Aprendizaje Automático

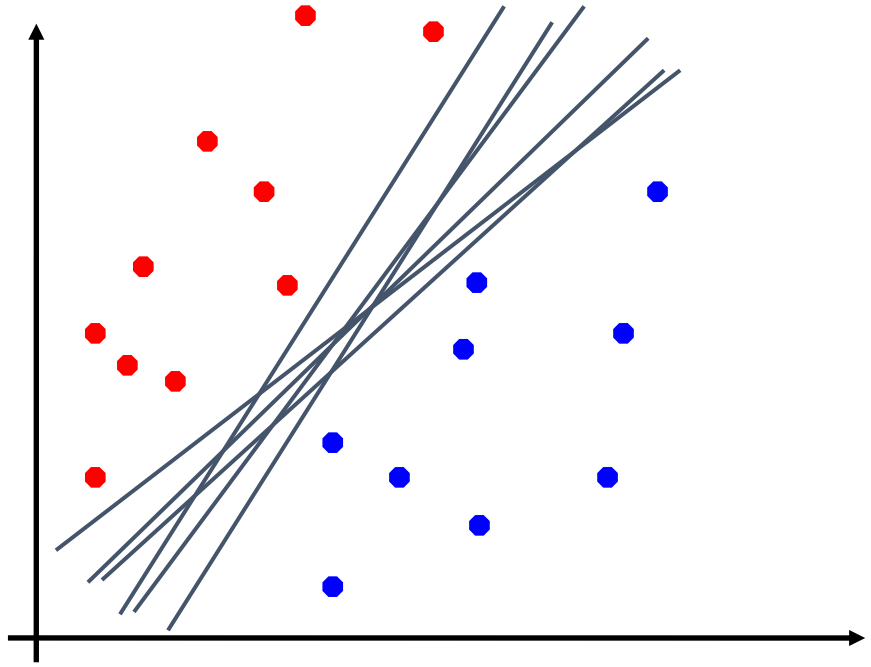
Departamento de Informática – UC3M

Tutorial 5 – Búsqueda hiperparámetros en SVM

2025

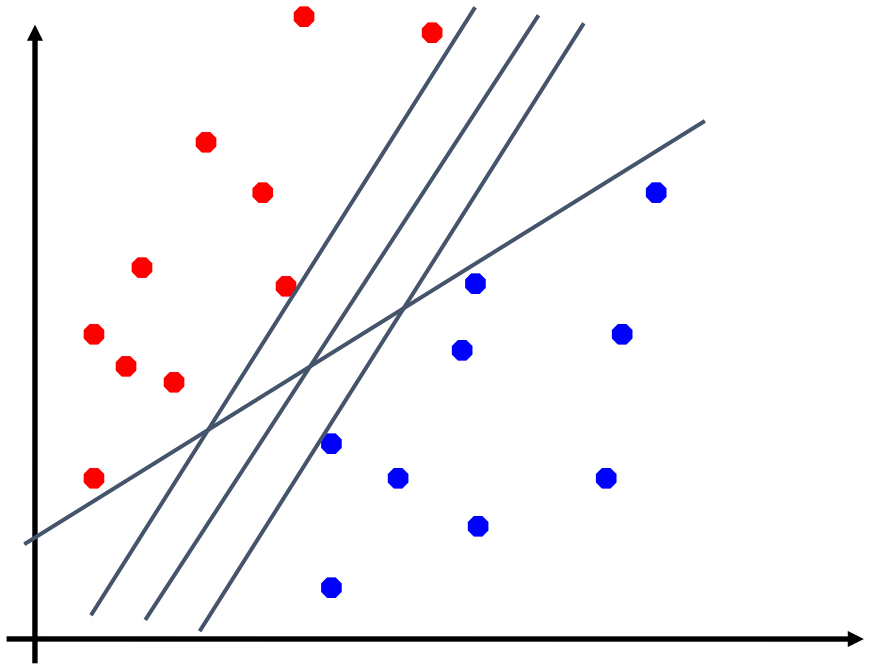
Tutorial 5

Recordando teoría



Tutorial 5

Recordando teoría



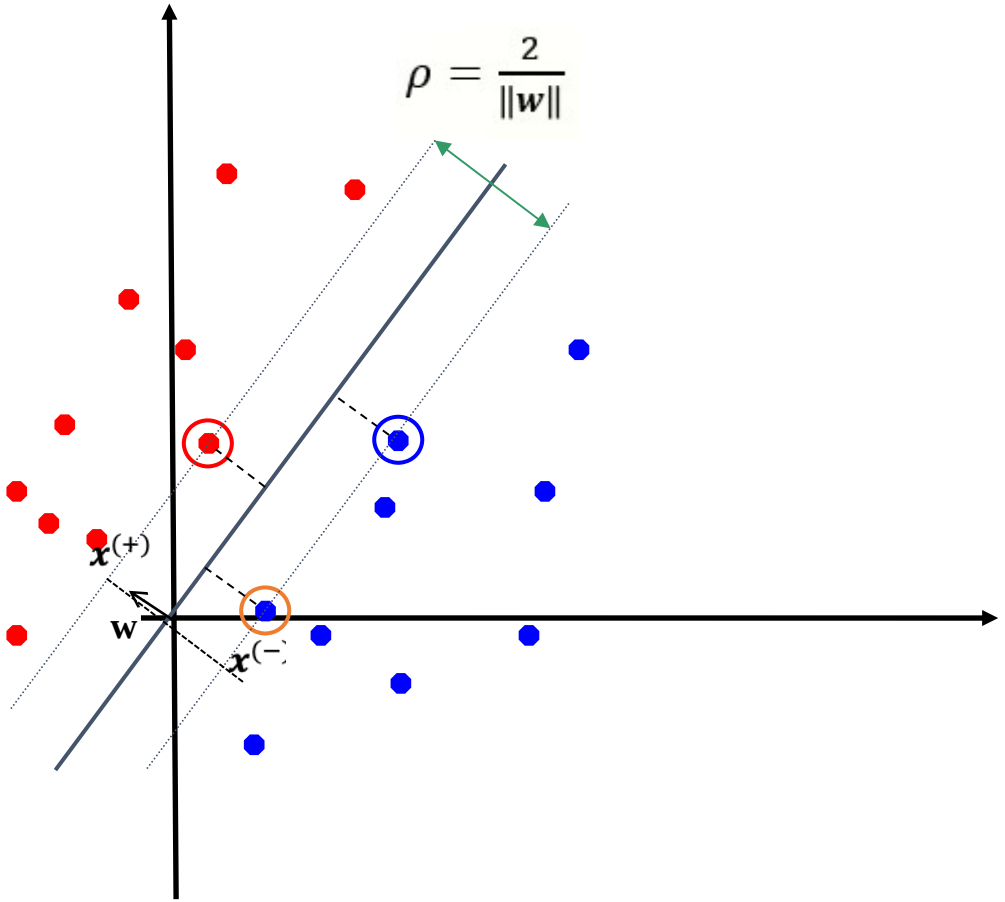
