Hands-on Deep Learning - Aula 1

Introdução, Ambiente Computacional e Shallow Learning

Camila Laranjeira[1], Hugo Oliveira[12], Keiller Nogueira[12]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Minas Gerais

[2]Interest Group in Pattern Recognition and Earth Observation (PATREO)
Universidade Federal de Minas Gerais

21 de Julho, 2018

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG

Agenda

Agenda

Agenda

Instrutores

- Camila Laranjeira[1]

- Hugo Oliveira[2]

[1] http://homepages.dcc.ufmg.br/~camilalaranjeira/
[2] http://homepages.dcc.ufmg.br/~oliveirahugo/

Especialistas e Coordenador

- Keiller Nogueira[3]

- Virgínia Mota[4]

- Jefersson Santos[5]

[3] http://homepages.dcc.ufmg.br/~keiller.nogueira/
[4] http://homepages.dcc.ufmg.br/~virginiaferm/
[5] http://homepages.dcc.ufmg.br/~jefersson/

Objetivos do Curso

- Introduzir os conceitos básicos de aprendizado profundo usando redes neurais artificiais.
- Apresentar ferramentas, *frameworks* e ambientes de programação para desenvolvimento de soluções com *Deep Learning*.
- **Introduzir técnicas consolidadas e exercitar na prática formas de adaptação para outros problemas.**

Ementa

- Aula 1 - 21/07
  - Boas Vindas
  - Ambiente
  - Python & bibliotecas
  - Machine Learning
  - Redes Neurais
- Aula 2 - 28/07
  - Redes Neurais
  - Classificação de imagens
  - Redes Neurais Convolucionais (CNN)

Ementa do Curso

- Aula 3 - 04/08
  - Redes Neurais Convolucionais (CNN)
  - Segmentação
  - Detecção
- Aula 4 - 11/08
  - Redes Recorrentes
- Aula 5 - 18/08
  - Redes Generativas
- Aula 6 - 25/08
  - Conclusão
  - Desafio Final

Atividades

- Exercícios para casa
  - Não serão cobrados, apenas para prática individual.
- Desafio Final: Propor uma aplicação que use alguma das arquiteturas aprendidas no curso.
  - Grupos de qualquer tamanho (podendo ser individual)
  - Submeter proposta via Fórum (Google) até **29/07**

Agenda

Introdução a Deep Learning

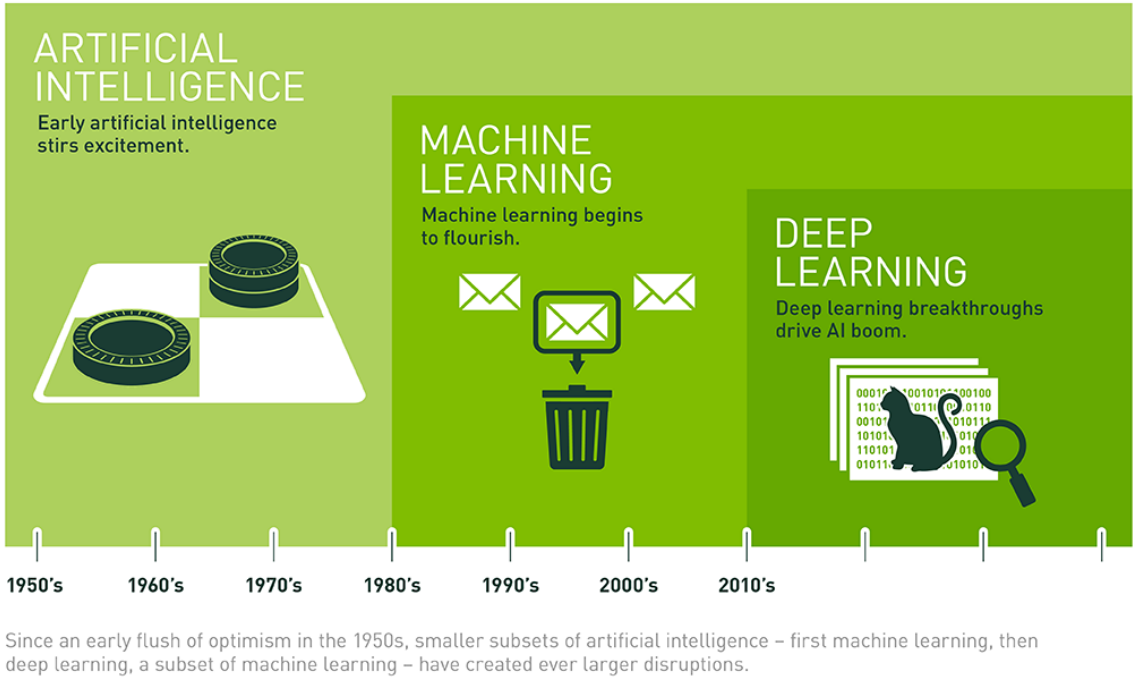


Figura: Ilustrando a relação entre Deep Learning, Aprendizado de Máquina e Inteligência Artificial [1]

O que é Aprendizado de Máquina?

Definição Abstrata [6]

Um algoritmo capaz de aprender com a experiência, nos permitindo lidar com tarefas muito difíceis para serem solucionadas com algoritmos explicitamente programados por seres humanos.

Definição Prática

Um algoritmo parametrizado, capaz de encontrar o conjunto de parâmetros que melhor se aproxima do comportamento desejado.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org

O que motivou Deep Learning?

- O mundo real não é estruturado. Dados possuem alta variabilidade, podendo também haver oclusões, entre outras perturbações
- Um desafio para a Inteligência Artificial é aprender representações e modelos capazes de lidar com tais adversidades

O que motivou Deep Learning?

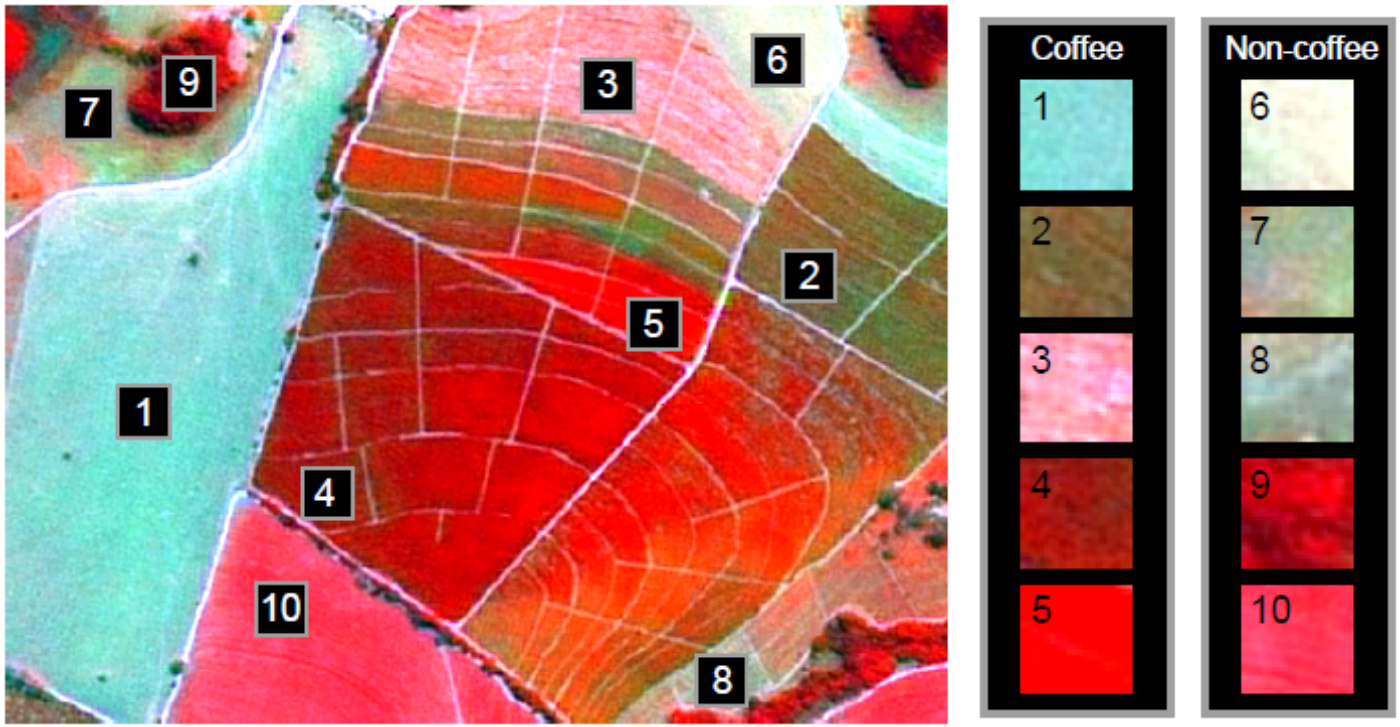


Figura: Exemplo de aplicação onde o dado possui diferentes variações.

O que motivou Deep Learning?

- Métodos existentes extraiam essas informações com um conceito pré-definido, sem observar diretamente os dados.
- Deep Learning surgiu graças a demanda por métodos que observassem os dados para extrair as informações com diferentes abstrações.

O que motivou Deep Learning?

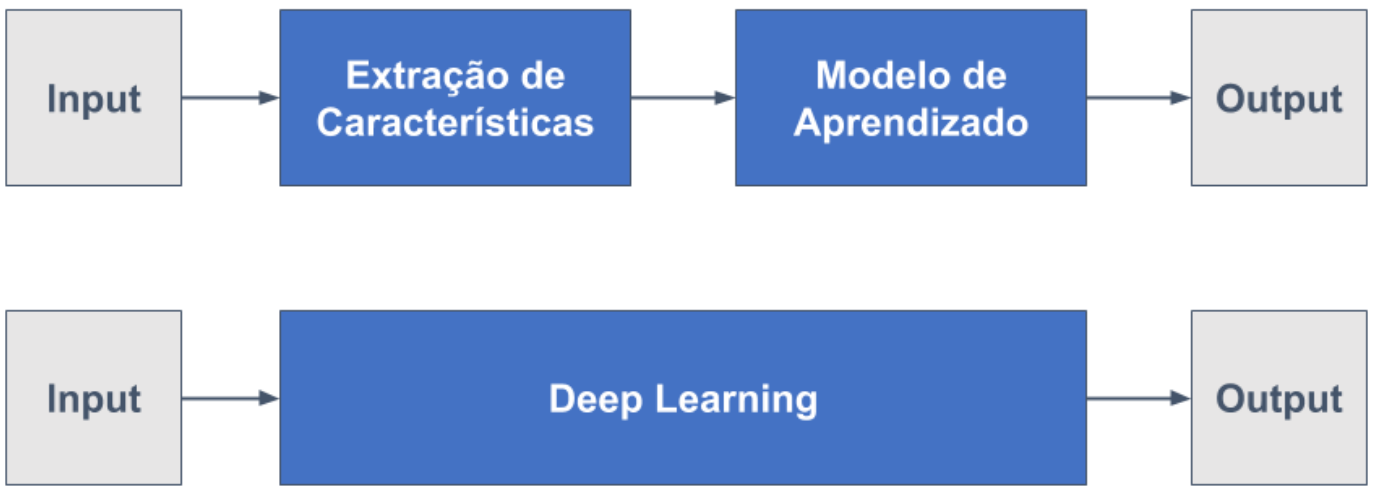
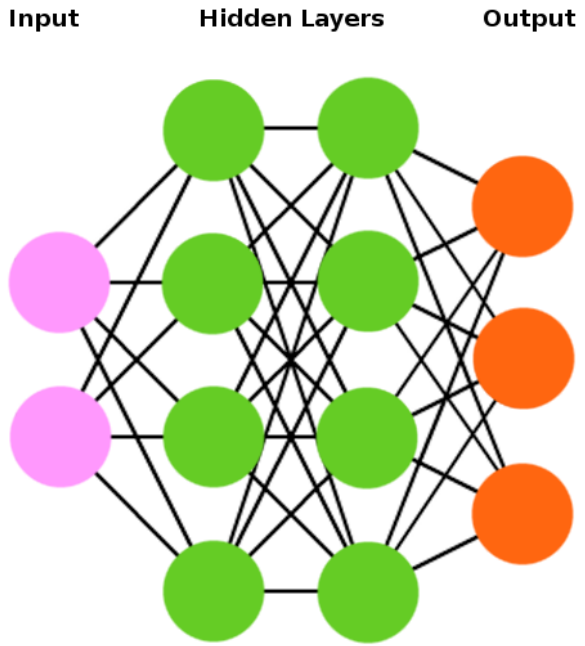


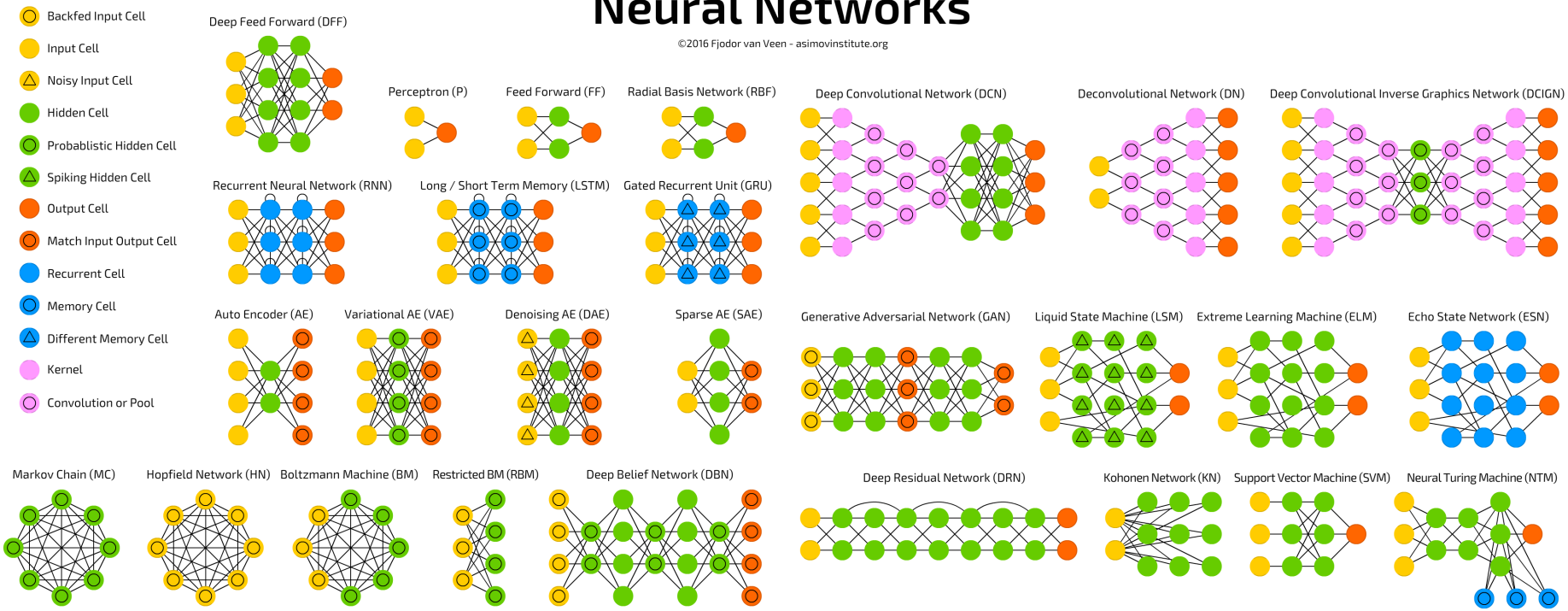
Figura: Diferença entre Aprendizado de Máquina tradicional e Deep Learning.

O que é Deep Learning?

- Tipo de Aprendizado de Máquina
- Baseado em Redes Neurais profundas
  - Mais de uma camada escondida
- Vasta gama de aplicações e **variações do modelo neural**



“Zoológico” de Arquiteturas



O que é Deep Learning?

Goodfellow [2] define Deep Learning como:

Uma técnica que permite ao computador construir representações complexas (de alto-nível) a partir de outras representações mais simples (de baixo-nível).

Lecun, Bengio e Hinton [3] definem Deep Learning como:

Uma técnica que permite modelos computacionais compostos por múltiplas camadas a aprender representação do dado com múltiplos níves de abstração (do baixo ao alto-nível).

O que é Deep Learning?

- Num geral, Deep Learning é um técnica de aprendizado de máquina voltada para redes neurais artificiais que aprendem a extrair informações de diferentes níveis de abstração observando diretamente o dado
  - Baixo-nível, como bordas e cantos em imagens
  - Médio-nível, partes inteiras de objetos, como um nariz ou uma boca
  - Alto-nível, que representa todo o objeto como, por exemplo, um rosto

O que é Deep Learning?

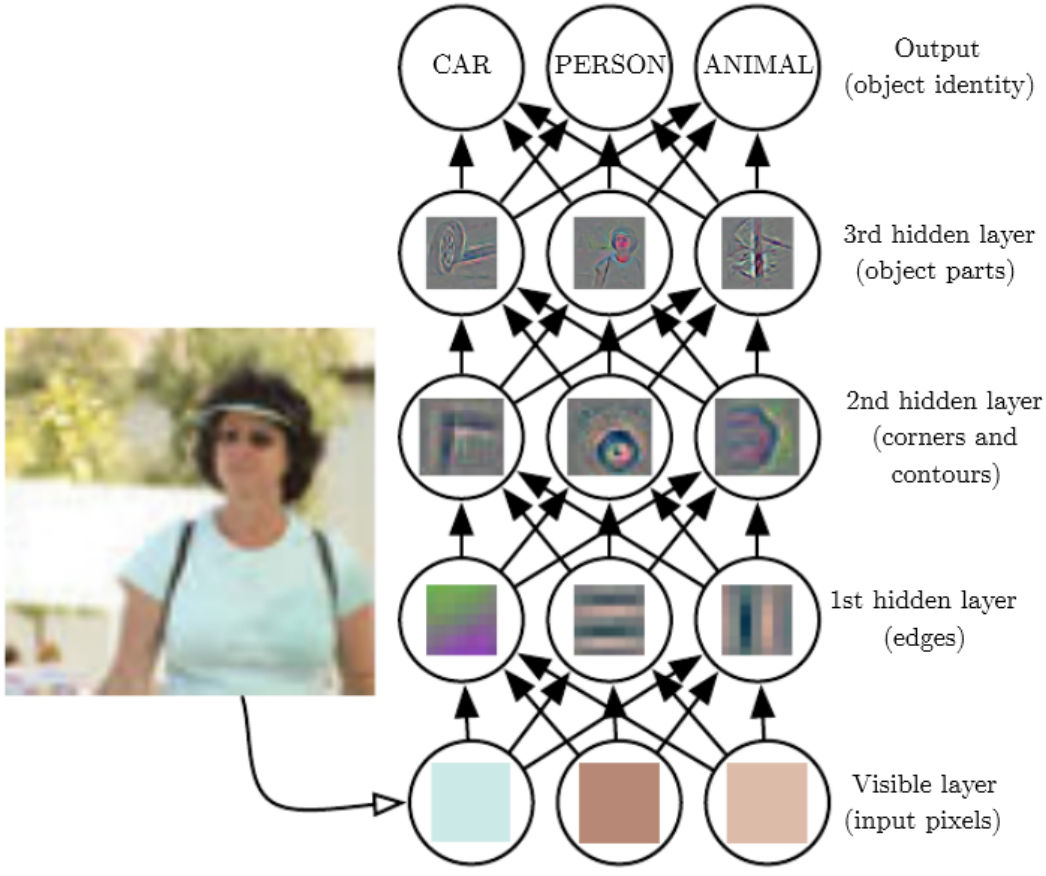
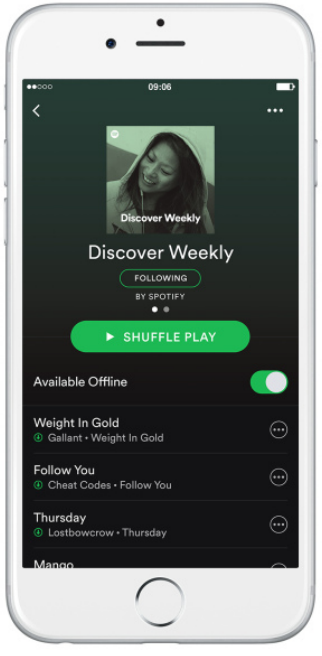


Figura: Exemplo [2] mostrando como Deep Learning parte de representações simples para mais complexas

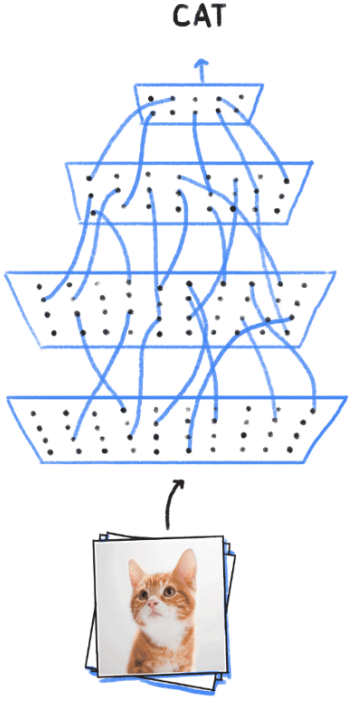
Aplicações

- Transferência de Estilo

- Recomendação de conteúdo

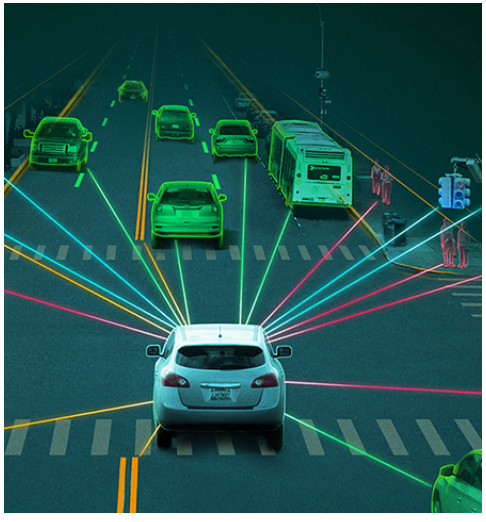


- Classificação



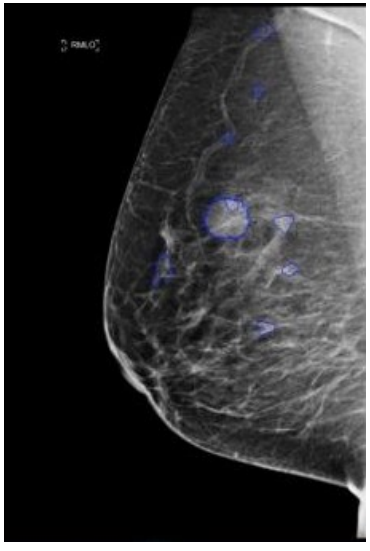
Aplicações

- Carros Autônomos



- Vigilância

- Imagens Médicas



O que é Deep Learning

Importância do Deep Learning

É difícil compreender a importância do avanço tecnológico que representa a chegada do Deep Learning sem explicar como funciona Shallow Learning

Agenda

Ambiente Computacional

- Python
  - Linguagem e biblioteca padrão
- Anaconda
  - Gerenciador de pacotes e ambientes
- Jupyter
  - Ambiente de programação, plots e anotações
- Pytorch
  - Framework de processamento vetorial e Deep Learning
- Colaboratory
  - Serviço de Cloud Gratuito da Google

Python[7]

- Linguagem interpretada
- Pensada para prototipação rápida
- Alta legibilidade
- *Binds* simples com outras linguagens
- Funciona em várias plataformas
- Fortemente e dinamicamente tipada

[7] https://www.python.org/

Python[7]

- Vasta biblioteca padrão
- Bibliotecas externas incluem uma vasta gama de aplicações computacionais
  - Processamento vetorial (Numpy)
  - Processamento científico (Scipy)
  - Visualização de dados (matplotlib)
  - Processamento de imagens (Scikit-Image)
  - Machine Learning (Scikit-Learn)
  - Deep Learning (Pytorch, Tensorflow, Theano, Caffe...)

