



Especialização

Big Data Analítica: Mineração de Banco de Dados

# Classificador Support Vector Machines ou Máquinas de Suporte Vetorial ou Máquinas de Vetores de Suporte (SVM)

Nádia Félix Felipe da Silva, Dra.

# Roteiro

- Introdução
  - O que é SVM?
  - Motivação
  - Exemplo simples
- SVM
  - Dados linearmente separáveis
  - Dados linearmente inseparáveis
- Exemplo

# Introdução

- Método supervisionado de aprendizado de máquina
- Apresenta resultados melhores que muitos métodos populares de classificação

# Introdução

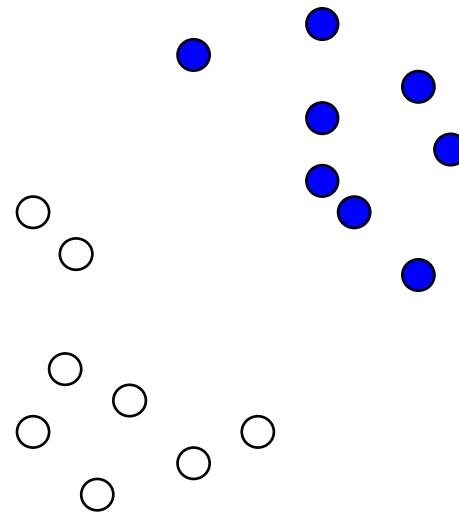
- 1968: base matemática
  - Teoria de Lagrange
- [Vapnik et al, 1992] Primeiro artigo
- [Vapnik et al, 1998] Definição detalhada
- Última década
  - Série de artigos com aplicações de SVM
  - Série de artigos com otimizações de SVM
    - SMO, por exemplo

# Introdução

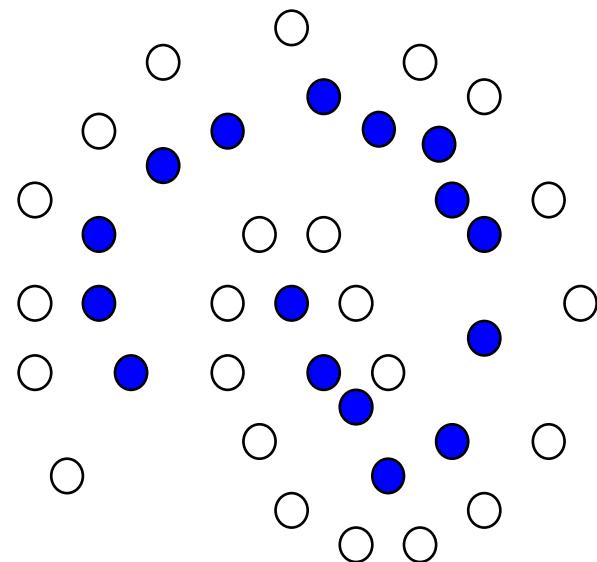
- SVM são utilizadas em diversas áreas:
  - Bioinformática
  - Reconhecimento de assinaturas
  - Classificação de texto e imagens
  - Identificação de spams
  - Reconhecimento de padrões diversos
  - Identificação de dados replicados

# Motivação da SVM

COMO SEPARAR AS DUAS CLASSES?

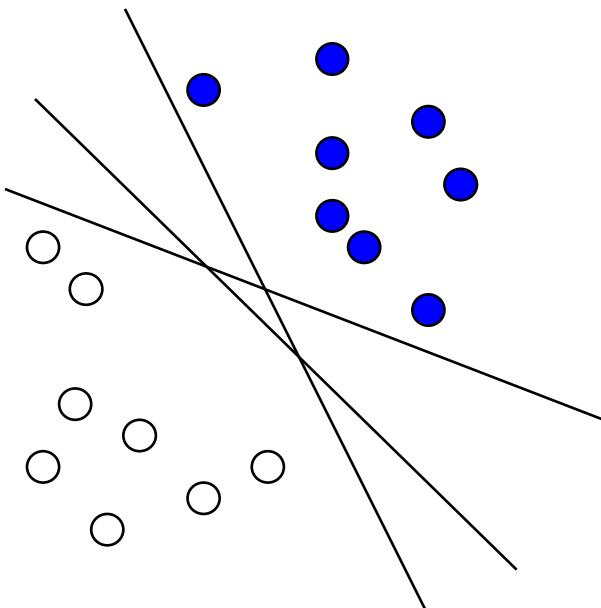


COMO SEPARAR AS DUAS CLASSES?

