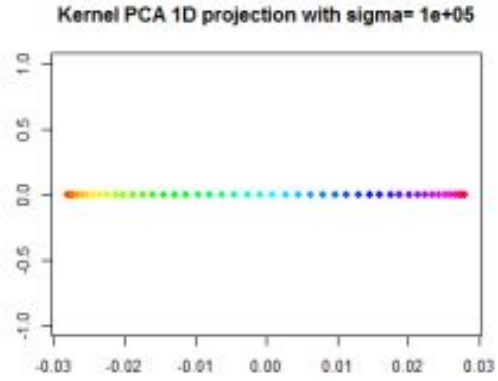


Original 3D globe data





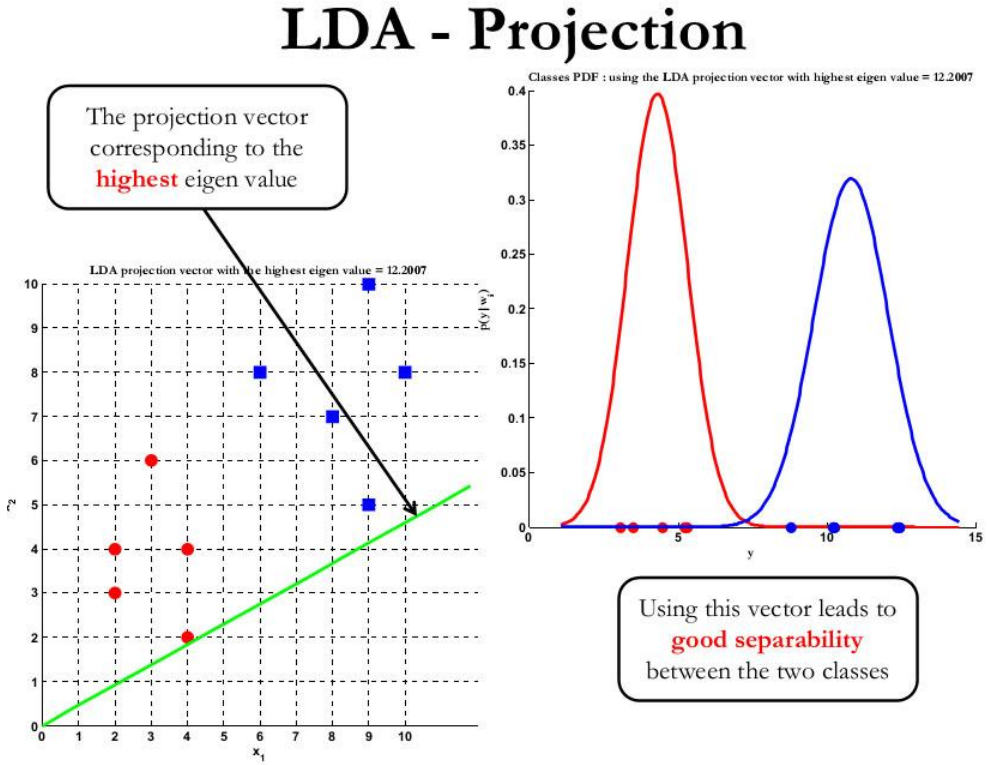




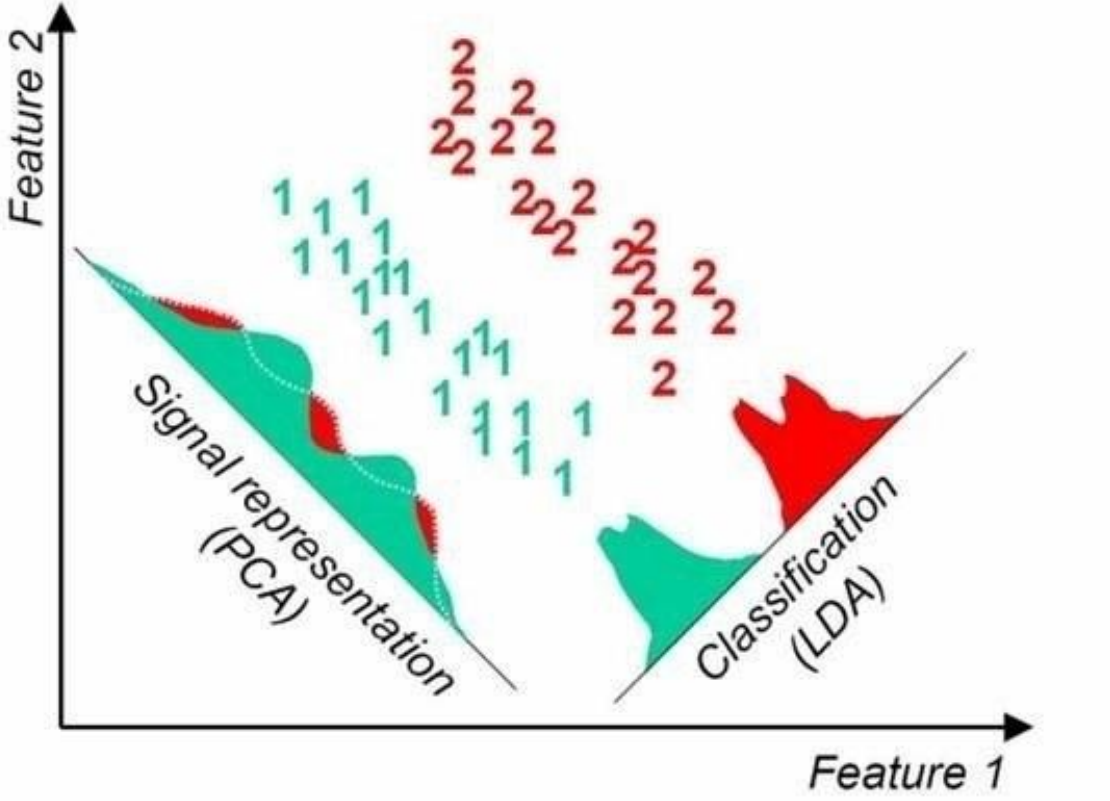
LDA - Linear Discriminant Analysis

- LDA é um método de Feature Extraction;
- Das n variáveis independentes, LDA extrai p < n novas variáveis independentes que mais separam as classes da variável dependente, baseado em estatística.
- As densidades de probabilidades das classes são maximizadas na projeção LDA;
- Deste modo, é considerado um método de Aprendizado Supervisionado*.