[7] https://www.python.org/

Anaconda[8]

- Gerenciador de pacotes relacionados ao Python
- Já vem com vários pacotes importantes para programação científica
  - Numpy, Scipy, Scikit-Image, Scikit-Learn, matplotlib, Jupyter...
- Ambientes virtuais

[8] https://anaconda.org/anaconda/python

Jupyter[9]

- IDE de acesso via browser
- Visualização em tempo de execução de plots, imagens, anotações...
- Funções custom do IPython
- Acesso direto às documentações das bibliotecas

[9] http://jupyter.org/

Pytorch[10]

- Framework de Machine Learning focado em Deep Learning
- Implementação baseada no Torch (em Lua)
- Comunidade grande e participativa
  - Muitos modelos implementados pela comunidade podem ser encontrados na internet
- Possui várias funcionalidades úteis (i.e. Data Loader, Agendador de Learning Rate, Data Augmentation...)

[10] https://pytorch.org/

Google Colaboratory[11]

- Construído em cima do Jupyter
- Serviço de Cloud gratuito para desenvolvimento de I.A.
- Tesla K80 GPU disponível por até 12 horas contínuas
- Permite diferentes frameworks (Keras, Tensorflow, Pytorch)

[11] https://colab.research.google.com/

Agenda

Estruturas de Dados Básicas

- Listas (*lists*)
- Tuplas (*tuples*)
- Conjuntos (*sets*)
- Dicionários (*dictionaries*)

Cheat Sheet Python Basics

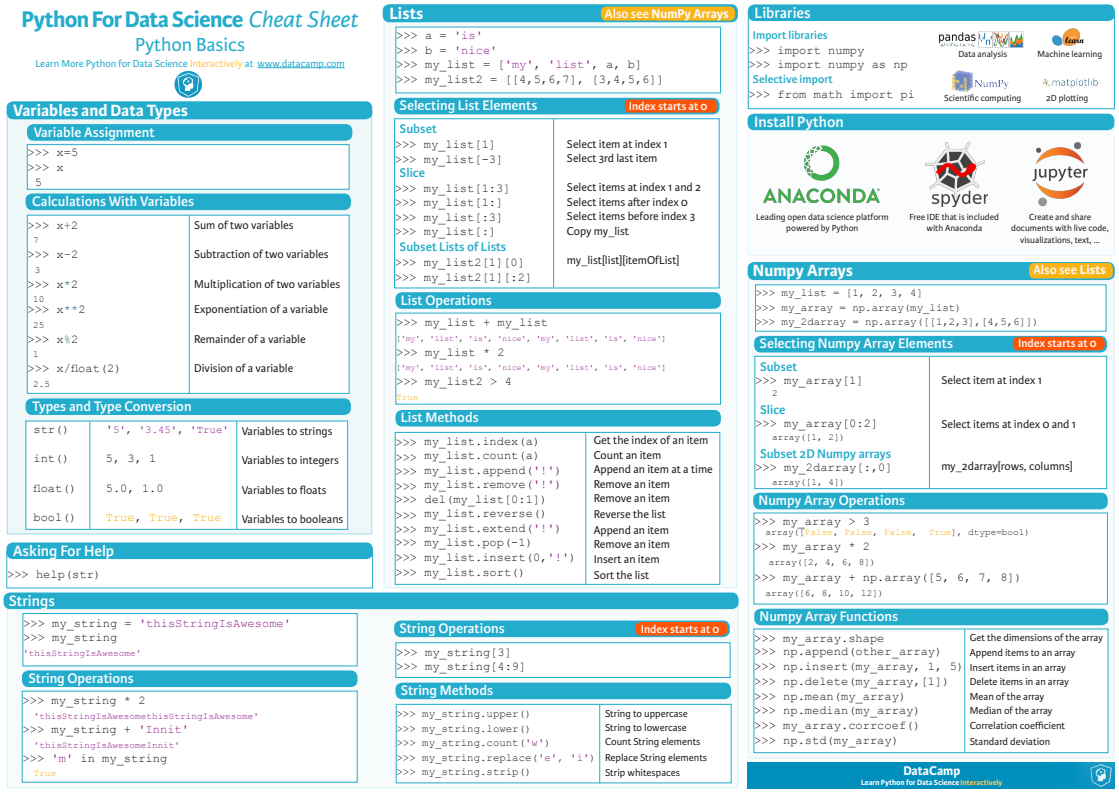


Figura: Cheat Sheet de funções básicas do Python[12].

[12] http://datacamp-community.s3.amazonaws.com/e30fbcd9-f595-4a9f-803d-05ca5bf84612

Estruturas de Dados Básicas

Demo - Python Collections

Collections_Python.ipynb

Introdução ao Numpy/Scipy

- Operações vetoriais/matriciais em Python
- Base para a maioria do Python Científico
- Funções que cobrem uma vasta gama de operações vetoriais

Cheat Sheet Numpy

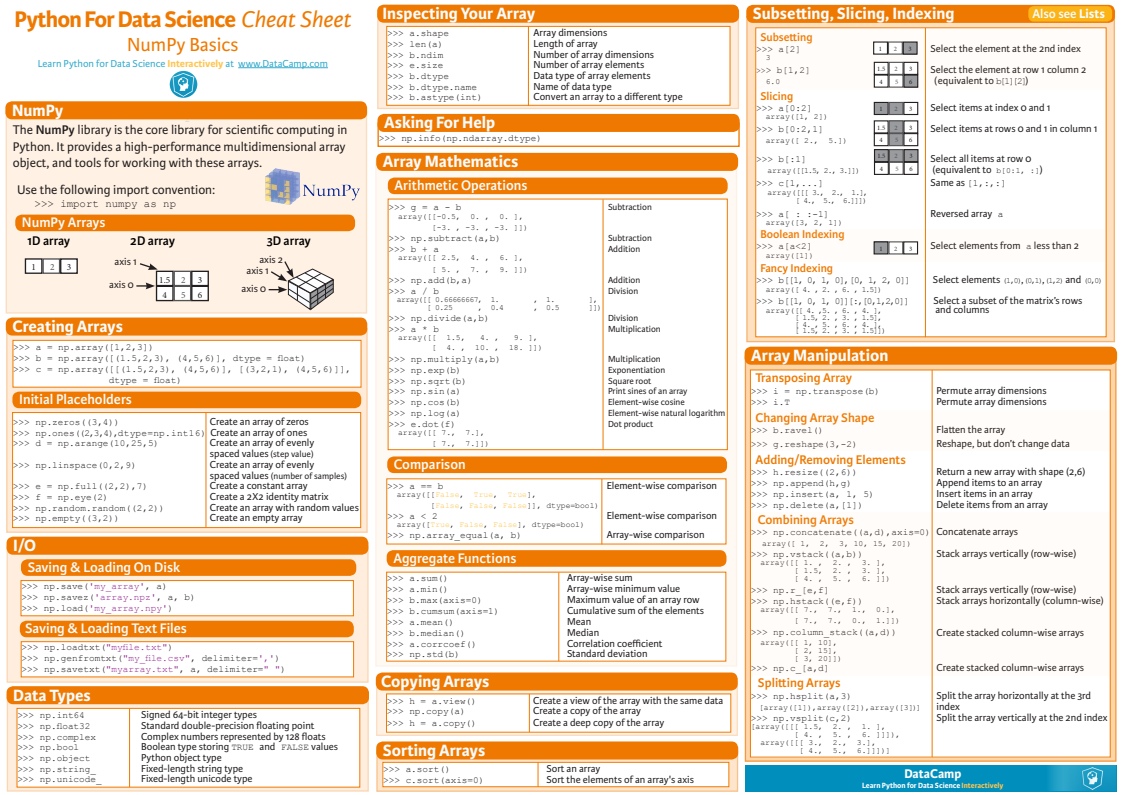
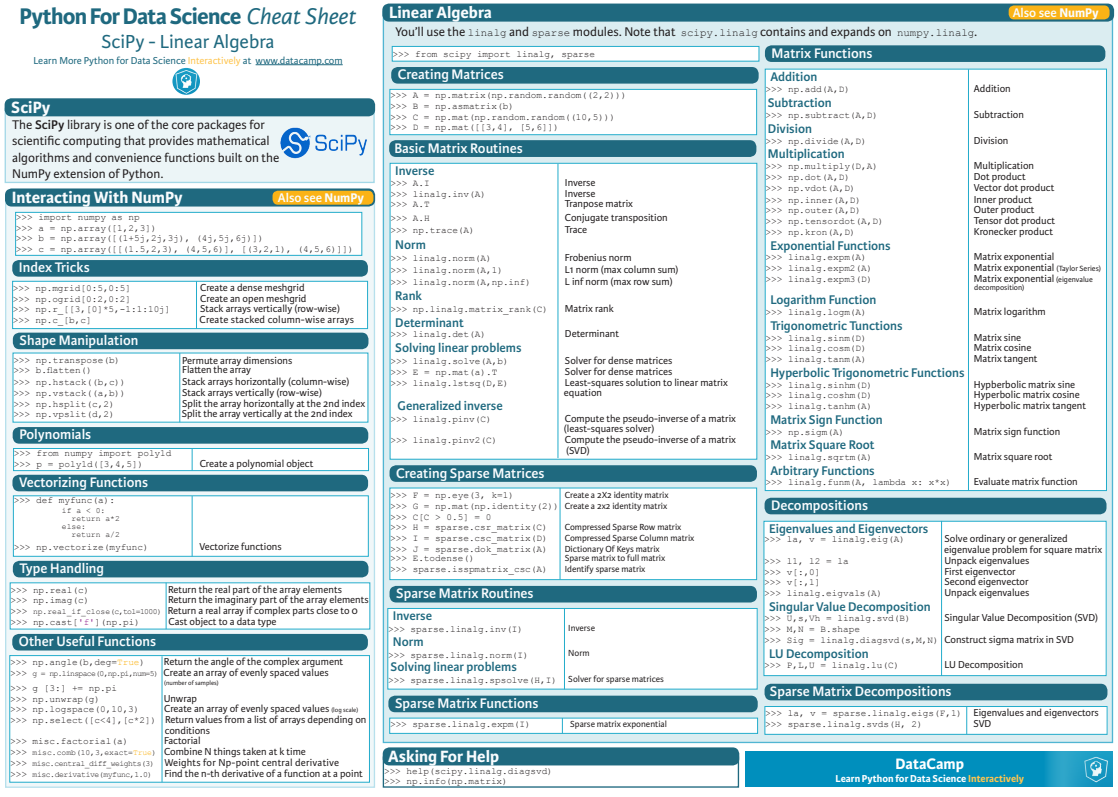


Figura: Cheat Sheet de funções do Numpy[13].

[13] http://datacamp-community.s3.amazonaws.com/e9f83f72-a81b-42c7-af44-4e35b48b20b7

Cheat Sheet Scipy



Python For Data Science *Cheat Sheet*

SciPy - Linear Algebra

Learn More Python for Data Science Interactively at www.datacamp.com

SciPy

The **SciPy** library is one of the core packages for scientific computing that provides mathematical algorithms and convenience functions built on the NumPy extension of Python.

Interacting With NumPy — Also see NumPy

```
>>> import numpy as np
>>> a = np.array([1,2,3])
>>> b = np.array([(1+5j,2j,3j), (4j,5j,6j)])
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]])
```

Index Tricks

`>>> np.mgrid[0:5,0:5]`	Create a dense meshgrid
`>>> np.ogrid[0:2,0:2]`	Create an open meshgrid
`>>> np.r_[[3,[0]*5,-1:1:10j]`	Stack arrays vertically (row-wise)
`>>> np.c_[b,c]`	Create stacked column-wise arrays

Shape Manipulation

`>>> np.transpose(b)`	Permute array dimensions
`>>> b.flatten()`	Flatten the array
`>>> np.hstack((b,c))`	Stack arrays horizontally (column-wise)
`>>> np.vstack((a,b))`	Stack arrays vertically (row-wise)
`>>> np.hsplit(c,2)`	Split the array horizontally at the 2nd index
`>>> np.vpslit(d,2)`	Split the array vertically at the 2nd index

Polynomials

`>>> from numpy import polyld`	
`>>> p = polyld([3,4,5])`	Create a polynomial object

Vectorizing Functions

```
>>> def myfunc(a):
        if a < 0:
          return a*2
        else:
          return a/2
>>> np.vectorize(myfunc)
```

Vectorize functions

Type Handling

`>>> np.real(c)`	Return the real part of the array elements
`>>> np.imag(c)`	Return the imaginary part of the array elements
`>>> np.real_if_close(c,tol=1000)`	Return a real array if complex parts close to 0
`>>> np.cast['f'](np.pi)`	Cast object to a data type

Other Useful Functions

`>>> np.angle(b,deg=True)`	Return the angle of the complex argument
`>>> g = np.linspace(0,np.pi,num=5)`	Create an array of evenly spaced values (number of samples)
`>>> g [3:] += np.pi`	
`>>> np.unwrap(g)`	Unwrap
`>>> np.logspace(0,10,3)`	Create an array of evenly spaced values (log scale)
`>>> np.select([c<4],[c*2])`	Return values from a list of arrays depending on conditions
`>>> misc.factorial(a)`	Factorial
`>>> misc.comb(10,3,exact=True)`	Combine N things taken at k time
`>>> misc.central_diff_weights(3)`	Weights for Np-point central derivative
`>>> misc.derivative(myfunc,1.0)`	Find the n-th derivative of a function at a point

Linear Algebra — Also see NumPy

You'll use the `linalg` and `sparse` modules. Note that `scipy.linalg` contains and expands on `numpy.linalg`.

```
>>> from scipy import linalg, sparse
```

Creating Matrices

```
>>> A = np.matrix(np.random.random((2,2)))
>>> B = np.asmatrix(b)
>>> C = np.mat(np.random.random((10,5)))
>>> D = np.mat([[3,4], [5,6]])
```

Basic Matrix Routines

Inverse

`>>> A.I`	Inverse
`>>> linalg.inv(A)`	Inverse
`>>> A.T`	Tranpose matrix
`>>> A.H`	Conjugate transposition
`>>> np.trace(A)`	Trace

Norm

`>>> linalg.norm(A)`	Frobenius norm
`>>> linalg.norm(A,1)`	L1 norm (max column sum)
`>>> linalg.norm(A,np.inf)`	L inf norm (max row sum)

Rank

`>>> np.linalg.matrix_rank(C)`	Matrix rank

Determinant

`>>> linalg.det(A)`	Determinant

Solving linear problems

`>>> linalg.solve(A,b)`	Solver for dense matrices
`>>> E = np.mat(a).T`	Solver for dense matrices
`>>> linalg.lstsq(D,E)`	Least-squares solution to linear matrix equation

Generalized inverse

`>>> linalg.pinv(C)`	Compute the pseudo-inverse of a matrix (least-squares solver)
`>>> linalg.pinv2(C)`	Compute the pseudo-inverse of a matrix (SVD)

Creating Sparse Matrices

`>>> F = np.eye(3, k=1)`	Create a 2X2 identity matrix
`>>> G = np.mat(np.identity(2))`	Create a 2x2 identity matrix
`>>> C[C > 0.5] = 0`	
`>>> H = sparse.csr_matrix(C)`	Compressed Sparse Row matrix
`>>> I = sparse.csc_matrix(D)`	Compressed Sparse Column matrix
`>>> J = sparse.dok_matrix(A)`	Dictionary Of Keys matrix
`>>> E.todense()`	Sparse matrix to full matrix
`>>> sparse.issparse(C)`	Identify sparse matrix

Sparse Matrix Routines

Inverse

`>>> sparse.linalg.inv(I)`	Inverse

Norm

`>>> sparse.linalg.norm(I)`	Norm

Solving linear problems

`>>> sparse.linalg.spsolve(H,I)`	Solver for sparse matrices

Sparse Matrix Functions

`>>> sparse.linalg.expm(I)`	Sparse matrix exponential

Matrix Functions

Addition

`>>> np.add(A,D)`	Addition

Subtraction

`>>> np.subtract(A,D)`	Subtraction

Division

`>>> np.divide(A,D)`	Division

Multiplication

`>>> np.multiply(D,A)`	Multiplication
`>>> np.dot(A,D)`	Dot product
`>>> np.vdot(A,D)`	Vector dot product
`>>> np.inner(A,D)`	Inner product
`>>> np.outer(A,D)`	Outer product
`>>> np.tensordot(A,D)`	Tensor dot product
`>>> np.kron(A,D)`	Kronecker product

Exponential Functions

`>>> linalg.expm(A)`	Matrix exponential
`>>> linalg.expm2(A)`	Matrix exponential (Taylor Series)
`>>> linalg.expm3(D)`	Matrix exponential (eigenvalue decomposition)

Logarithm Function

`>>> linalg.logm(A)`	Matrix logarithm

Trigonometric Functions

`>>> linalg.sinm(D)`	Matrix sine
`>>> linalg.cosm(D)`	Matrix cosine
`>>> linalg.tanm(A)`	Matrix tangent

Hyperbolic Trigonometric Functions

`>>> linalg.sinhm(D)`	Hyperbolic matrix sine
`>>> linalg.coshm(D)`	Hyperbolic matrix cosine
`>>> linalg.tanhm(A)`	Hyperbolic matrix tangent

Matrix Sign Function

`>>> np.signm(A)`	Matrix sign function

Matrix Square Root

`>>> linalg.sqrtm(A)`	Matrix square root

Arbitrary Functions

`>>> linalg.funm(A, lambda x: x*x)`	Evaluate matrix function

Decompositions

Eigenvalues and Eigenvectors

`>>> la, v = linalg.eig(A)`	Solve ordinary or generalized eigenvalue problem for square matrix
`>>> l1, l2 = la`	Unpack eigenvalues
`>>> v[:,0]`	First eigenvector
`>>> v[:,1]`	Second eigenvector
`>>> linalg.eigvals(A)`	Unpack eigenvalues

Singular Value Decomposition

`>>> U,s,Vh = linalg.svd(B)`	Singular Value Decomposition (SVD)
`>>> M,N = B.shape`	
`>>> Sig = linalg.diagsvd(s,M,N)`	Construct sigma matrix in SVD

LU Decomposition

`>>> P,L,U = linalg.lu(C)`	LU Decomposition

Sparse Matrix Decompositions

`>>> la, v = sparse.linalg.eigs(F,1)`	Eigenvalues and eigenvectors
`>>> sparse.linalg.svds(H, 2)`	SVD

Asking For Help

```
>>> help(scipy.linalg.diagsvd)
>>> np.info(np.matrix)
```

DataCamp — Learn Python for Data Science Interactively

Figura: Cheat Sheet de funções do Scipy[14].

[14] http://datacamp-community.s3.amazonaws.com/5710caa7-94d4-4248-be94-d23dea9e668f

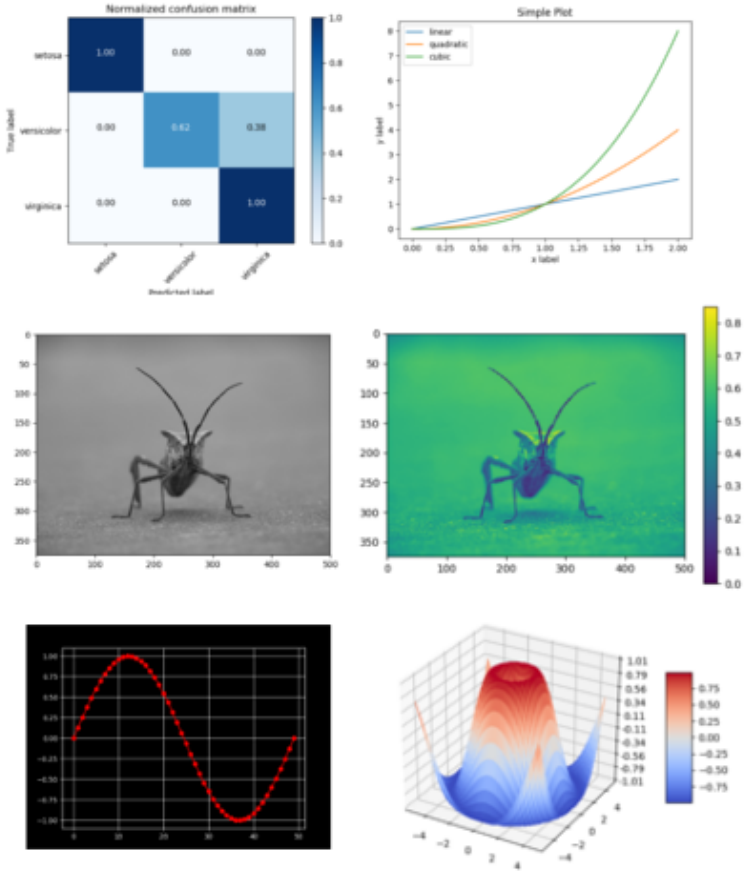
Introdução ao Numpy e Scipy

Demo - Introduction to Numpy/Scipy

Introduction_Numpy_Scipy.ipynb

Introdução ao Matplotlib

- Biblioteca mais popular do Python para visualização de dados
- Grande variedade de gráficos
- Liberdade para editar diferentes aspectos da figura



Cheat Sheet Matplotlib

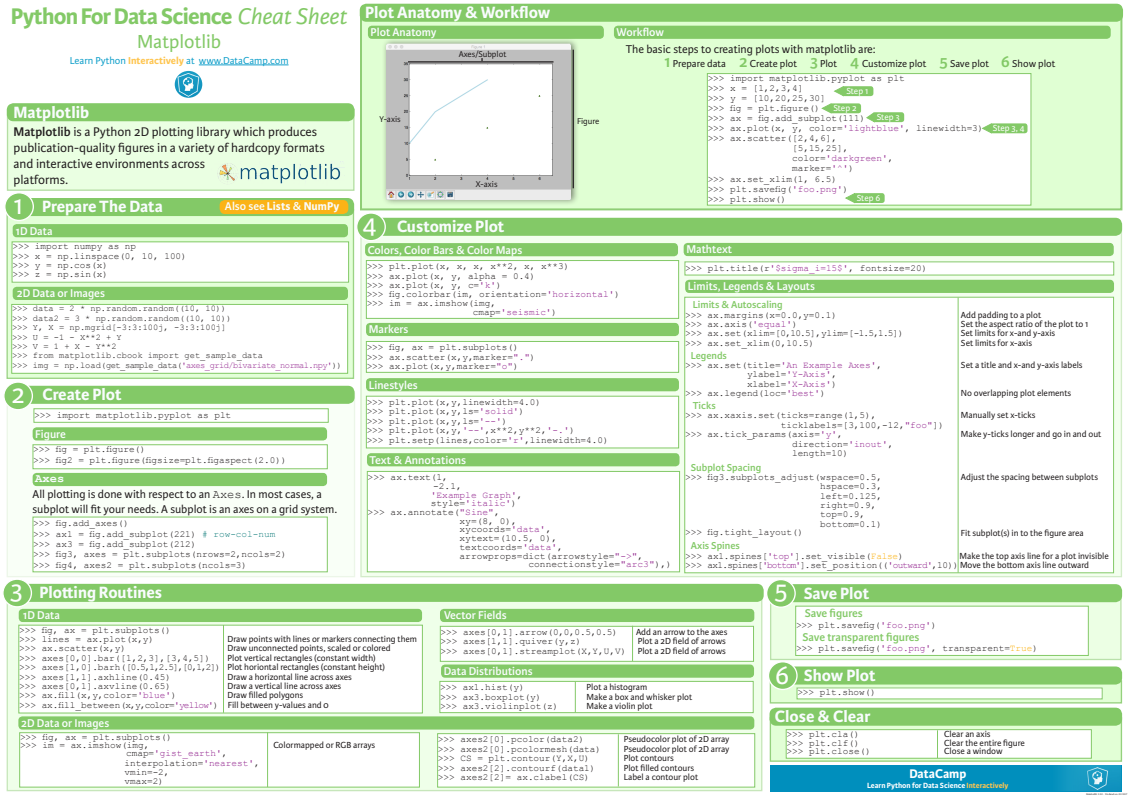


Figura: Cheat Sheet de funções da Matplotlib[15].