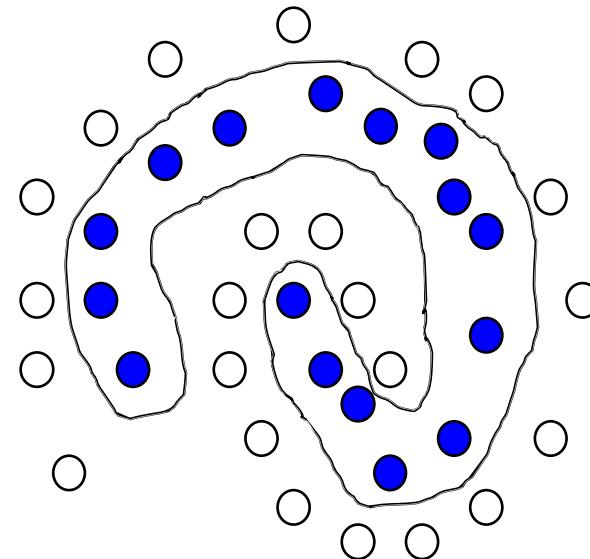


# Motivação da SVM

RETA / PLANO / HIPERPLANO



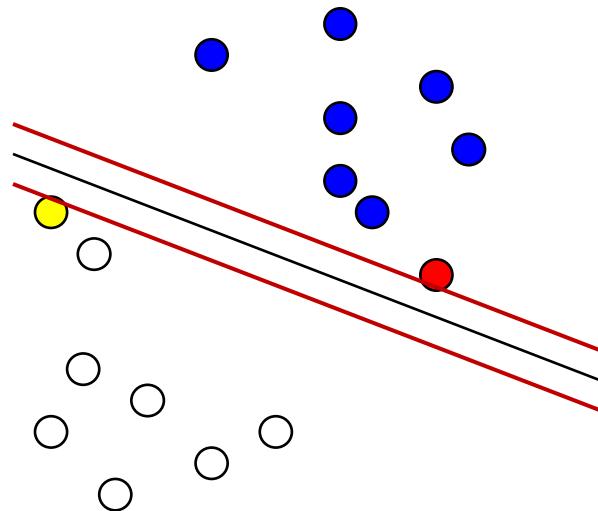
?



Qual o hiperplano ótimo?  
Menor erro de classificação

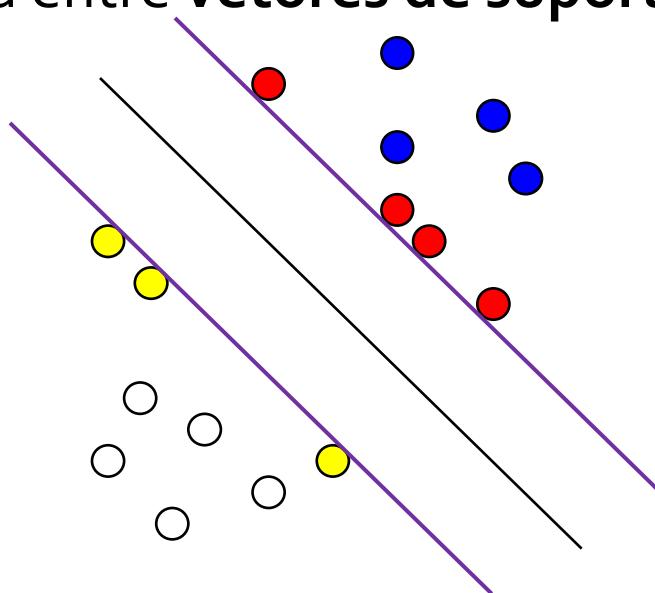
# Conceitos de SVM

- Qual o **hiperplano ótimo**?
  - Menor erro de classificação
  - Maior **margem**
    - Distância entre **vetores de suporte** e o hiperplano