LDA - Linear Discriminant Analysis



PCA x LDA

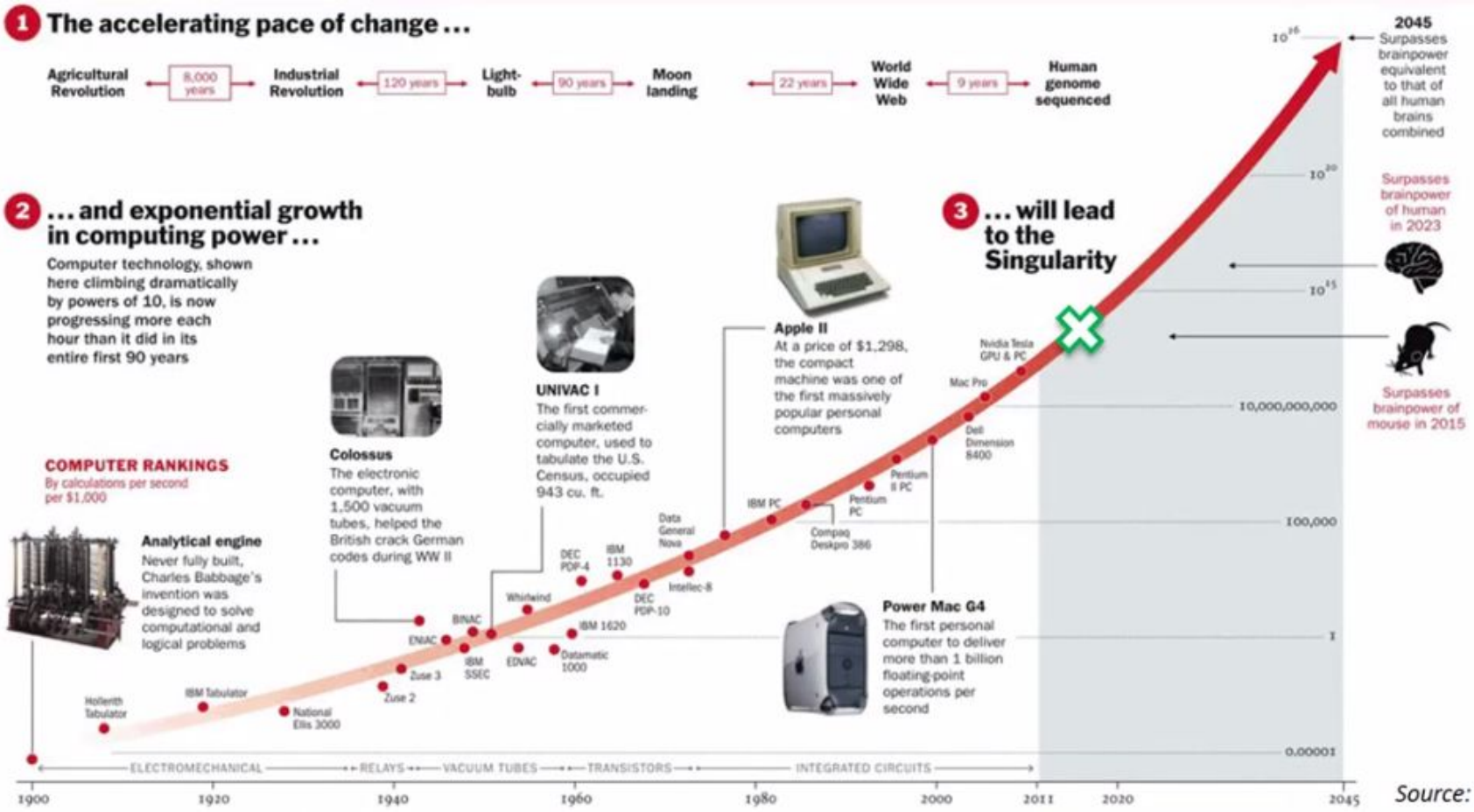

Feature 2
Feature 1
Signal representation (PCA)
Classification (LDA)

Aprendizado Supervisionado
Redes Neurais

Redes Neurais - Contextualização Histórica

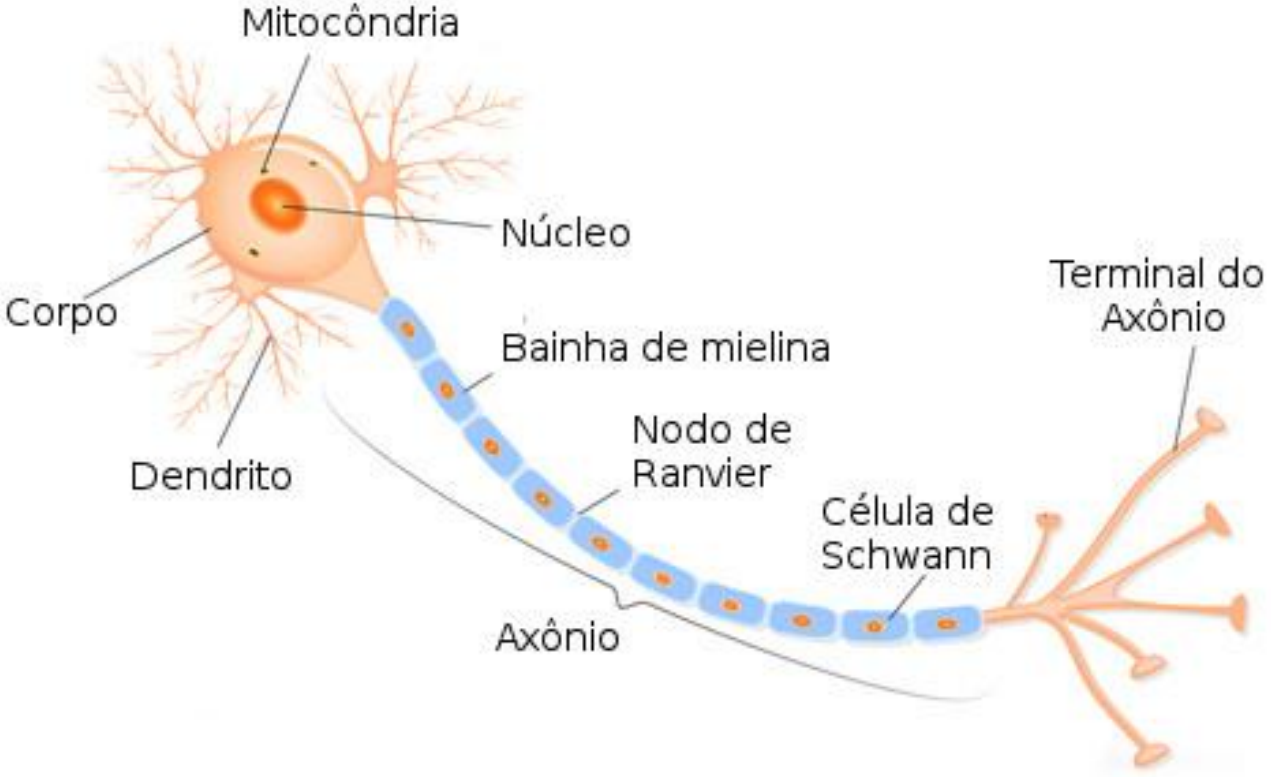
- Primeiro neurônio artificial: 1943, McCulloch & Pitts, chamado de Percéptron.
- Caiu no esquecimento na década de 60 devido ao fato das portas lógicas serem adotadas em detrimento dos neurônios artificiais como unidade básica de processamento (problema do XOR)
- Retornaram na década de 80 com o avanço do desenvolvimento da capacidade de processamento e de memória dos computadores digitais.
- Hardware dedicado: redes neurais profundas acarretaram o desenvolvimento de Unidades Gráficas de Processamento (GPU's) dedicadas para lidar com esse tipo de arquitetura.