15 http://datacamp-community.s3.amazonaws.com/28b8210c-60cc-4f13-b0b4-5b4f2ad4790b

Introdução ao Matplotlib

Demo - Introduction to Matplotlib

Introduction_Matplotlib.ipynb

Agenda

Extração de Features

- Antes de Deep Learning, extração de features era necessária para lidar com dados não estruturados
- Features boas são desenhadas visando realçar e compactar as propriedades importantes do objeto
  - É difícil saber quais propriedades são importantes para o domínio e a tarefa em mãos
- Diferentes domínios possuem diferentes tipos de features

Extração de Features

- Alguns domínios possuem features naturais
  - Informações de prontuário médico (i.e. idade, sexo, peso, presença/ausência de sintomas etc)
  - Mercado imobiliário (i.e. bairro, andar, número de quartos, número de banheiros etc)
- Outros domínios precisam de algoritmos de extração de features
  - Imagens (i.e. forma, textura, cor etc)
  - Sinais Temporais (i.e. densidade espectral, frequências mais importantes, amplitude média etc)

Tipos de Features

- Features Categóricas
  - Possui divisão bem definida entre categorias (i.e. bairro – Ouro Preto/Liberdade/Savassi/Lourdes, tipo – apartamento/casa/kitnet, cozinha – s/n, cor – branca/amarela/azul/verde, piscina – s/n etc)
- Features Numéricas
  - Valores contínuos ou discretos que possuem uma relação numérica com valores próximos (i.e. andar, número de quartos, metros quadrados, número de banheiros etc)

Extração de Features em Texto

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

- Que características parecem relevantes em um conteúdo textual?

Extração de Features em Texto

"Essa é uma frase contendo diferentes flexões gramaticais!"

- Tokenização e filtragem
  - {"Essa", "é", "uma", "frase", "contendo", "diferentes", "flexões", "gramaticais" }
  - Dependendo do contexto, algumas palavras são descartadas no processo de filtragem. Essas são as chamadas *stop words*.
- Lematização (opcional)
  - { "Essa", "ser", "uma", "frase", "conter", "diferente", "flexão", "gramatical" }

Extração de Features em Texto

- Dicionários (Bag of Words)
  - A cada palavra é atribuído um valor inteiro (0, 1, 2, 3...)
  - Palavras com códigos próximos não necessariamente possuem significados/propriedades próximas
  - As features mais comuns são compostas das frequências dos termos

Extração de Features em Texto

"Nesse exemplo, a ordem das palavras depende da frequência que cada palavra aparece."

ID	String	Frequência
0	palavra	2
1	da	2
2	ordem	1
...	...	...
N	cada	1

Extração de Features em Texto

- Term-Frequency (TF)
  - Contar as ocorrências de cada palavra em valor absoluto seria fortemente impactado pelo tamanho do texto.
  - Atribui-se portanto um peso intitulado frequência do termo, que pode ser dado por $t_f = \frac{count_w}{N}$, sendo N o número total de palavras do texto, e $count_w$ o número de ocorrências da palavra.
  - Dessa forma a frequência de palavras é dada de forma relativa **a cada documento**.

Extração de Features em Texto

- Term-Frequency – Inverse Document-Frequency (TF-IDF)
  - Para refinar o peso dado pela frequência de termos t_f, é levado em consideração a frequência das palavras ao longo de **todos os documentos**.
    - Palavras muito comuns trazem pouca informação ao texto (i.e. “e”, “a”, “o”, “um”, “uma”, “de”, “que” etc)
  - Ambos os fatores compõe uma feature de texto amplamente utilizada chamada TF-IDF.

Extração de Features em Texto

- Scikit-learn

From text

The `sklearn.feature_extraction.text` submodule gathers utilities to build feature vectors from text documents.

`feature_extraction.text.CountVectorizer` ([...])	Convert a collection of text documents to a matrix of token counts
`feature_extraction.text.HashingVectorizer` ([...])	Convert a collection of text documents to a matrix of token occurrences
`feature_extraction.text.TfidfTransformer` ([...])	Transform a count matrix to a normalized tf or tf-idf representation
`feature_extraction.text.TfidfVectorizer` ([...])	Convert a collection of raw documents to a matrix of TF-IDF features.

Extração de Features em Texto

Demo - Text Features

Text_Features.ipynb

Extração de Features em Imagens

- Imagens possuem muita redundância entre pixels vizinhos
- Como se livrar da informação redundante e extrair apenas as características mais importantes?

Extração de Features em Imagens

- Algumas categorias principais de features em imagens
  - Bordas
  - Frequência
  - Cor
  - Textura
  - Forma
- Diferentes domínios da área de Visão Computacional usam diferentes combinações desses features
- Dependendo da tarefa, os features podem ser extraídos da imagem toda (features globais) ou de patches (features locais)

Extração de Features em Imagens

- Bordas
  - Canny
  - Sobel
  - Prewitt
  - Laplace

Extração de Features em Imagens

- Frequências
  - Wavelets
  - Discrete Fourier Transform (DFT)
  - Discrete Cosine Transform (DCT)
  - Spectral Density

Extração de Features em Imagens

- Features de Cor
  - Global Color Histogram (GCH)
  - Color Coherent Vector (CCV)
  - Border/Interior Pixel Classification (BIC)
  - Color-Based Clustering (CBC)

Extração de Features em Imagens

- Features de Textura
  - Gray-Level Co-occurrence Matrix (GLCM)
  - Local Binary Patterns (LBP)
  - Wavelets
  - Homogeneous Texture Descriptor (HTD)
  - Quantized Compound Change Histogram (QCCH)

Extração de Features em Imagens

- Features de Forma
  - Histogram of Oriented Gradients (HOG)
  - Wavelets
  - Scale-Invariant Feature Transform (SIFT)
  - Speeded Up Robust Features (SURF)
  - Shape Index
  - Extended Morphological Profiles (EMP)

Extração de Features em Imagens

- A maioria dos features codifica informações com mais de uma semântica
  - BIC codifica informação tanto de cor quanto de textura
  - Wavelets codificam tanto informação de textura quanto de frequência
  - SIFT codifica tanto informação de textura quanto de forma

Features em Imagens

Demo - Image Features

Image_Features.ipynb

Agenda

Shallow Learning

- Abordagem clássica para a aprendizagem de máquina
- Modelos estatísticos para o reconhecimento automático de padrões
- Usa algoritmos de extração de características como pré-processamento

Shallow Learning

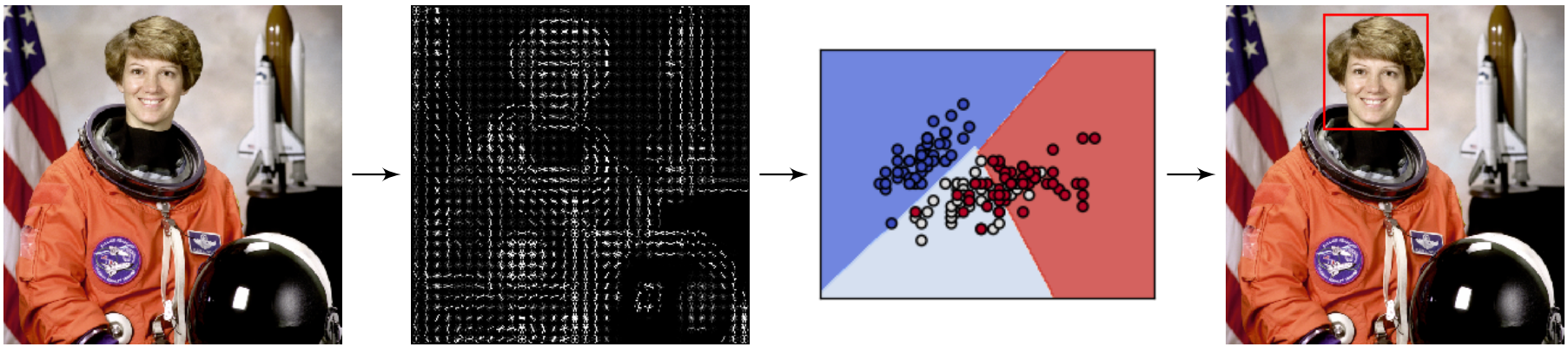


Figura: Pipeline padrão de um algoritmo de Machine Learning clássico.

Agenda

Seleção de Modelos

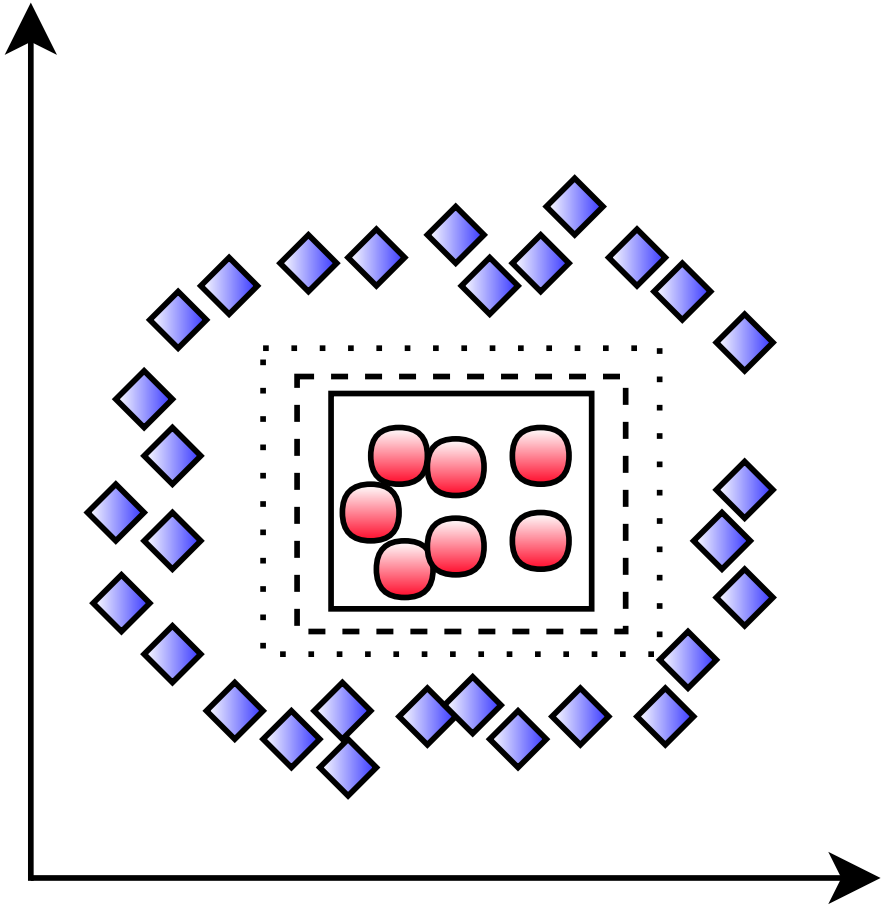


Figura: Qual fronteira de decisão escolher dentre todas as possíveis?

Seleção de Modelos

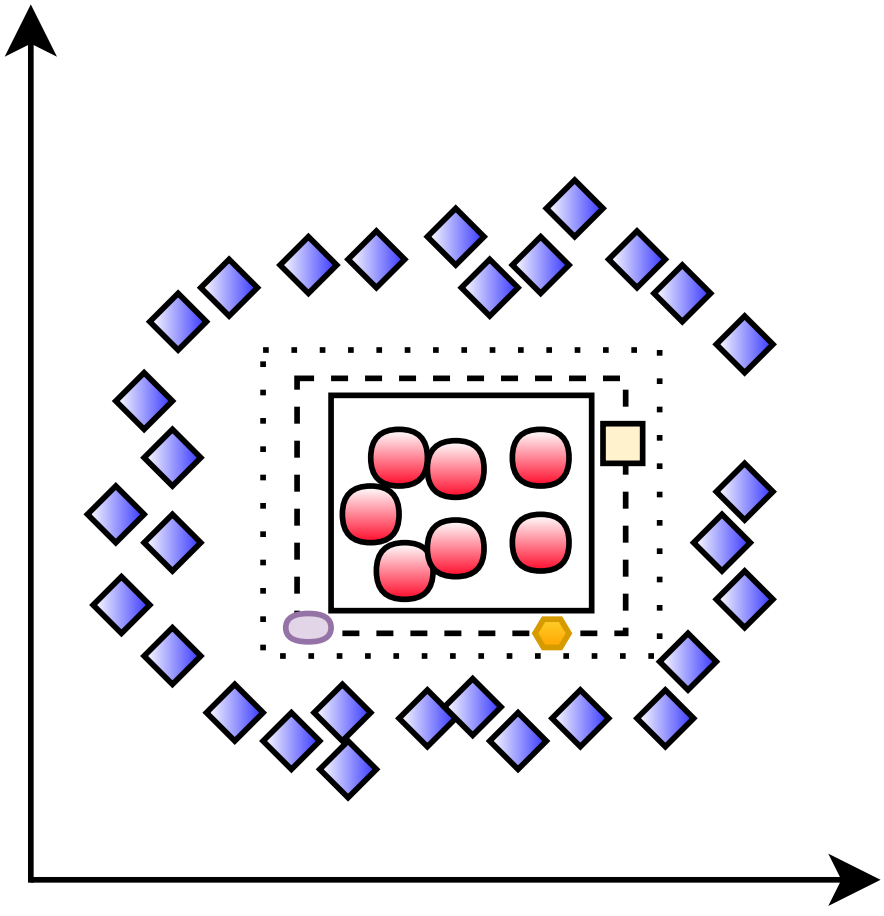


Figura: Qual fronteira de decisão escolher dentre todas as possíveis? Essa fronteira é generalista o bastante?

Seleção de Modelos

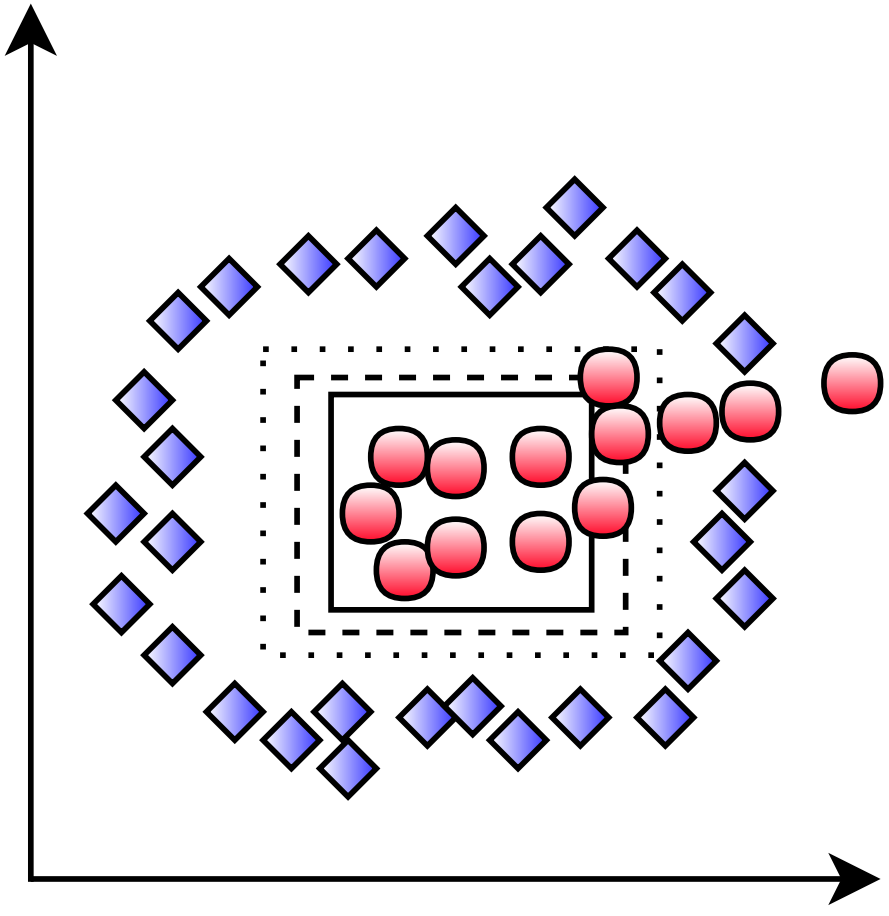


Figura: Qual fronteira de decisão escolher dentre todas as possíveis?

Seleção de Modelos

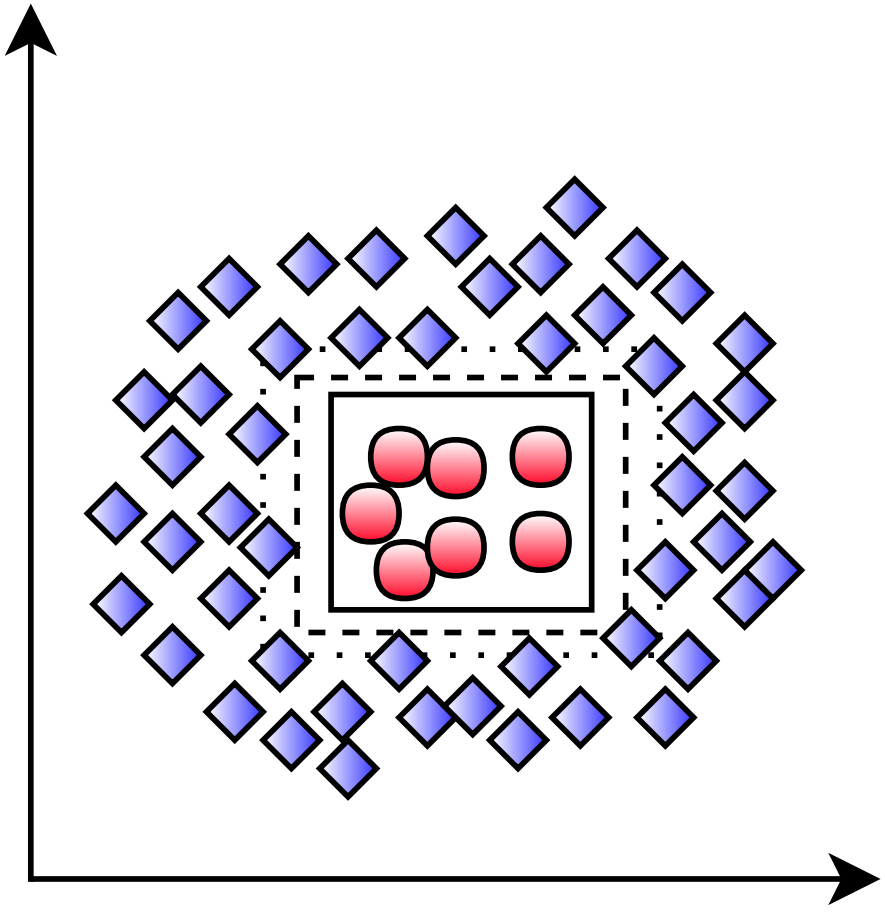


Figura: Qual fronteira de decisão escolher dentre todas as possíveis?

Seleção de Modelos

- Esse exemplos nos fazem ponderar sobre a real natureza de Machine Learning
  - Inferência sobre um conjunto de dados limitado é um problema complexo
  - Não há um único modelo a ser otimizado

Machine Learning

Suposições dos Modelos de Learning

Quantas e quais suposições fazer?

Número de Parâmetros

Figura: Trade-off do número de parâmetros de um modelo.

Número de Parâmetros

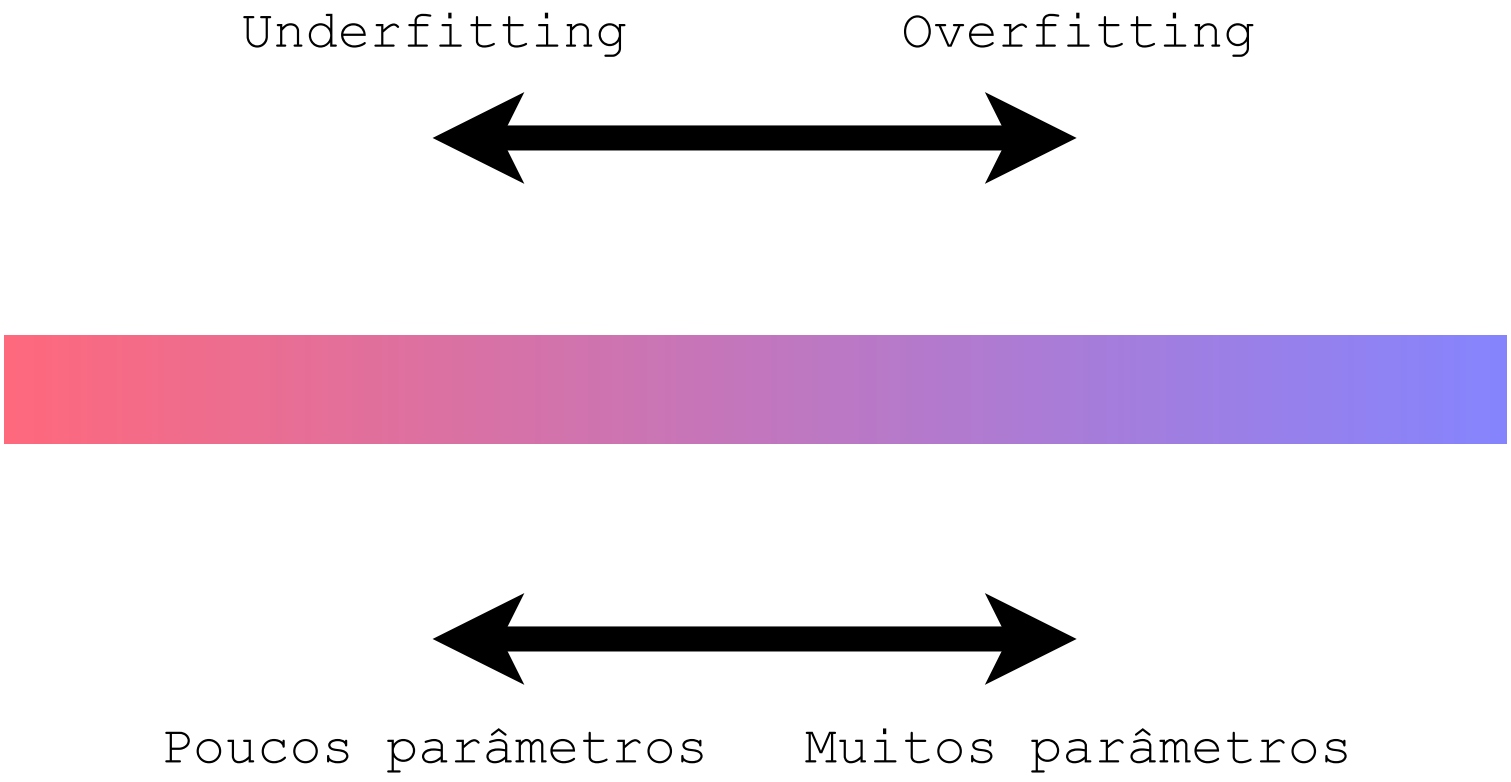


Figura: Trade-off do número de parâmetros de um modelo.

