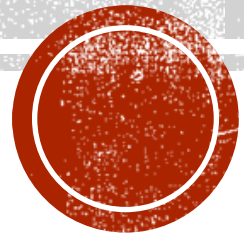
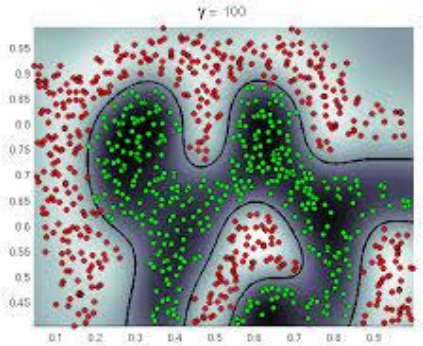
# APRENDIZAJE SUPERVISADO

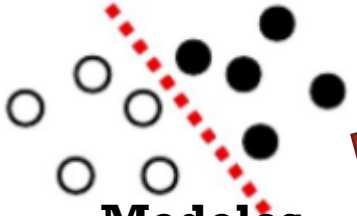


Javier Diaz, PhD
UNIVERSIDAD ICESI

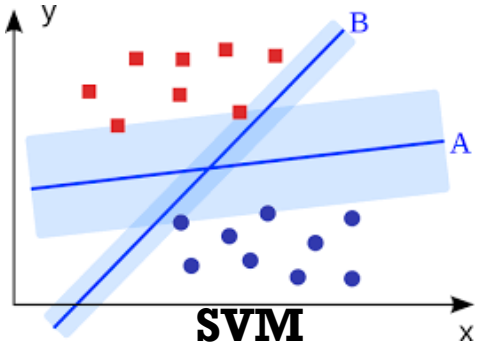
# AGENDA



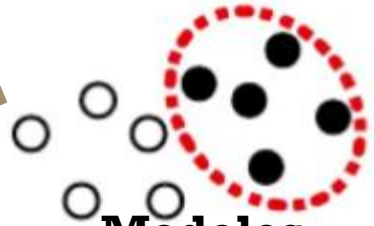
**Aprendizaje Supervisado**



**Modelos discriminantes**

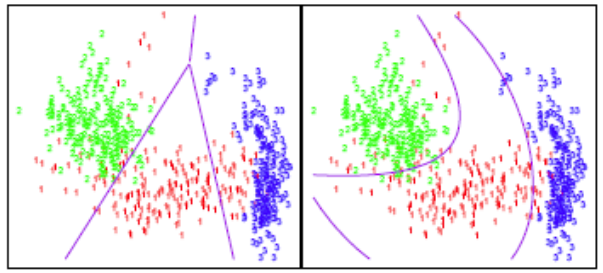



**SVM**

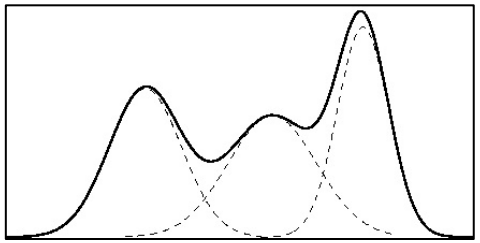


**Modelos generativos**

**GDA**



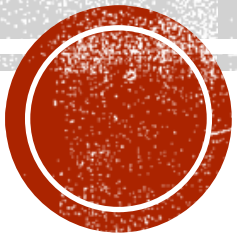
**LDA** **QDA**



**GMM**

Javier Diaz, PhD
UNIVERSIDAD ICESI

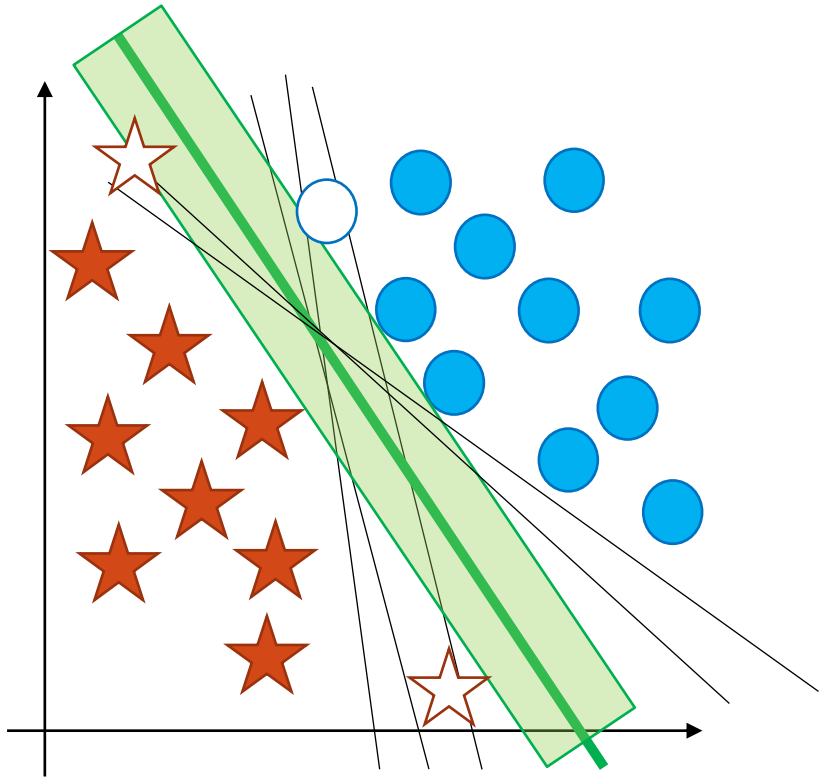
# SVM



Javier Diaz, PhD

UNIVERSIDAD ICESI

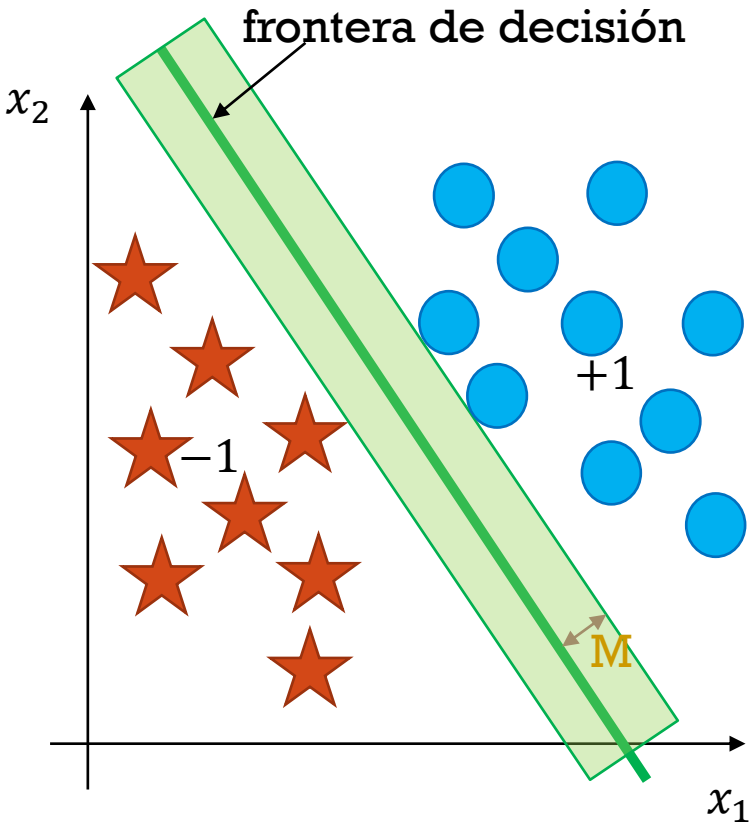
# CLASIFICADOR DE MARGEN MÁXIMO



- Muchos clasificadores de hiperplano pueden llegar a separar linealmente 2 clases.
- Los SVMs buscan una frontera de decisión con el **mayor margen posible** (a través de un objetivo de optimización). Entre más lejos esta un punto correctamente clasificado de la frontera, mejor.
