Kerneles SVM

Dr. Misael López Ramírez

m.lopezramirez87@gmail.com

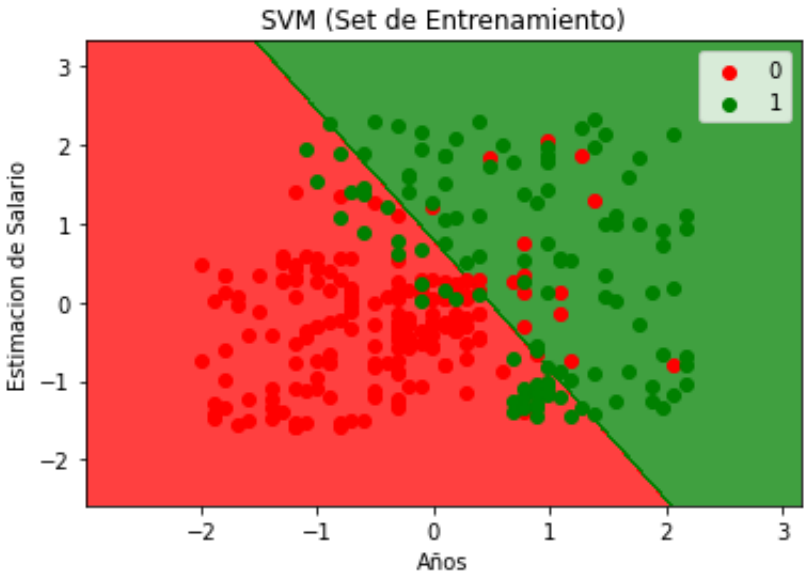
m.lopezramirez@ugto.mx

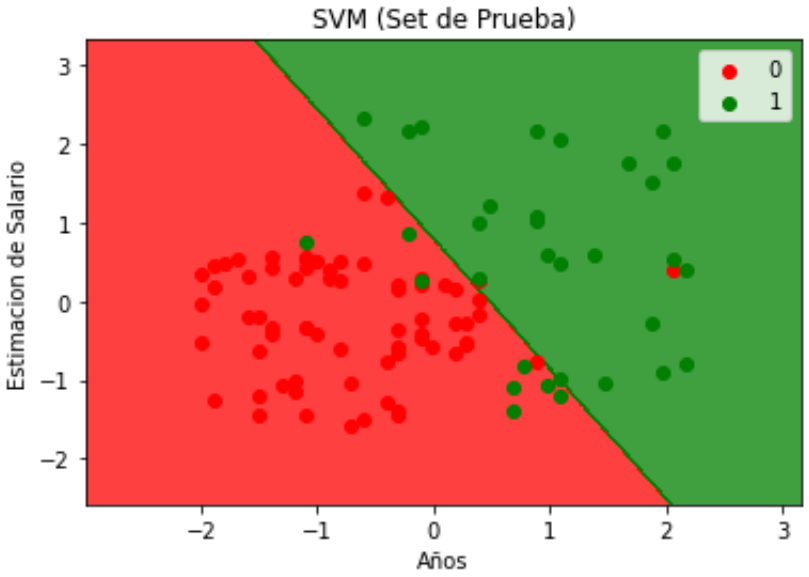
Tel: 464 128 09 28

SVM

```python
# Entrenamiento SVM
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

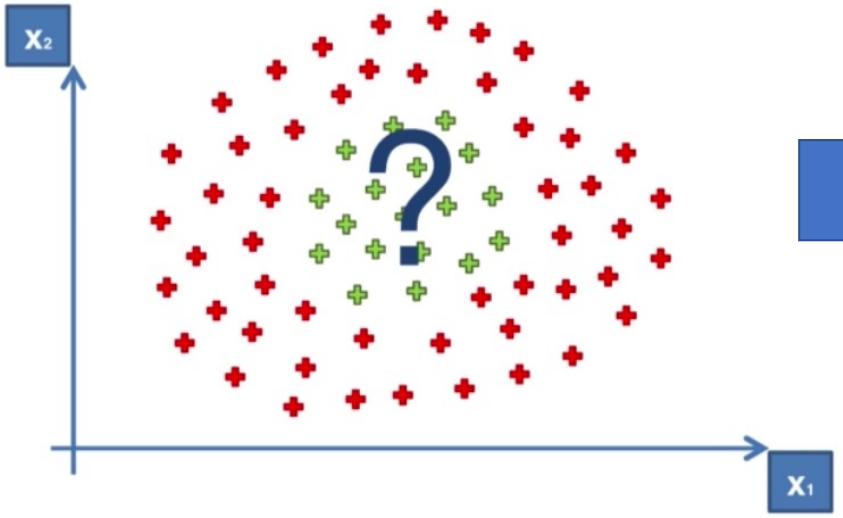
Resultados