Número de Parâmetros

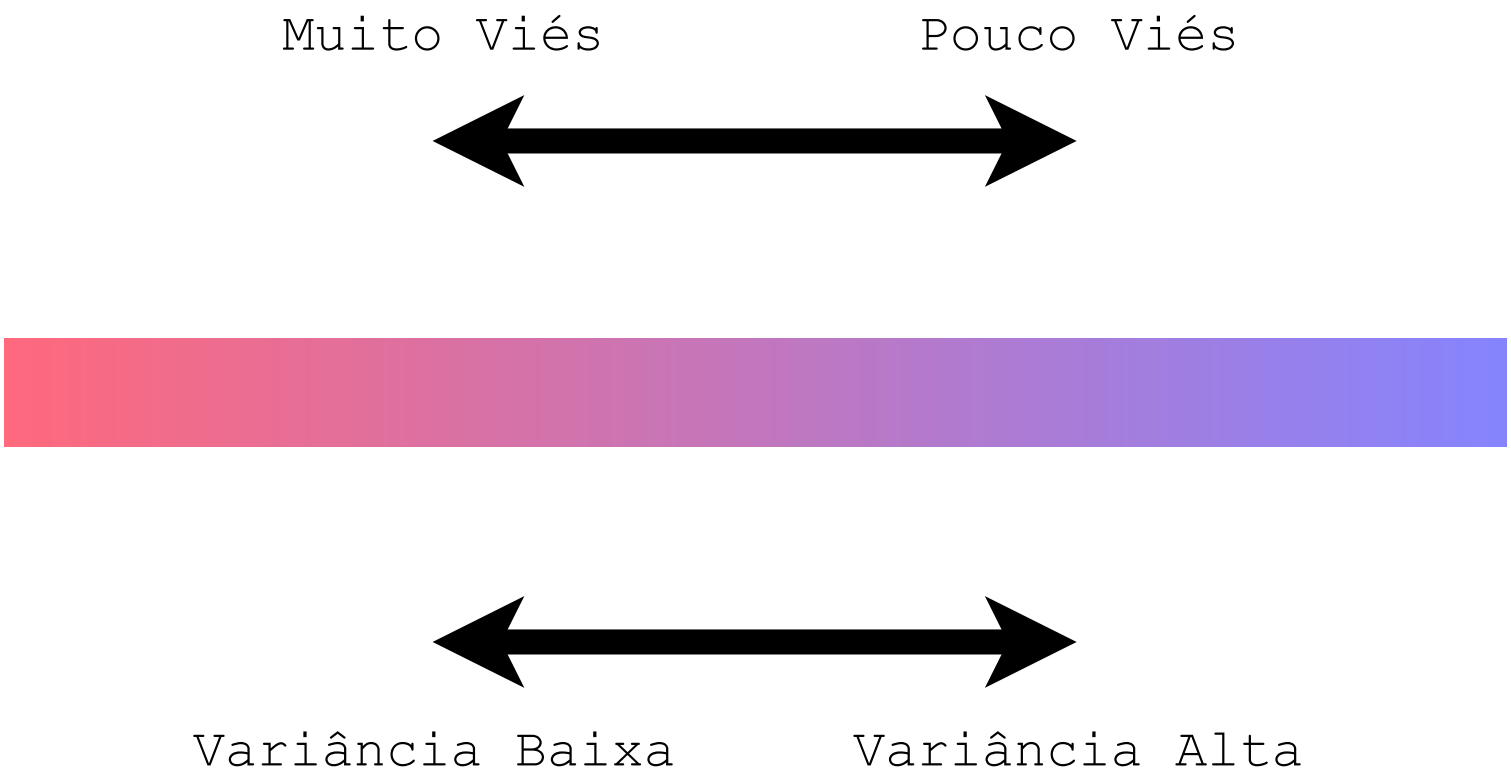


Figura: Trade-off do número de parâmetros de um modelo.

Número de Parâmetros

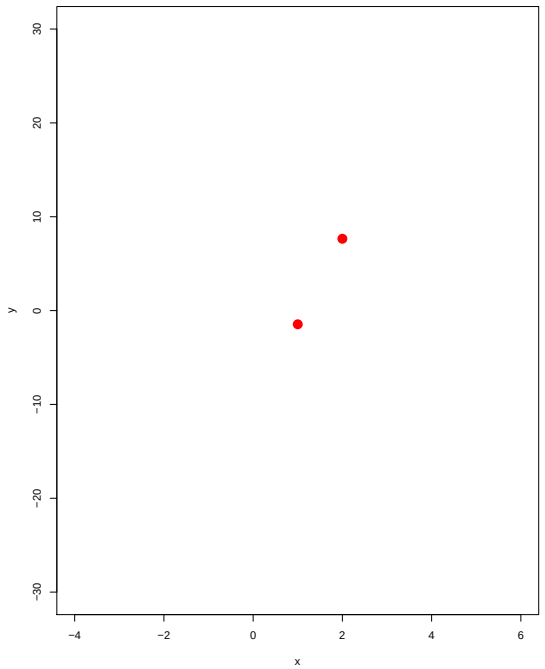


Figura: Número de parâmetros vs. número de amostras.

Número de Parâmetros

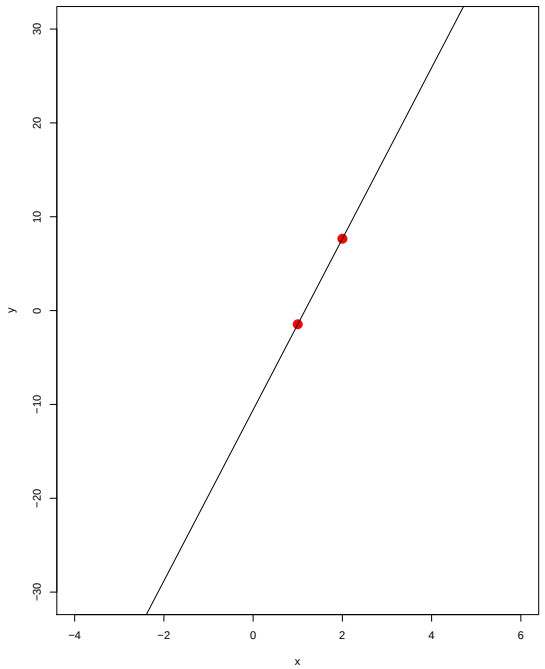


Figura: Número de parâmetros vs. número de amostras.

Número de Parâmetros

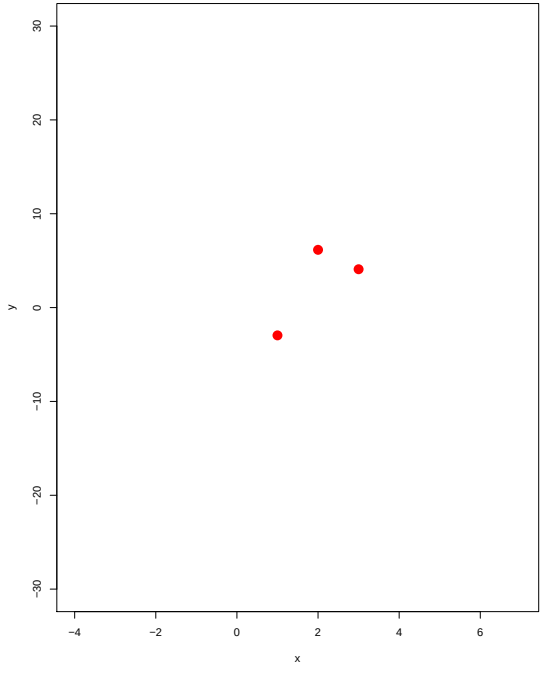


Figura: Número de parâmetros vs. número de amostras.

Número de Parâmetros

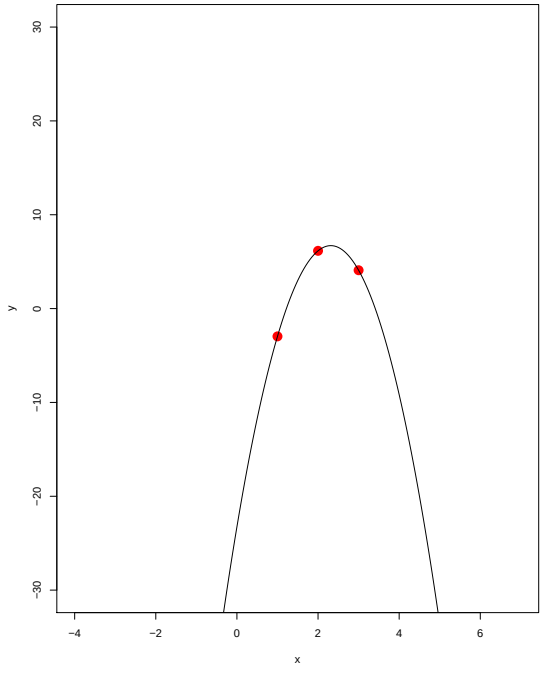


Figura: Número de parâmetros vs. número de amostras.

Número de Parâmetros

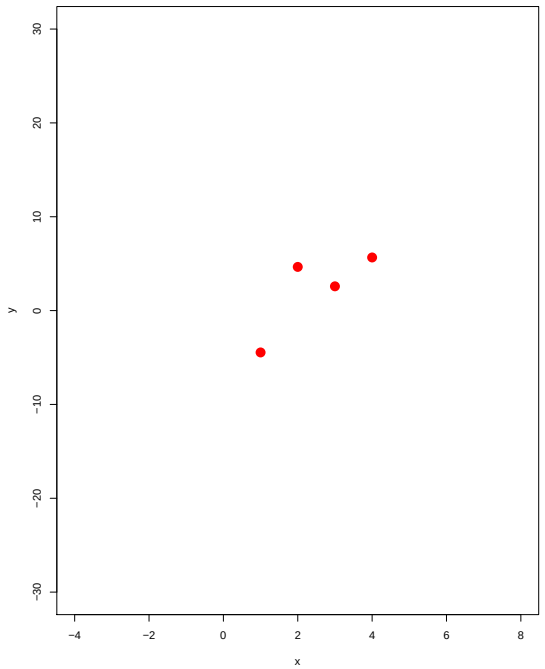


Figura: Número de parâmetros vs. número de amostras.

Número de Parâmetros

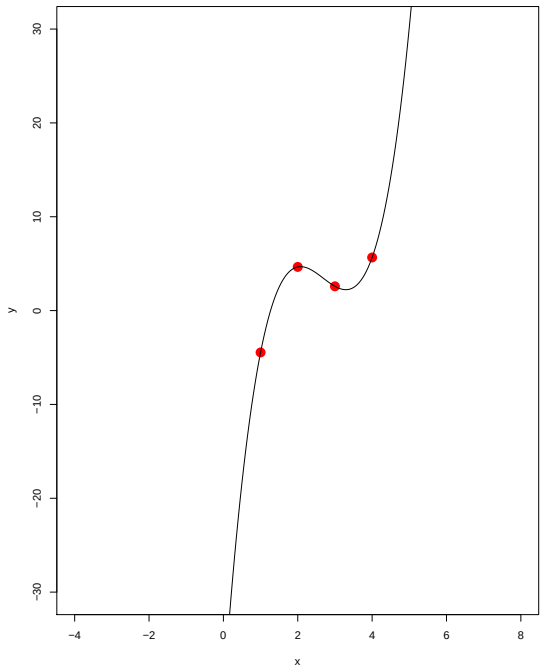


Figura: Número de parâmetros vs. número de amostras.

Número de Parâmetros

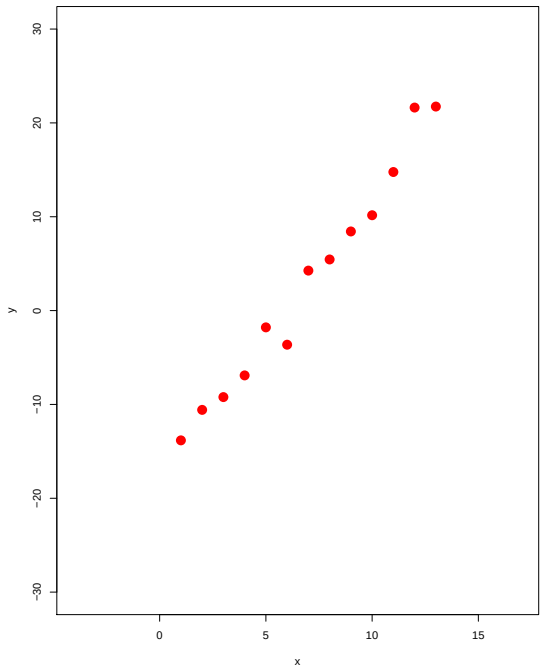


Figura: Conjunto de 13 pontos a serem fitados.

Número de Parâmetros

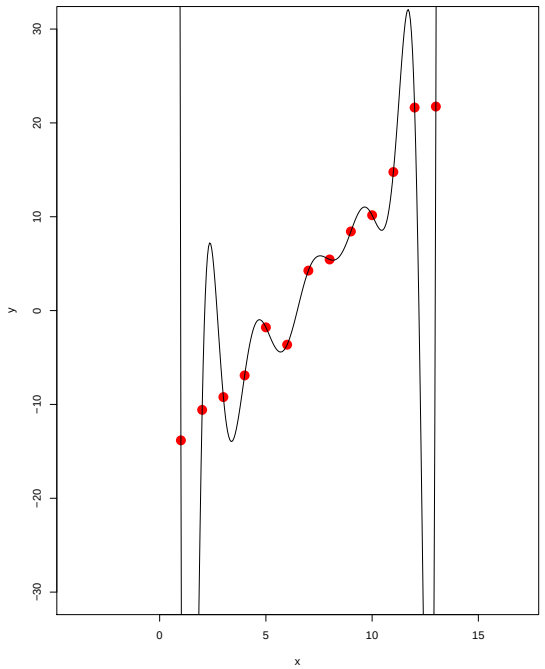


Figura: Fit com um polinômio de 12º.

Número de Parâmetros

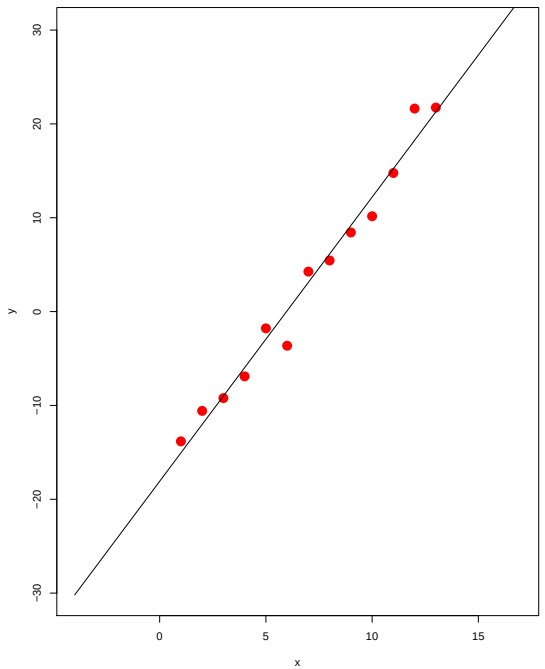
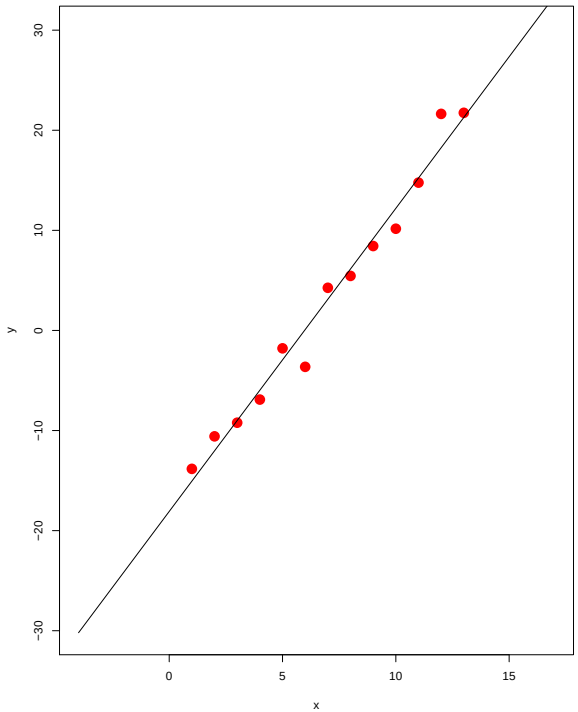
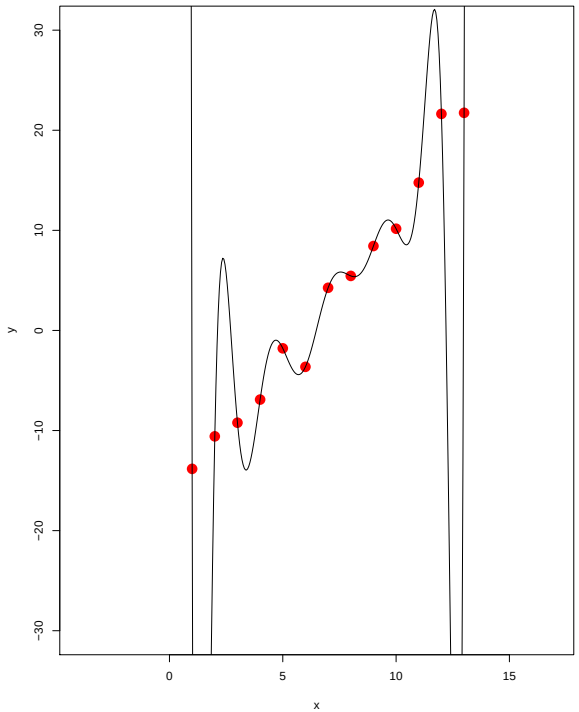


Figura: Fit com um polinômio de 1º.

Número de Parâmetros



(a) Polinômio de 1º grau.



(b) Polinômio de 12º graus.

Figura: Exemplos de fits de curvas polinomiais.

Número de Parâmetros

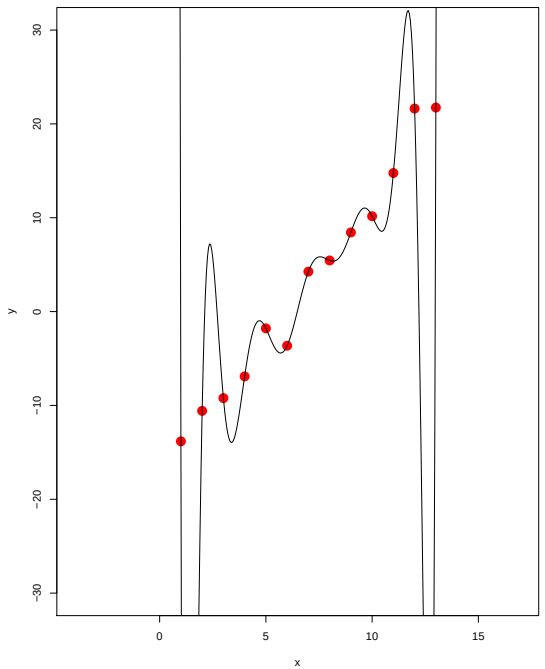


Figura: Exemplo de modelo overfitado (alta variância).

Número de Parâmetros

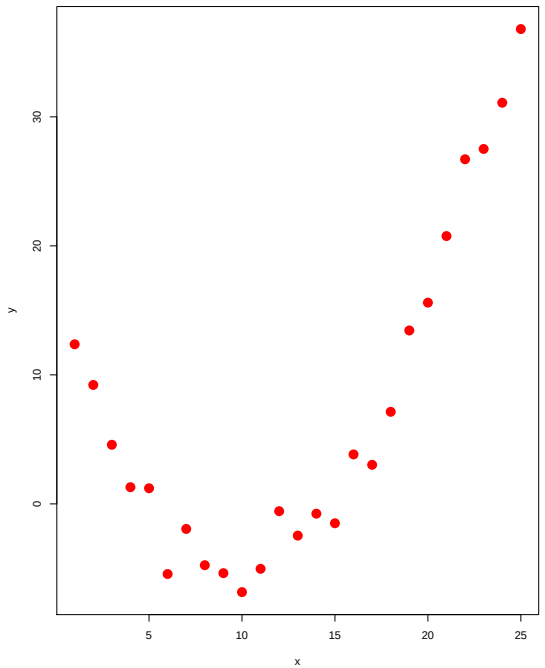


Figura: Conjunto de pontos a serem fitados.

Número de Parâmetros

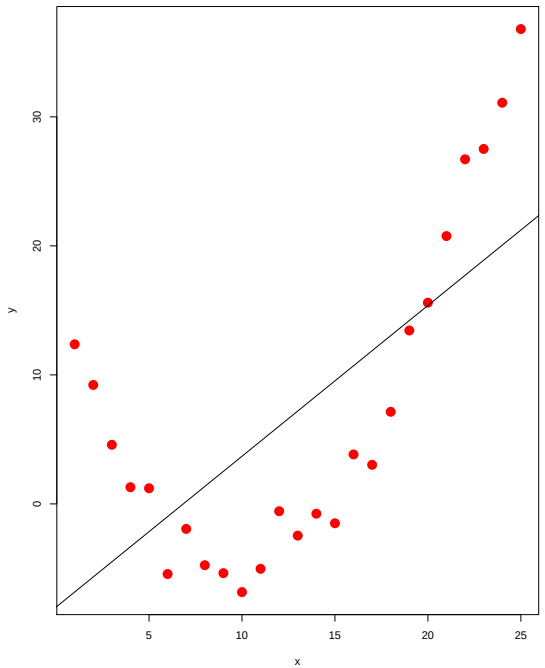


Figura: Fit com um polinômio de 1º grau.

Número de Parâmetros

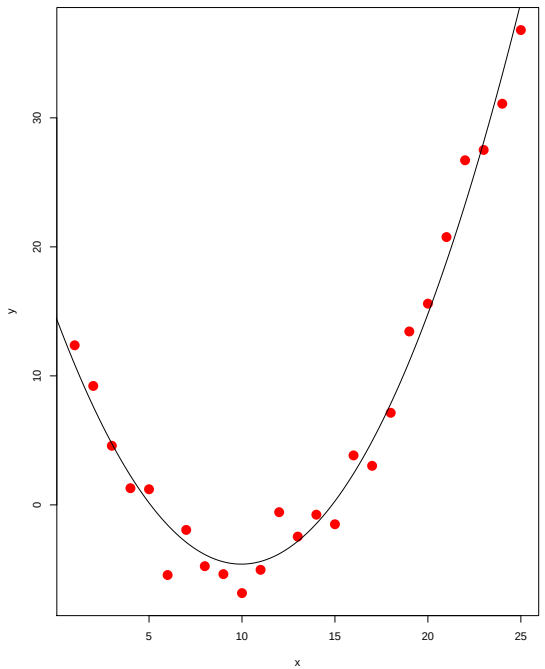
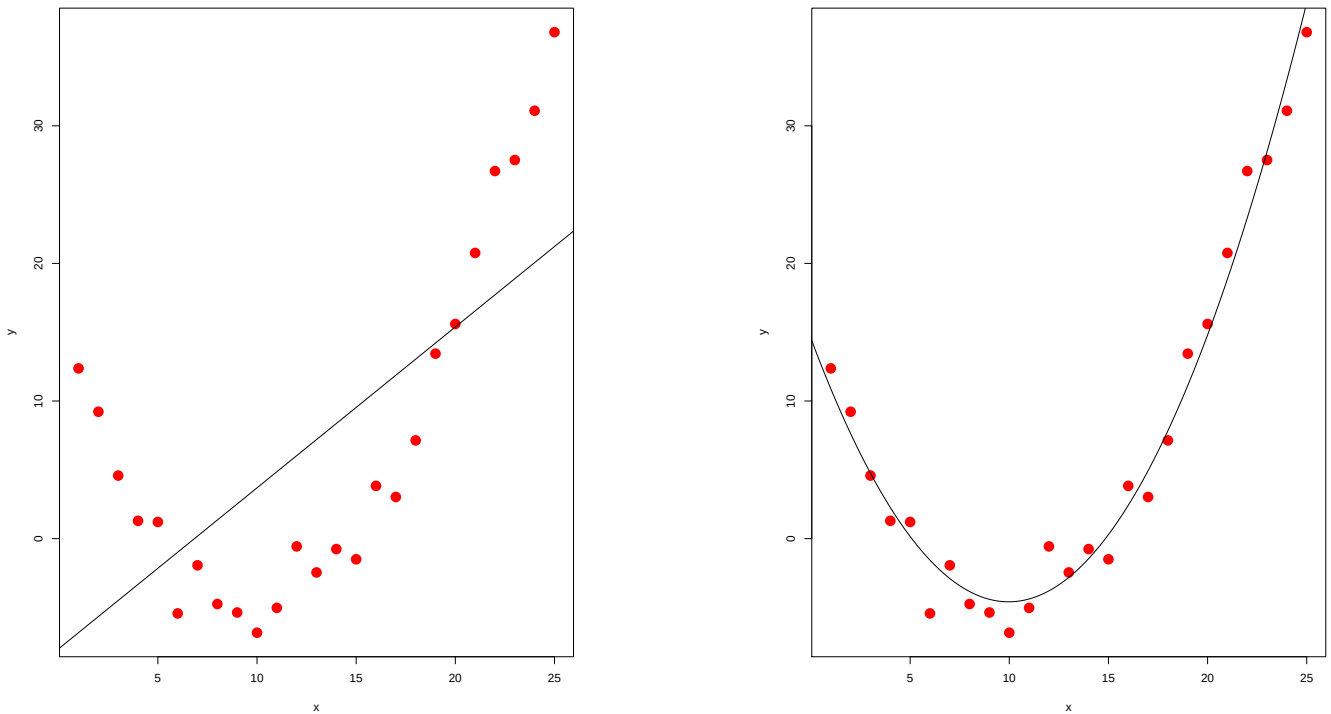


Figura: Fit com um polinômio de 2º grau.