- Los SVMs son una extensión de un **clasificador de margen máximo**, un discriminador linear binario.

Javier Diaz, PhD
UNIVERSIDAD ICESI

# CLASIFICADOR DE MARGEN MÁXIMO




- Hay dos categorías: la azul, a la que le asignamos un valor positivo de +1, y la roja, con un valor negativo de -1
- La **frontera de decisión** se define a partir de una ecuación: $\boldsymbol{\omega}^T \boldsymbol{x} + b = 0$, con $\boldsymbol{\omega}$ siendo el vector de parámetros de la línea, perpendicular a la misma.
- Las ecuaciones de los límites del margen $M$ son:

$$\boldsymbol{\omega}^T \boldsymbol{x} + b = -M \qquad \text{y} \qquad \boldsymbol{\omega}^T \boldsymbol{x} + b = +M$$

dado $i$, con clase $\boldsymbol{y}^{(i)} \in \{-1, +1\}$, tenemos:

$$y^{(i)}\left(\boldsymbol{\omega}^T \boldsymbol{x}^{(i)} + b\right) \geq M$$

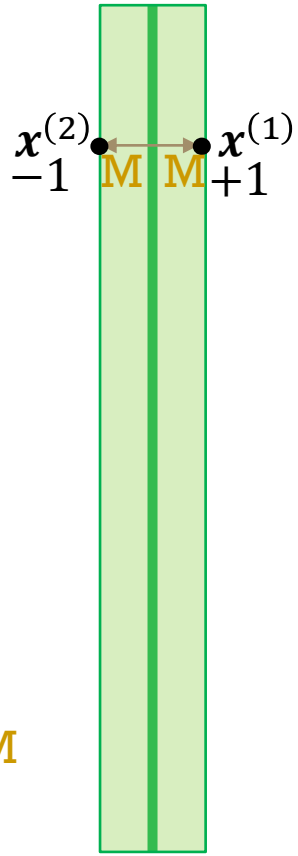
- Se debe buscar entonces es los parámetros $\boldsymbol{\omega}$ y $b$ que hagan que la frontera de decisión esté lo más lejos posible de los datos, para que el **margen** $\boldsymbol{M}$ sea lo mayor posible (un "cojín de tranquilidad").