Tutorial 5

Recordando teoría

Elegir el hiperplano que tenga el máximo margen

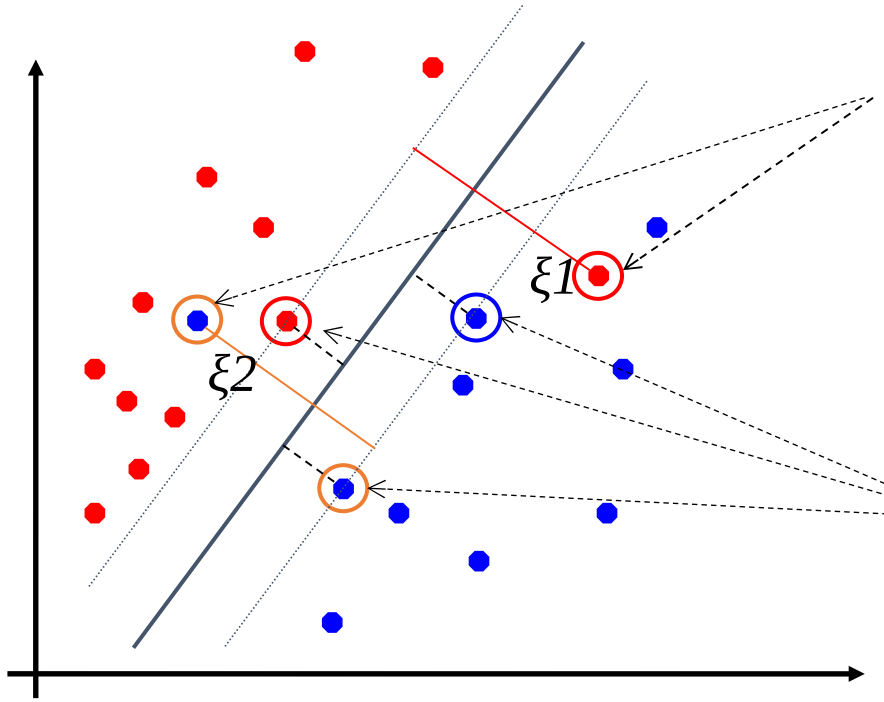
Vectores soporte

Margen. Distancia perpendicular desde los vectores soporte.



Tutorial 5

Recordando teoría



Instancias inclasificables y variables de holgura (slack)