# Conceitos de SVM

- Qual o **hiperplano** ótimo?
  - Menor erro de classificação
  - Maior **margem**
    - Distância entre **vetores de suporte** e o hiperplano



# Problema a resolver

- Qual o melhor “hiperplano separador” ?
    - Margem estreita: mínimas perturbações no hiperplano podem produzir um hiperplano não-separador.
    - Margem grande: a propriedade de “separação” de classes é preservada por pequenas perturbações no hiperplano.
  - Conclusão: *Melhor* hiperplano separador é o que apresenta a maior margem.
- 
- Encontrar o melhor hiperplano separador = Problema de *Otimização* com *restrições*

# Como funciona para dados linearmente separáveis?

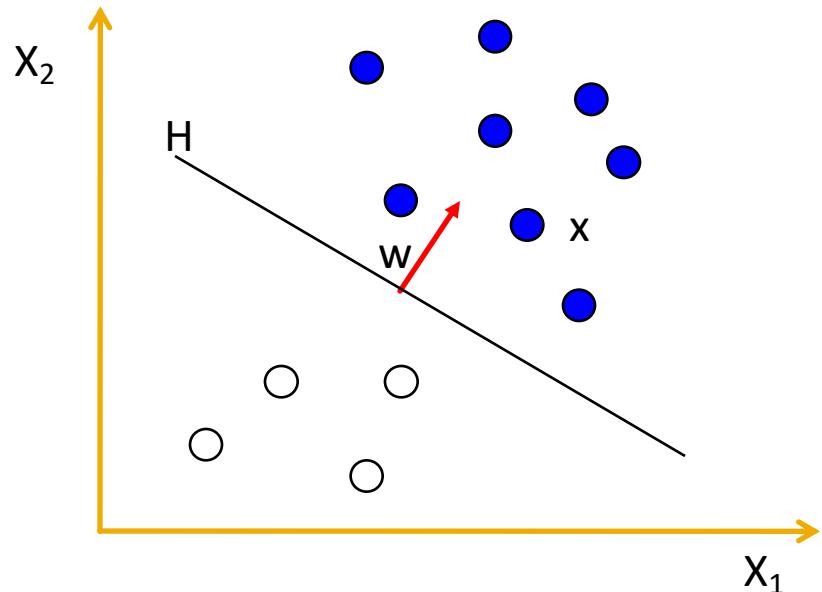
- Dados de treinamento
  - Tuplas no formato  $(X_1, X_2, \dots, X_n, Y)$ 
    - Atributos  $X_i$
    - Classe  $Y (+1, -1)$
- Conjunto dito linearmente separável, se existir um hiperplano  $H$  (no espaço de entrada) que separe as tuplas de classes diferentes
- Determinar os **vetores de suporte**
- Encontrar o **hiperplano ótimo**
  - Com maior **margem**

# O Hiperplano (H)

$$H: \bar{w} \cdot \bar{x} + b$$

$\bar{w} \cdot \bar{x} + b > 0 \rightarrow \text{classe} = \text{círculo preenchido}$