# CLASIFICADOR DE MARGEN MÁXIMO

Vectores

- Vamos a escoger dos puntos ($x^{(1)}$ y $x^{(2)}$), uno en cada límite del margen ($y^{(1)} = +1$, $y^{(2)} = -1$), cuyo vector sea perpendicular a la frontera, y sustraemos sus dos ecuaciones:

$$\omega^T x^{(1)} + b = +1$$
$$\omega^T x^{(2)} + b = -1$$
$$\omega^T (x^{(1)} - x^{(2)}) = 2$$




- Queremos encontrar el vector $(x^{(1)} - x^{(2)})$, pero no podemos dividir por un vector ($\omega$), vamos a dividir por su largo, normalizándolo:

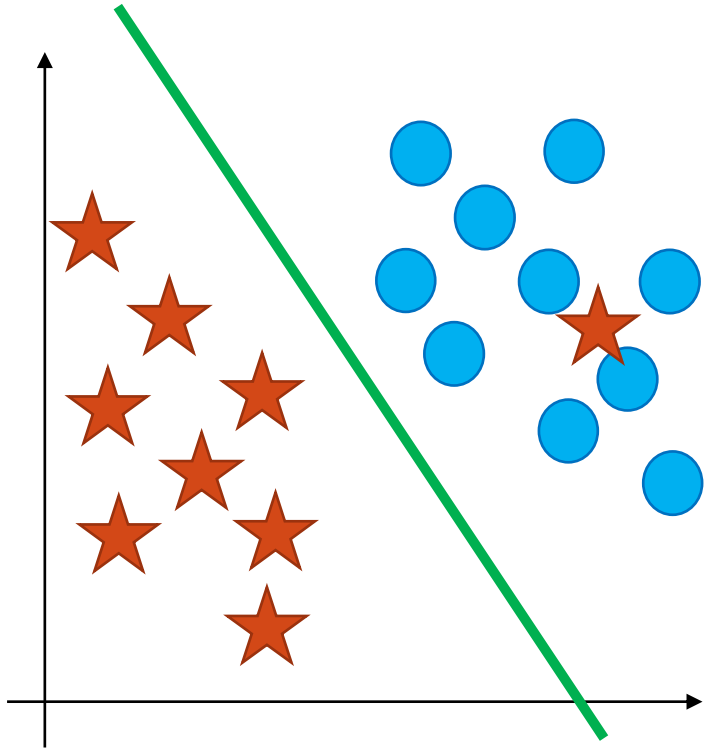
Ancho del margen

$$\frac{\omega^T}{\|\omega\|}(x^{(1)} - x^{(2)}) = \frac{2}{\|\omega\|}$$

- En el lado izquierdo tenemos la proyección de $(x^{(1)} - x^{(2)})$ sobre el vector de parámetros normalizado. Como son paralelos, y uno es un vector unitario (suma del cuadrado de sus coeficientes es igual a 1), equivale al largo de la diferencia de los puntos en la dirección del vector de pesos, es decir, el ancho del margen, que vamos a maximizar:
  - Necesitamos una maximización de $2/\|\omega\|$, equivalente a una minimización de $\frac{1}{2}\|\omega\|^2$
  - Sujeto a la clasificación de cada instancia $i$, mas allá del margen dada por: $y^{(i)}(\omega^T x^{(i)} + b) \geq 1$

M

# CLASIFICADOR DE MARGEN MÁXIMO



- Problema de optimización:
  - Minimizar de $\frac{1}{2}\|\boldsymbol{\omega}\|^2$, sujeto a: $y^{(i)}\left(\boldsymbol{\omega}^T \boldsymbol{x}^{(i)} + b\right) \geq 1$
- **Programación cuadrática**: equivale a maximizar la función $Q(\alpha)$:

  $Q(\alpha) = \sum_i \alpha^{(i)} - \frac{1}{2}\sum_{i,j} \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)T}\boldsymbol{x}^{(j)}$,

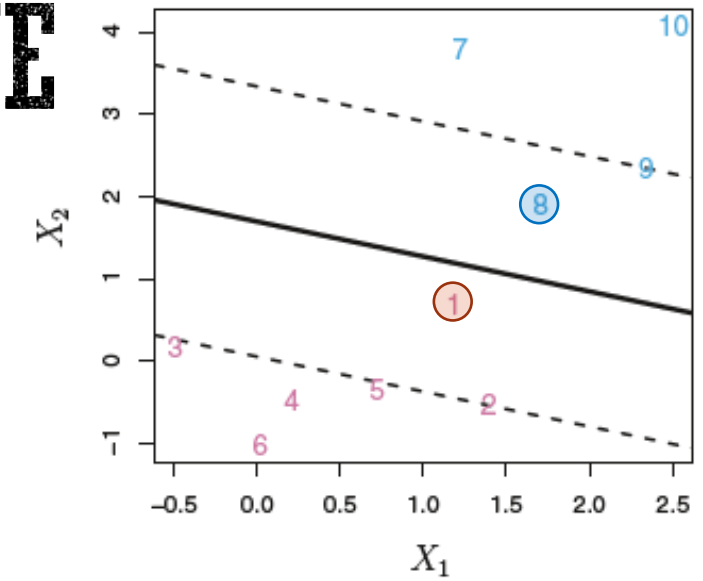
  donde $\alpha^{(i)}, \alpha^{(j)} \geq 0$ y $\sum_i \alpha^{(i)} y^{(i)} = 0$,