SVM

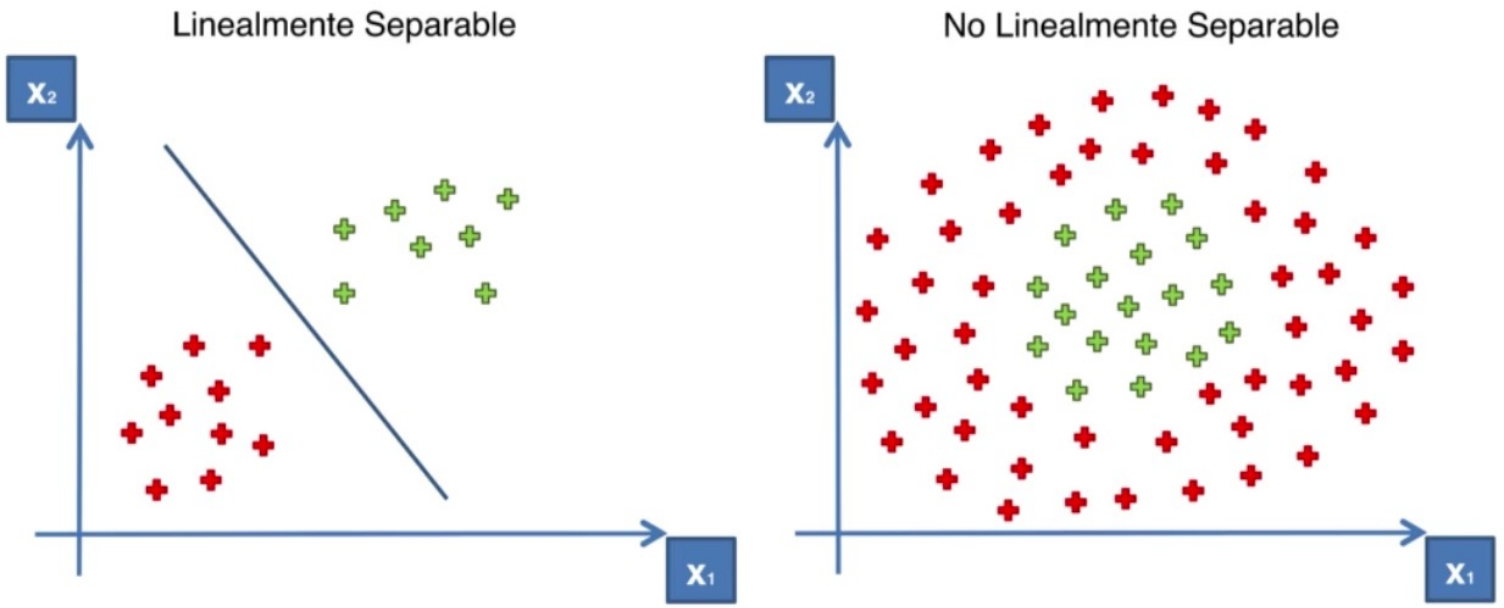
Por que Utilizarlos



Porque esta nube de puntos no es LINEALMENTE SEPARABLE

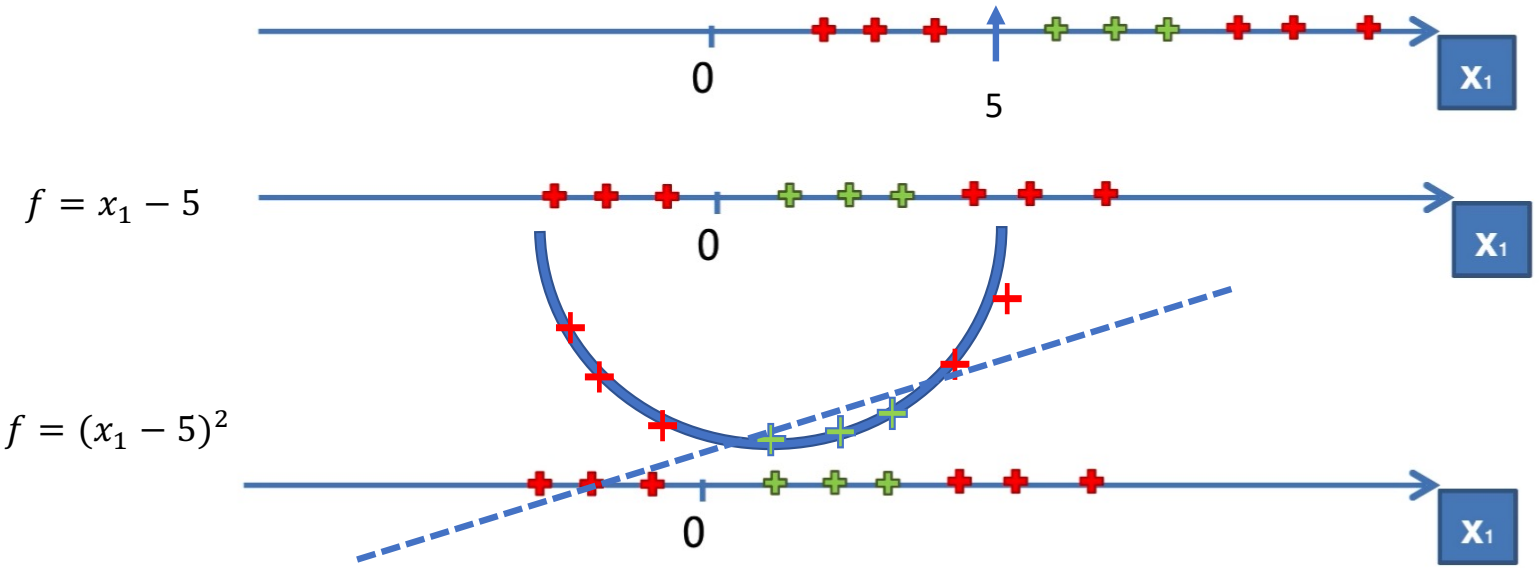
SVM

Por que Utilizarlos

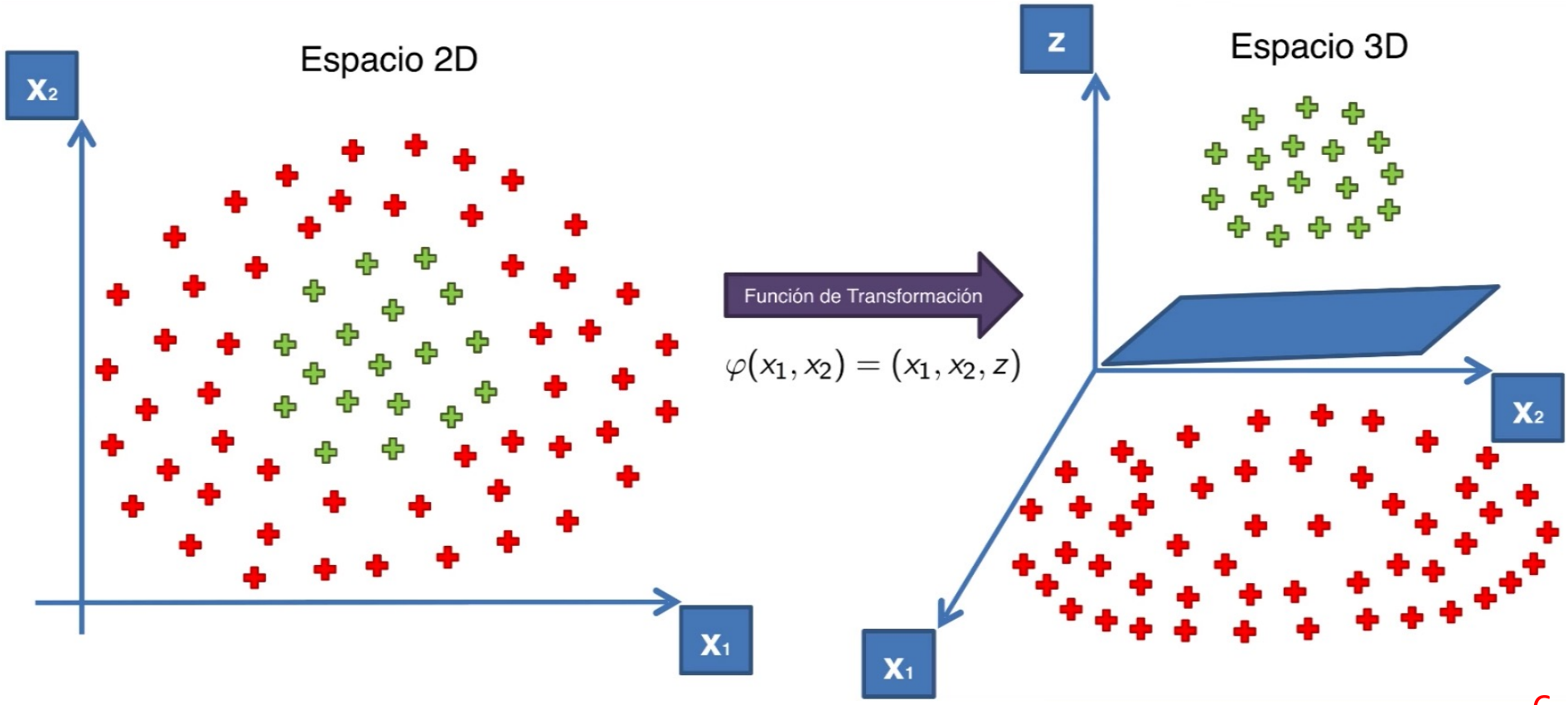


Idea del Uso de Kernel

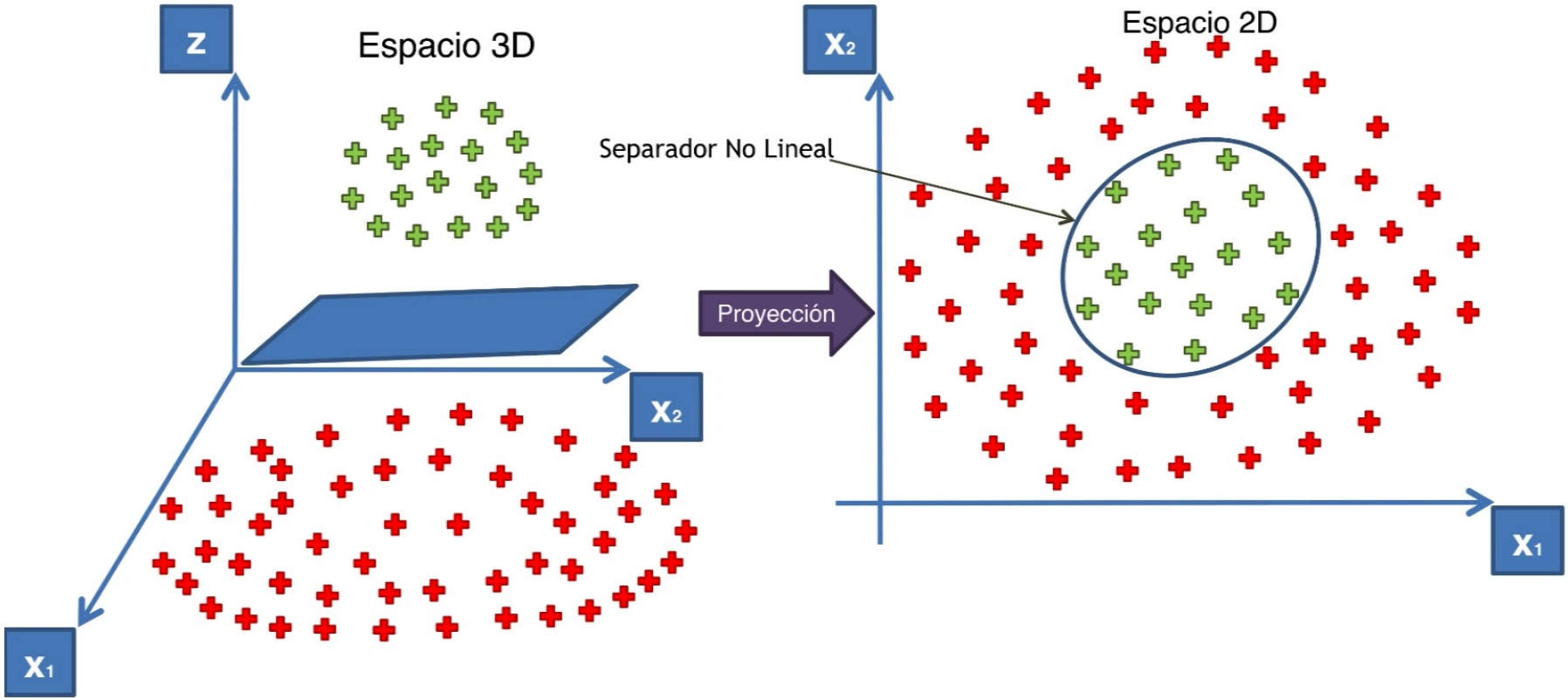
- Aumentando dimensiones