$\bar{w} \cdot \bar{x} + b < 0 \rightarrow \text{classe} = \text{círculo vazio}$



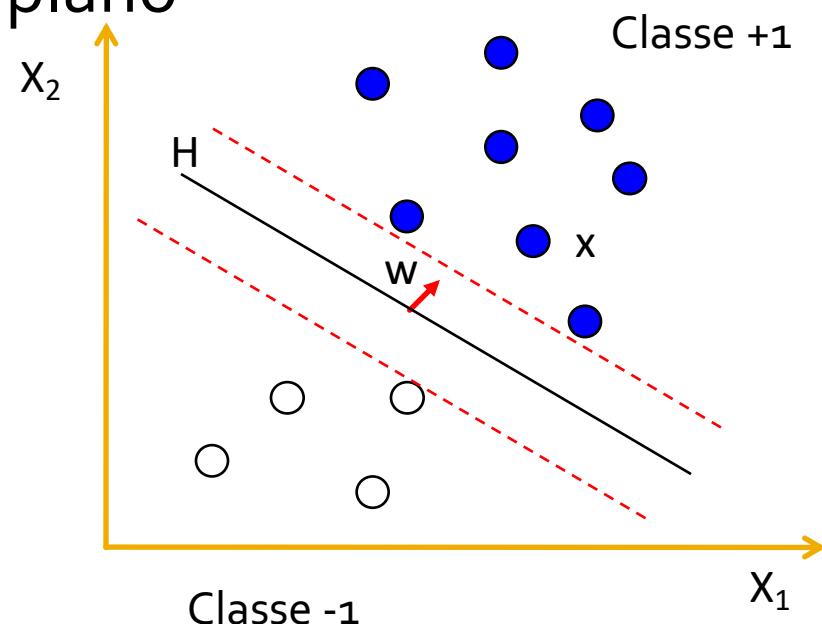
# O Hiperplano (H)

$$H: \bar{w} \cdot \bar{x} + b$$

$\bar{w} \cdot \bar{x} + b > 0 \rightarrow \text{classe} = \text{círculo preenchido}$

$\bar{w} \cdot \bar{x} + b < 0 \rightarrow \text{classe} = \text{círculo vazio}$

- $b$  = deslocamento do hiperplano a partir da origem
- $w$  = vetor ortogonal ao hiperplano