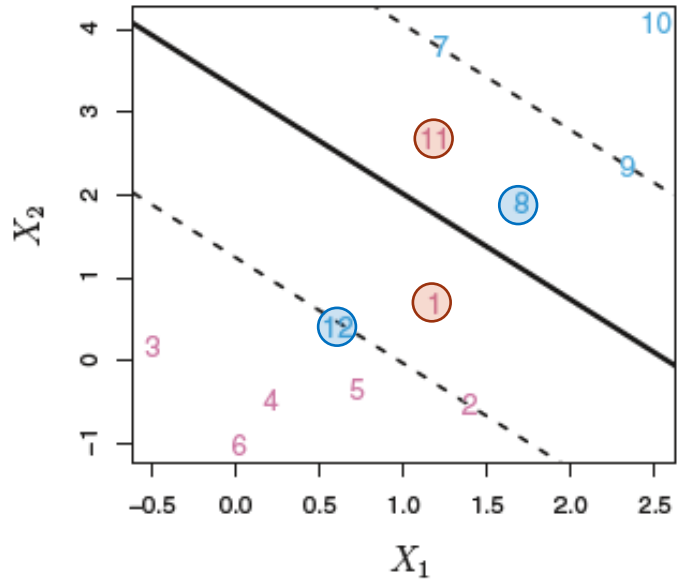
  lo que nos permite encontrar, $\boldsymbol{\omega} = \sum_i \alpha^{(i)} y^{(i)} \boldsymbol{x}^{(i)}$, y de ahí, $b$
- **No lo vamos a hacer!**
  - Nos interesa $\boldsymbol{x}_i^{(i)T}\boldsymbol{x}^{(j)}$, un producto entre los vectores de dos instancias, que se puede interpretar como una **medida de similitud** entre ellas.
  - La mayoría de los $\alpha^{(i)}$ son iguales a cero → solo algunos de los vectores $\boldsymbol{x}_i^{(i)}$ son importantes y van a dar el **soporte** necesario para tener una **máquina** que obtiene los parámetros, **¿Cuáles creen ustedes que son estos?**
  - **El problema puede no ser linealmente separable**

    **→ clasificadores de margen suave**

# CLASIFICADOR DE MARGEN SUAVE

- **Separabilidad lineal**: requisito para el clasificador de margen máximo, que exista un hiperplano separador. De lo contrario, el margen será negativo, y no existe un clasificador de margen máximo.
- **IDEA**: Seguir teniendo una frontera de decisión lineal, pero permitiendo sobrepasarla en una cierta medida, clasificando erróneamente algunas instancias de entrenamiento que violan el margen, mejorando la clasificación de las demás instancias.
- Extender el **clasificador de margen máximo**, usando un **margen suave**
- La idea es también prevenir que el clasificador sea **muy sensible** a un solo punto que pueda causar cambios dramáticos al hiperplano separador (→ **overfitting**).
- A los **vectores de soporte** del margen, se añaden los de los puntos que violan el margen o se encuentran mal clasificados (los errores)




ISLR, 2015

Javier Diaz, PhD
UNIVERSIDAD ICESI

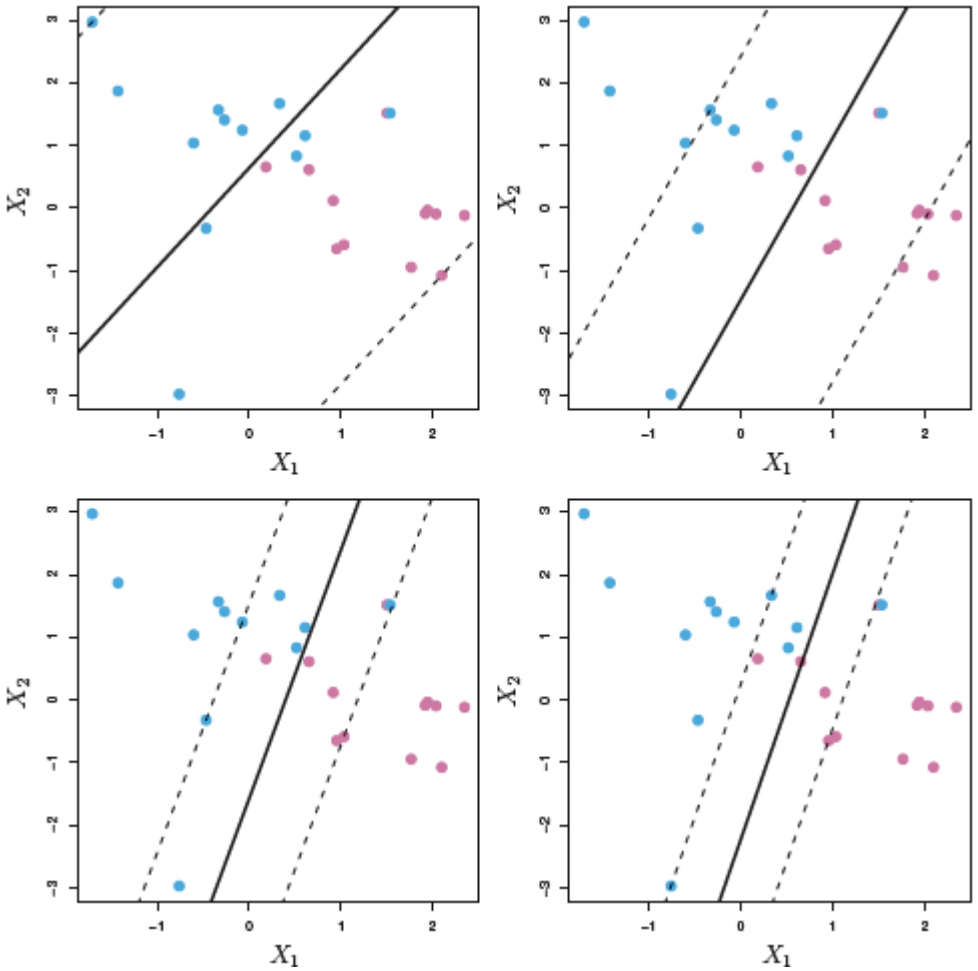
# CLASIFICADOR DE MARGEN SUAVE

- Se cambia la restricción de optimización a:

$$y^{(i)}\left(\boldsymbol{\omega}^T \boldsymbol{x}^{(i)} + b\right) \geq M * (\mathbf{1} - \epsilon^{(i)}),$$
$$\text{con } \epsilon^{(i)} \geq 0, \quad \text{y} \quad \sum_{i=1}^{n} \epsilon^{(i)} \leq C$$

Donde

- $\epsilon_i$ : **errores** (distancia con el límite correcto del margen), 0 si se respeta el margen, >1 si se sobrepasa la frontera de decisión
- $C$: **total de error aceptado** (magnitud acumulada de violación del margen). Controla el **overfitting**. Se utiliza cross-validation para especificar el valor de $C$.
- Con $C = 0$, se tiene un clasificador de margen máximo. Entre mas grande es $C$, mas grande el margen

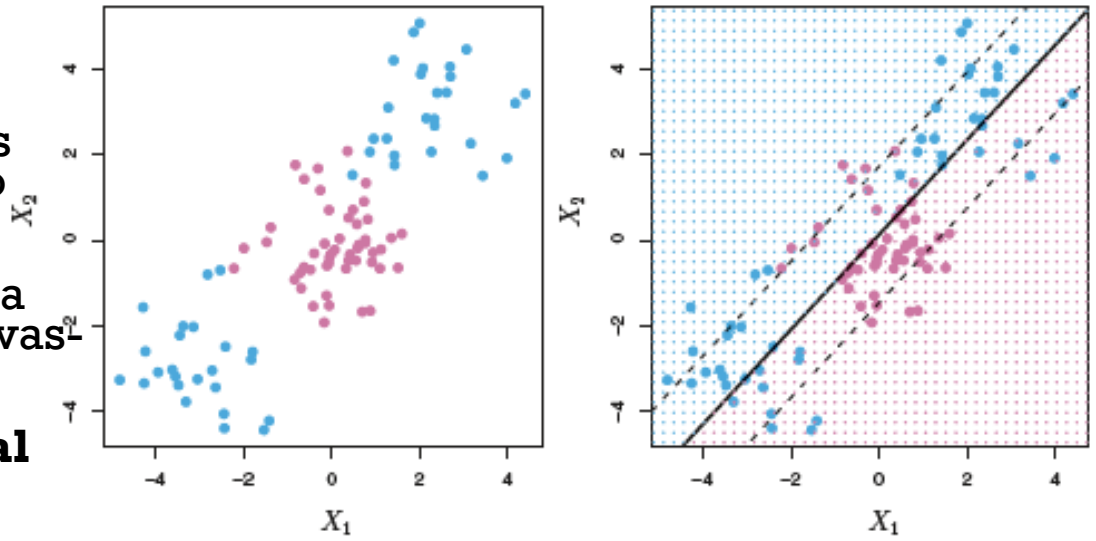



ISLR, 2015

Javier Diaz, PhD
UNIVERSIDAD ICESI

# SVM - CLASIFICACIÓN

- **Fronteras de separación no lineales:** Se podrían crear nuevas variables predictivas, modificando las originales, utilizando polinomios de mayor orden o términos de interacción
- Las **máquinas de vectores de soporte** aumentan la dimensionalidad del espacio de variables predictivas
- Puede necesitarse un **espacio de infinitas dimensiones**, para llegar a la **separabilidad lineal**




ISLR, 2015

→ ¿Cómo hago para que no vuelva **inmanejable** computacionalmente?

- Se aplica una función $\phi$ de transformación de aumento dimensional vectorial, por ejemplo,

$$\phi\left(\left[x_1^{(i)}, x_2^{(i)}\right]\right) = \left[x_1^{(i)2}, x_2^{(i)2}, \sqrt{2}x_1^{(i)}x_2^{(i)}\right]$$

- No estamos agregando información nueva, ni perdiendo información

→ ¿Cómo sé que tipo de transformación de aumento dimensional puedo utilizar?

Javier Diaz, PhD
UNIVERSIDAD ICESI

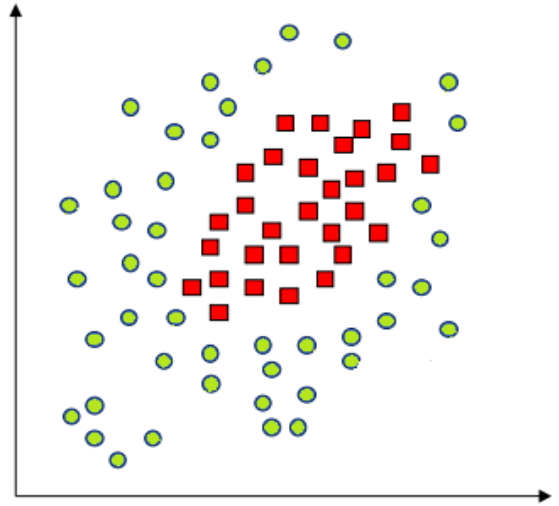
# SVM - CLASIFICACIÓN

- Calculemos el producto punto de dos puntos $x^{(1)}$ y $x^{(2)}$ en la nueva representación 3D del ejemplo dada por el, retomando el $\phi$ de ejemplo :