Vectores de soporte

Minimizar

- **sin** *slack*

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

- **con** *slack*

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i$$

Parámetro C

*soft*SVM

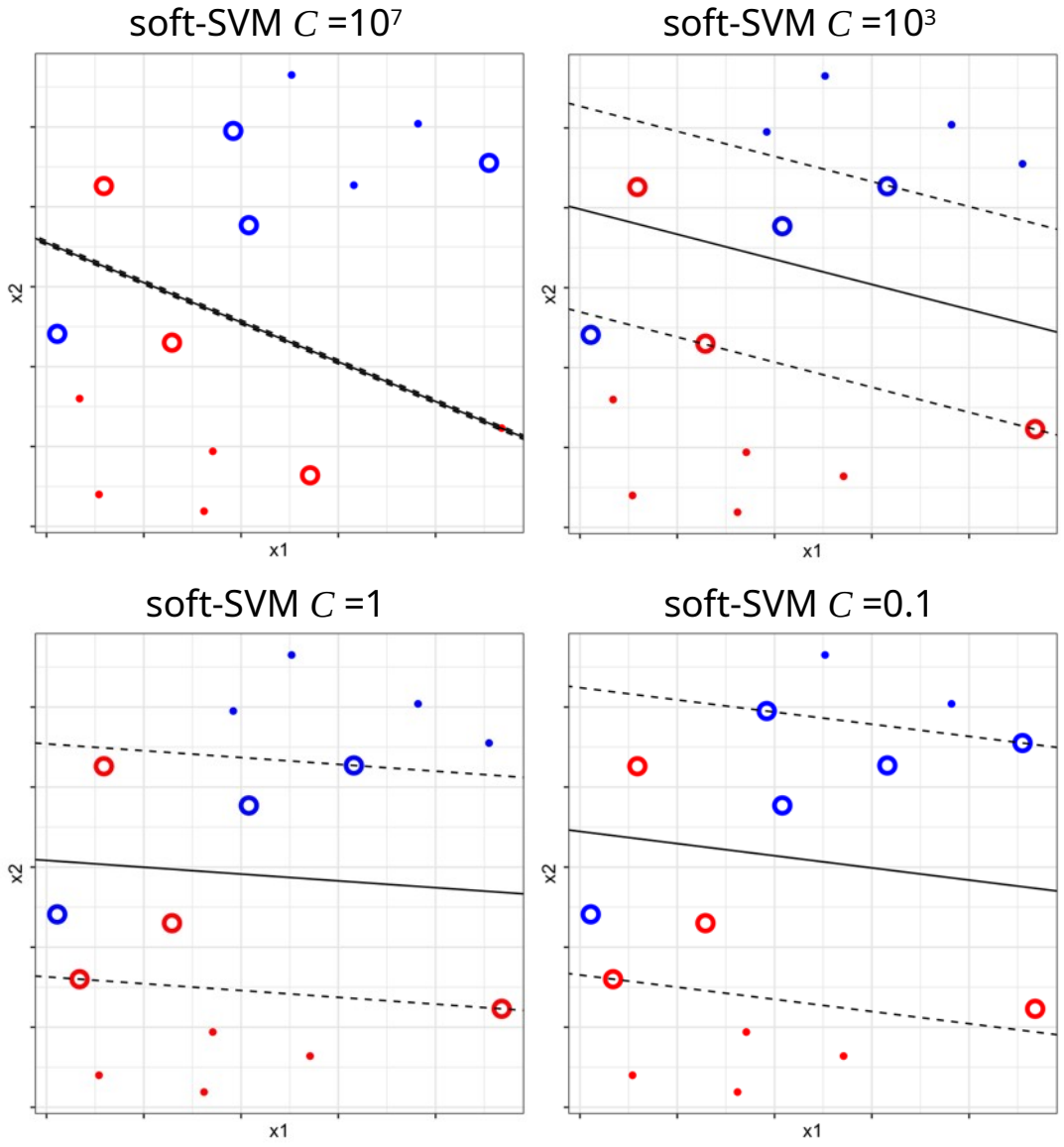
Tutorial 5

Recordando teoría

- Hay dos objetivos contrapuestos, márgenes grandes suponen holguras grandes y viceversa

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i$$

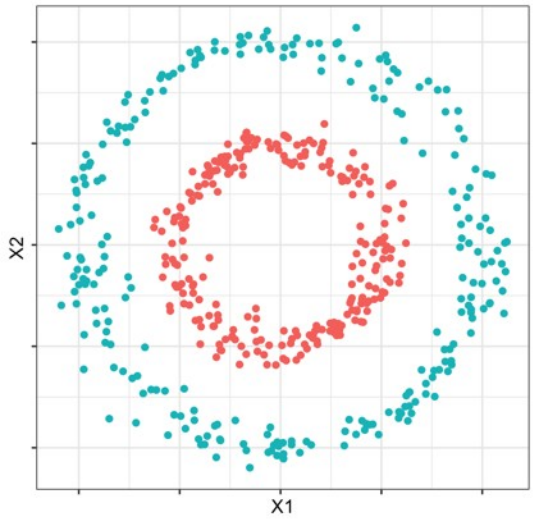
- Si C tiene un valor grande
  - Mucha importancia a que todos los datos de entrenamiento se clasifiquen correctamente (*slack variables* tienden a cero)
  - Si hay ruido, puede haber overfitting
- Si C tiene un valor pequeño ocurre lo contrario
  - Si es demasiado pequeño, puede ocurrir que haya demasiados datos de entrenamiento mal clasificados (muchas *slack variables* con valores altos)