Redes Neurais - Hardware dedicado

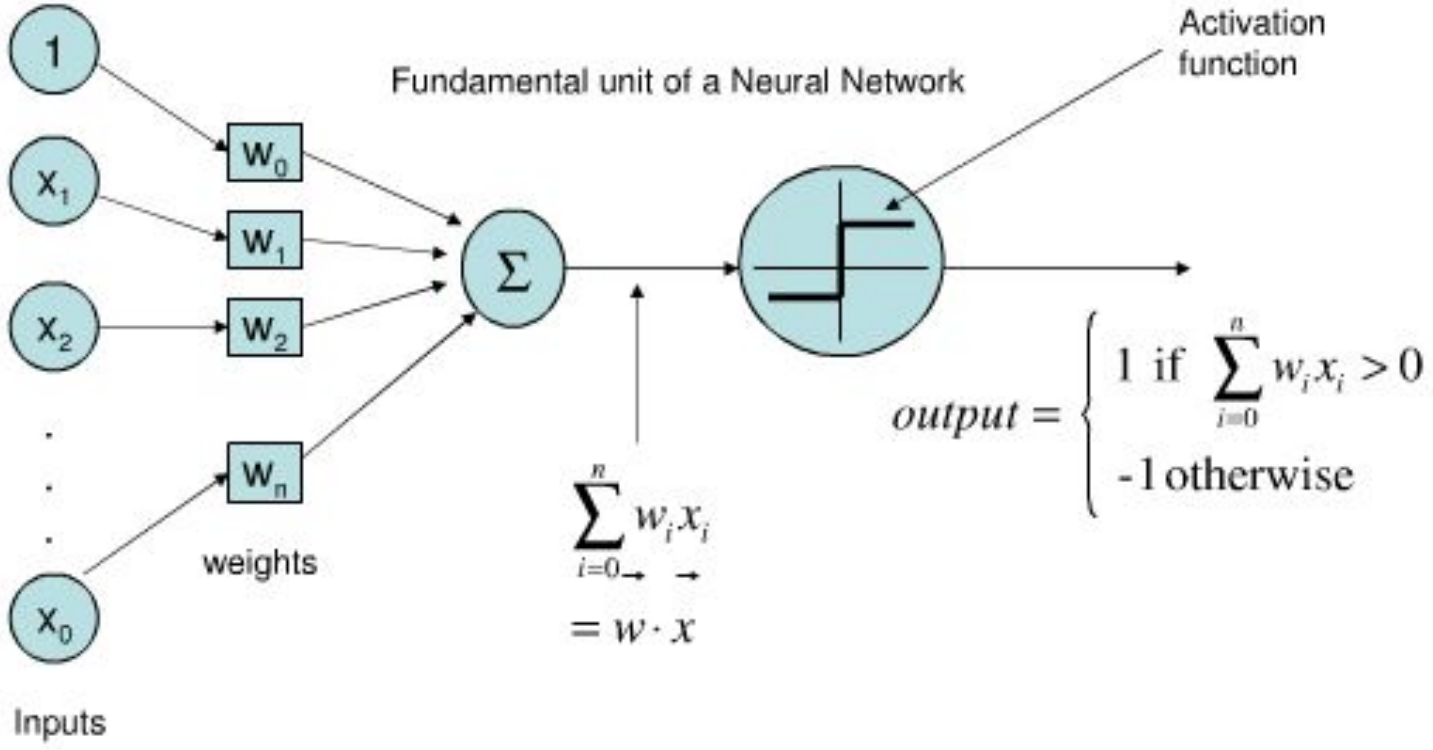


Source: Time Magazine

Redes Neurais - Percéptron, inspiração biológica

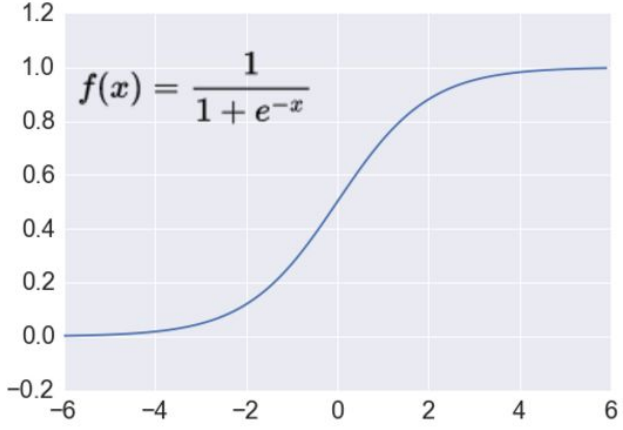


Redes Neurais - Percéptron

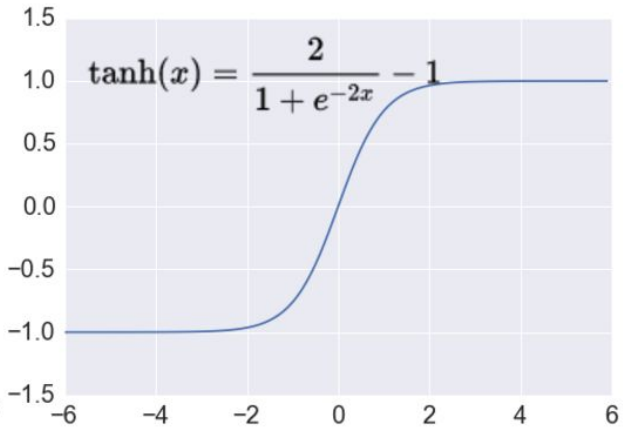


Redes Neurais - Funções de ativação

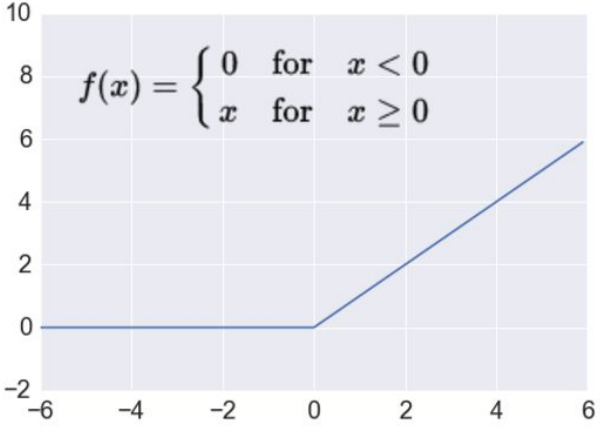
Sigmoid



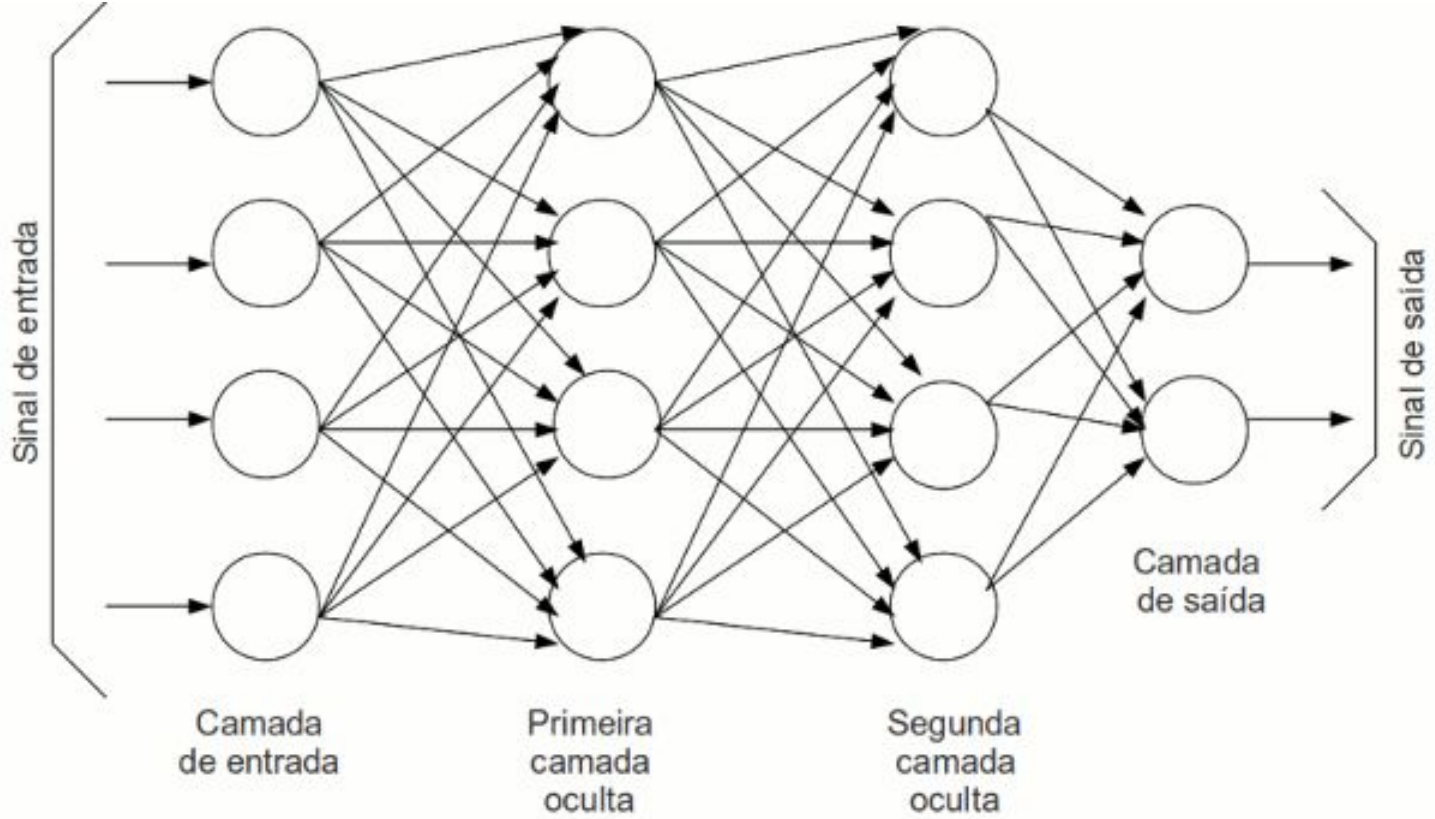
TanH



ReLU



Redes Neurais - Paradigma conexionista



Redes Neurais - Treinando a rede

1 - Inicializar os pesos aleatoriamente (valores entre 0 e 1);

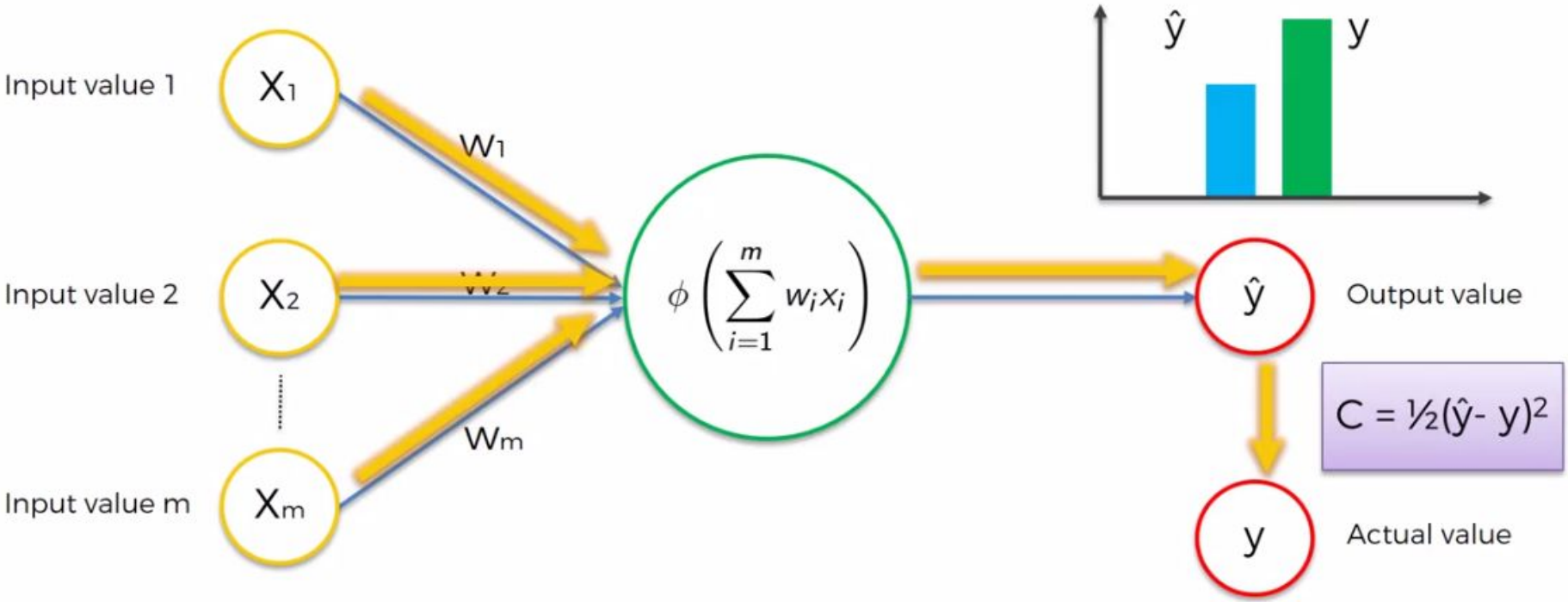
2 - Entrar com os dados na rede, da esquerda para a direita (**Forward propagation**);