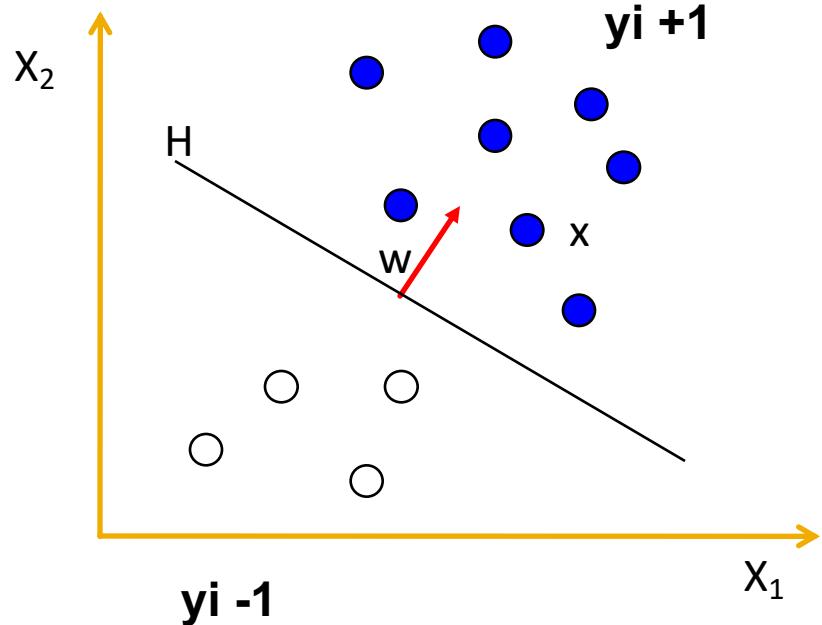


# O Hiperplano ( $H$ )

$$H: \bar{w} \cdot \bar{x} + b$$

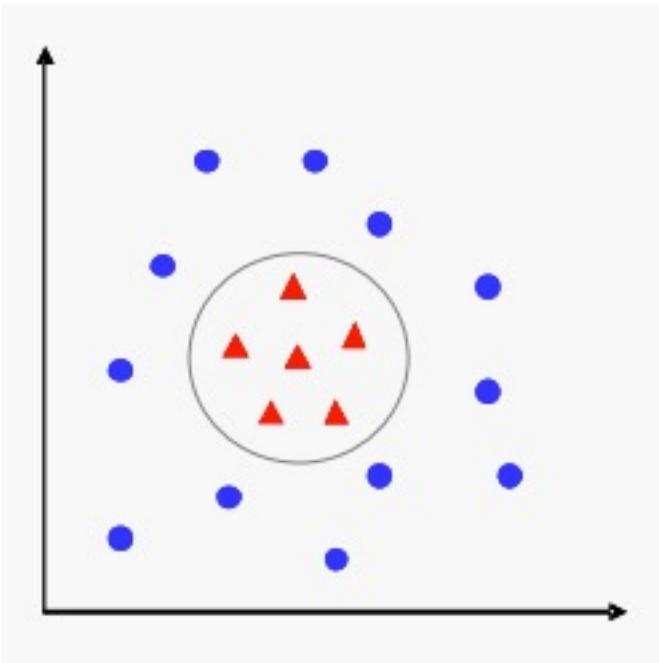
$$\bar{w} \cdot \bar{x} + b > 0 \rightarrow y_i = +1$$

$$\bar{w} \cdot \bar{x} + b < 0 \rightarrow y_i = -1$$



# SVM não linear

- Muitos conjuntos de dados são não lineares



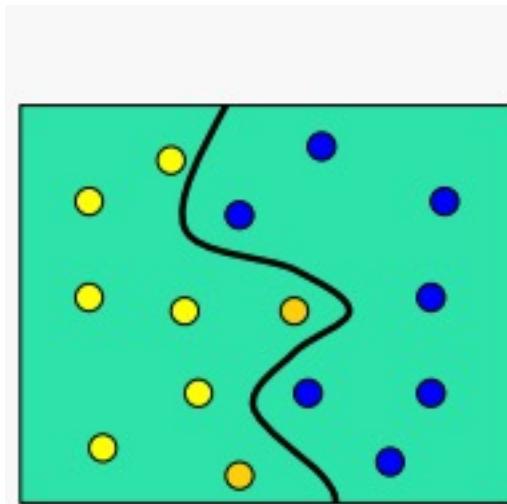
# SVM não linear

- Mapeamento dos dados para espaço de maior dimensão
  - Teorema de Cover

*Conjunto de dados não linearmente separáveis em um espaço pode ser transformado para outro espaço em que, com alta probabilidade, os objetos se tornam linearmente separáveis*

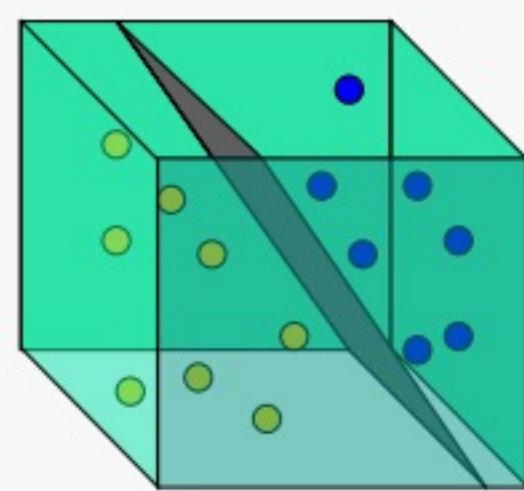
# SVM não linear

- Ideia: mapeia dados para espaço de maior dimensão e encontra o hiperplano separador: Função  $\Phi$