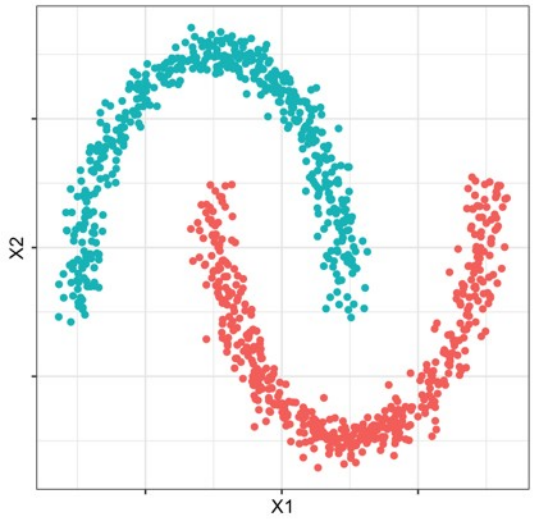


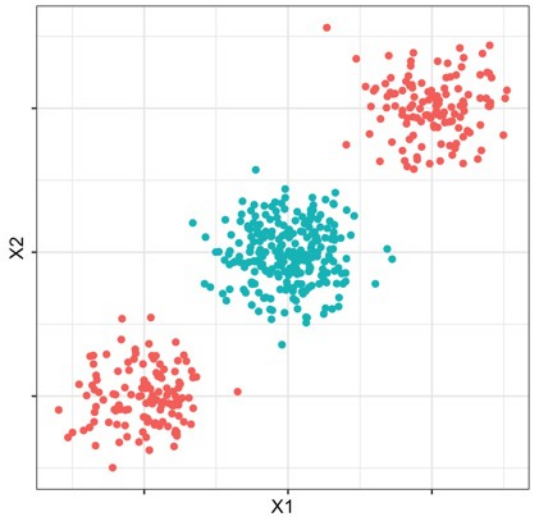
Tutorial 5

Recordando teoría

- Modelo lineal
  - los datos son separables linealmente (*hard margin*)
  - los datos no son separables linealmente (*soft margin*)
- Modelo no lineal
  - kernels



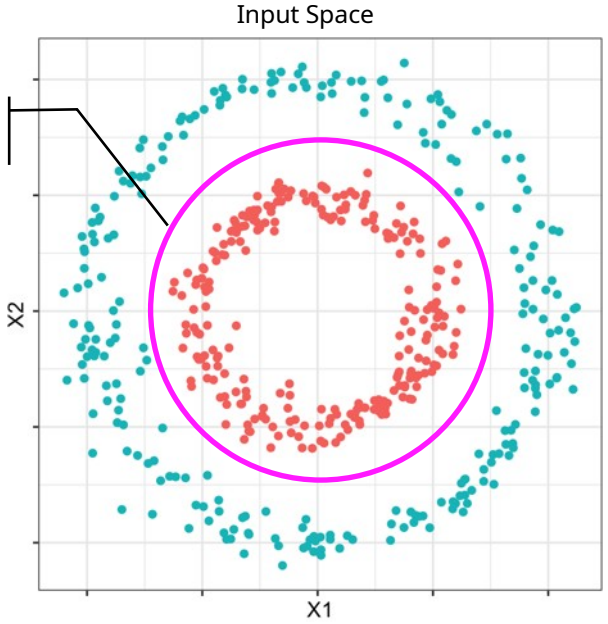




Tutorial 5

Recordando teoría

Espacio original
Espacio de entrada



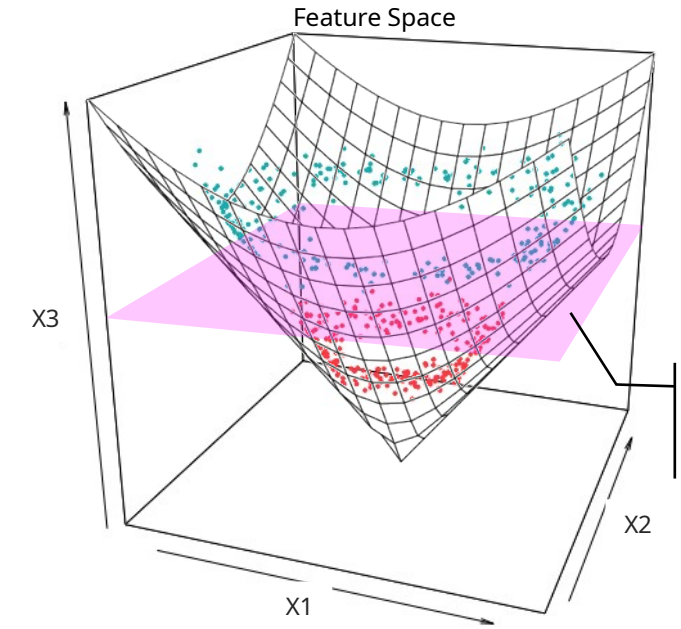
Frontera de Decisión no lineal

$\phi: (x_1, x_2) \rightarrow \left(x_1, x_2, \sqrt{x_1^2 + x_2^2}\right)$

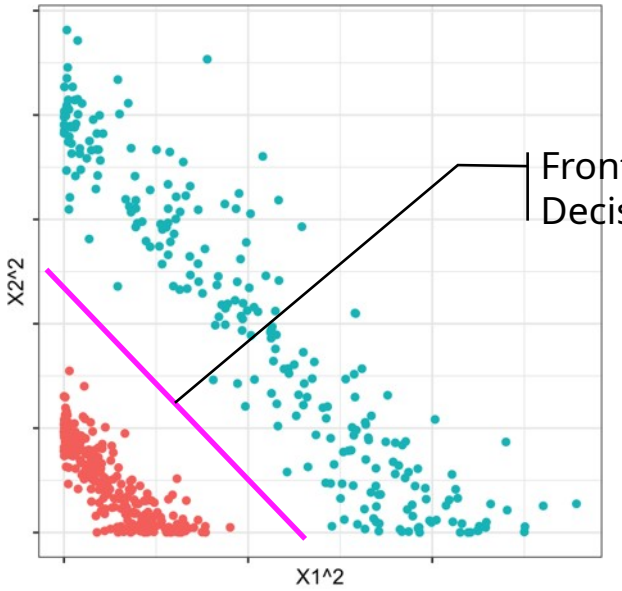
$\phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$