Número de Parâmetros



(a) Polinômio de 1º grau. **(b)** Polinômio de 2º grau.

Figura: Exemplos de fits de curvas polinomiais.

Número de Parâmetros

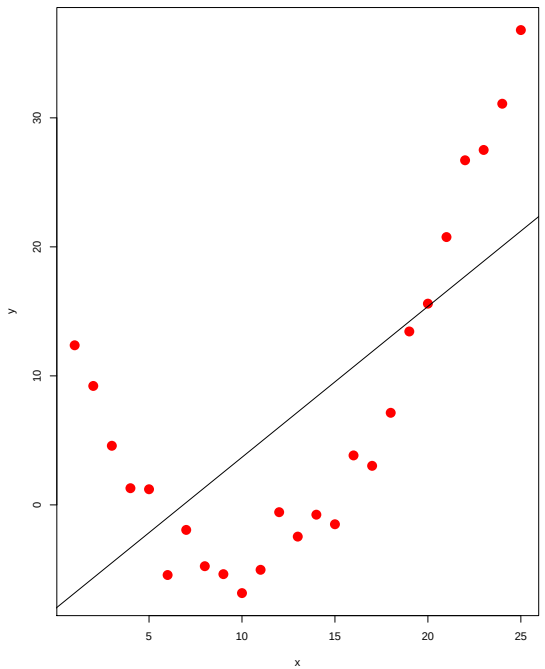


Figura: Exemplo de modelo underfitado (alto viés).

Número de Parâmetros

- Sempre é possível com *N samples* fitar um polinômio de grau (N-1) que passe exatamente pelos pontos
- Como escolher o grau do polinômio?
  - $y = a.x + b$
  - $y = a.x^2 + b.x + c$
  - $y = a.x^3 + b.x^2 + c.x + d$

Número de Parâmetros

Número de Parâmetros de um modelo

Em Machine Learning, apesar da maioria dos modelos não serem polinomiais, existe esse mesmo trade-off entre viés e variância. Por um lado, um modelo com poucos parâmetros pode ser simples demais para modelar os padrões dos dados; por outro lado, um modelo complexo demais consegue memorizar todas as variações do conjunto de treino, falhando em capturar os padrões realmente importantes dos dados.

Fit Polinomial

Demo - Polynomial Fit

Poly_Fit.ipynb

Agenda

Mapa de Métodos

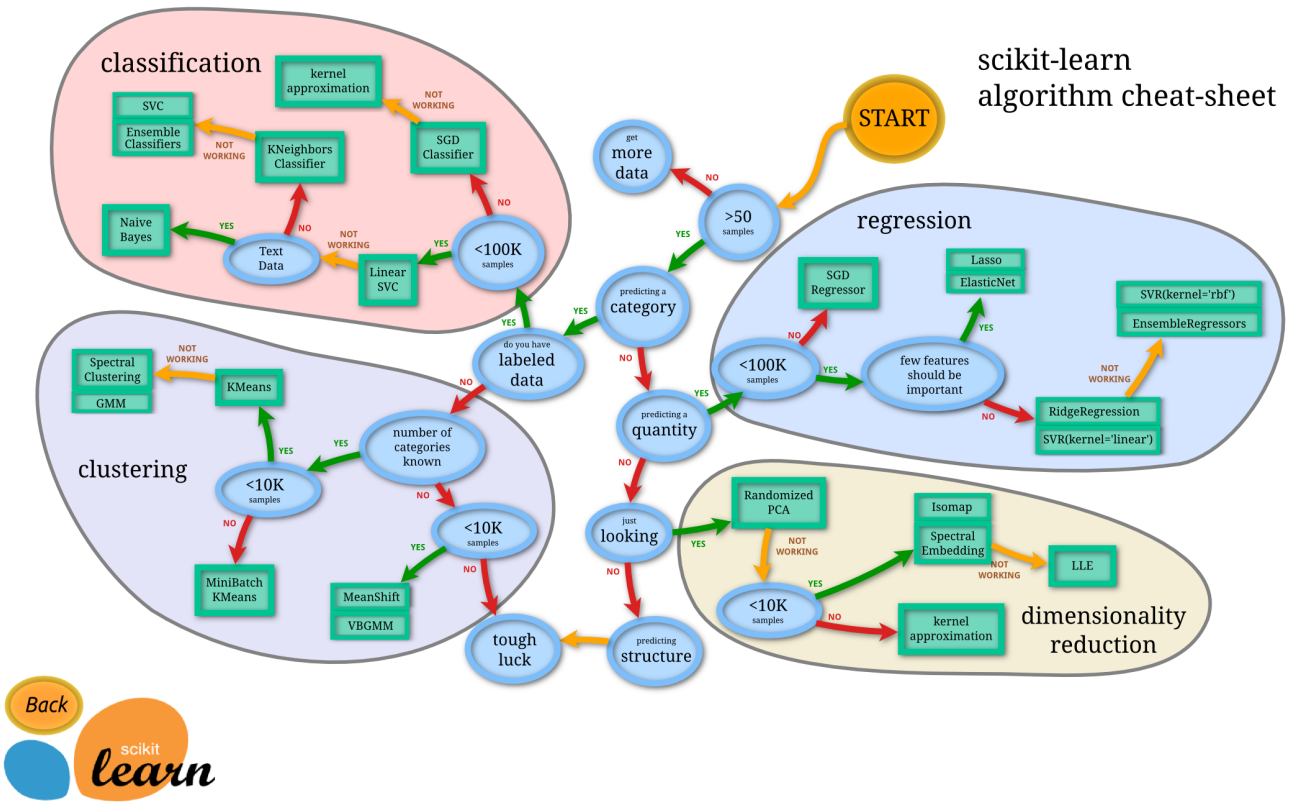


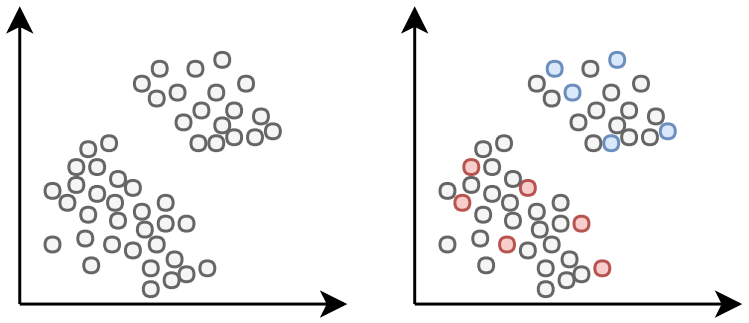
Figura: Mapa do scikit-learn[16] indicando os métodos indicados para cada tipo de problema.

[16] http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Tipos de Rotulação

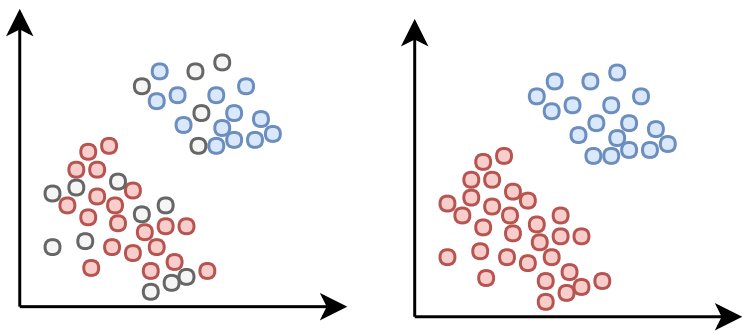
- Algoritmos Não-Supervisionados
  - Nenhum rótulo
- Algoritmos Fracamente Supervisionados
  - Poucas amostras rotuladas e muitas não rotuladas
- Algoritmos Semi-Supervisionados
  - Amostras rotuladas e não rotuladas
- Algoritmos Supervisionados
  - Amostras rotuladas

Tipos de Rotulação



(a) Não-Supervisionado

(b) Fracamente Supervisionado



(c) Semi-Supervisionado

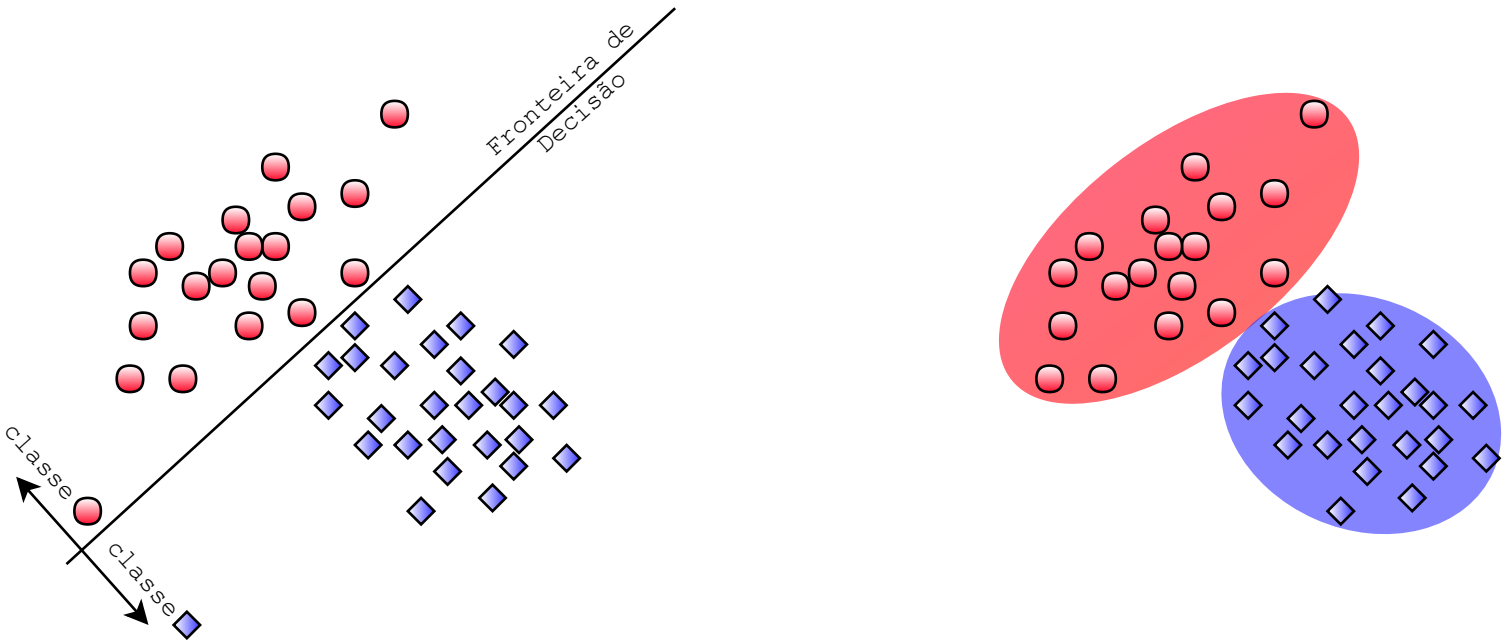
(d) Supervisionado

Figura: Algoritmos de acordo com os tipos de rotulação.

Tipos de Modelagem

- Modelos Generativos
  - Modelam a distribuição que gerou os dados
- Modelos Discriminativos
  - Modelam fronteiras de decisão entre diferentes classes

Tipos de Modelagem



(a) Modelo Discriminativo

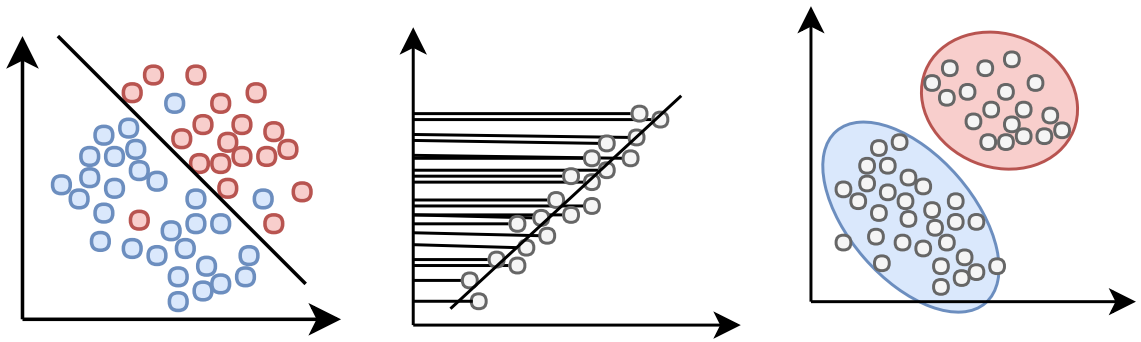
(b) Modelo Generativo

Figura: Tipos de Modelagem.

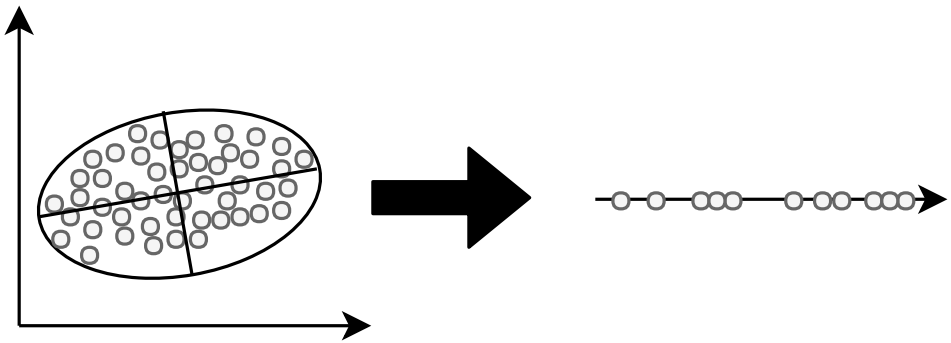
Tipos de Tarefas

- Classificação
  - Predizem classes
- Regressão
  - Predizem valores numéricos
- Clusterização
  - Acham os clusters naturais presentes no conjunto de dados
- Redução de Dimensionalidade
  - Diminuem a dimensionalidade de dados para propósitos de análise ou visualização por humanos

Tipos de Tarefas



(a) Classificação **(b)** Regressão **(c)** Clusterização



(d) Redução de Dimensionalidade

Figura: Algoritmos de acordo com os tipos de tarefas.

Notas sobre os Slides

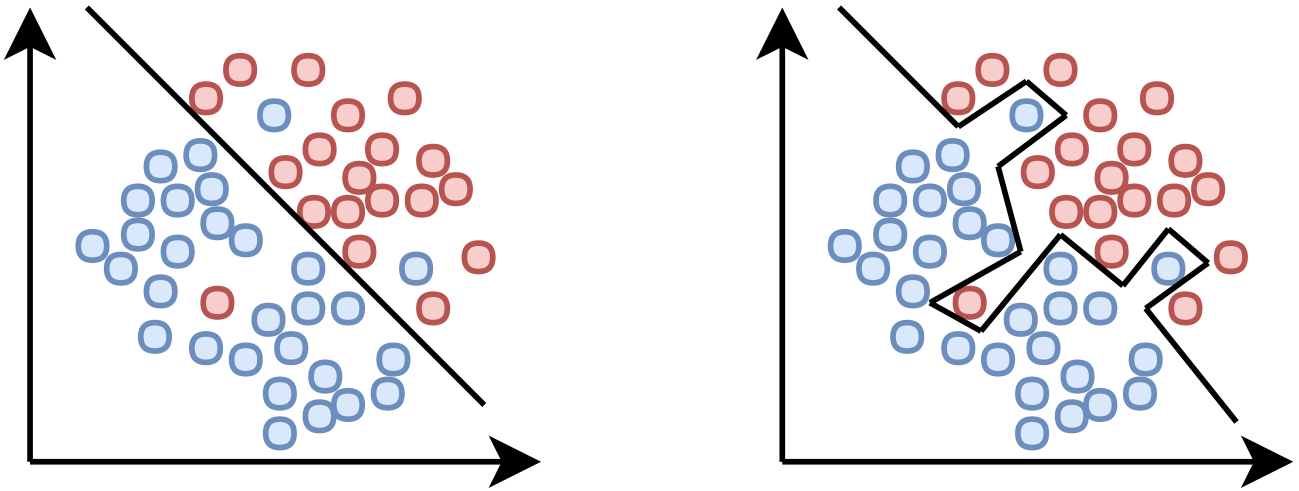
Notas sobre os Slides

Algumas das figuras usadas na seção a seguir sobre algoritmos de classificação foram adaptadas de outros conjuntos de slides. Mais especificamente, foram usados materiais do Prof. Jefersson Santos (DCC/UFMG), Prof. Alexander Ihler (UC/Irvine) e Prof. Ethem Alpaydin (Bogaziçi University).

Classificação

- Separação de amostras em classes
- Requer dados rotulados
- Otimização de uma fronteira de decisão

Overfitting em um Classificador



(a) (b)

Figura: Overfitting em um classificador.

Algoritmos de Classificação

- K Nearest Neighbors (KNN)
- Decision Trees (DT)
- Random Forest (RF)
- Support Vector Machine (SVM)
- AdaBoost

K Nearest Neighbors

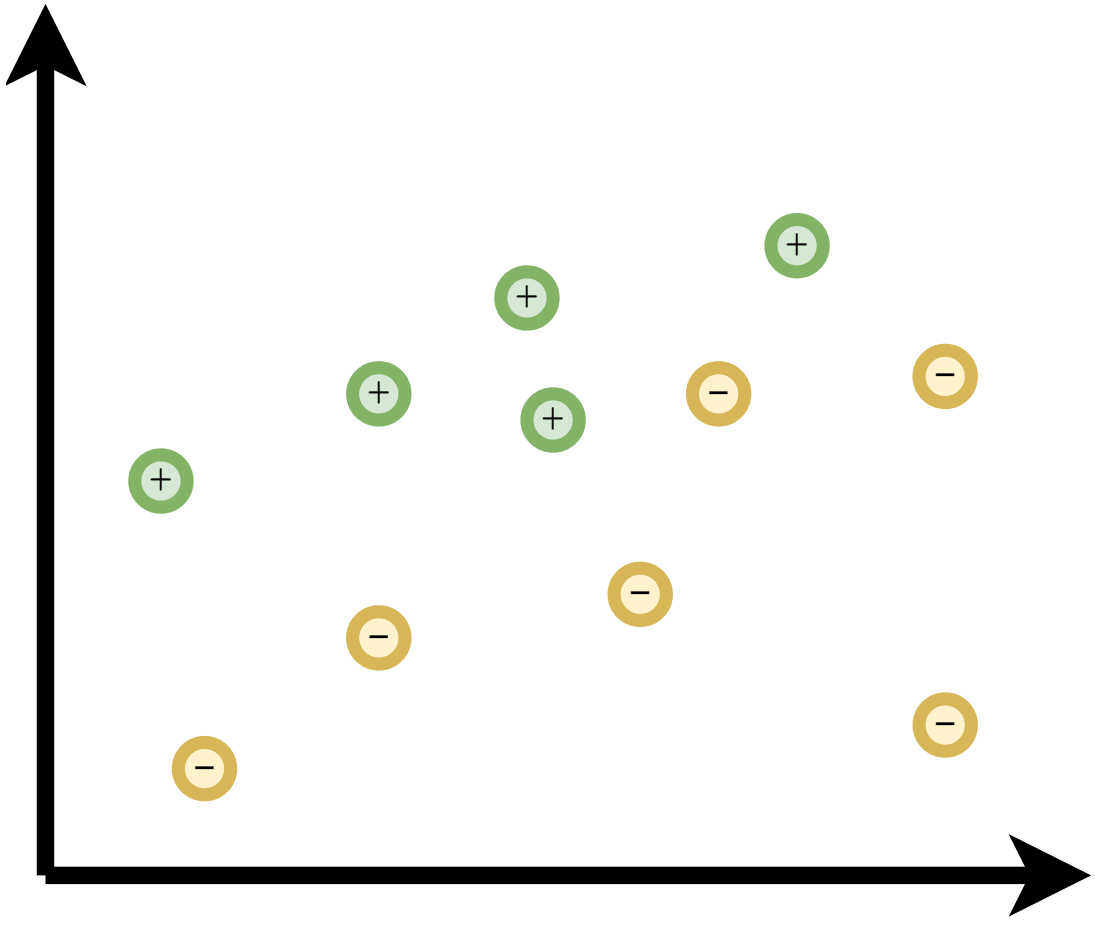


Figura: Exemplo de KNN.

K Nearest Neighbors

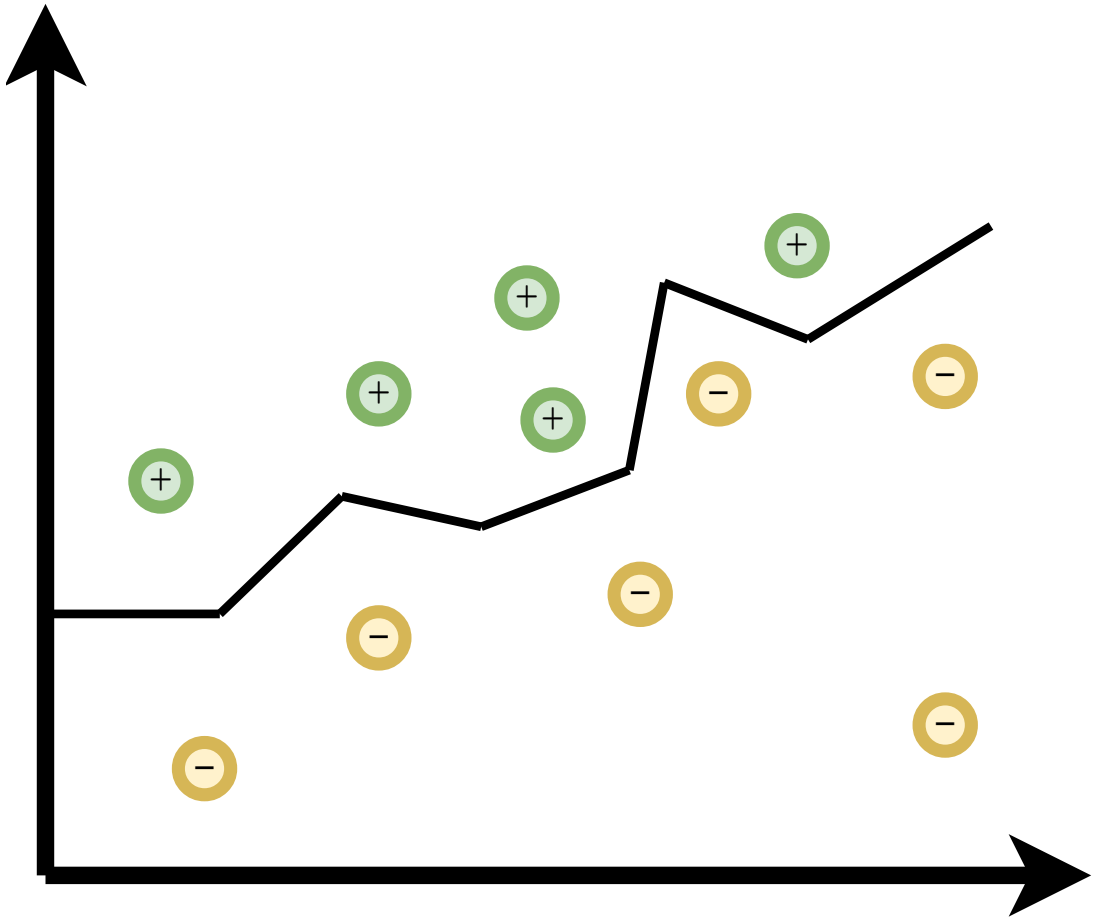


Figura: Exemplo de KNN.