3 - Calcular o Erro Quadrático Médio (ou outro erro utilizado);

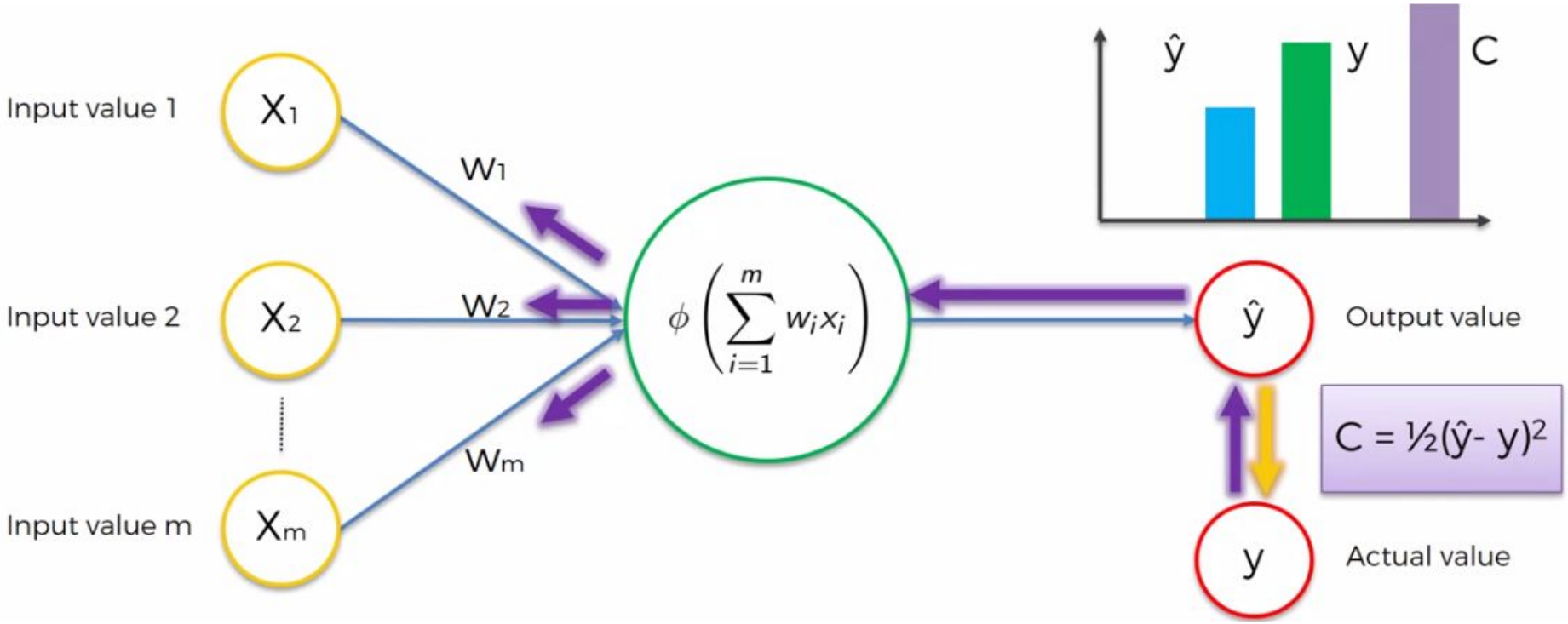
4 - Realizar **Backpropagation** (direita para a esquerda), atualizando cada um dos pesos;

5 - Repita 2 a 5 até convergência.