14/10/2022

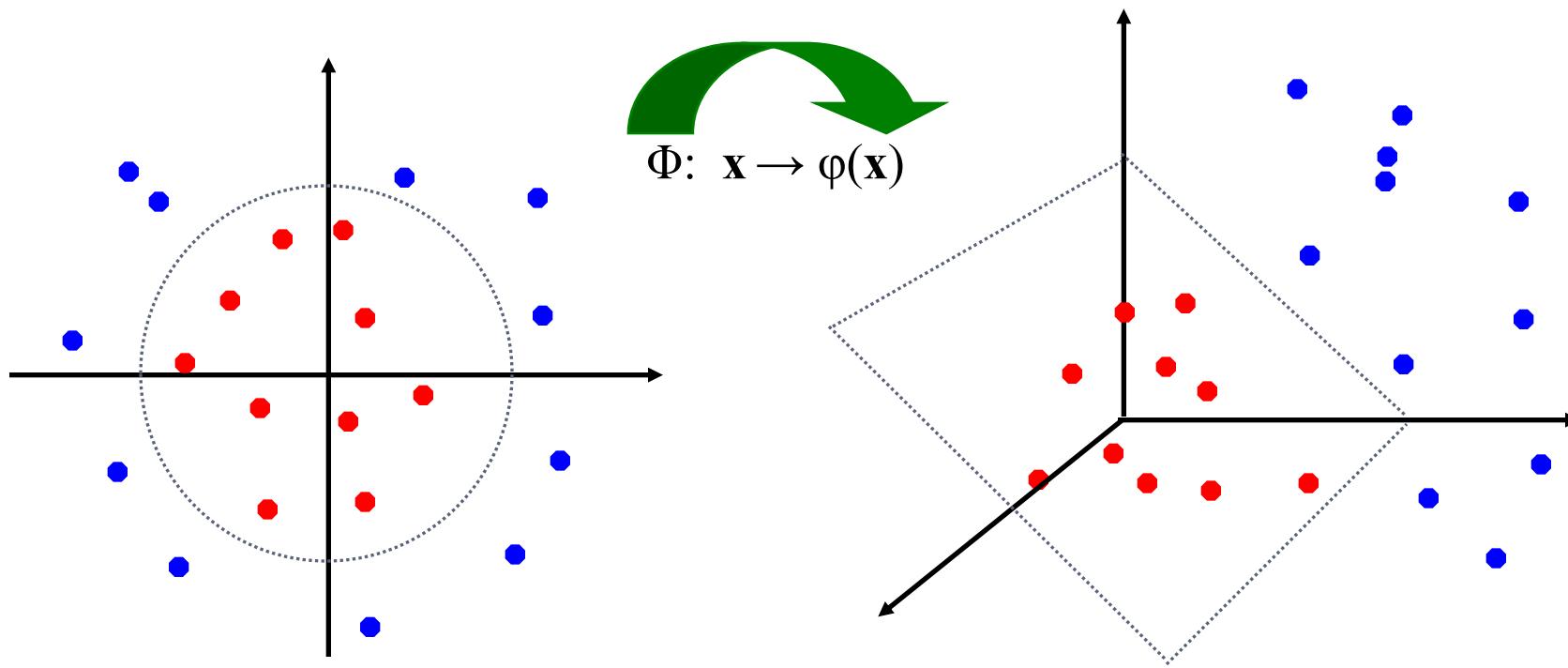
Espaço de entradas



Espaço de características

# SVM não linear

- Premissa: O espaço de atributos original sempre pode ser mapeado para algum espaço de características superior onde o dataset é separável



# SVM não linear

- A computação da função  $\Phi$  pode ser extremamente custosa e inviável dependendo do conjunto de atributos possíveis
- Para fins de simplicidade utiliza-se uma função kernel:
  - A função kernel faz um problema, que originalmente é não separável, separável.
  - Mapeia os dados em um espaço melhor representável.

# Kernels

- Linear
- Polinomial  $K(\mathbf{x}, \mathbf{y}) = (\mathbf{1} + \mathbf{x}^\top \mathbf{y})^d$
- RBF

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

- Sigmoidal

# Classificador SVM

- A obtenção de um classificador por meio do uso de SVMs envolve então a escolha de uma função kernel, além de parâmetros dessa função e da constante de regularização  $c$ .
- Segundo observado em Hsu et al. (2003), uma boa escolha inicial é empregar o kernel RBF e o kernel sigmoidal pode ter comportamento semelhante ao RBF.

