**MACHINE LEARNING**

Mr.[Diego Pedro Pinto Roa](mailto:dpinto@pol.una.py)

Support Vector Machines

**\_\_\_**

By

* [Ivan Daniel Weiss Van Der Pol](mailto:ivanweisslml@fpuna.edu.py)
* [Emilio Ginzo Benitez](mailto:emilioginzo@fpuna.edu.py)
* [Cesar Augusto Rodas Espinola](mailto:cesar_rodas@fpuna.edu.py)



# 

[**Support Vector Machines for binary classification**](#_b5znwjn64i3x)

[Introduction](#_56h9h4ugiyft)

[Clasificación con una línea](#_ydc8jjp9mj4)

[Cuando los Grupos no se Pueden Separar con una Línea](#_o65k7w4qlx5d)

[Clasificación en Múltiples Categorías](#_sn94ntdekbhs)

[Aprendiendo con Ruido y Márgenes Flexibles](#_br15xk2jnap)

[Multi-class classification](#_euaxavj7dkmv)

[Learning with noise: soft margins](#_2jerncyg9ifm)

[Algorithmic implementation of Support Vector Machines](#_qzeeg2uo8hef)

[Case Study 1: training a Support Vector Machine](#_qa4kkozc72i8)

[Case Study 2: predicting disease progression](#_8p8gl3arrwxp)

[Case Study 3: drug discovery through active learning](#_3bruumg96d3z)

[**Kernel-based Models**](#_f3esucjsnq7b)

[Introduction](#_5ks9q8fcnjga)

[Other kernel-based learning machines](#_bjdeoverdi1z)

[Introducing a confidence measure](#_2jr15txvl471)

[One class classification](#_98p6pavd44o5)

[Regression: learning with real-valued labels](#_bohshq6q3n1z)

[Structured output learning](#_os8qqki4trld)

[**Learning with Kernels**](#_4p9a57w81pks)

[Introduction](#_eah89w9bw30d)

[Properties of kernels](#_rjv4vckik2xz)

[Simple kernels](#_uzr8xdxnb0tm)

[Kernels for strings and sequences](#_of6k9j23079a)

[Kernels for graphs](#_mu087l14kok4)

[Multiple kernel learning](#_gdeedsbz202h)

[Learning kernel combinations via a maximum margin approach](#_avdcdg481r7b)

[Algorithmic approaches to multiple kernel learning](#_ih8py7fz39qx)

[Case Study 4: protein fold prediction](#_e89pgaojnsk6)

[**Videography**](#_8b77ylm9lg5e)

# 

# Support Vector Machines for binary classification

### Introduction

Imagina que tenemos un juego donde clasificamos cosas en dos grupos. Por ejemplo, podemos clasificar si un animal es un perro o un gato. Una máquina de vectores de soporte nos ayuda a hacer esto de una manera especial.

### Clasificación con una línea

Pensemos en una línea que separa a los perros de los gatos en una imagen. La línea está dibujada de manera especial para estar justo en el medio, lo más lejos posible de los animales.

#### Puntos Especiales

Algunos animales están muy cerca de la línea. Estos son como los superhéroes de nuestro juego. La línea se ajusta para   
estar más cerca de estos animales especiales.

#### Fórmula y Elementos

La fórmula para la línea que usamos en la máquina de vectores de soporte es:

w⋅x+b=0

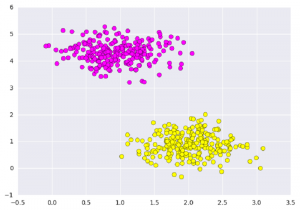
**w** son los "pesos" que nos dicen la dirección de la línea.

**x** son los datos de entrada que estamos clasificando.

B es un número especial llamado "sesgo" o "desplazamiento" que nos dice dónde está la línea en el espacio.

### 

#### Ejemplo



**from** sklearn.datasets **import** make\_blobs

# creating datasets X containing n\_samples

# Y containing two classes

X, Y **=** make\_blobs(n\_samples**=**500, centers**=**2,

random\_state**=**0, cluster\_std**=**0.40)

**import** matplotlib.pyplot as plt

# plotting scatters

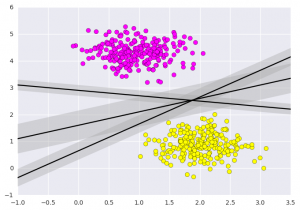
plt.scatter(X[:, 0], X[:, 1], c**=**Y, s**=**50, cmap**=**'spring');

plt.show()

# creating linspace between -1 to 3.5

xfit **=** np.linspace(**-**1, 3.5)

# plotting scatter

plt.scatter(X[:, 0], X[:, 1], c**=**Y, s**=**50, cmap**=**'spring')

# plot a line between the different sets of data

**for** m, b, d **in** [(1.0, 0.65, 0.33),  
 (0.5, 1.60, 0.55),  
 (**-**0.2, 2.9, 0.2)]:

yfit **=** m **\*** xfit **+** b

plt.plot(xfit, yfit, '-k')

plt.fill\_between(xfit, yfit **-** d, yfit **+** d, edgecolor**=**'none',

color**=**'#AAAAAA', alpha**=**0.4)

plt.xlim(**-**1, 3.5);

plt.show()

### 

### Cuando los Grupos no se Pueden Separar con una Línea

Imagina que en nuestro juego, los perros y los gatos están muy mezclados y no podemos dibujar una línea para separarlos. Es como si los juguetes estuvieran todos revueltos y no podemos dividirlos con una línea.