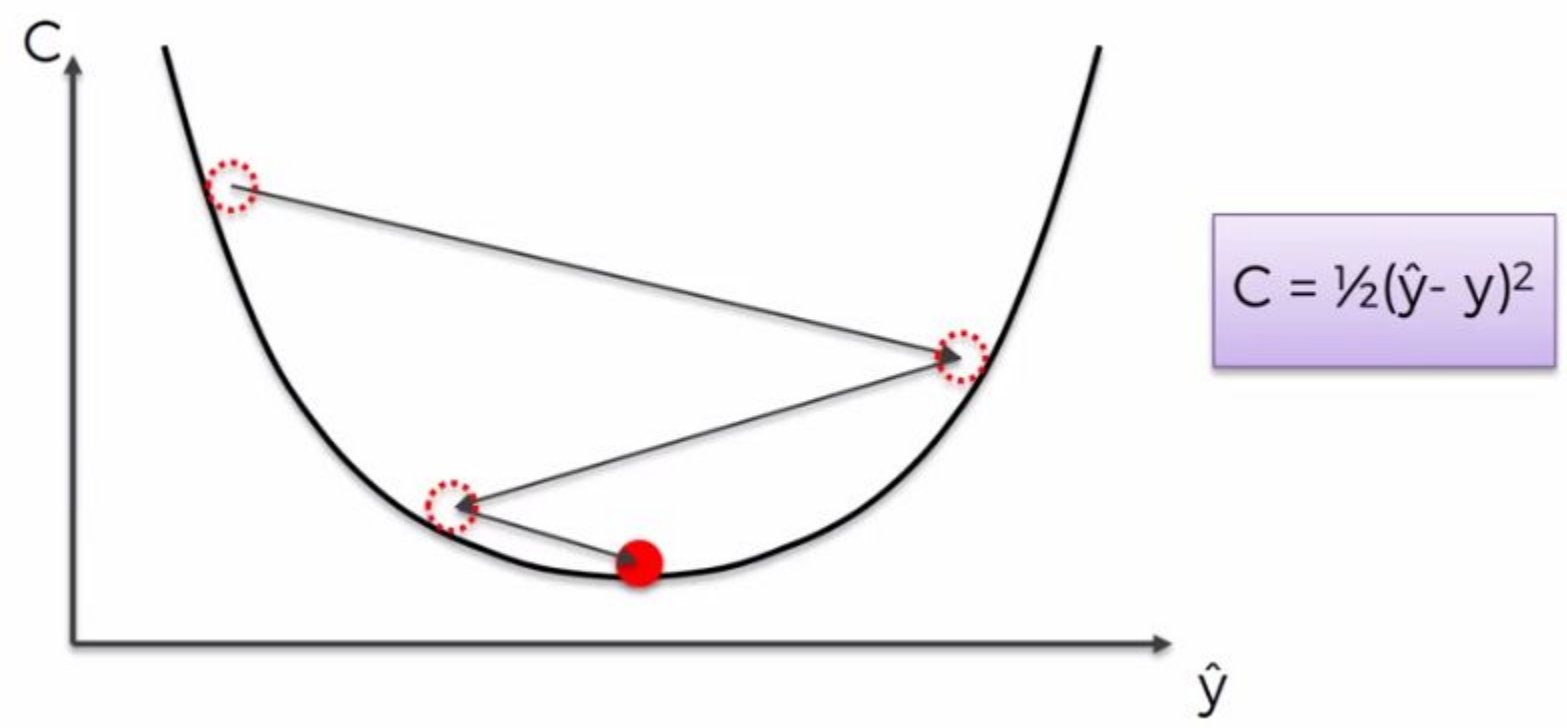
Redes Neurais - Forward Propagation



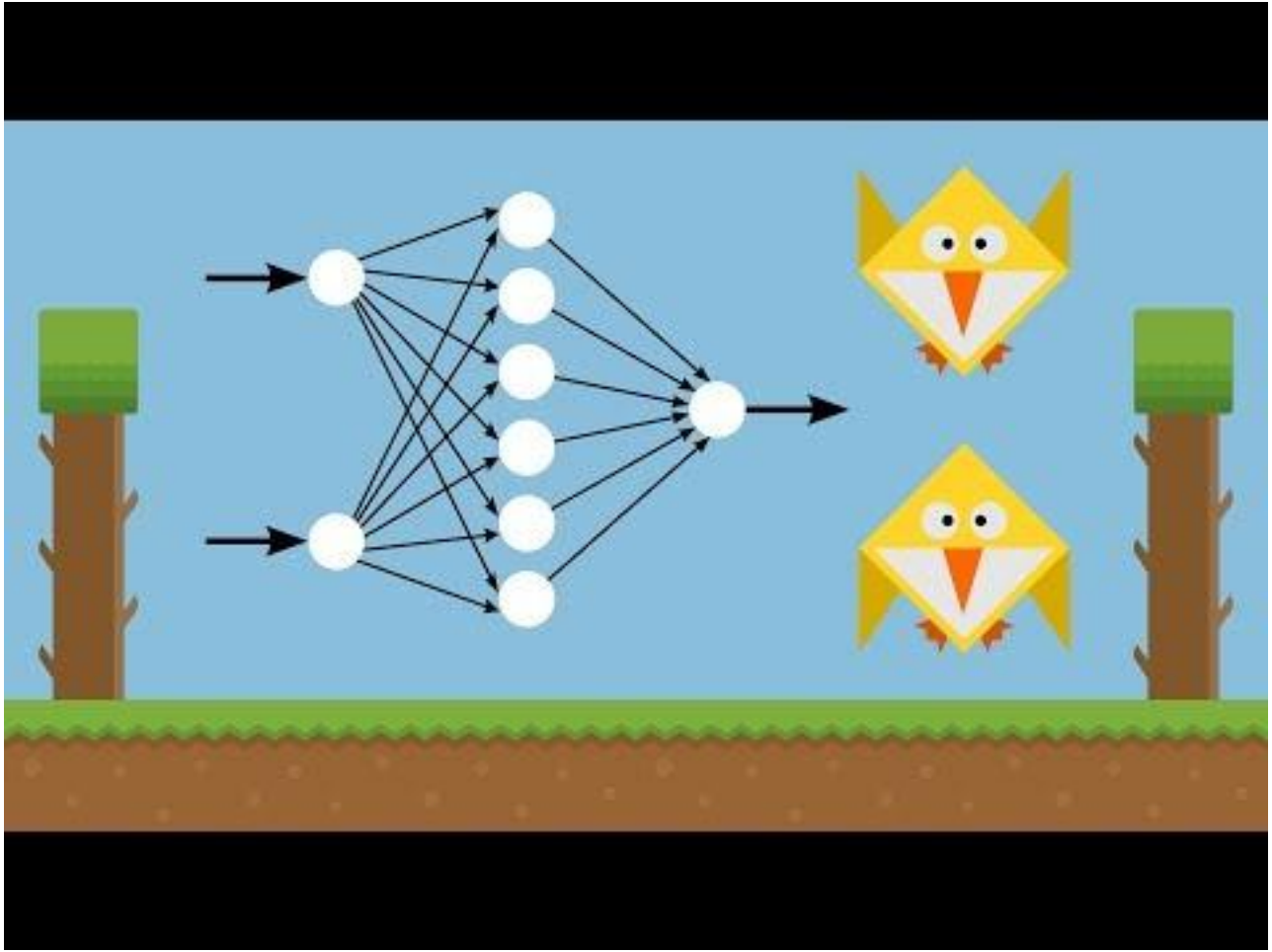
Redes Neurais - Backpropagation

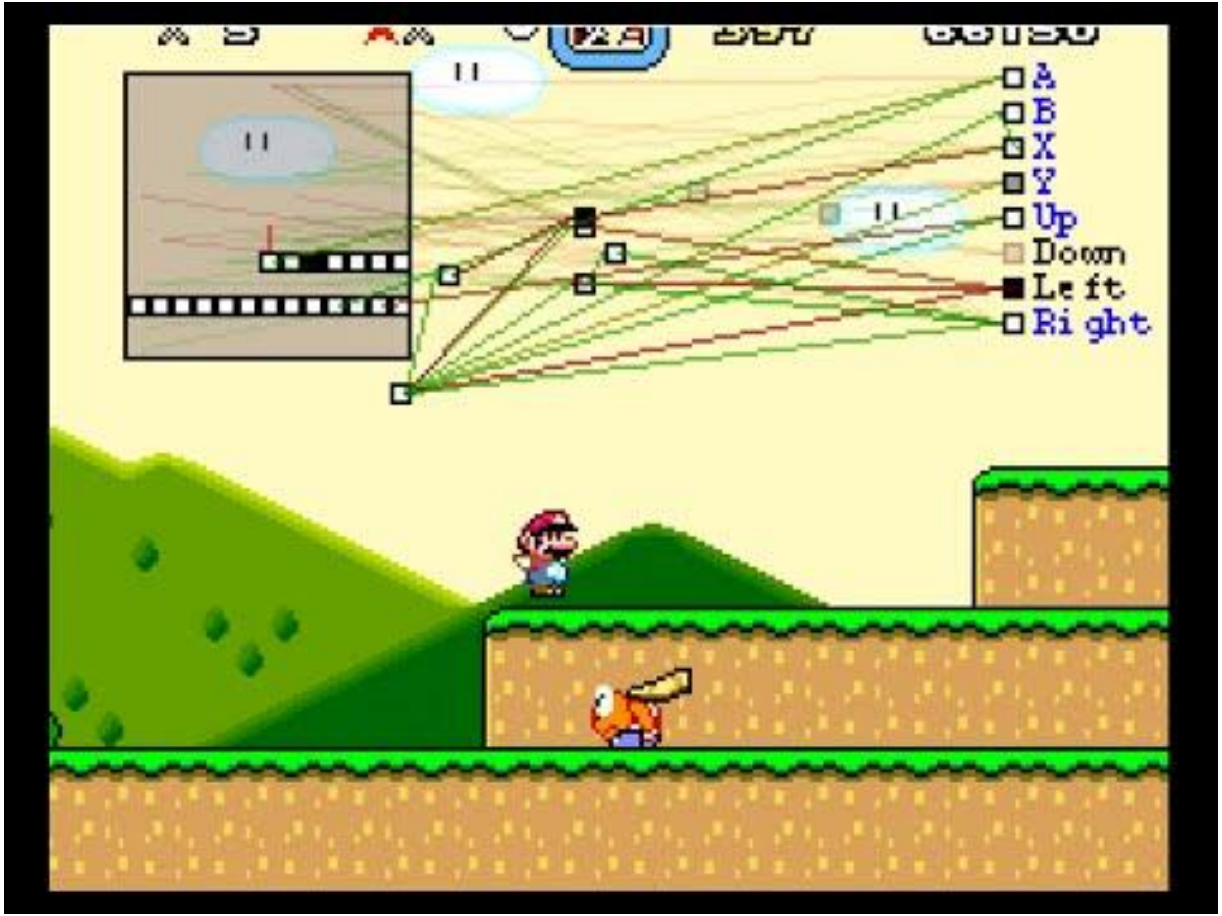


Redes Neurais - Descida de Gradiente



$$C = \frac{1}{2}(\hat{y} - y)^2$$




A
B
X
Y
Up
Down
Left
Right

Redes Neurais Convolucionais

- Redes Neurais Profundas: o nome é dado devido à grande quantidade de camadas (> 5), e à grande quantidade de neurônios artificiais.
- Camadas: Convolução, Max Pooling, Dropout, Flatten e Full Connected.
- Sua grande vantagem está na extração automática de atributos nas camadas de convolução e maxpooling, não sendo necessário realizar previamente esta etapa.
- Grande processamento e custo computacional - Recomenda-se uso de GPUs.