SVM a una Dimensión Superior



SVM a una Dimensión Superior



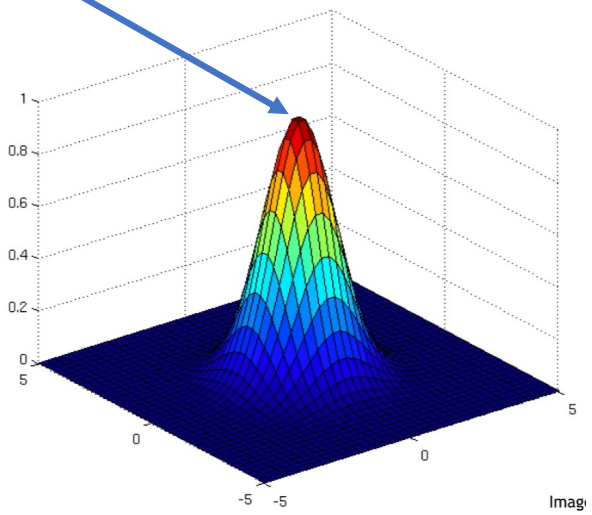
SVM a una Dimensión Superior

Transformar una variable a un Espacio de Dimension Superior puede ser muy costoso computacionalmente

Kernel Gaussiano RBF

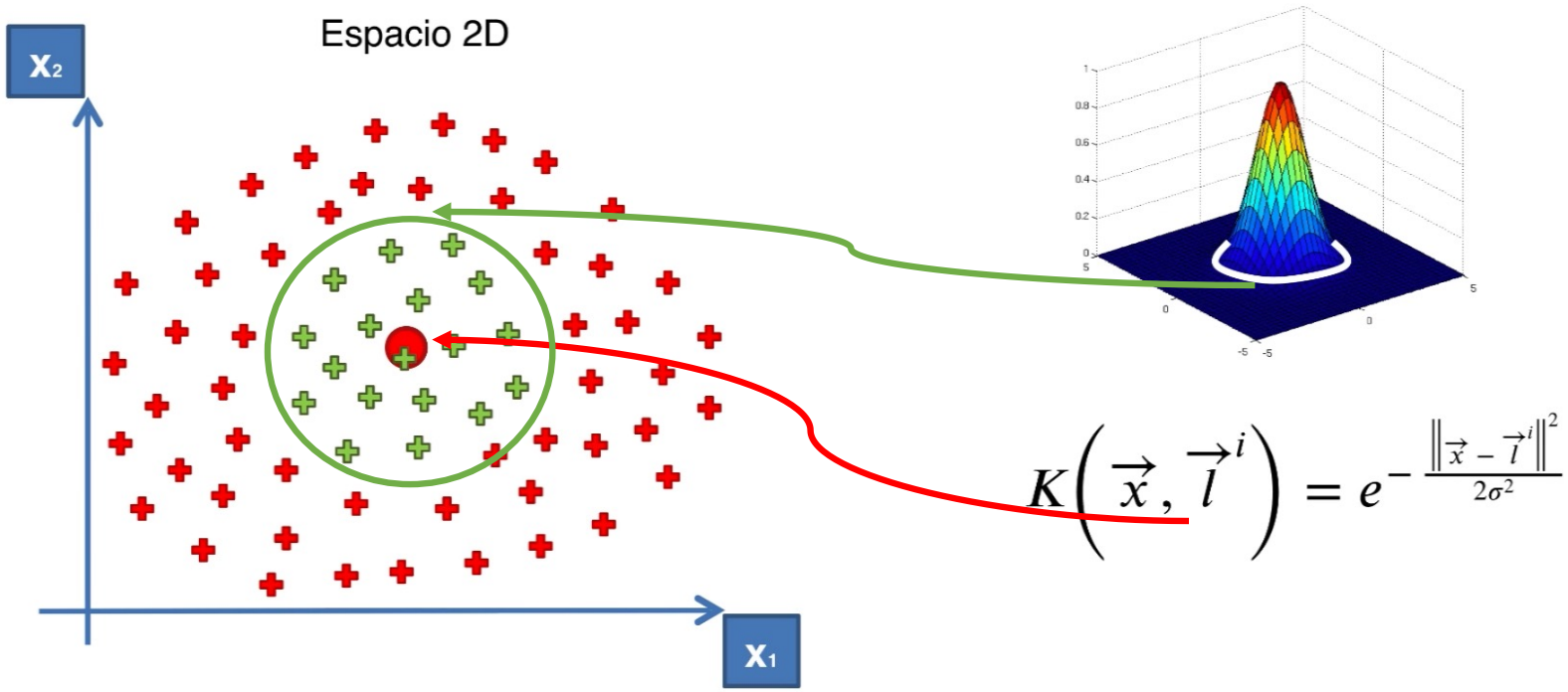