Para resolver este problema, usamos algo especial llamado "truco del kernel". Es como encontrar un camino secreto en el juego que nos ayuda a separar a los perros y a los gatos de una manera diferente.

### 

#### ¿Qué es el Truco del Kernel?

El truco del kernel es una idea mágica que nos permite separar a los perros y a los gatos de una manera más creativa. En lugar de una línea, imaginamos formas curvas o incluso dibujos mágicos que nos ayudan a dividir a los animales.

#### Cómo Encontramos Estas Curvas?

Pensamos en Nuevas Dimensiones: En lugar de solo mirar hacia adelante y atrás, imaginamos otras direcciones especiales. Estas direcciones nos ayudan a ver el juego desde ángulos diferentes.

| Tenemos los datos en base a 2 dimensiones (x,y) | Agmentamos nuestros datos con propiedades no lineales para tener mas dimensiones ( |
| --- | --- |
|  |  |

| Encontramos la linea magica (hyperplano) en esta dimension nueva | Proyectar a el plano original |
| --- | --- |
|  |  |

## 

### Clasificación en Múltiples Categorías

Cuando tenemos más de dos grupos para clasificar, necesitamos usar un enfoque especial llamado "clasificación uno contra todos" para ayudarnos.

#### Categorías Especiales

Perros (Categoría 1): triangulo

Gatos (Categoría 2): cuadrado

Pájaros (Categoría 3): cruz

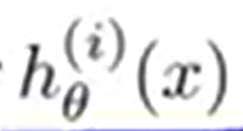
#### 1 VS TODOS

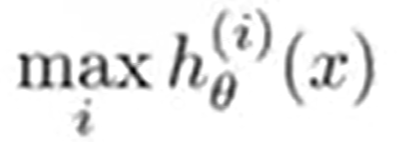
##### 

**Haciendo Equipos de Clasificación**: Imagina que nuestros perros y gatos ahora tienen amigos, como pájaros. Tenemos que clasificar a todos estos amigos en grupos.

**Calculando Equipos Especiales**: Creamos un equipo para cada amigo, por ejemplo, un equipo de perros, otro de gatos y uno para pájaros. Para cada equipo, dibujamos una línea o una curva especial para separar a ese amigo de los demás.

**Prediciendo Amigos**: Cuando llega un nuevo amigo a nuestro juego, vemos en qué equipo encaja mejor. La línea o curva que está más cerca de nuestro amigo nos dice a qué equipo pertenece.

En este metodo se tienen 2+ grupos para clasificar esto es imposible con solo 1 hiperplano  
Por lo tanto lo que hacemos es calcular un hiperplano para cada grupo (i)  
Para predecir la probabilidad de x pertenezca a la clase (i)  
Y luego cuando un nuevo dato X se procesa se determina a cual grupo (i) pertenece de a cuerdo a cual cual maximiza su pertenencia



| **TABLA ORIGINAL** | **Se encuentra el hiperplano para perros** | **Se encuentra el hiperplano para gatos** | **Se encuentra el hiperplano para pajaros** |
| --- | --- | --- | --- |
|  |  |  |  |

#### 1 VS 1

**Amigos en Competencia**: Imagina que tenemos dos amigos en una competencia especial, por ejemplo, un perro y un gato.

**Dibujando una Línea entre Amigos**: Dibujamos una línea imaginaria entre el perro y el gato. Esta línea nos ayuda a separarlos visualmente.

**Decidiendo Quién Gana**: Cuando llega un nuevo amigo, lo comparamos con el perro y el gato. Si está más cerca del perro en la línea, decimos que es más parecido al perro. Si está más cerca del gato, decimos que es más parecido al gato.

**Muchas Competencias**: Repetimos este proceso de competencias entre todos nuestros amigos para saber quién es más parecido a quién.

### Aprendiendo con Ruido y Márgenes Flexibles

A veces, en nuestro juego, los datos están un poco desordenados o "ruidosos". Es como cuando hay un poco de confusión entre nuestros amigos. Para ayudarnos, usamos algo especial llamado "márgenes flexibles".

**¿Qué Son los Márgenes Flexibles?**

Imagina que nuestros amigos (los animales) están jugando en un parque. Queremos trazar líneas especiales para separarlos, pero a veces algunos amigos están cerca de la línea, ¡casi en la línea de otro equipo!

Los márgenes flexibles nos permiten tener un poco de espacio entre la línea y nuestros amigos. Así, si un amigo está cerca de la línea, no estamos muy seguros de su equipo, ¡le damos un poco de espacio!

**Ejemplo con Perros y Gatos**

Supongamos que un perro y un gato están jugando cerca de la línea de separación. Con los márgenes flexibles, decimos: "Está bien, si estás cerca de la línea, ¡aún eres parte de tu equipo!". Esto nos ayuda a evitar errores cuando algunos amigos están muy cerca de la línea.

## 

# Videography

[Support Vector Machine (SVM) in 2 minutes](https://www.youtube.com/watch?v=_YPScrckx28&pp=ygUXU3VwcG9ydCBWZWN0b3IgTWFjaGluZXM%3D&ab_channel=VisuallyExplained)

[The Kernel Trick in Support Vector Machine (SVM)](https://www.youtube.com/watch?v=Q7vT0--5VII&ab_channel=VisuallyExplained)