Redes Neurais Convolucionais

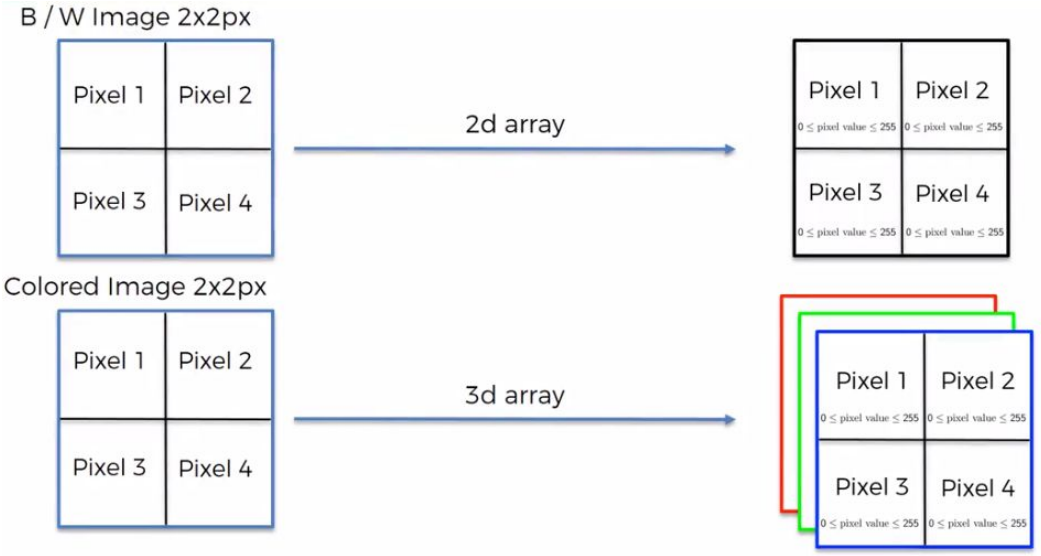
- O grande forte de redes neurais profundas é em reconhecimento de imagem e visão computacional.
- Suas principais aplicações são: Sistemas de veículos autônomos, reconhecimento de voz e imagem, classificação de objetos em imagens, geração automática de texto, análise de comportamento em vídeo, jogos automatizados, entre muitas outras.
- Trabalharemos com foco em reconhecimento de imagens.

Redes Neurais Convolucionais - Imagens

A representação de imagens em um computador é dada por uma matriz de pixels, como na figura abaixo

88	82	84	88	85	83	80	93	102
88	80	78	80	80	78	73	94	100
85	79	80	78	77	74	65	91	99
38	35	40	35	39	74	77	70	65
20	25	23	28	37	69	64	60	57
22	26	22	28	40	65	64	59	34
24	28	24	30	37	60	58	56	66
21	22	23	27	38	60	67	65	67
23	22	22	25	38	59	64	67	66

Representação em pixels



Representação em RGB

Redes Neurais Convolucionais - Convolução

Como dito anteriormente, a camada de convolução é responsável pela extração de atributos das imagens. Isto é feito pela operação de convolução

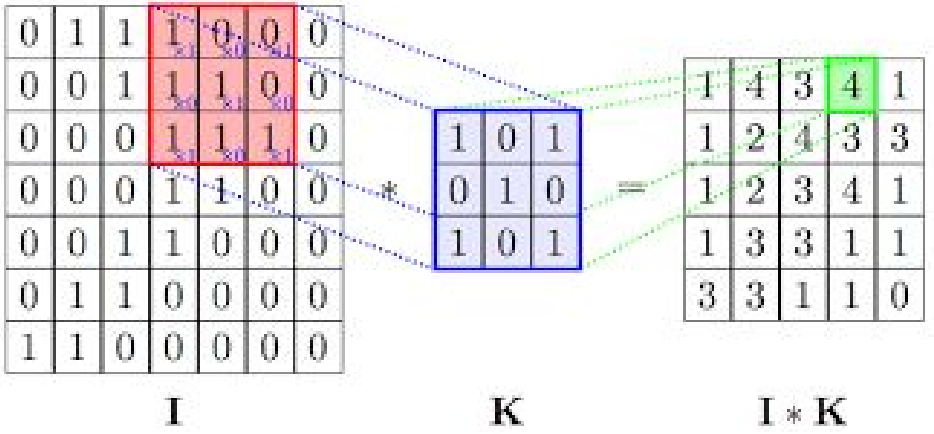
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t - \tau)\, d\tau$$

Redes Neurais Convolucionais - Convolução

O filtro K representa uma máscara (usualmente 3x3) que percorre toda a matriz de pixels da imagem I realizando a operação de convolução.

Os valores coincidentes entre I e K são multiplicados e somados, gerando um valor chave do atributo extraído.

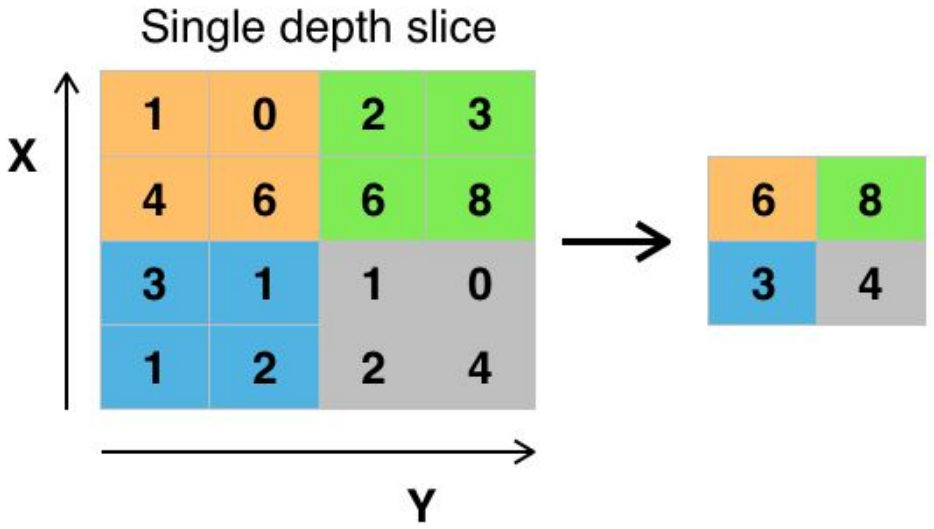
O filtro percorre a imagem inteira.



Redes Neurais Convolucionais - Max Pooling

É criado um filtro que percorre a camada de convolução, mas este filtro apenas retém o maior valor encontrado na camada.

Podemos ver na imagem que o filtro tem tamanho 2x2 (este tamanho pode ser variado). Assim, novamente, ele preserva a informação relevante extraindo o maior valor.



Redes Neurais Convolucionais - Dropout e Flatten

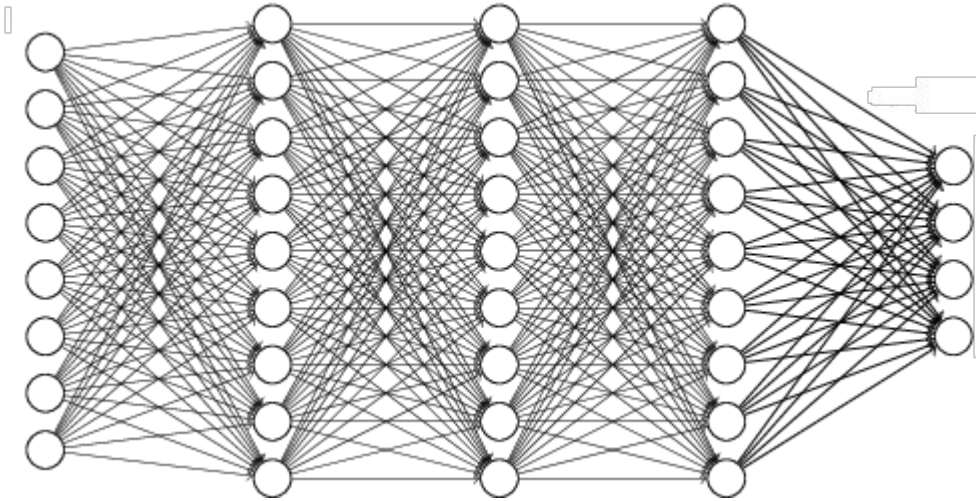
- Dropout - uma estratégia para evitar overfitting, consiste em setar randomicamente uma fração de inputs em 0 para cada iteração no processo de treinamento.

- Flatten - Transforma a matriz final (após convolução e max pooling), em um vetor que irá alimentar a próxima camada, Full Connected.