$\phi: (x_1, x_2) \rightarrow (x_1^2, x_2^2, 1)$

Espacio transformado
Espacio de características



Frontera de Decisión lineal

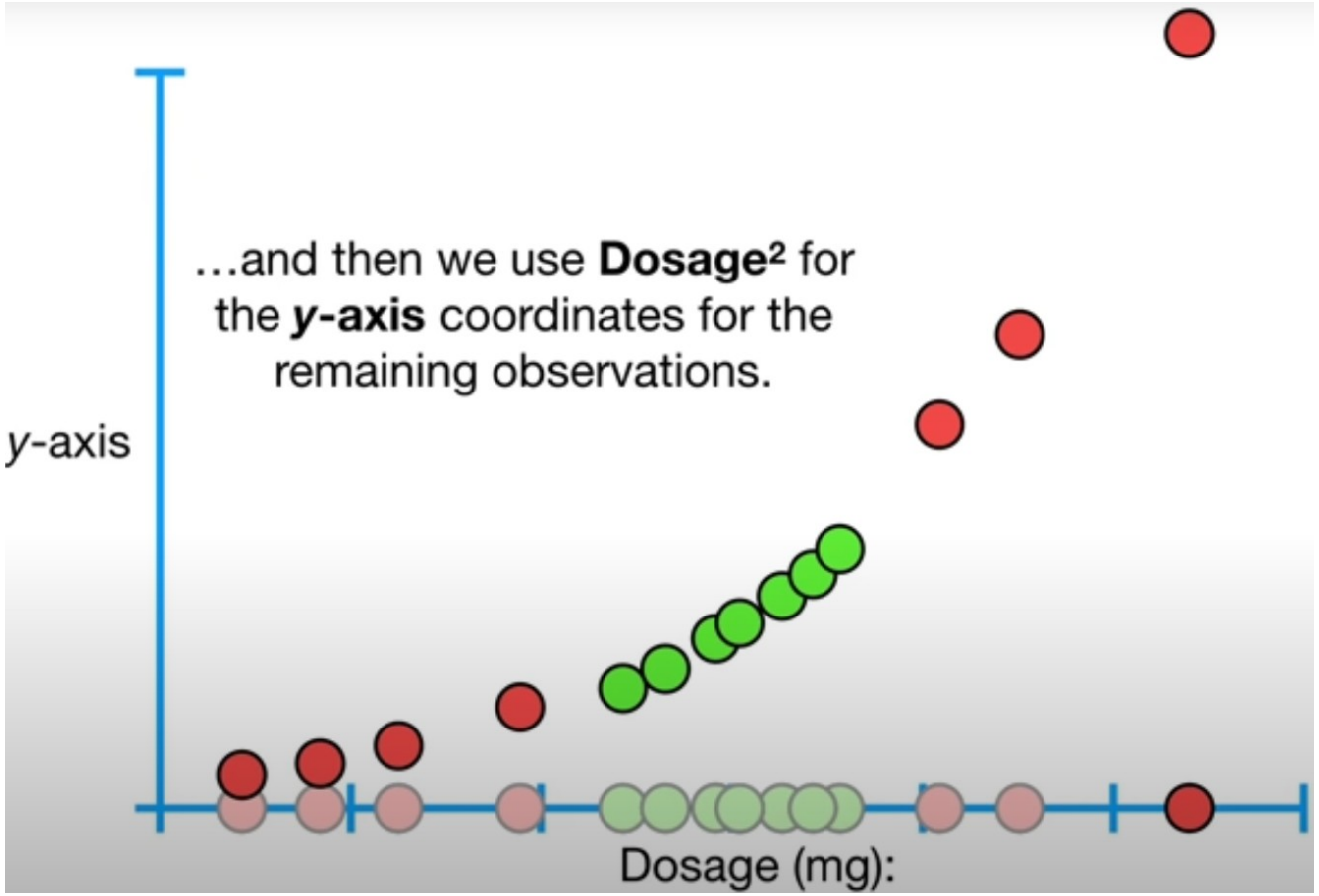


Frontera de Decisión lineal

Se pasa de la formulación:
$w_2 x_1 + w_1 x_2 = 0$ que no separa linealmente
a
$w_2 x_1^2 + w_1 x_2^2 + w_0 = 0$ que sí lo hace

Tutorial 5

Recordando teoría



Máquinas de Soporte Vectorial

Separación no lineal → Separación lineal

- La frontera de decisión

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$

pasar a ser:

$$\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b = 0$$

- El problema de optimización

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}, y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right) \geq 1, \forall\{(x_i, y_i)\}$$

pasa a ser:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}, y_i\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b\right) \geq 1, \forall\{(x_i, y_i)\}$$

- El clasificador

$$y = sign\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{z} + b\right)$$

pasa a ser:

$$y = sign\left(\mathbf{w}^{\mathrm{T}}\phi(\boldsymbol{z}) + b\right)$$

que en términos de multiplicadores de Lagrange (proceso de optimización) es:

$$y = sign\left(\sum_{i=1}^{n} \lambda_i y_i \phi(x_i) \cdot \phi(\mathbf{z}) + b\right)$$

Máquinas de Soporte Vectorial

Kernel Trick – Truco del Kernel

El cálculo es costoso:

$$\phi(x_i) \cdot \phi(\mathbf{z})$$

Es un producto escalar que mide la similitud entre vectores.