$$K\left(\vec{x}, \vec{l}^{\,i}\right) = e^{-\frac{\left\|\vec{x} - \vec{l}^{\,i}\right\|^2}{2\sigma^2}}$$

Media= Punto Central

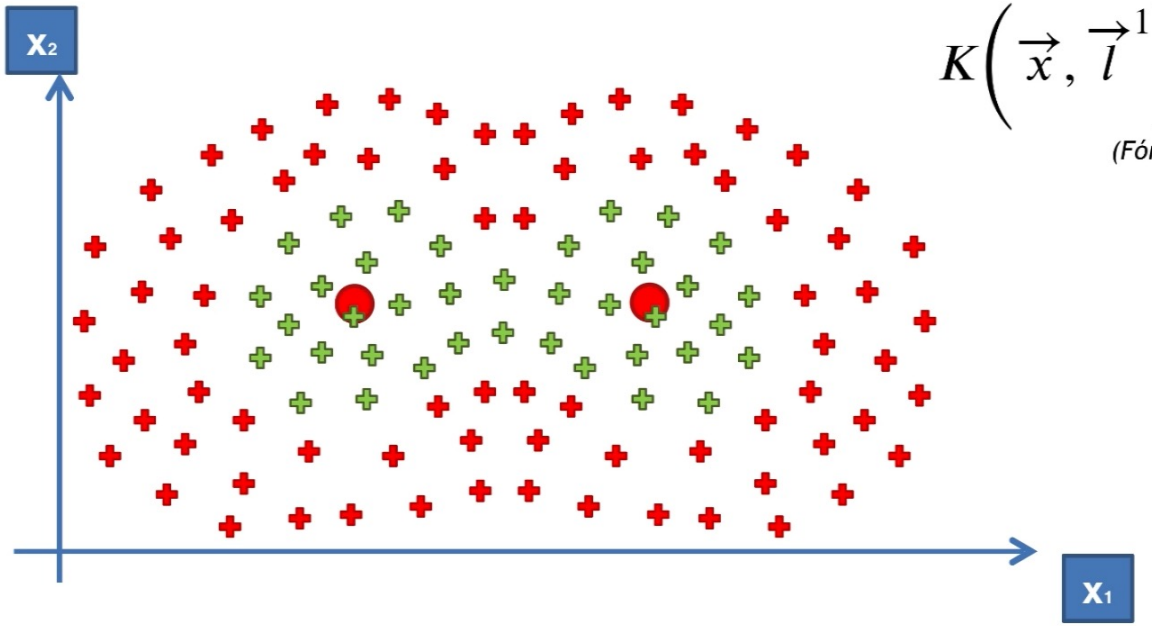


Kernel Gaussiano RBF

Media= Punto Central



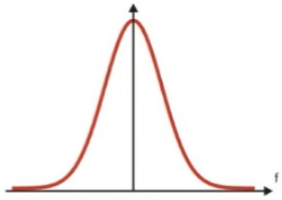
Kernel Gaussiano RBF



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

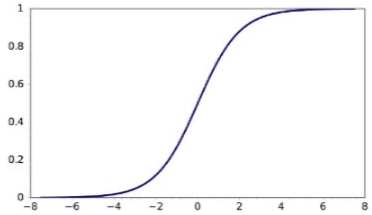