$$\phi\left(x^{(1)}\right)^T \cdot \phi\left(x^{(2)}\right) = \left[x_1^{(1)2}, x_2^{(1)2}, \sqrt{2}x_1^{(1)}x_2^{(1)}\right]^T \cdot \left[x_1^{(2)2}, x_2^{(2)2}, \sqrt{2}x_1^{(2)}x_2^{(2)}\right]$$
$$= x_1^{(1)2}x_1^{(2)2} + 2x_1^{(1)}x_1^{(2)}x_2^{(1)}x_2^{(2)} + x_2^{(1)2}x_2^{(2)2} = \left(x_1^{(1)}x_1^{(2)} + x_2^{(1)}x_2^{(2)}\right)^2$$
$$= \boxed{(x^{(1)T} \cdot x^{(2)})^2}$$

Función kernel

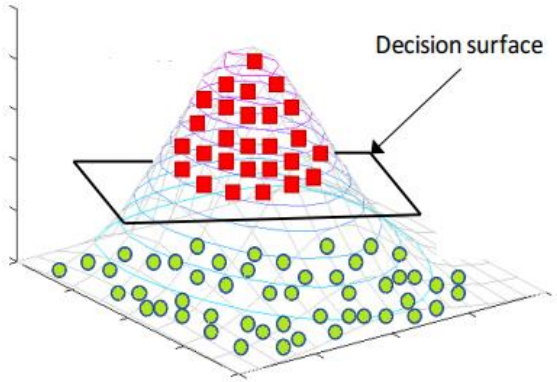
→ Al utilizar el cambio de representación anterior, tenemos que el producto punto de los dos vectores es igual al cuadrado del producto punto!



- "**The Kernel Trick**": Utilizar una **función "kernel"** $K(x^{(i)}, x^{(j)})$, que representa una noción de similitud entre 2 instancias en un espacio de mayor dimensión, de tal manera que **no es necesaria la transformación explícita $\phi$!**

$$Q(\alpha) = \sum_i \alpha^{(i)} - \frac{1}{2}\sum_{i,j} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}x^{(j)T}x^{(j)}, \quad \text{es una instancia de}$$

$$Q(\alpha) = \sum_i \alpha^{(i)} - \frac{1}{2}\sum_{i,j} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\boxed{K(x^{(i)}, x^{(j)})},$$



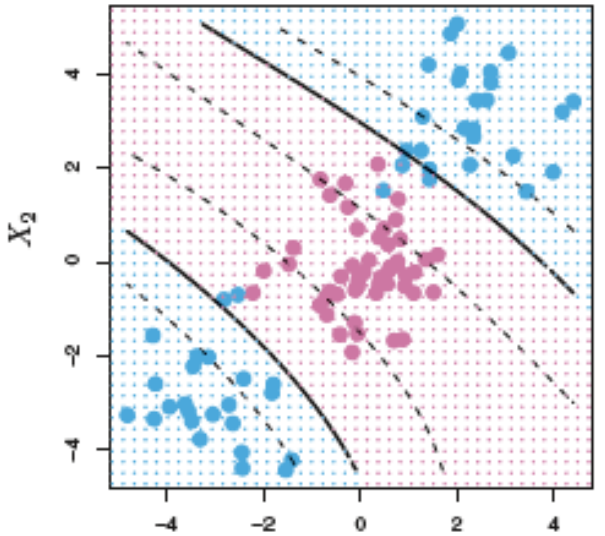

https://www.hackerearth.com/blog/developers/simple-tutorial-svm-parameter-tuning-python-r/

Javier Diaz, PhD
UNIVERSIDAD ICESI

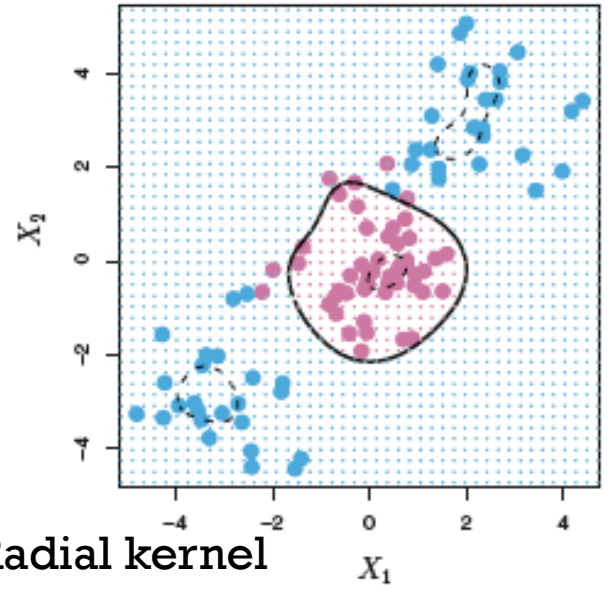
# SVM - CLASIFICACIÓN

- Dependiendo del dominio de aplicación, se pueden utilizar diferentes funciones Kernel, mapeando las instancias originales a un espacio dimensional, donde sí es posible encontrar un hiperplano lineal discriminante
- Las funciones Kernel mas comunes son:
  - Lineal: $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \boldsymbol{x}^{(i)T} \cdot \boldsymbol{x}^{(j)} = \sum_p x_p^{(i)} x_p^{(j)}$
  - Polinomial: $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \left(\boldsymbol{x}^{(i)T} \cdot \boldsymbol{x}^{(j)} + c\right)^d$
  - RBF : $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = exp\left(-\gamma \left\| \boldsymbol{x}_p^{(i)} - \boldsymbol{x}_p^{(j)} \right\|^2\right)$ o