Truco... utilizo unas funcionales Kernel (similitud).

Cambiamos:

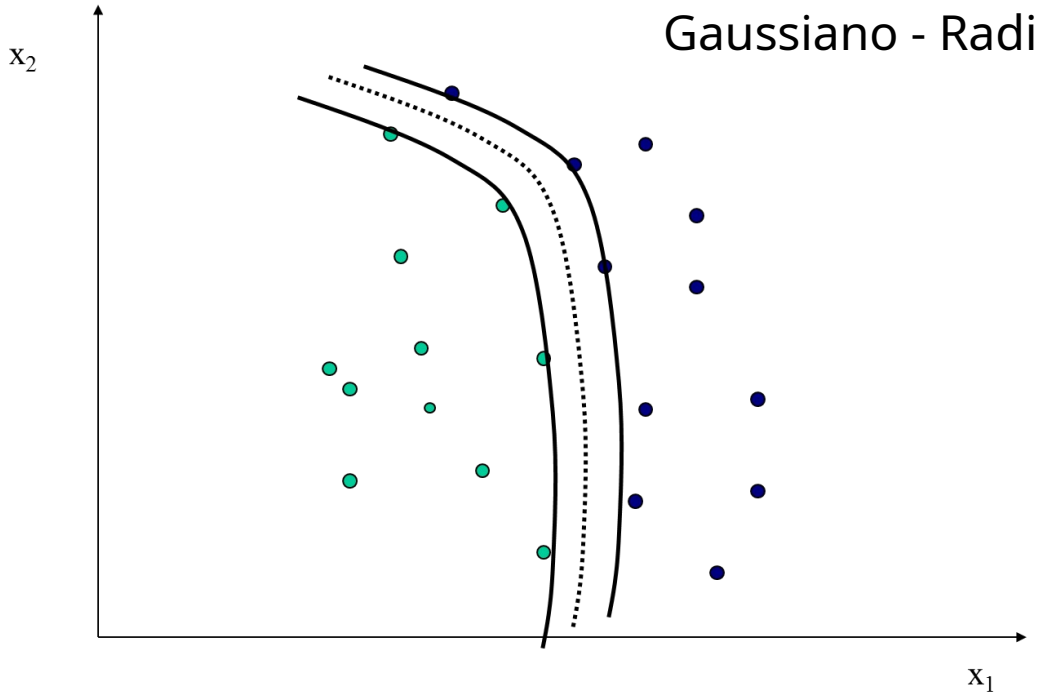
$$y = sign\left(\sum_{i=1}^{n} \lambda_i y_i \phi(x_i) \cdot \phi(\mathbf{z}) + b\right)$$

Por

$$y = sign\left(\sum_{i=1}^{n} \lambda_i y_i K(x_i, z) + b\right)$$

$$K(x_i, z) = e^{-\frac{\|x_i - z\|^2}{2\sigma^2}} = e^{-\gamma \|x_i - \mathbf{z}\|^2}$$