- No Weka, temos....

Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

Open file...

Open URL...

Open DB...

Generate...

Undo

Edit...

Save...

## Filter

Choose

None

Apply

## Current relation

Relation: iris-weka.filters.unsupervised.attribute.Re...  
 Instances: 150 Attributes: 3

## Attributes

All

None

Invert

Pattern

No.

Name

1  petallength2  petalwidth3  class

Remove

## Selected attribute

Name: petallength

Type: Numeric

Missing: 0 (0%) Distinct: 43

Unique: 10 (7%)

## Statistic

## Value

Minimum

1

Maximum

6.9

Mean

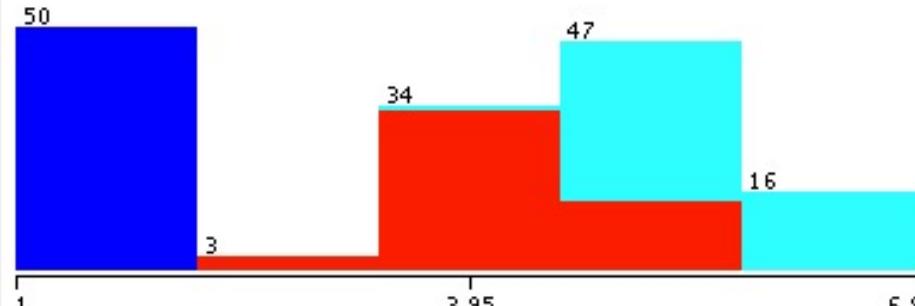
3.759

StdDev

1.764

Class: class (Nom)

Visualize All



## Status

OK

14/10/2022

Log



x 0

Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

**Classifier**

## ▼ functions

- 📄 GaussianProcesses
- 📄 IsotonicRegression
- 📄 LeastMedSq
- 📄 LibLINEAR
- 📄 LibSVM
- 📄 LinearRegression
- 📄 Logistic
- 📄 MultilayerPerceptron
- 📄 PaceRegression
- 📄 PLSClassifier
- 📄 RBFNetwork
- 📄 SimpleLinearRegression
- 📄 SimpleLogistic
- 📄 SMO
- 📄 SMOreg
- 📄 SPegasos
- 📄 VotedPerceptron
- 📄 Winnow

## ▶ lazy

## ▶ meta

**Filter...****Remove filter****Close****Classifier output****Status**

OK 14/10/2022

**Log**

x 0

Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

## Classifier

Choose

SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKern"

## Test options

 Use training set Supplied test set  Cross-validation Folds  Percentage split % (Nom) class 