    Gaussiano $= exp\left(-\frac{\left\|x_p^{(i)} - x_p^{(j)}\right\|^2}{2\sigma^2}\right), con\ \gamma = 1/2\sigma^2$
  - String: Distancia de Levenshtein
- Un Kernel válido, debe satisfacer las **condiciones de Mercer**, que garantizan que las funciones actúen como **distancias** (simétrica, continua, positiva semi-definida)



Polynomial kernel



Radial kernel

ISLR, 2015

Javier Diaz, PhD
UNIVERSIDAD ICESI

# SVM - CLASIFICACIÓN

Clasificación con más de 2 clases:

- Uno contra uno:
  - Para cada par de clases crear un clasificador: $n(n-1)/2$ modelos
  - Decidir la clase obtenida en el mayor número de modelos
- Uno contra todos:
  - Crear $n$ modelos, uno para cada clase, comparándolo con una categoría conformada por el resto de clases.
  - Decidir la clase cuya proyección $\boldsymbol{\omega}^T \boldsymbol{x} + b$ esté lo más alejada del margen

Los SVM **no** producen intrínsecamente las probabilidades de clasificación para cada clase, pero existe un procedimiento ("Platt scaling", Platt, 1999) que permite estimarlas. Python lo incluye en su clase **SVC**.

Platt, John (1999). "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods."



# TALLER: SVM DE CLASIFICACIÓN EN PYTHON

- Ejecutar el notebook con el taller de **SVM** en Python
  - Dataset: **Datos sintéticos** (introducción) y **Heart** (caso de estudio)
  - Análisis exploratorio de datos
  - Pretratamiento con **PCA** en scikit-learn
  - Entrenamiento de modelos de clasificación **SVM** con margen suave con y sin kernels en Scikit-learn
  - Análisis de la influencia de los parámetros
    - C, de regularización en el control del overfitting
    - Kernel, con el tipo de función de similitud en el espacio de mayor dimensionalidad
    - Parámetros específicos al kernel
  - Búsqueda del mejor Pipeline de procesamiento con la mejor configuración del clasificador

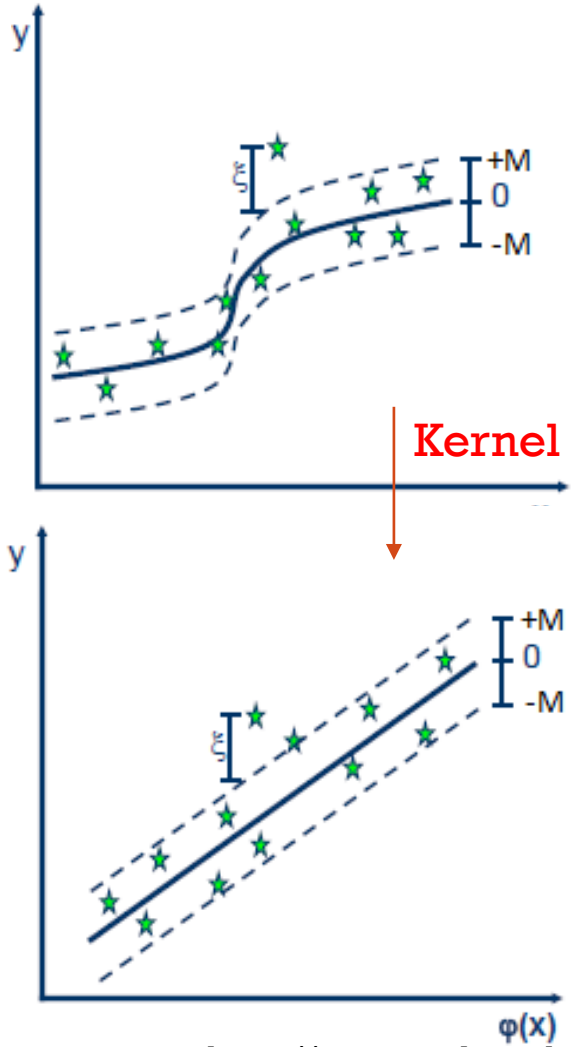
Javier Diaz, PhD
UNIVERSIDAD ICESI

# SVM - REGRESIÓN

La idea de los SVM de clasificación se puede extender también a la **regresión** (Vapnik, 1996)**:**