Gaussiano - Radial



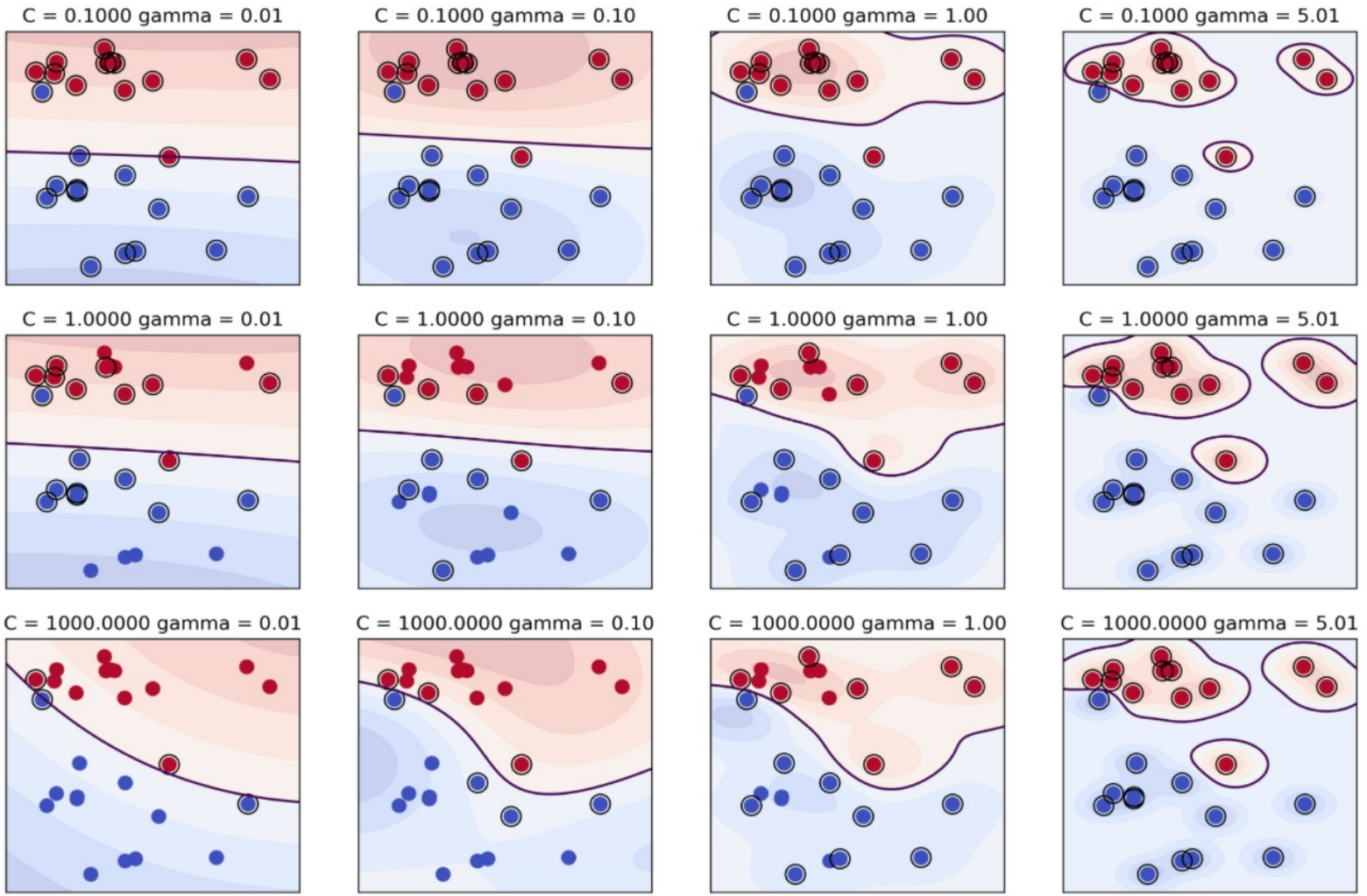
Tutorial 5

$$K(\vec{x}_i.\vec{x}) = e^{-\gamma\|\vec{x}_i-\vec{x}\|^2}$$ Kernel Radial / Gaussiano

gamma

- Gamma pequeño, modelo demasiado simple (casi lineal)
- Gamma grande, sobreaprendizaje (radio de influencia limitado al vector de soporte)

C



Tutorial 5

Metodología SVMs

1) Escalar (normalizar) atributos
2) Ajuste de hiperparámetros:
   a) Parámetro **C** (coste): valores grandes tienden a sobreajustar, valores pequeños a infra-ajuetar.
   b) Kernel:
      - Lineal (polinómico de orden 1): ninguno
      - Gaussiano: gamma. **Gammas** grandes tienden a sobreajustar.
   - En el artículo "A practical guide to Support Vector Classifier" recomiendan:
      - C en el rango $[2^{-5};2^{15}]$
      - gamma en el rango $[2^{-15};2^{3}]$
      - pero puede depender del problema.
      - Con espacios de búsqueda loguniform
      - Nota: en scikit-learn gamma suele tener un buen default γ = 1/(M*var)

        Donde M = número de atributos y var = varianza de la todos los atributos

Tutorial 5

Biblioteca sklearn

- En *Scikit Learn* pueden encontrarse tres implementaciones distintas del algoritmo Suport Vector Machine:
  - Las clases `sklearn.svm.SVC` y `sklearn.svm.NuSVC` permiten crear modelos SVM de clasificación empleando kernel ***lineal***, *polinomial*, ***radial*** *o sigmoide*. La diferencia es que **SVC** controla la regularización a través del hiperparámetro **C**, mientras que NuSVC lo hace con el número máximo de vectores soporte permitidos.
  - La clase `sklearn.svm.LinearSVC` permite ajustar modelos SVM con kernel lineal. Es similar a SVC cuando el parámetro kernel='linear', pero utiliza un algoritmo más rápido.
- Para regresión: `sklearn.svm.SVR`, `sklearn.svm.NuSVR` y `sklearn.svm.LinearSVR`.

Tutorial 5

Parámetros

- **C** : float, default=1.0 Parámetro de regularización. La fuerza de la regularización es inversamente proporcional a C. Debe ser estrictamente positivo.
- **gamma** : **Se usa en kernel Radial**.

$$K(\mathbf{u}, \mathbf{v}) = e^{-\frac{\|\boldsymbol{u}-\mathbf{v}\|^2}{2\sigma^2}} = e^{-\gamma\|\boldsymbol{u}-\mathbf{v}\|^2}$$

$$\gamma = \frac{1}{2\sigma^2}$$

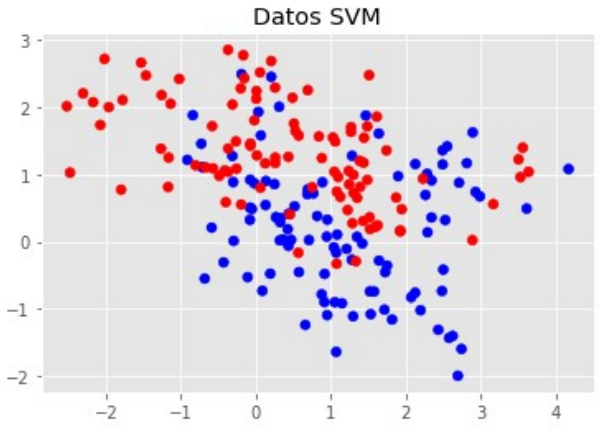
Define cuánta curvatura queremos en la frontera de decisión:

- Gamma alta significa más curvatura.
- Gamma baja significa menos curvatura.