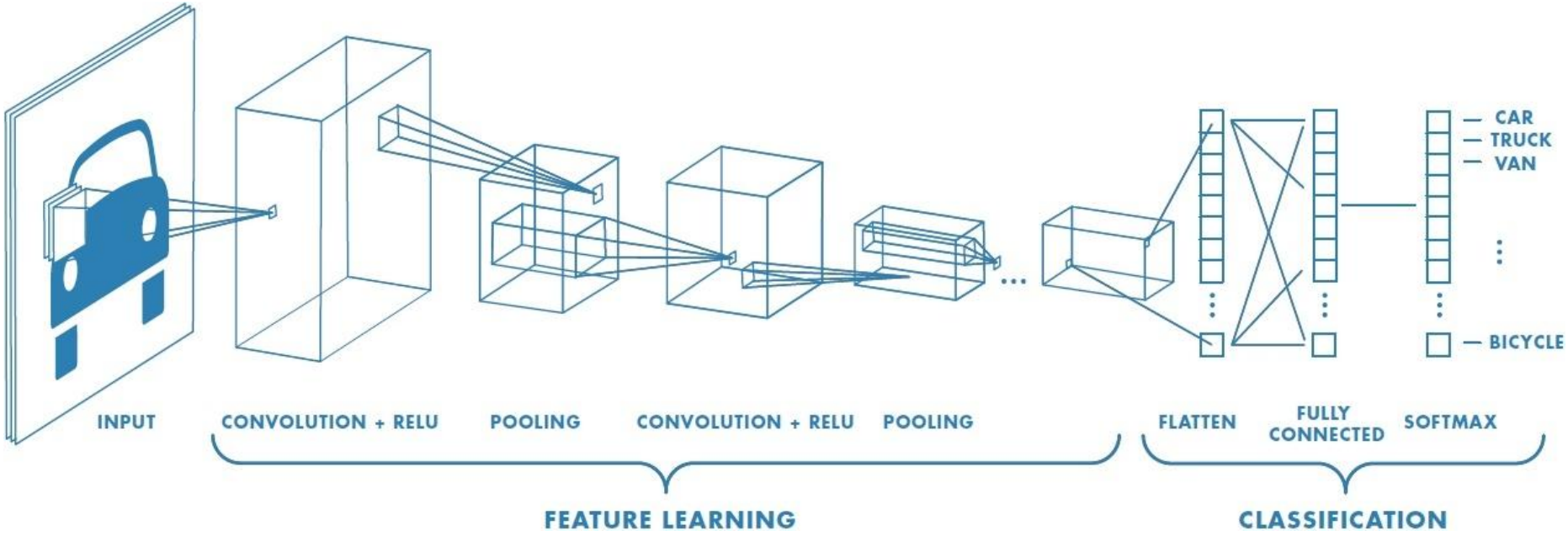
Redes Neurais Convolucionais - Full Connected

Essa camada consiste em uma Rede Neural Artificial que fará a classificação da nossa imagem.

Consiste em várias camadas de neurônios tipo Percéptron conectados à todos os neurônios da camada seguinte, e assim por diante.



Redes Neurais Convolucionais - Modelo Completo



http://scs.ryerson.ca/~aharley/vis/conv/flat.html

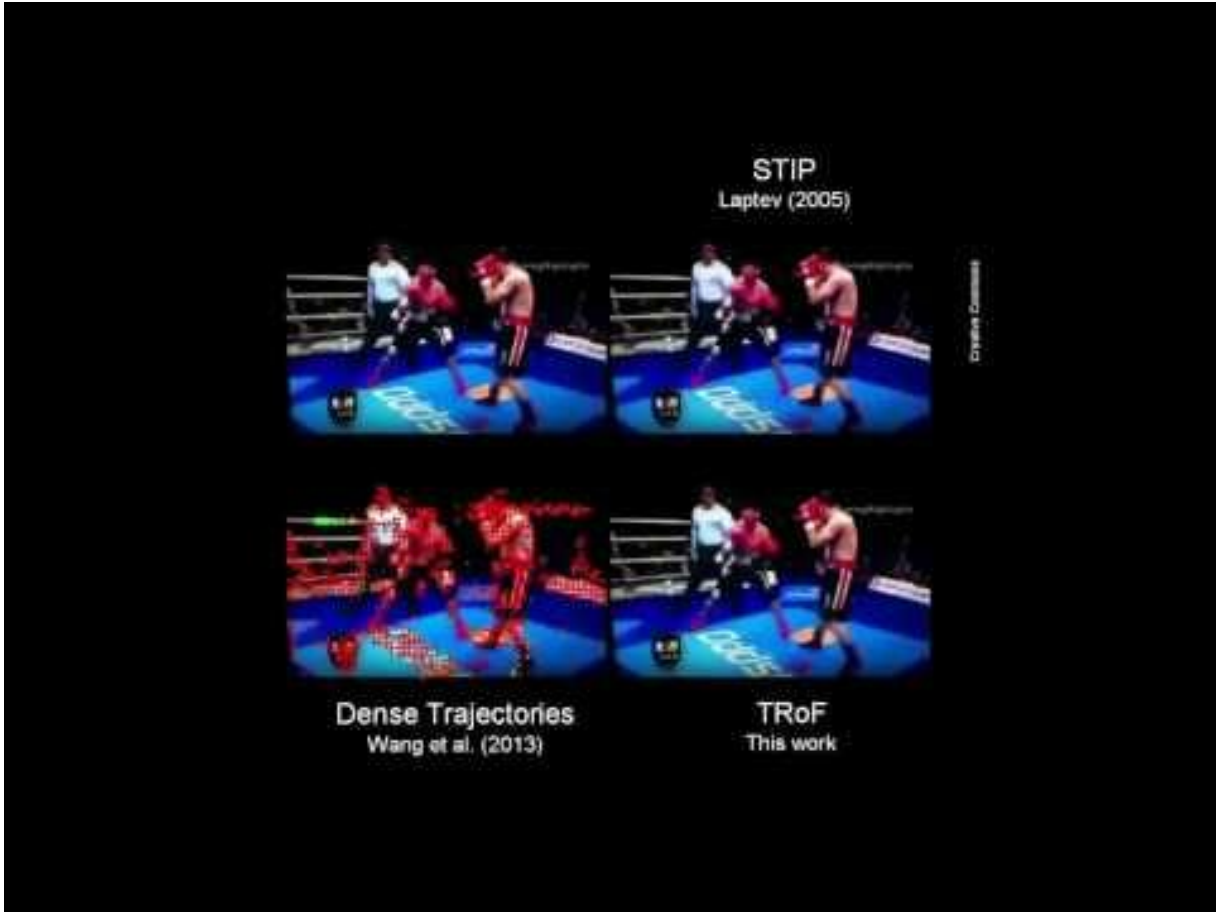

TWO MINUTE
PAPERS
WITH KÁROLY ZSOLNAI-FEHÉR (KZF)
AI LEARNS
SEMANTIC STYLE TRANSFER
Disclaimer: I was not part of this research project, I am merely providing commentary on that work.

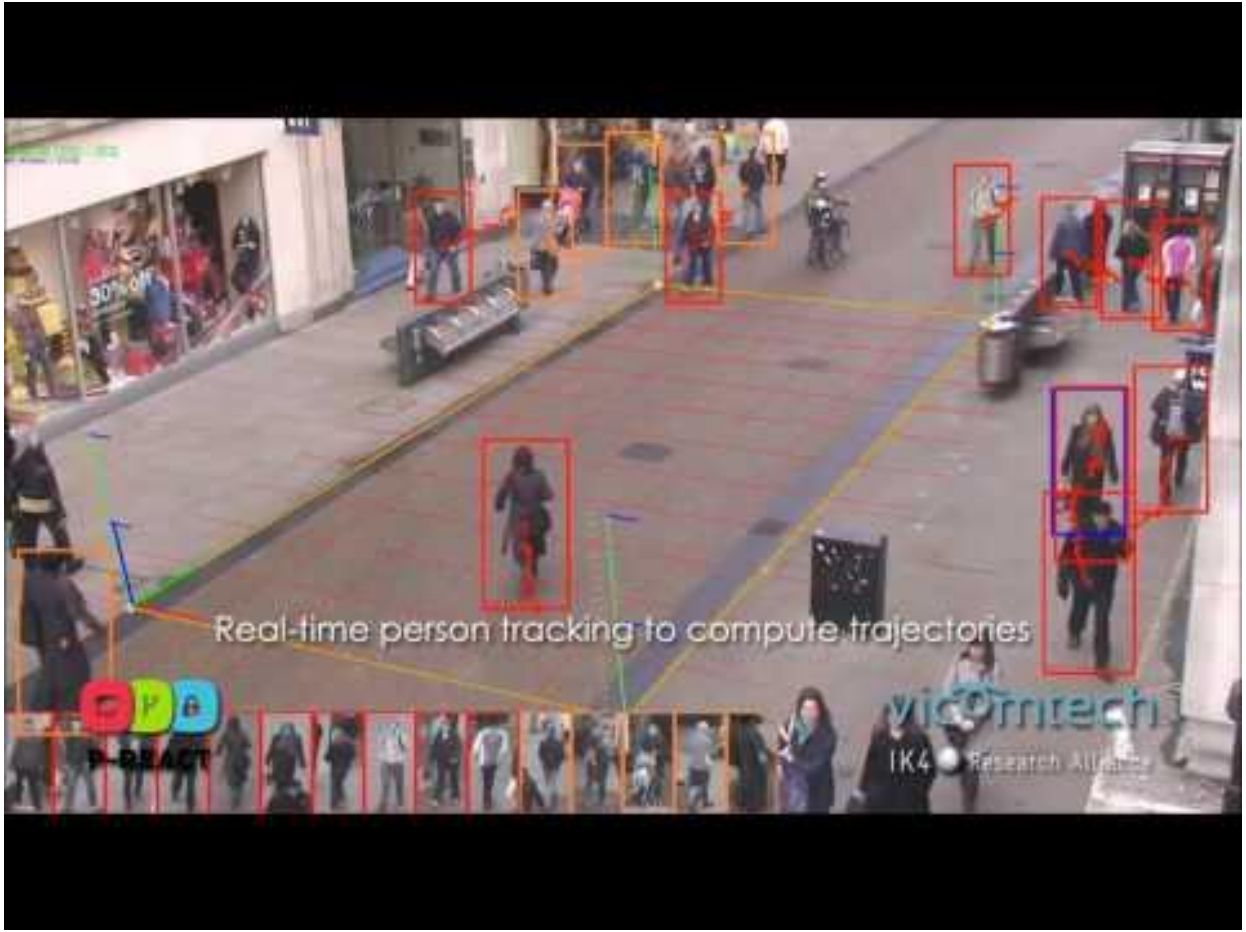

Our Prediction
Chinese

Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning

Sangdoo Yun[1] Jongwon Choi[1] Youngjoon Yoo[2]
Kimin Yun[3] and Jin Young Choi[1]