(Fórmula Simplificada)

Tipos de kernel



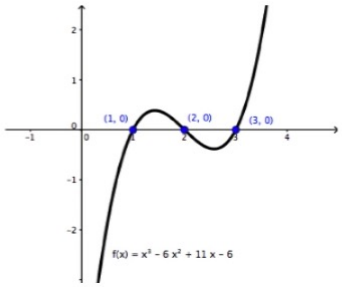
Kernel Gaussiano RBF

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



Kernel Sigmoide

$$K(X, Y) = \tanh\left(\gamma \bullet X^T Y + r\right)$$



Kernel Polinómico

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

sklearn.svm.SVC

class `sklearn.svm.`**SVC**(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

C : *float, default=1.0*
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, default=3*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Changed in version 0.22: The default value of `gamma` changed from 'auto' to 'scale'.

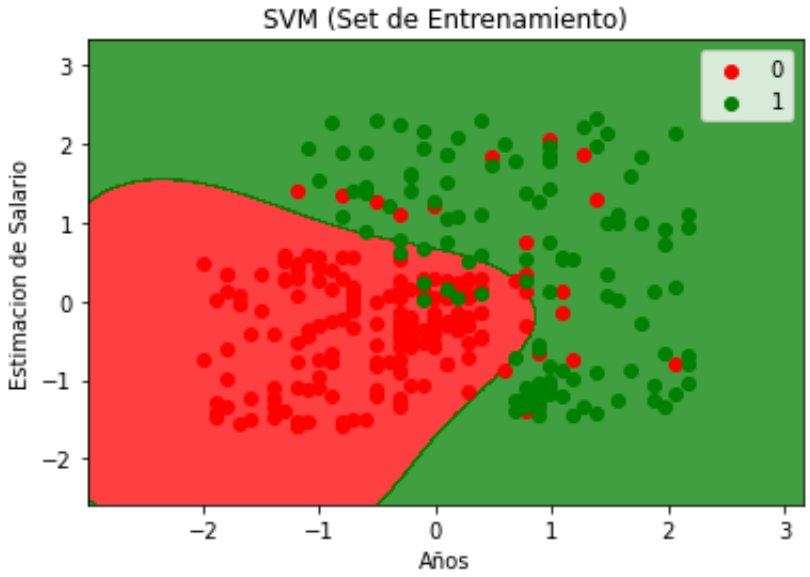