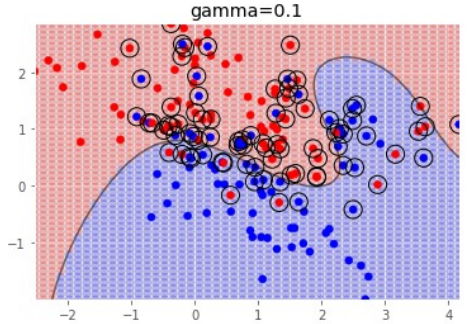
El parámetro gamma define hasta dónde llega la influencia de un único ejemplo de entrenamiento, donde valores bajos significan 'lejos' y valores altos significan 'cerca'.

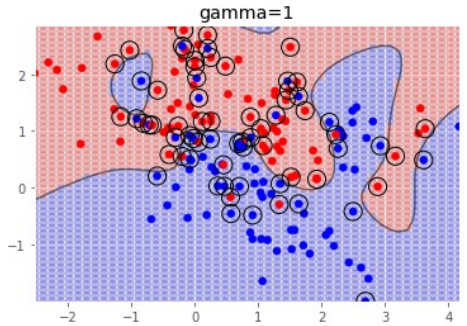
Tutorial 5

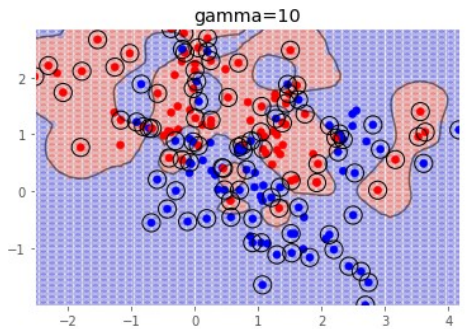
Clasificación

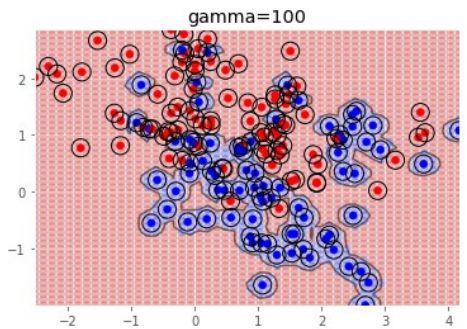


Radial
Gamma [0.1, 1, 10, 100]

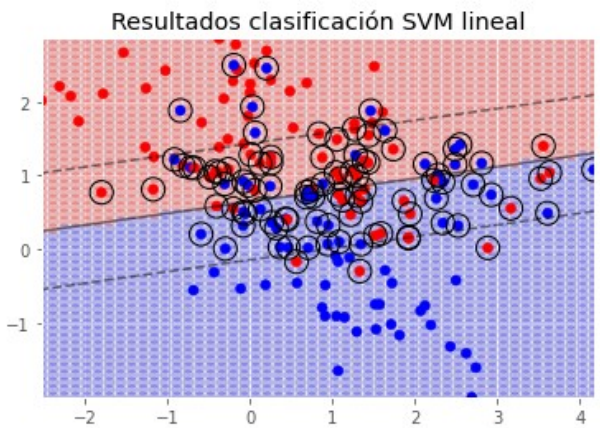








Lineal
(GridSearch/RandomSearch)



Tutorial 5

Regresión (pipeline)

```
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
# This is the preprocessing pipeline: SVMs need scaling
scaler = StandardScaler()
svr = SVR()
pipe_regr = Pipeline([
    ('scale', scaler),
    ('SVM', svr)])
np.random.seed(42)
pipe_regr.fit(X=X_train, y=y_train)
```

Tutorial 5

Regresión (pipeline y GridSearch)

Nombre de parámetros

```
param_grid = {'SVM__C': [0.1, 1, 10, 100],
              'SVM__gamma': [0.01, 0.1, 1]}
inner = KFold(n_splits=3, shuffle=True, random_state=42)
hpo_regr = GridSearchCV(pipe_regr,
                        param_grid,
                        scoring='neg_mean_squared_error’,
                        cv=inner,n_jobs=4, verbose=1)
np.random.seed(42)
hpo_regr.fit(X=X_train, y=y_train)
```

Práctica

Problemas de balanceo

- Tenemos muchas más muestras de personas que no abandonan (80%) que de personas que abandonan (20%).
- Gestión del desbalanceo:
  1. Split estratificado (stratify=y):

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

  2. Utilizar como métrica (balanced_accuracy), **media de los recall**:

```
grid = RandomizedSearchCV(…
                          scoring = ‘balanced_accuracy’,
                          …)
```

  3. Algunos métodos (class_wight = ‘balanced’)

```
lr = LogisticRegression(…, class_wight=‘balanced’, …)
```

- Opcional plus: SMOTE (*Imbalance learn*)