## Result list (right-click for options)

## Classifier output



## Status

OK 14/10/2022

## Classifier

Choose

**SMO** -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKern

## Test options



weka.gui.GenericObjectEditor

- Use training set
- Supplied test set
- Cross-validation
- Percentage split

More

(Nom) class

Start

## Result list (right-click to copy)

## Status

OK

## About

Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

More

Capabilities

buildLogisticModels

False

c

1.0

checksTurnedOff

False

debug

False

epsilon

1.0E-12

filterType

Normalize training data

kernel

Choose

**PolyKernel** -C 250007 -E 1.0

numFolds

-1

randomSeed

1

toleranceParameter

0.001

Log



Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

## Classifier

Choose

SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKern

## Test options



weka.gui.GenericObjectEditor

- Use training set
- Supplied test set
- Cross-validation
- Percentage split

More

(Nom) class

Start

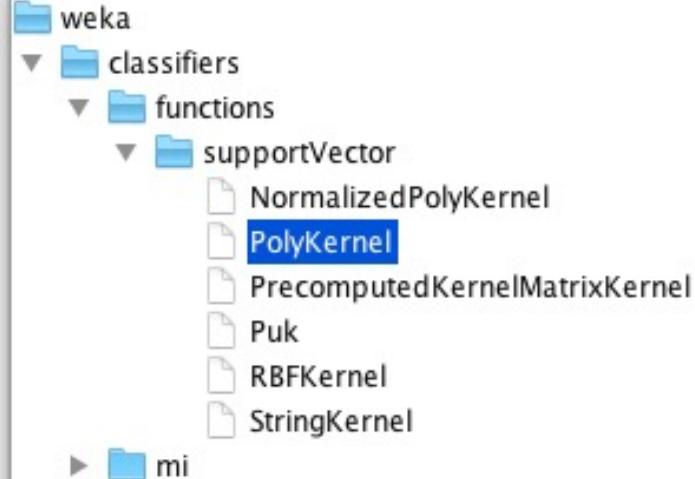
Result list (right-click to copy)

Status

OK

14/10/2022

buildLogisticModels



checksTurnedOff

debug

epsilon

filterType

kernel

numFolds

randomSeed

Log



Preprocess

Classify

Cluster

Associate

Select attributes

Visualize

## Classifier

Choose

`LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -seed 1`

## Test options

 Use training set Supplied test set

Set...

 Cross-validation

Folds

10

 Percentage split

%

66

More options...

## Classifier output

(Nom) class



Start

Stop

## Result list (right-click for options)

## Status

OK

14/10/2022

Log



# SVM em python



```
from sklearn import svm
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target

classifier = svm.SVC(kernel='linear', C=1,
                      random_state=0)

scores = cross_val_score(classifier, X, Y, cv=10)
```

# SVM em python

- ```
svm = SVC(kernel='rbf',  
random_state=0, gamma=1, C=1)
```
- ```
svm = SVC(kernel='poly',  
random_state=0, gamma=1, C=1)
```

# Grid Search

- Ajuste de parâmetros de algoritmos de aprendizado de máquina

```
1 # -*- coding: utf-8 -*-
2
3 from sklearn import datasets
4 from sklearn.model_selection import train_test_split
5 from sklearn.model_selection import GridSearchCV
6 from sklearn.metrics import classification_report
7 from sklearn.svm import SVC
8
9 # Loading the Digits dataset
10 digits = datasets.load_digits()
11
12 # To apply an classifier on this data, we need to flatten the image, to
13 # turn the data in a (samples, feature) matrix:
14 n_samples = len(digits.images)
15 X = digits.images.reshape((n_samples, -1))
16 y = digits.target
17
18 # Split the dataset in two equal parts
19 X_train, X_test, y_train, y_test = train_test_split(
20     X, y, test_size=0.5, random_state=0)
21
22
23 #Set the parameters by cross-validation
24 tuned_parameters = [{ 'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
25                      'C': [1, 10, 100, 1000]}, 
26                      { 'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
27
28
29 scores = ['precision', 'recall']
```

# Grid Search

```
50
51 for score in scores:
52     print("# Tuning hyper-parameters for %s" % score)
53     print()
54
55     clf = GridSearchCV(SVC(), tuned_parameters, cv=5,
56                         scoring='%s_macro' % score)
57     clf.fit(X_train, y_train)
58
59     print("Best parameters set found on training set:")
60     print()
61     print(clf.best_params_)
62     print()
63     print("Grid scores on training set:")
64     print()
65     means = clf.cv_results_['mean_test_score']
66     stds = clf.cv_results_['std_test_score']
67     for mean, std, params in zip(means, stds, clf.cv_results_['params']):
68         print("%0.3f (+/-%0.03f) for %r"
69               % (mean, std * 2, params))
70     print()
71
72     print("Detailed classification report:")
73     print()
74     print("The model is trained on the full training set.")
75     print("The scores are computed on the full evaluation set.")
76     print()
77     y_true, y_pred = y_test, clf.predict(X_test)
78     print(classification_report(y_true, y_pred))
79     print()
80
```