- Se busca incluir dentro de las márgenes la mayor cantidad de puntos
- Se implementan funciones kernel que se adapten a las no linealidades de los datos a predecir
- Los puntos dentro de las márgenes no influyen
- Los vectores de soporte son los puntos que se encuentran por fuera de los límites de las márgenes (incluyendo los que están sobre las márgenes)
- Se busca minimizar de $\frac{1}{2}\|\boldsymbol{\omega}\|^2$, sujeto a:

$$\begin{cases} y^{(i)} - (\boldsymbol{\omega}^T \boldsymbol{x}^{(i)} + b) \leq M \\ (\boldsymbol{\omega}^T \boldsymbol{x}^{(i)} - b) - y^{(i)} \leq M \end{cases}$$




https://www.saedsayad.com/support_vector_machine_reg.htm

Javier Diaz, PhD
UNIVERSIDAD ICESI

# TALLER: SVM PARA REGRESIÓN

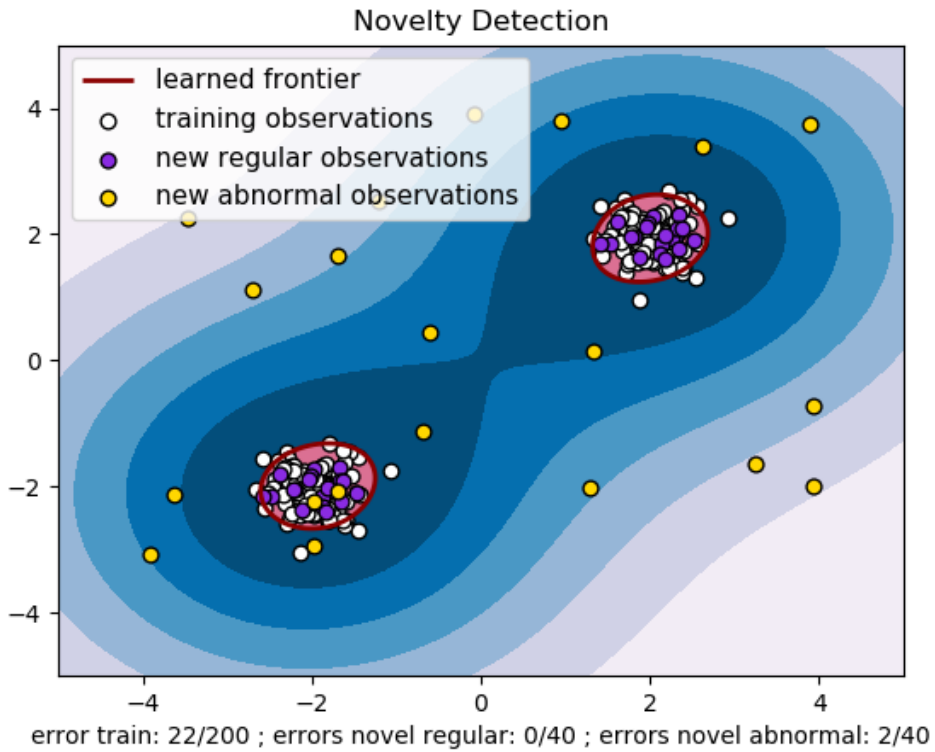
- Ejecutar el notebook con el taller de **SVR** en Python
  - Dataset: **Datos sintéticos**

Javier Diaz, PhD
UNIVERSIDAD ICESI

# SVM – DETECCIÓN DE ANOMALÍAS

También podemos extender la idea de los SVM al problema de detección de anomalías (tarea no supervisada)**: One-Class SVM**

- Estimación de una distribución de los datos en un espacio dimensional mayor
- Idea de similitud a datos del training set
- Se debe definir la amplitud del margen, que va a determinar la anomalidad de los datos




https://scikit-learn.org

Javier Diaz, PhD
UNIVERSIDAD ICESI

# TALLER: ONE-CLASS SVM PARA DETECCIÓN DE ANOMALÍAS

- Ejecutar el notebook con el taller de **One-Class SVM** en Python
  - Dataset: **Datos sintéticos**
  - Entrenamiento de modelos detección de anomalías tipo **One-Class SVM** en Scikit-learn
  - Análisis de la influencia de los parámetros
    - C, de regularización en el control del overfitting
    - Kernel, con el tipo de función de similitud en el espacio de mayor dimensionalidad (rbf vs. poly)
    - Parámetros específicos al kernel

Javier Diaz, PhD
UNIVERSIDAD ICESI

# SUPPORT VECTOR MACHINES

Conclusiones:

- Modelo que sirve para la clasificación, regresión y detección de outliers!
- Basado en la búsqueda del margen más ancho
- Funciona para encontrar patrones lineales y no lineales, gracias a la aplicación de funciones kernel,
- Escogencia de una función Kernel dado el dominio de aplicación, que permite proyectar los datos en espacios de mayor dimensión implícitamente, buscando similitudes entre ellos sin necesidad de modificar su representación
- Se controla el ajuste utilizando un parámetro de regularización $C$, que especifica la rigidez frente al error de entrenamiento
- Solo algunas instancias del dataset de entrenamiento son importantes (los vectores de soporte)



# REFERENCIAS


- *Introduction to Statistical Learning with Applications in R (ISLR)*, G. James, D. Witten, T. Hastie & R. Tibshirani, 2014, Springer
- *Python Data Science Handbook,* Jake VanderPlas, 2017, O'Reilly
- *The Elements of Statistical Learning,* T. Hastie & R. Tibshirani, 2009, Springer
- *Python Machine Learning (2nd ed.),* Sebastian Raschka, 2017, Packt