coef0 : *float, default=0.0*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

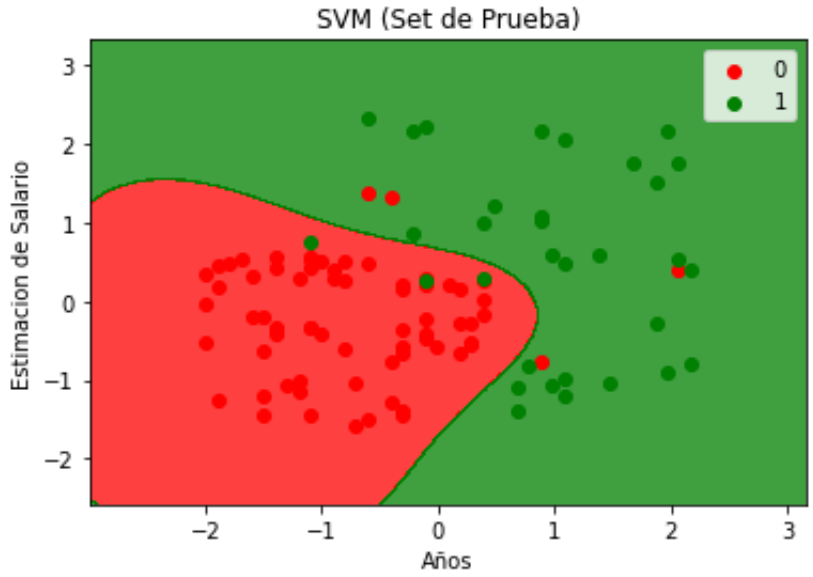
shrinking : *bool, default=True*
Whether to use the shrinking heuristic. See the User Guide.

sklearn.svm.SVC

class sklearn.svm.**SVC**(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*) [source]