K Nearest Neighbors

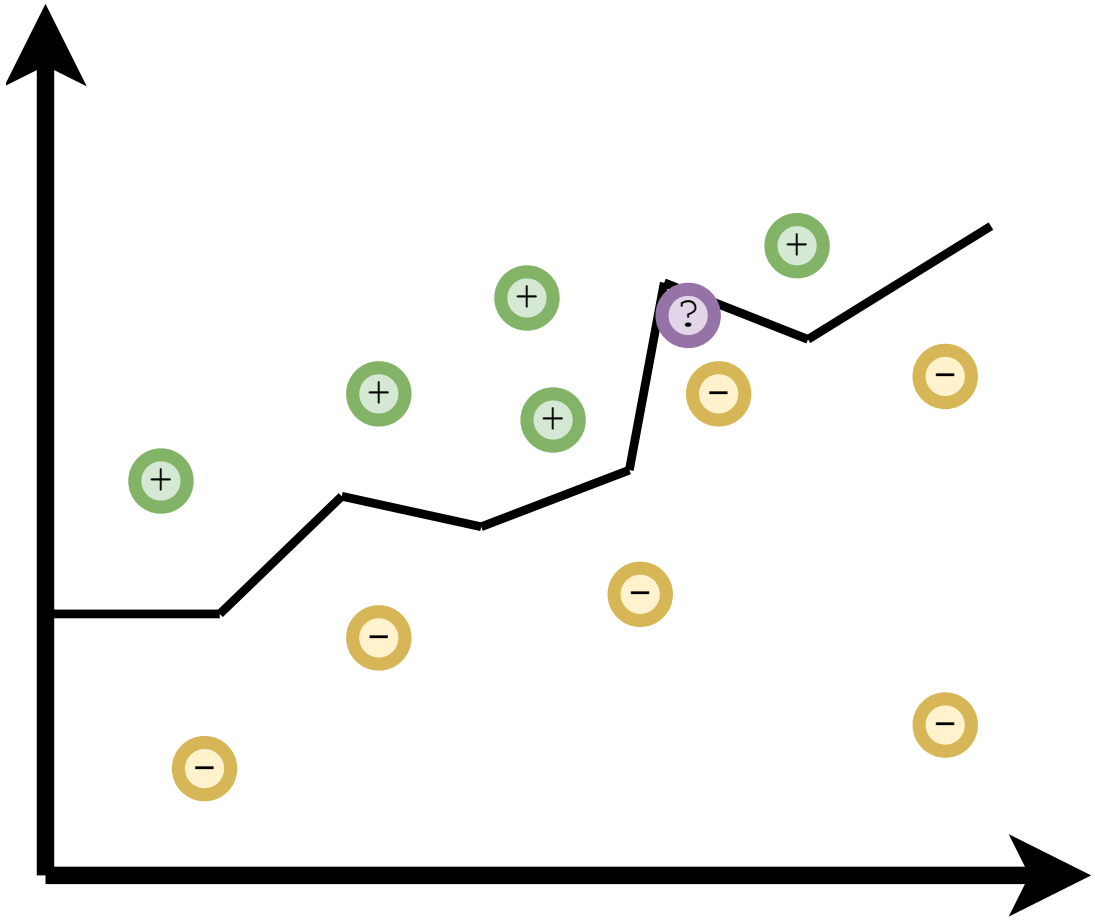


Figura: Exemplo de KNN.

K Nearest Neighbors

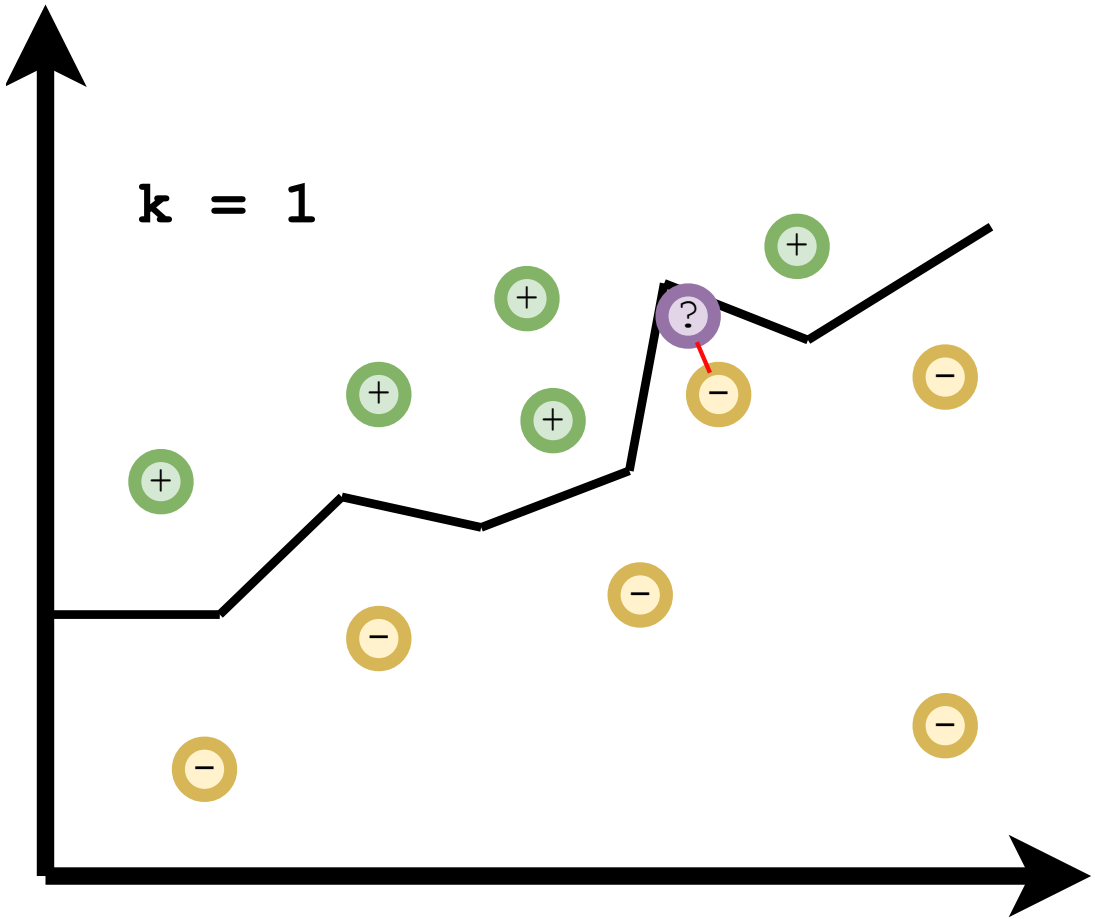


Figura: Exemplo de KNN.

K Nearest Neighbors

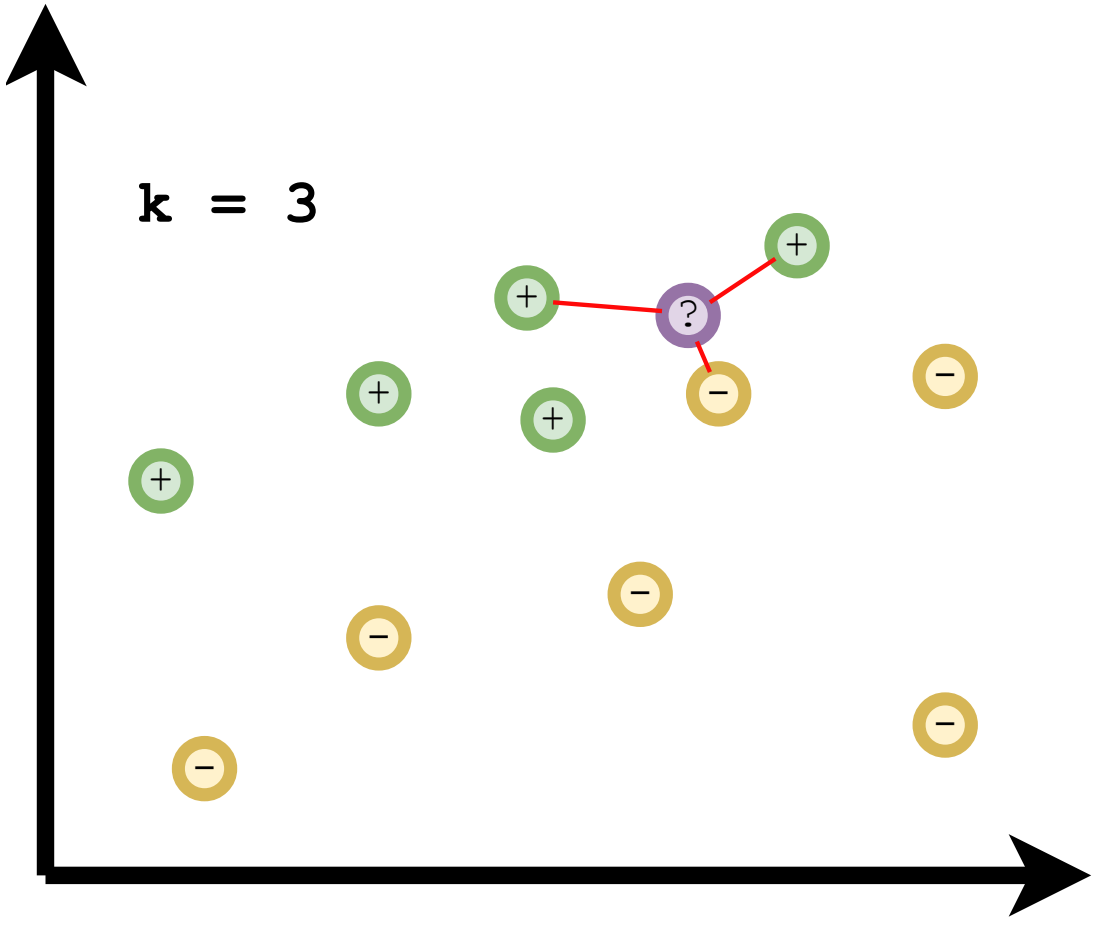


Figura: Exemplo de KNN.

K Nearest Neighbors

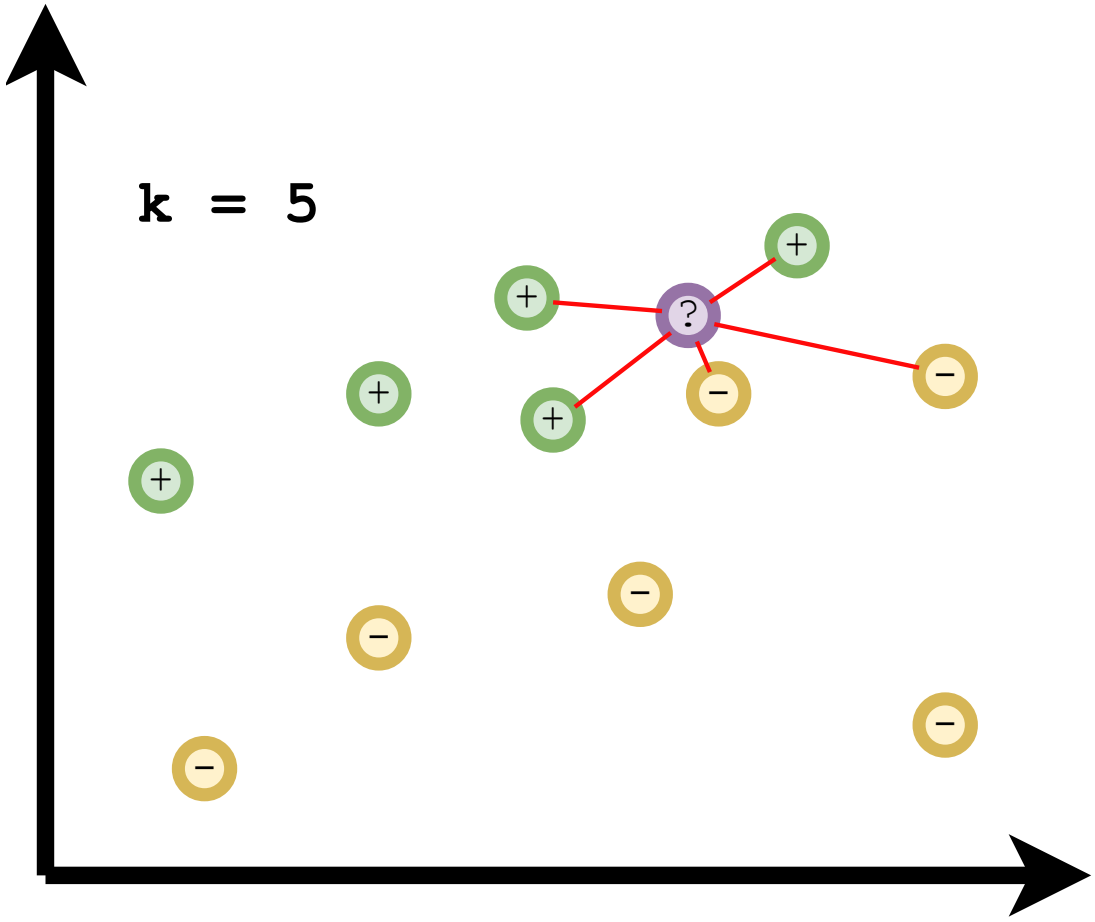


Figura: Exemplo de KNN.

K Nearest Neighbors

- Vantagens
  - Não possui etapa de treinamento
  - Processo de teste extremamente simples (distância euclideana)
  - Poucos hiperparâmetros a serem otimizados
- Desvantagens
  - Todas as amostras são salvas no modelo
  - Para o teste, uma busca deve ser feita com complexidade $\Theta(N)$
  - Problemas com generalização e *outliers*

KNN

Demo - KNN

KNN.ipynb

Decision Trees

- Naturalmente hierárquico
- Computacionalmente barato
- Algoritmo recursivo simples de se implementar

Exemplo de Decision Tree

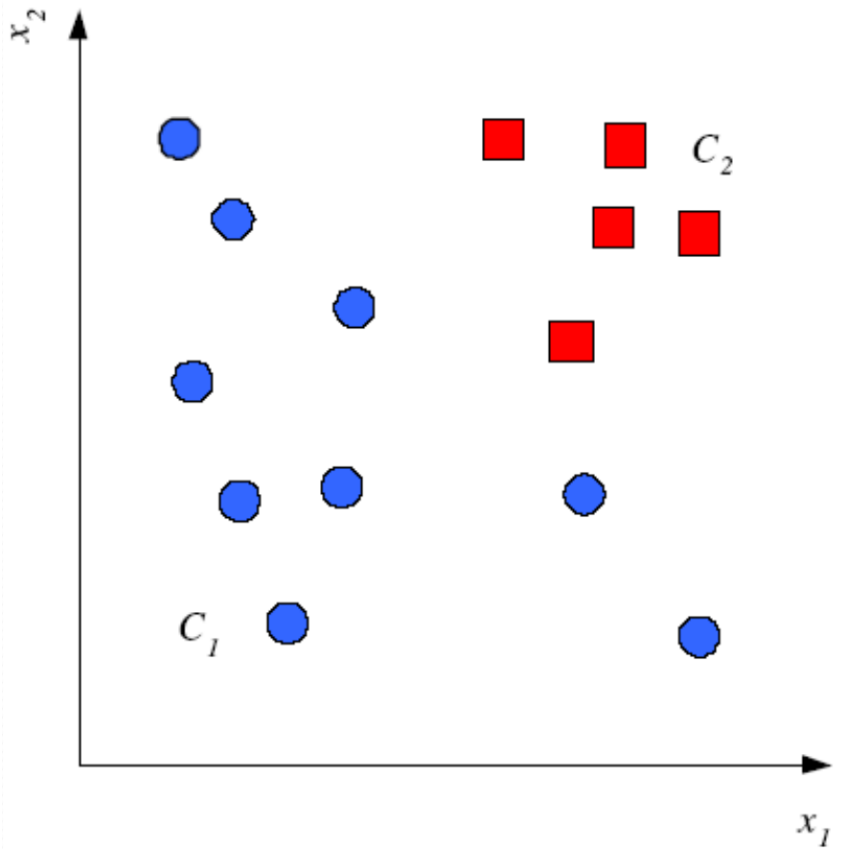


Figura: Exemplo de Decision Tree.

Exemplo de Decision Tree

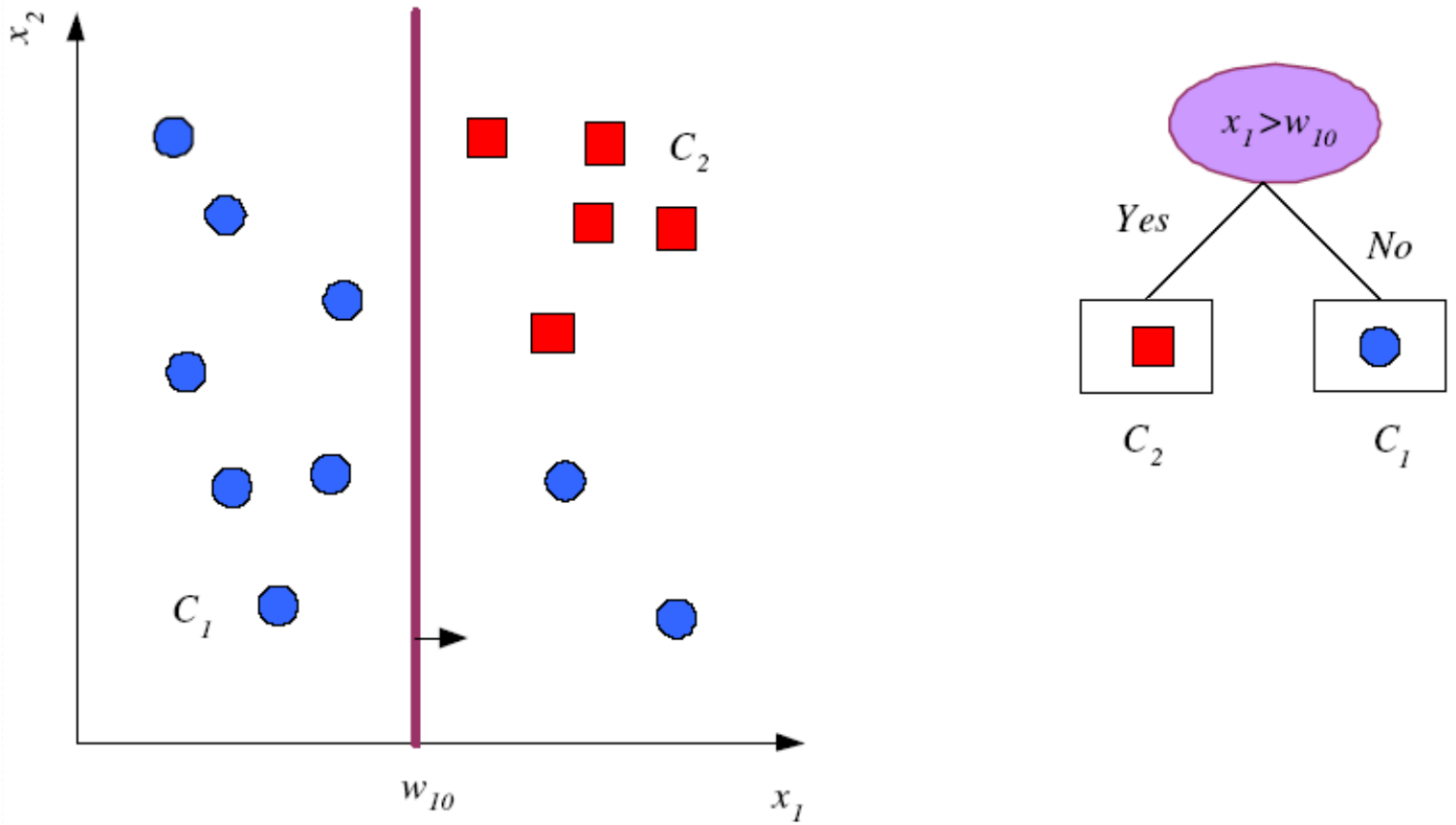


Figura: Exemplo de Decision Tree.

Exemplo de Decision Tree

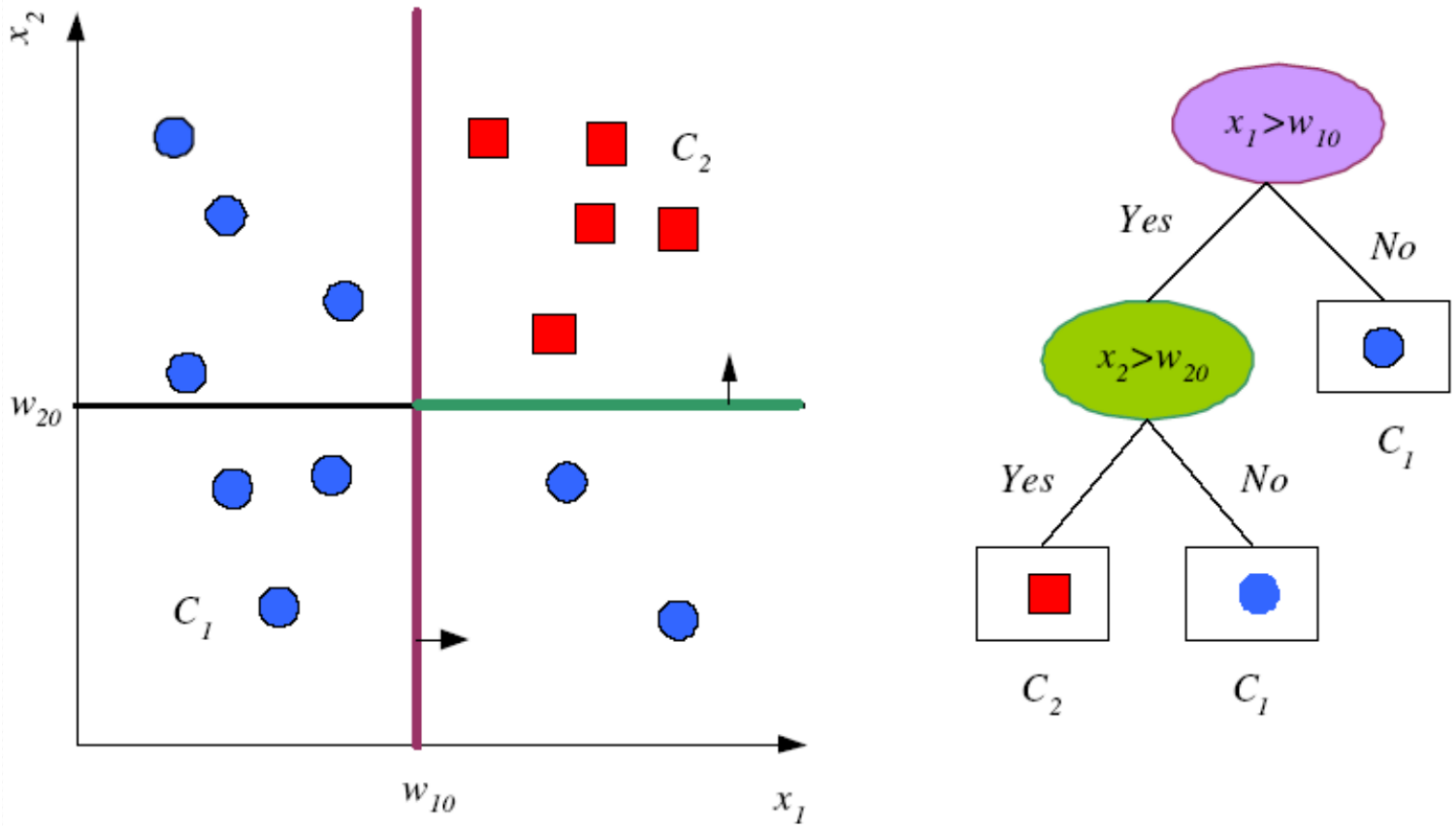


Figura: Exemplo de Decision Tree.