1. ASRI, Dept. of Electrical and Computer Eng., Seoul National University, South Korea
2. Graduate School of Convergence Science and Technology, Seoul National University, South Korea
3. Electronics and Telecommunications Research Institute (ETRI), South Korea


STIP
Laptev (2005)
Dense Trajectories
Wang et al. (2013)
TRoF
This work


Real-time person tracking to compute trajectories
P-REACT
vicomtech
IK4

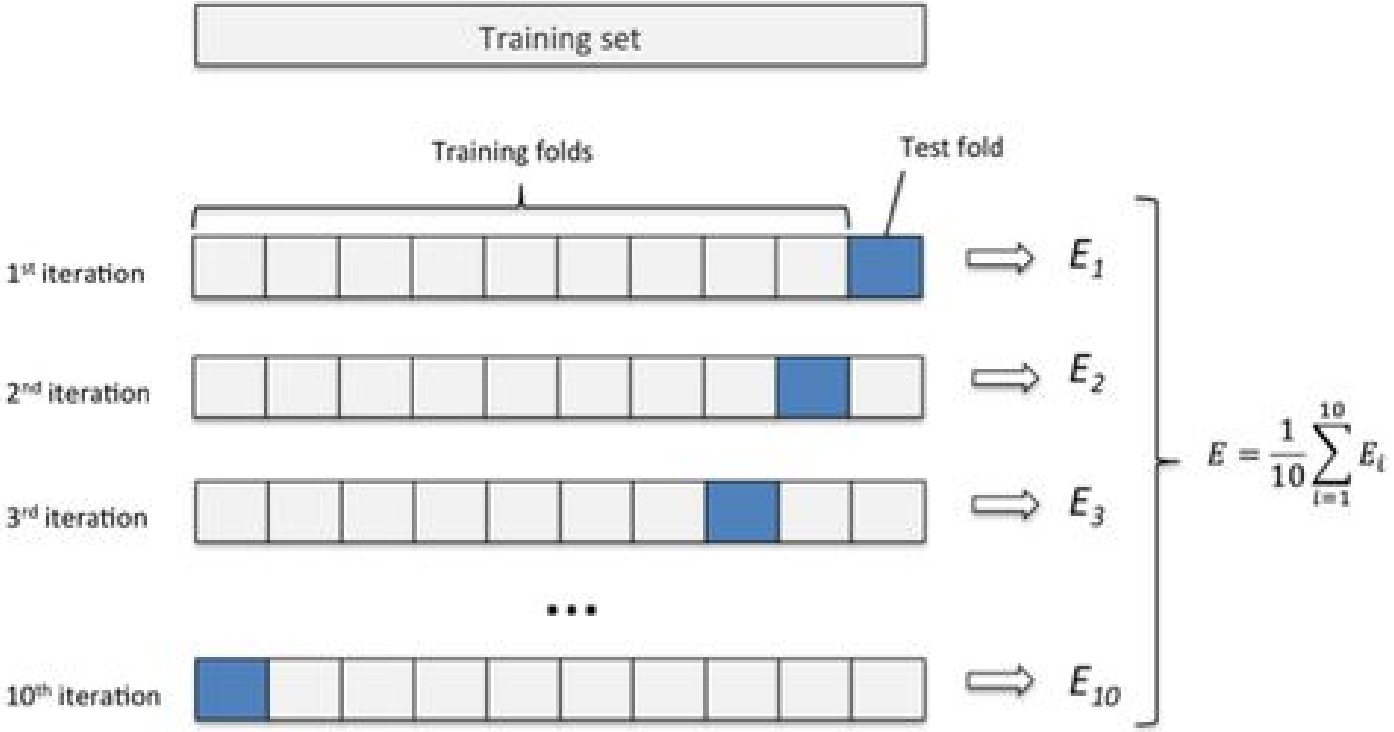

NVIDIA
and cruises right through

K-fold Cross Validation

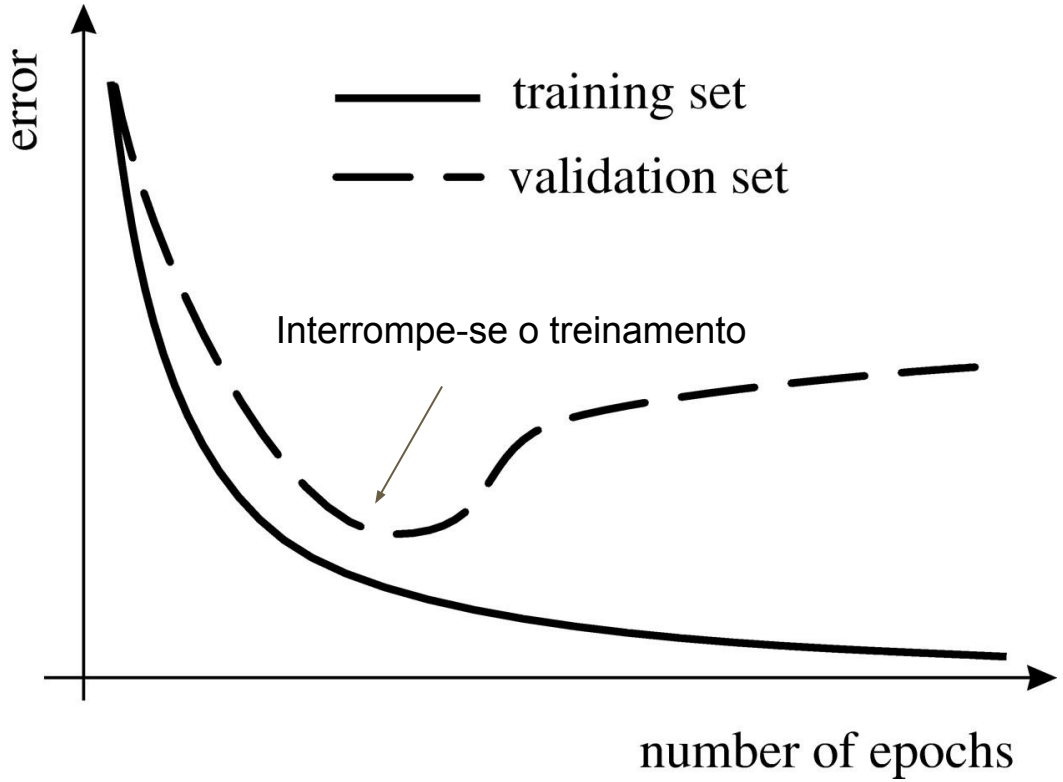
- De modo a melhor criar um modelo para um determinado conjunto de dados, é necessário aplicar algumas técnicas de validação, além de evitar o overfitting.

- Consiste em uma técnica onde são criadas K pastas para treinamento e validação.

K-fold Cross Validation



K-fold Cross Validation



K-fold Cross Validation

- Desta forma é possível treinar o modelo e validá-lo ao mesmo tempo, assim o modelo final pode ser tomado como a média dos k modelos treinados.
- É possível observar a variação dos erros de treinamento e validação. Deve-se interromper o treinamento quando o erro de validação começar a subir, de modo à evitar overfitting.

Referências

- Kirill Eremenko - Super DataScience
- UCI Machine Learning Repositorie (http://archive.ics.uci.edu/ml/index.php)
- Plataforma Kaggle (www.kaggle.com)
- Machine Learning - Plataforma Coursera, Andrew Ng
- “Machine Learning: a Probabilistic Perspective”, Kevin P. Murphy, 2012.
- “Pattern Recognition and Machine Learning”, Christopher M. Bishop, 2006.
- “Pattern Classification”, David G. Stork, Peter E. Hart, and Richard O. Duda, 2000.
- “Deep Learning”, Ian Goodfellow , Yoshua Bengio, Aaron Courville, 2016.
- “Hands-On Machine Learning with Scikit-Learn and TensorFlow”, Aurélien Géron, 2017.

Obrigado!

- Meus contatos:

hugo.padovani@cetax.com.br

www.github.com/hgpadovani