```
# Entrenamiento SVM
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

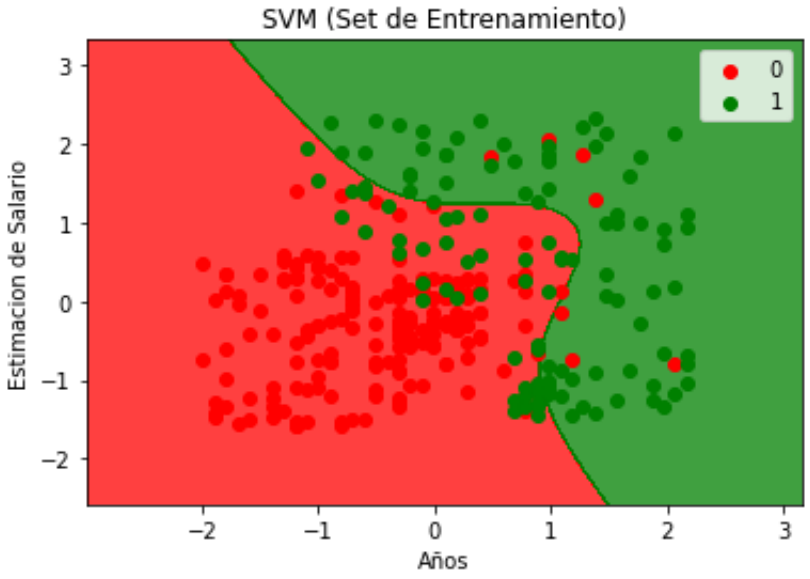


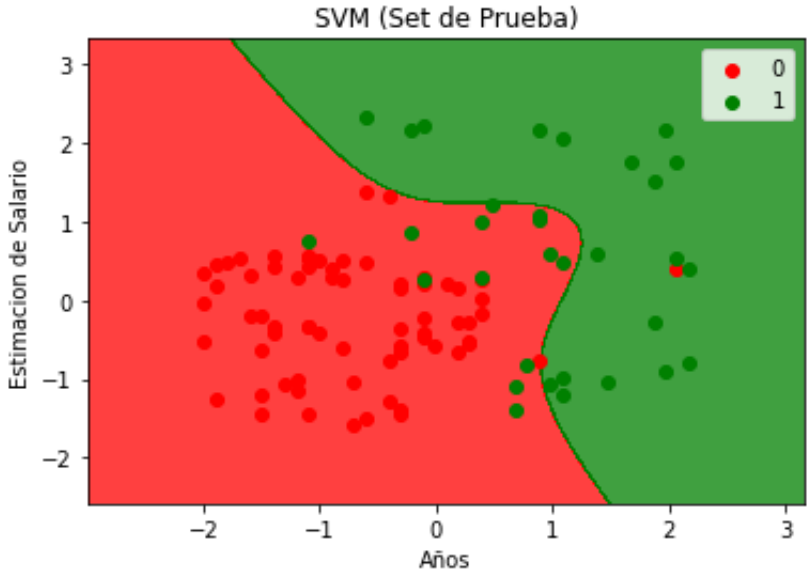


sklearn.svm.SVC

class sklearn.svm.**SVC**(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*) [source]

```
# Entrenamiento SVM
from sklearn.svm import SVC
classifier = SVC(kernel = 'poly', random_state = 0)
classifier.fit(X_train, y_train)
```

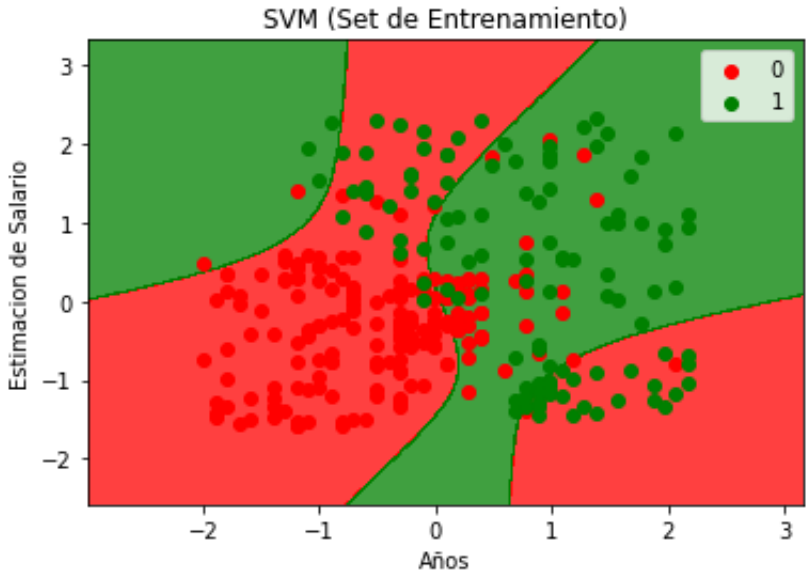


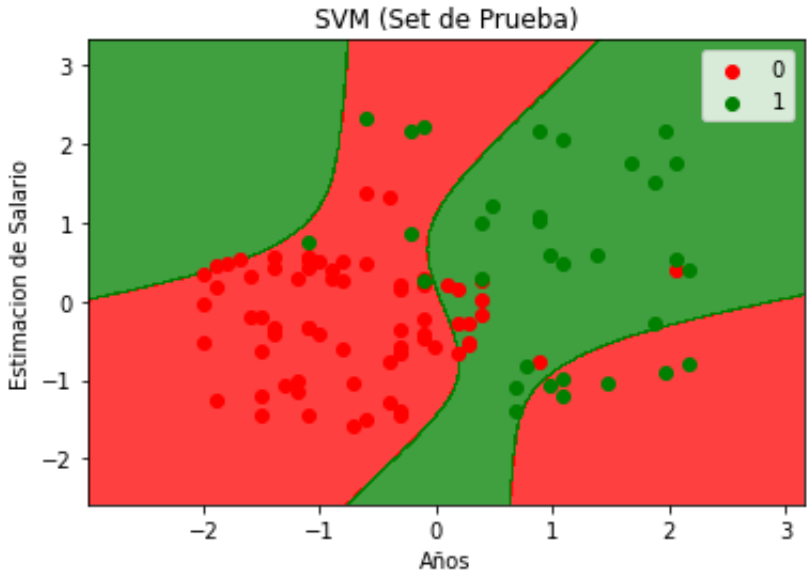


sklearn.svm.SVC

class sklearn.svm.**SVC**(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*) [source]

```
# Entrenamiento SVM
from sklearn.svm import SVC
classifier = SVC(kernel = 'sigmoid', random_state = 0)
classifier.fit(X_train, y_train)
```





Continuara.....