Exemplo de Decision Tree

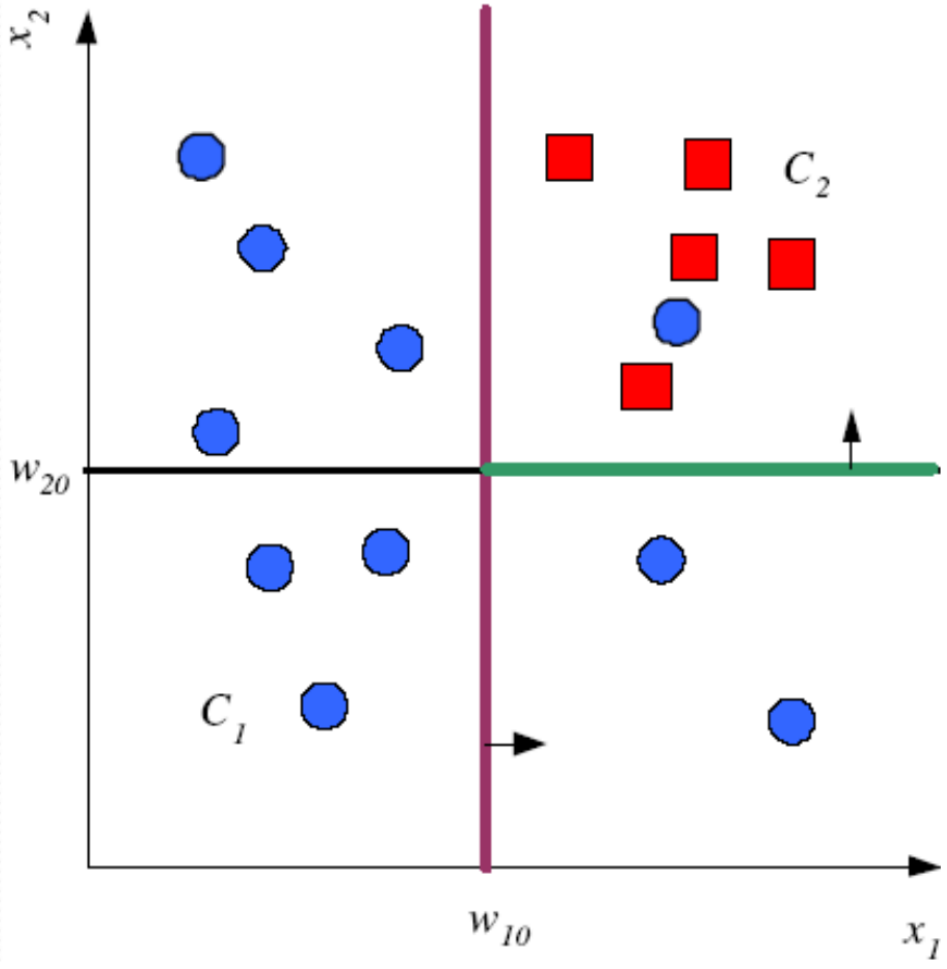


Figura: Exemplo de Decision Tree.

Exemplo de Decision Tree

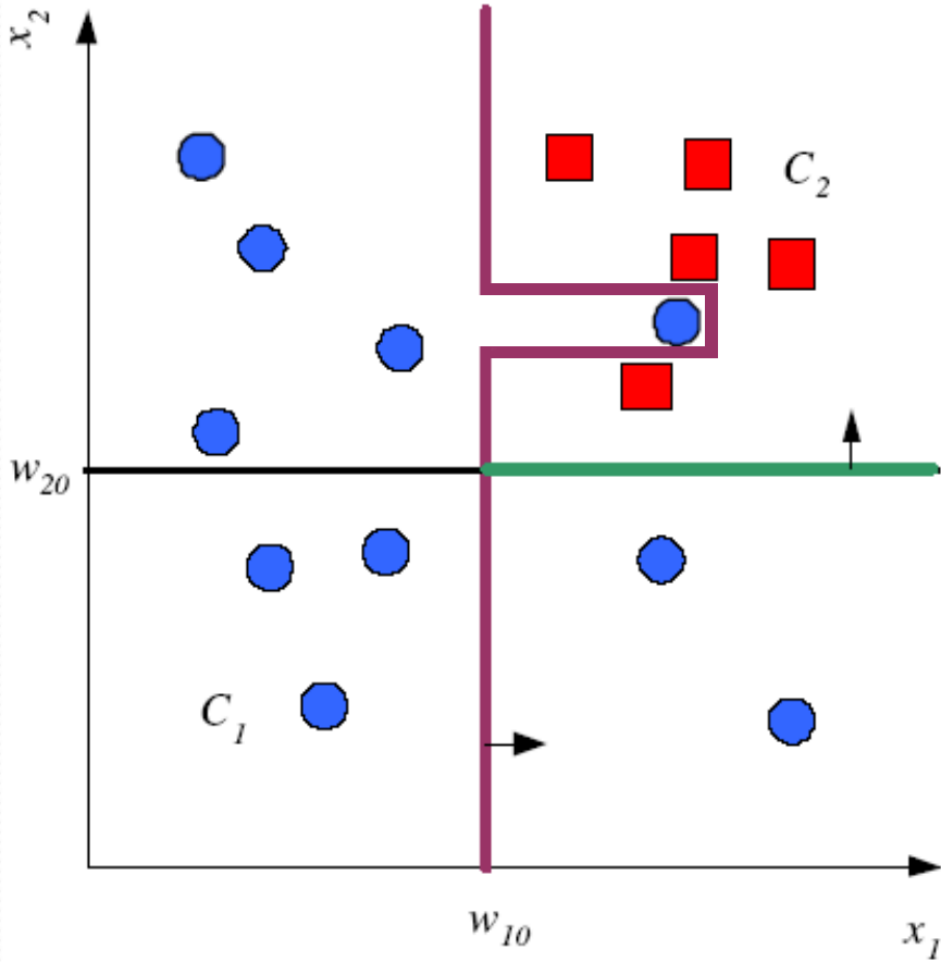


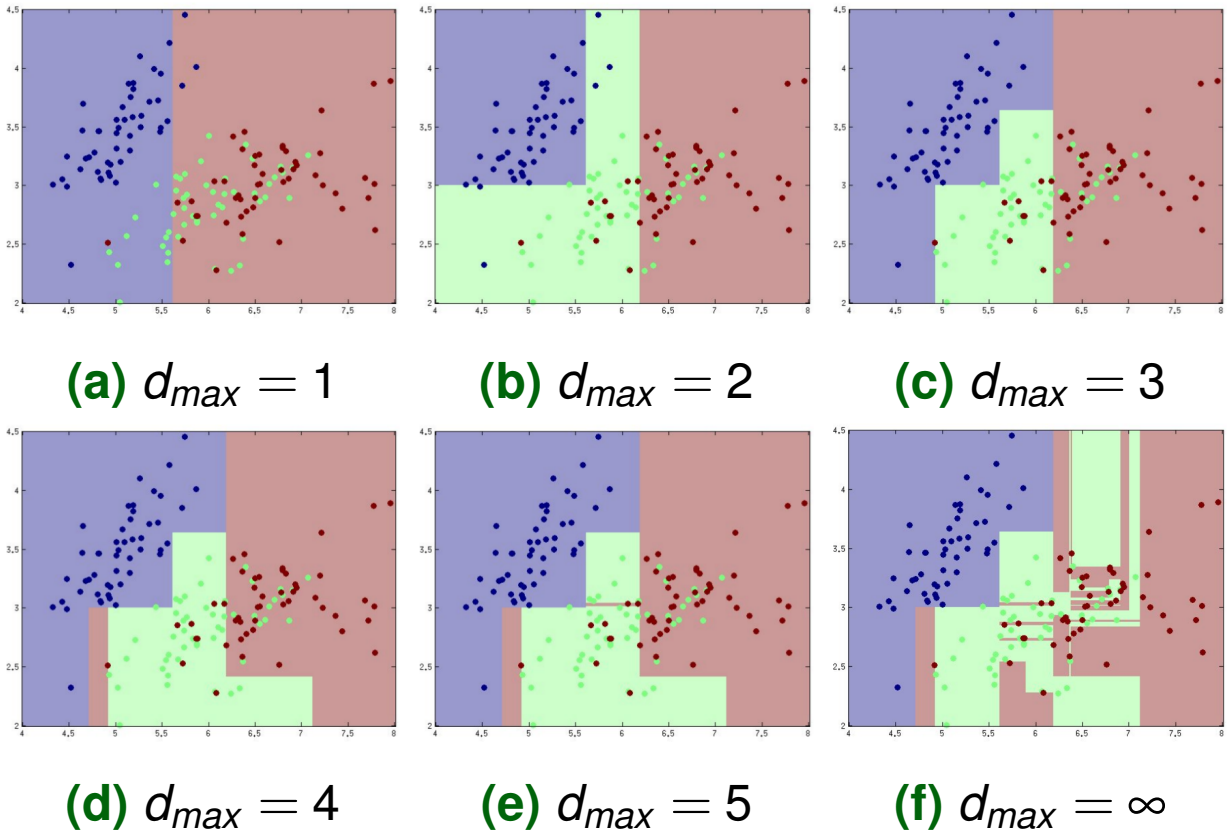
Figura: Exemplo de Decision Tree.

Decision Trees

Complexidade em Decision Trees

É possível enviesar o modelo controlando a altura máxima da árvore.

Controlando a Complexidade das DTs



(a) $d_{max} = 1$ **(b)** $d_{max} = 2$ **(c)** $d_{max} = 3$

(d) $d_{max} = 4$ **(e)** $d_{max} = 5$ **(f)** $d_{max} = \infty$

Figura: Compexidade das DTs.

Decision Trees

- Early Stop é normalmente utilizado para evitar overfit
  - Caso o modelo não seja impedido de crescer, ruído nos dados facilmente gera overfit
  - Um modelo é considerado bom e para de crescer quando os nós atingem um certo grau de pureza

Decision Trees

Pureza de Nós em uma Decision Tree

Como medir pureza nos nós de uma Decision Tree?

Decision Trees

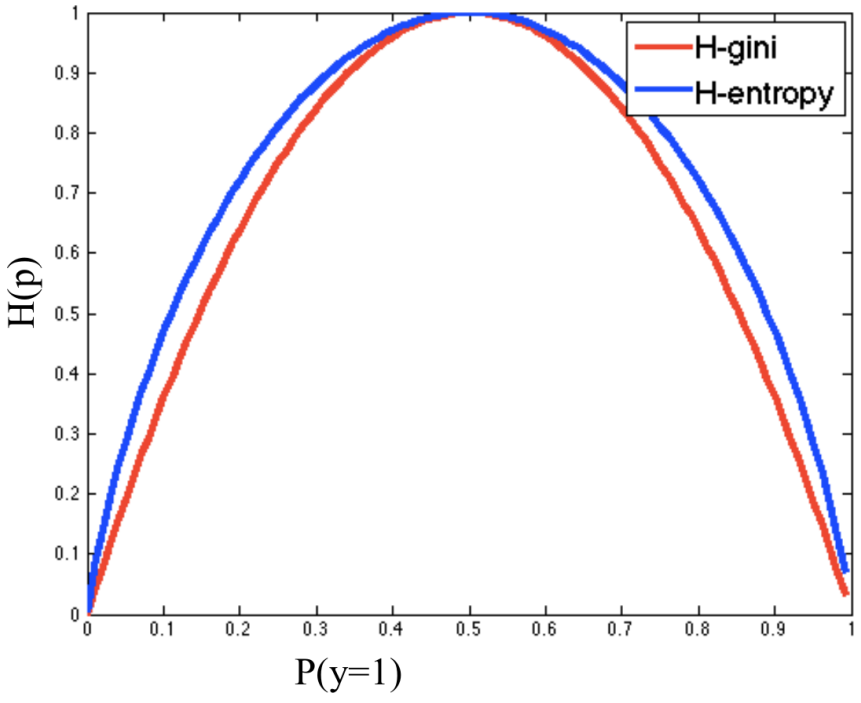


Figura: Pureza nos nós de uma Decision Tree.

Decision Trees

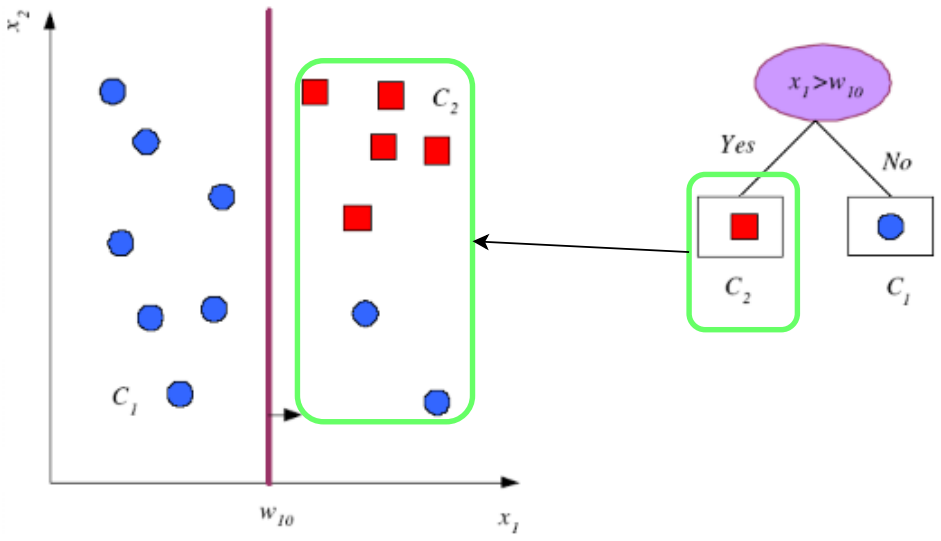


Figura: Pureza nos nós de uma Decision Tree.

- Classe vermelha: $\frac{5}{7} \approx 0.71$
- Classe azul: $\frac{2}{7} \approx 0.29$

Decision Trees

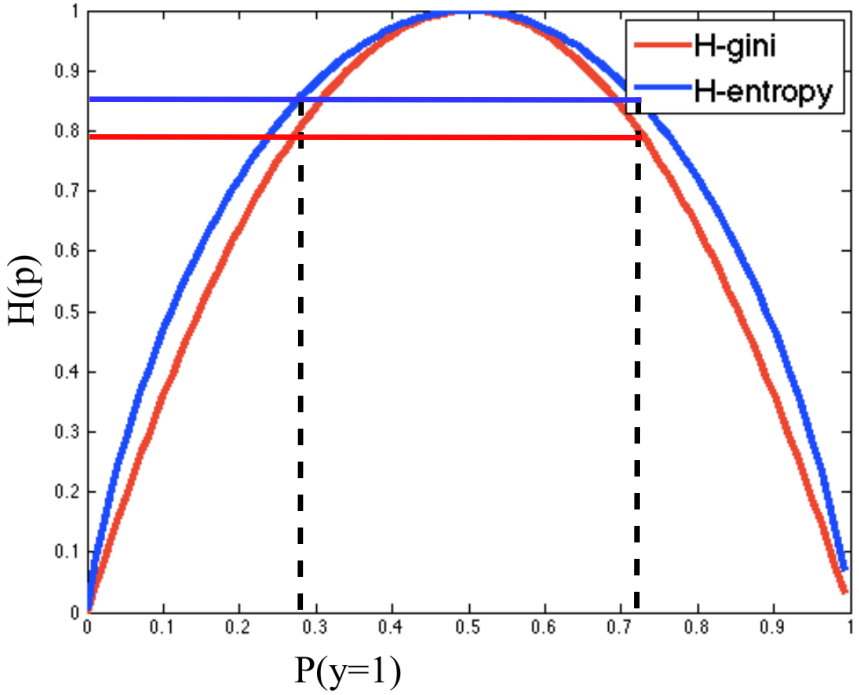


Figura: Pureza nos nós de uma Decision Tree.

- Gini: $G(p) = 1 - \sum_{i=1}^{J} p_i^2$
- Entropia: $H(p) = -\sum_{i=1}^{J} p_i \log_2 p_i$

Decision Trees

Exercício - Overfit/Unverfit em KNNs e Decision Trees

KNN_and_Decision_Trees.ipynb

Support Vector Machines

- Boa capacidade de generalização (maximização da margem)
- Robusto para dados com muitas dimensões
- Forte background matemático
- Possibilidade do uso de kernels
- Modelo bastante compacto após treinamento

Support Vector Machines

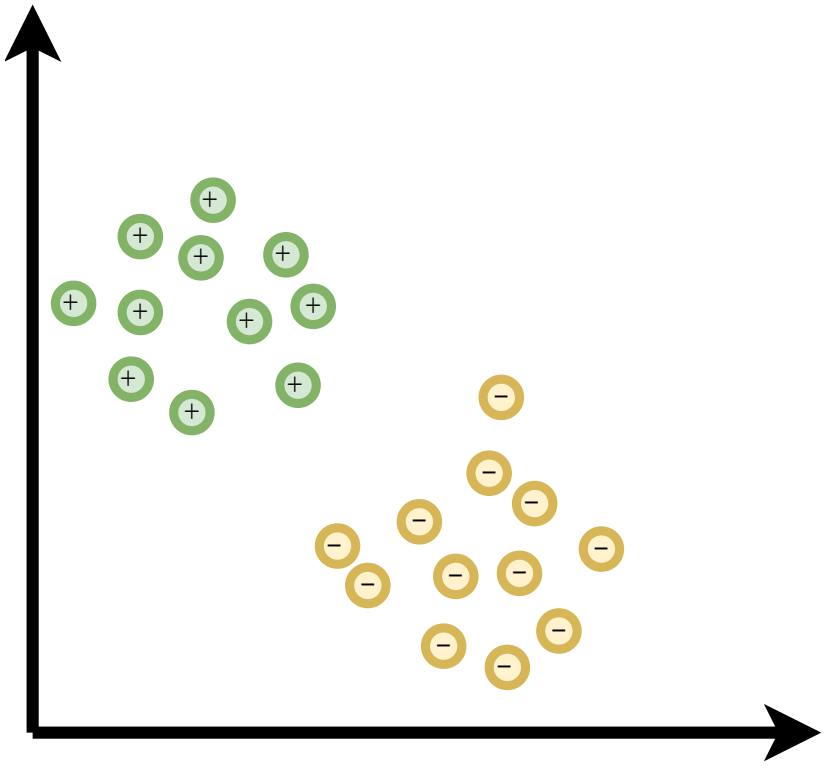


Figura: Classificação usando Support Vector Machines.

Support Vector Machines

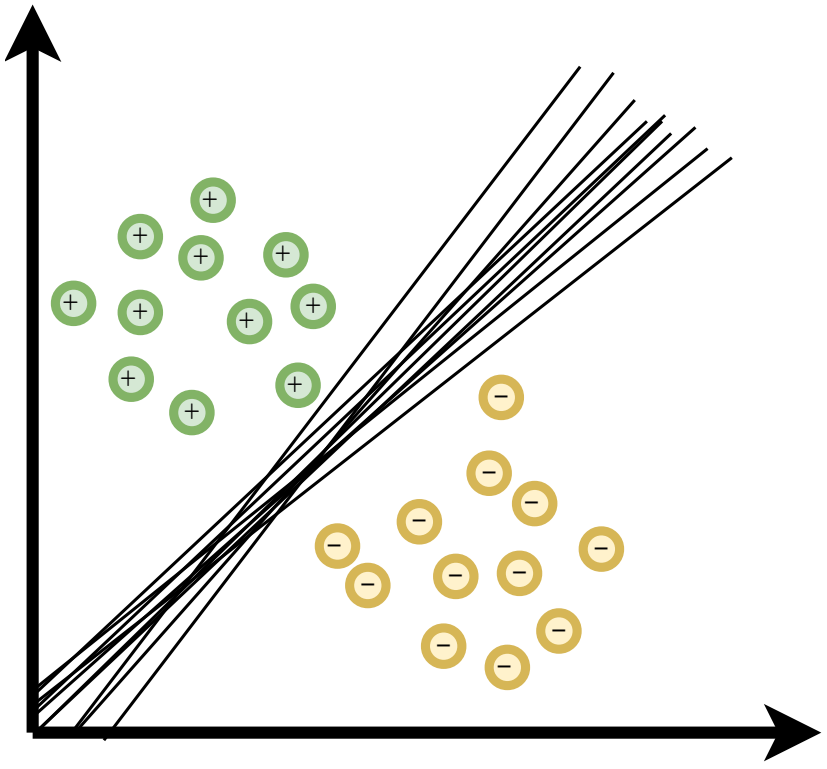


Figura: Classificação usando Support Vector Machines.

Support Vector Machines

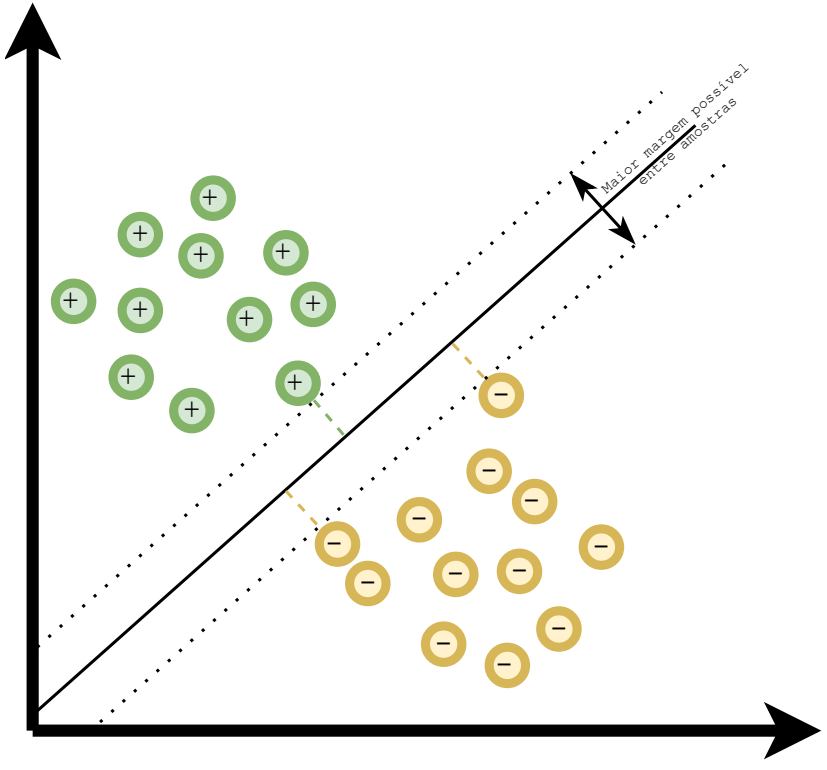


Figura: Classificação usando Support Vector Machines.

Support Vector Machines

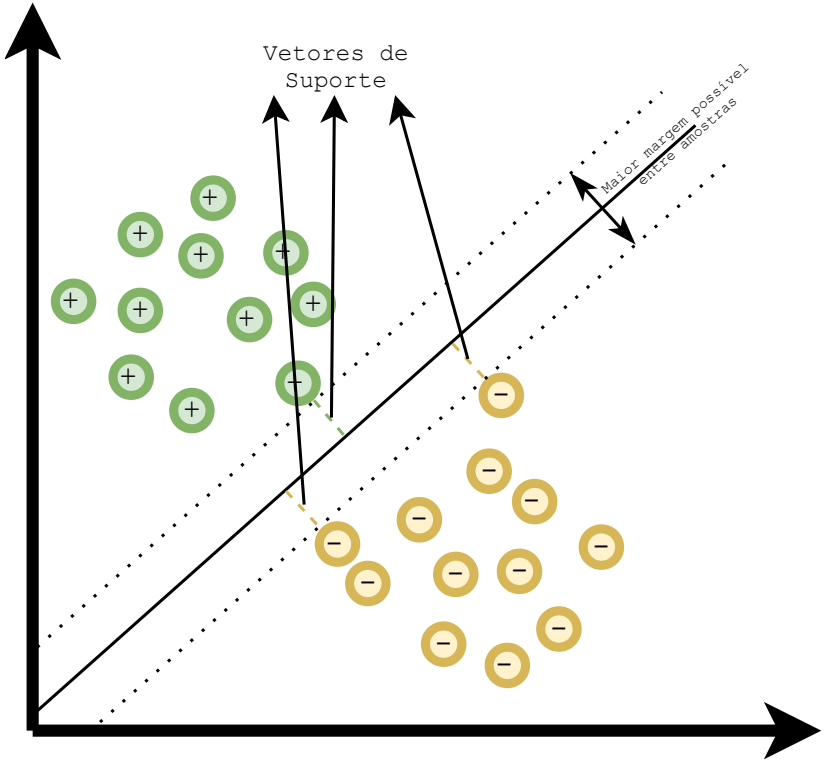


Figura: Classificação usando Support Vector Machines.

Support Vector Machines

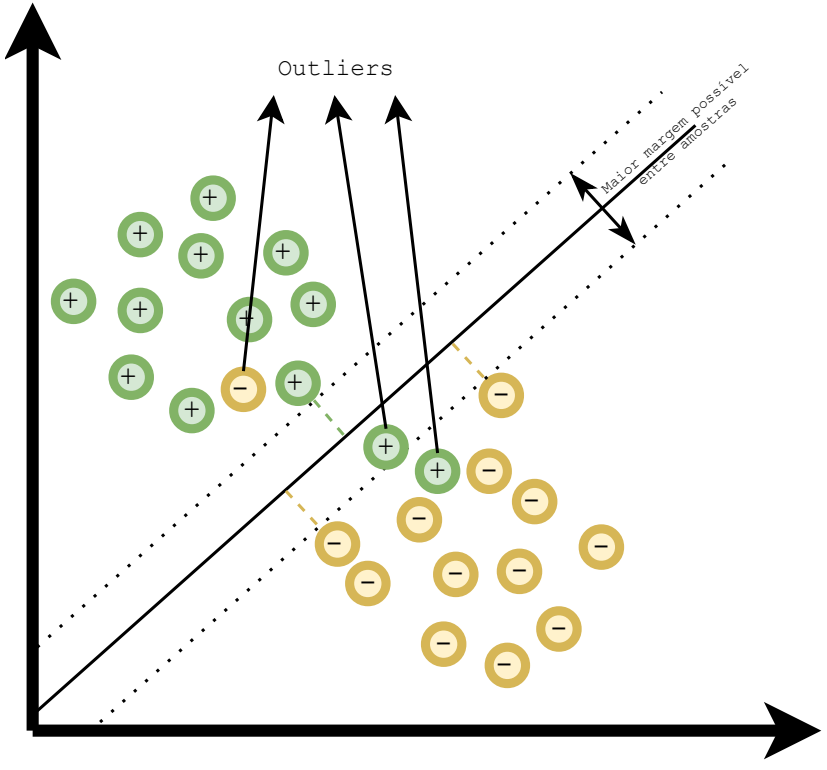


Figura: Classificação usando Support Vector Machines.

Kernels em Support Vector Machines

- Muitas vezes os dados de treinamento não são linearmente separáveis
- SVMs pertencem a um conjunto de algoritmos que podem contar com o uso de kernels
- Várias funções diferentes podem servir como kernels
  - Funções Polinomiais
  - Radial Basis Function (RBF)
  - Tangente Hiperbólica (tanh)

Kernels

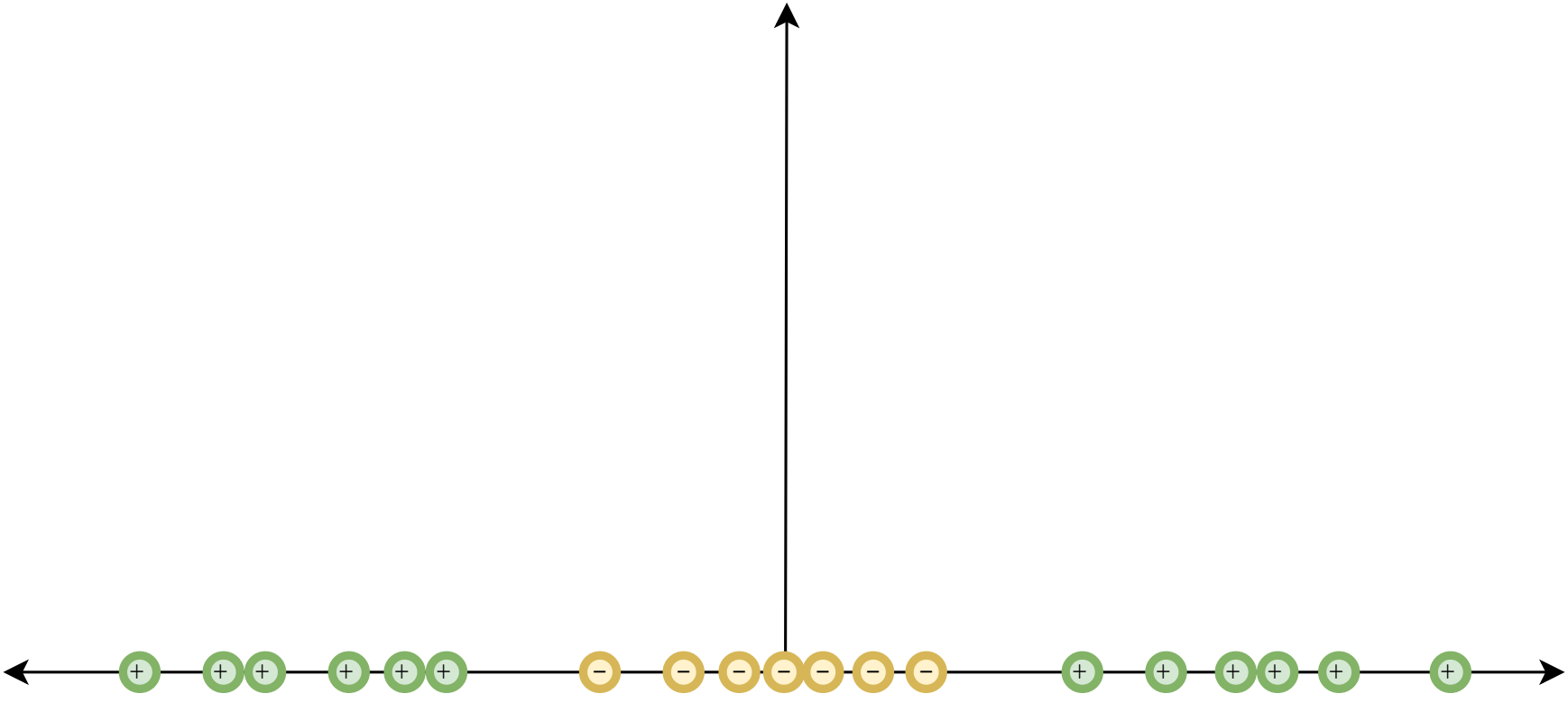


Figura: Kernels em Support Vector Machines.

Kernels

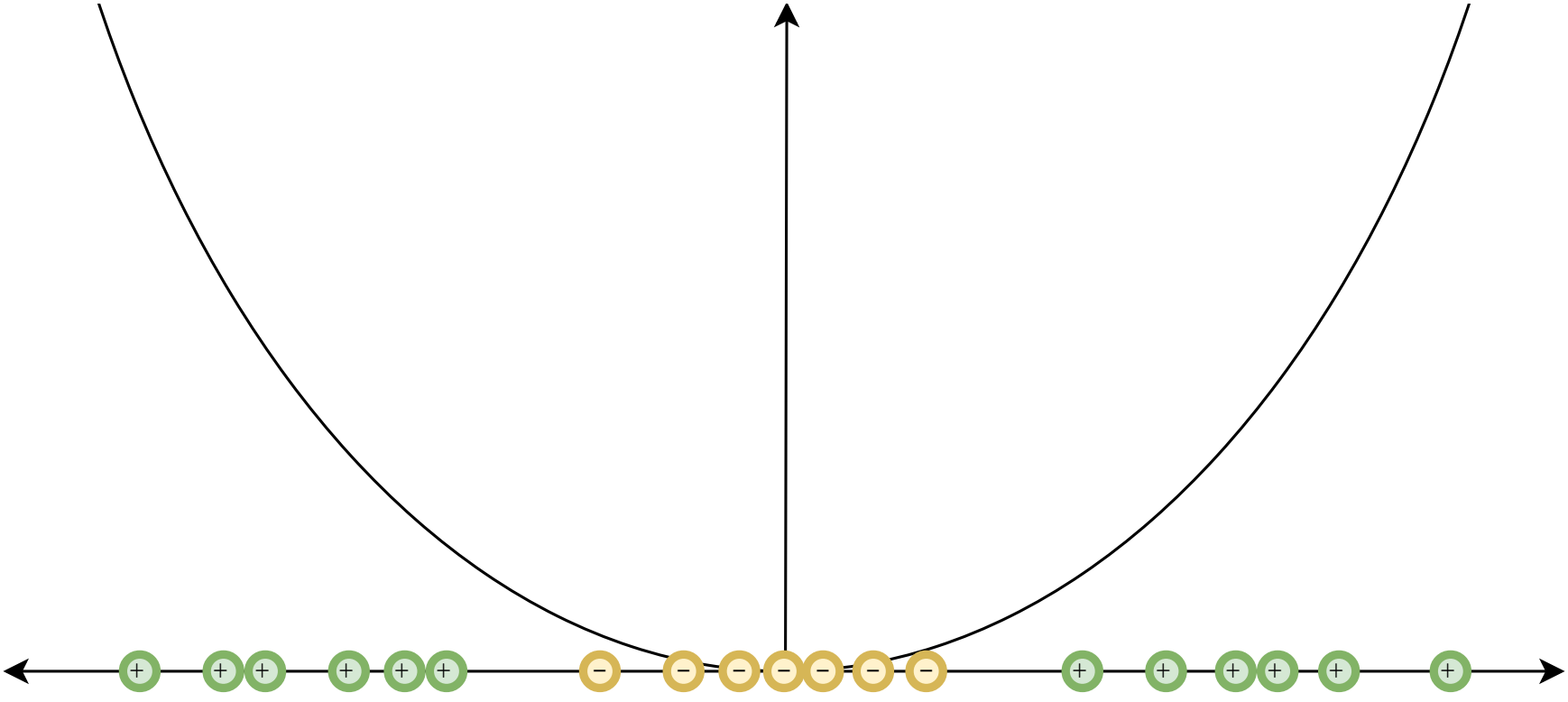


Figura: Kernels em Support Vector Machines.

Kernels

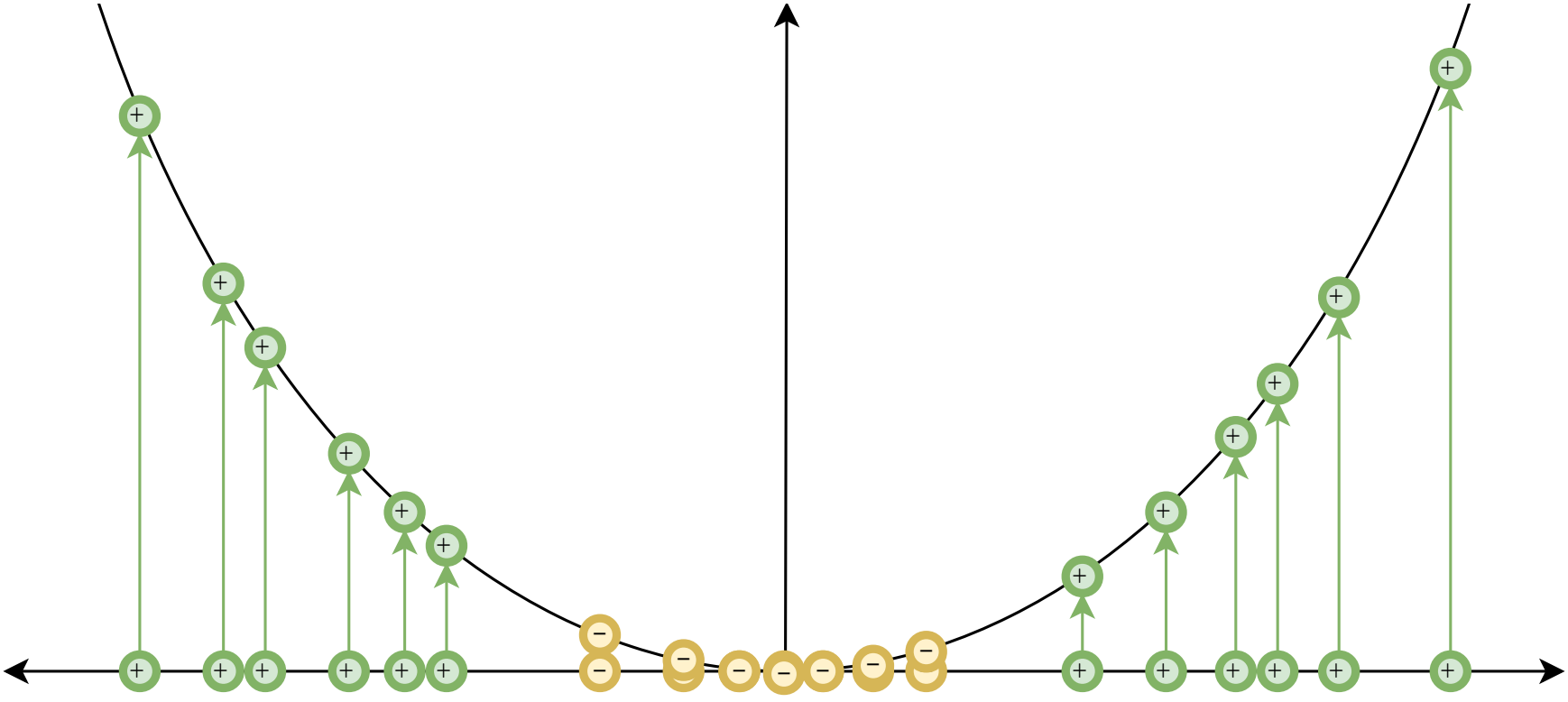


Figura: Kernels em Support Vector Machines.

Kernels

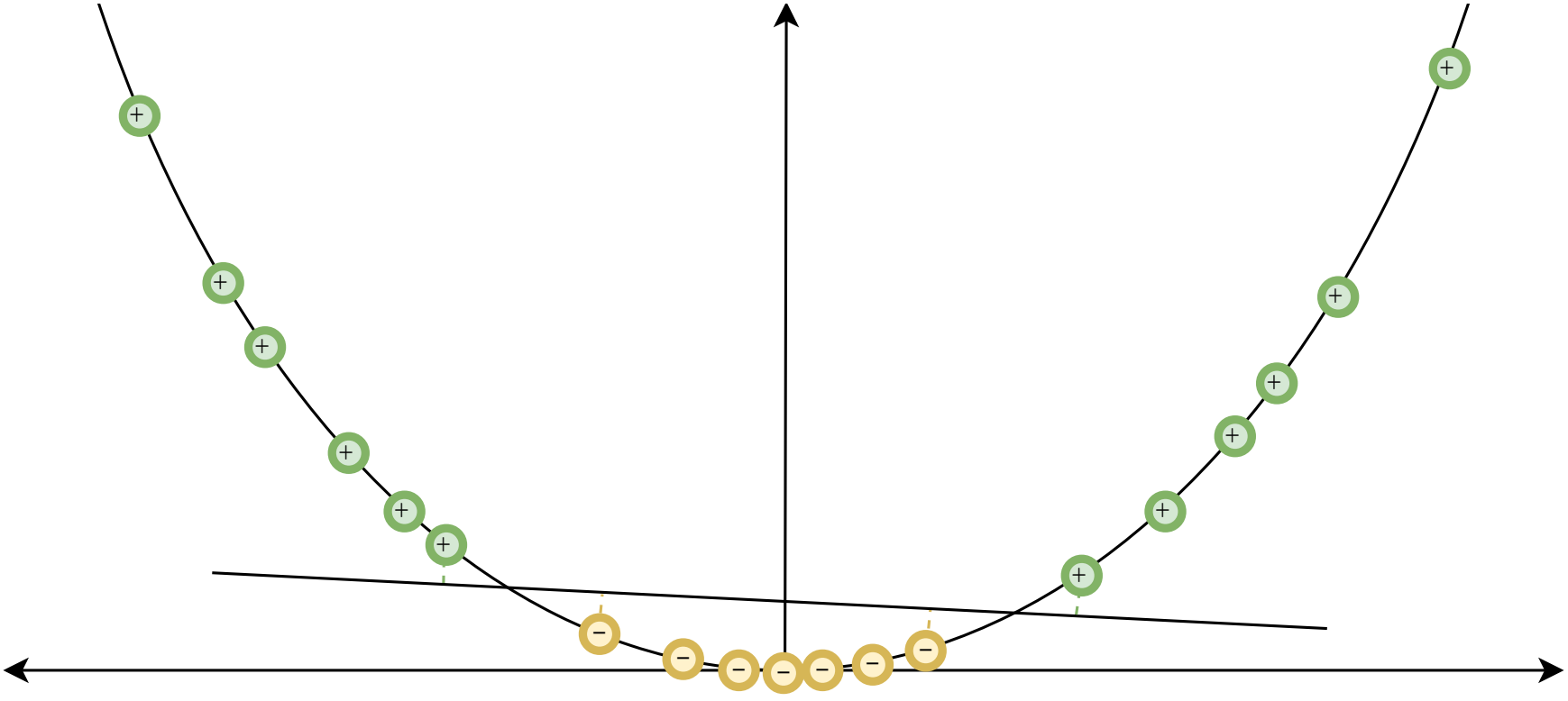


Figura: Kernels em Support Vector Machines.

SVM Kernels

Demo - SVM Kernels

SVM_Kernels.ipynb

Agenda

Avaliação de Modelos

Avaliação de Modelos

Como garantir que meu modelo está com um bom trade-off entre viés e variância?

Avaliação de Modelos

- É preciso quantificar a quantidade de erros de predição que o modelo de inferência gerou
- Métricas de erro

Avaliação de Modelos

- Métricas para problemas de Classificação
  - Acurácia
  - F1 (Dice) score
  - Intersection Over Union (Jaccard)
  - Precision
  - Recall

Avaliação de Modelos

- Métricas para problemas de Regressão
  - Variância Explicada
  - R^2
  - Mean Squared Error (MSE)
  - PSNR (processamento de sinais/imagens)

Avaliação de Modelos

- Métricas para problemas de Clusterização
  - Homogeneidade
  - Informação Mútua
  - Completude

Protocolos de Avaliação

- Conjuntos de Dados
  - Conjunto de Treino
    - Usado para treinar os modelos
  - Conjunto de Teste
    - Usado para avaliar o modelo

Protocolos de Avaliação

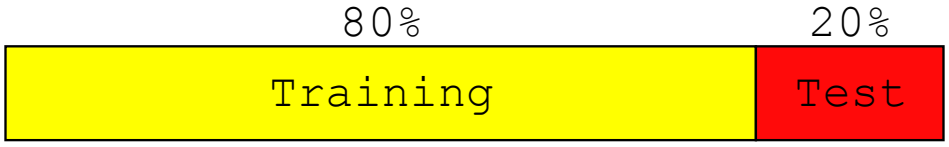


Figura: Divisão entre Treino e Teste.

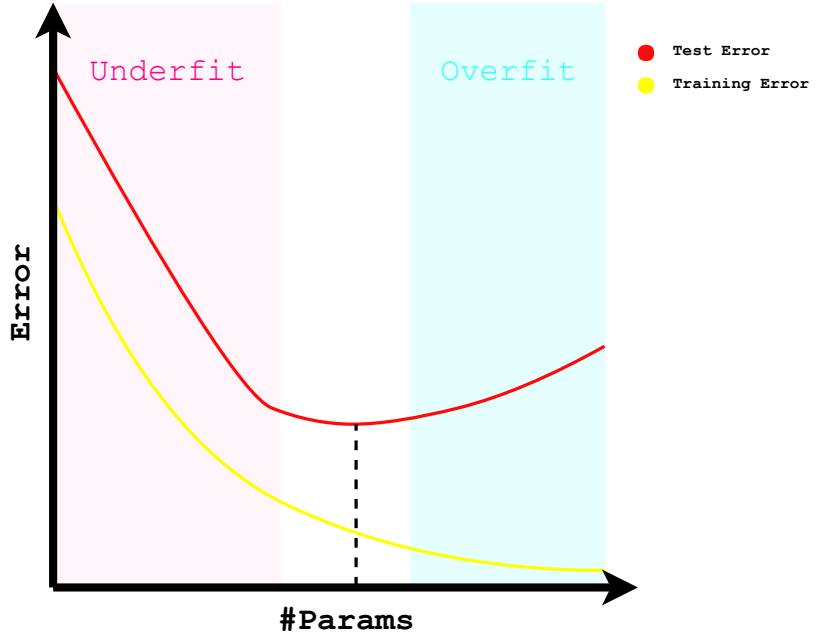


Figura: Erro vs. #Params.

Protocolos de Avaliação

- Conjuntos de Dados
  - Conjunto de Treino
    - Usado para treinar os modelos
  - Conjunto de Teste
    - Usado para avaliar os modelos
  - Conjunto de Validação
    - Usado para tunar os parâmetros dos modelos aos dados

Protocolos de Avaliação

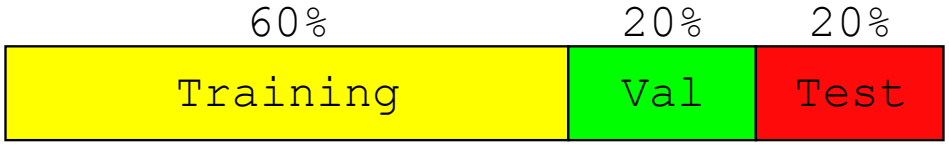


Figura: Divisão entre Treino, Validação e Teste.

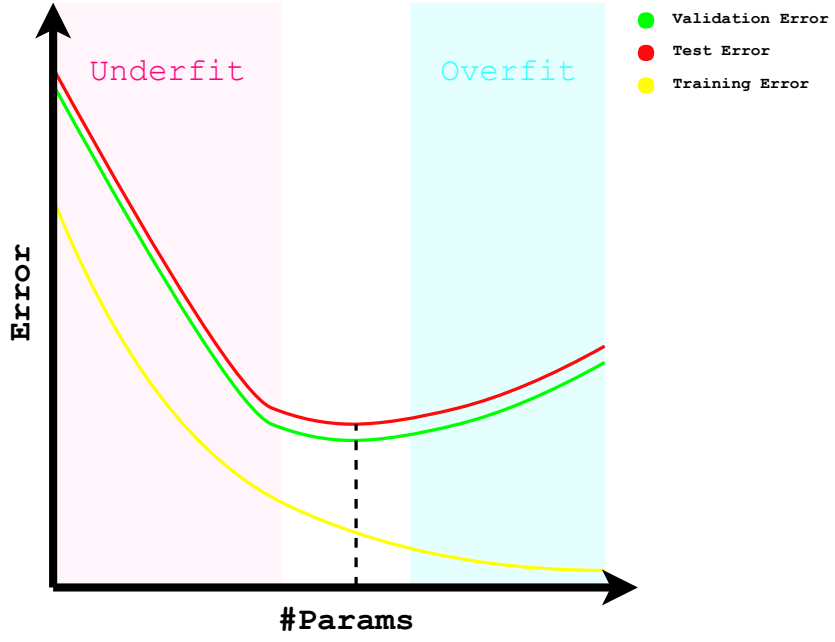


Figura: Erro vs. #Params.

Protocolo de Avaliação

Exercício - Full Protocol

Full_Protocol.ipynb

Agenda

MultiLayer Perceptron (MLP)

Demo - MultiLayer Perceptron (MLP)

MLP.ipynb

[1] NVIDIA.
Deep learning - nvidia developer, 2018.

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio.
Deep learning, volume 1.
MIT press Cambridge, 2016.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton.
Deep learning.
nature, 521(7553):436, 2015.