Tarea #5

“Clasificación de SVM con Función Radial”

INTELIGENCIA ARTIFICIAL

18:00 – 19:00 Am

Profesor:

JOSE MARIO RIOS FELIX

Alumno:

Estrada Valenzuela Jesús Ernesto

7to semestre

ING. En Sistemas Computacionales

21/09/2025

EXPLICACION PARA EXPOSICION :

1. ¿Qué es una Máquina de Vectores de Soporte?

Es un potente algoritmo de **aprendizaje supervisado** que se utiliza para **clasificación** y **regresión**.

El objetivo principal: Encontrar el "hiperplano" óptimo que mejor separa las distintas clases de datos.

Piénsalo como encontrar la "calle" más ancha posible que separe dos barrios (clases).

2. Conceptos Clave

- **Hiperplano:** Es la frontera de decisión. En 2D es una línea, en 3D es un plano.
- **Vectores de Soporte:** Los puntos de datos más cercanos al hiperplano. Son los que "soportan" y definen la frontera.
- **Margen:** La distancia entre el hiperplano y los vectores de soporte. SVM intenta **maximizar este margen**.

3. El Truco del Kernel (Datos No Lineales)

¿Qué pasa si los datos no se pueden separar con una línea recta?

El **"truco del kernel"** usa funciones matemáticas para transformar los datos a una dimensión superior donde sí sean separables.

- **Radial (RBF):** El más popular y flexible. Usado en nuestro ejemplo.
- **Lineal:** Para datos ya separables linealmente.
- **Polinómico:** Para fronteras de decisión curvas.

CODIGO DE EJEMPLO :

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import numpy as np
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)


svm_rbf = SVC(kernel='rbf', gamma=0.7, C=1.0)

svm_rbf.fit(X_train, y_train)


y_pred = svm_rbf.predict(X_test)


print("Exactitud del modelo:", accuracy_score(y_test, y_pred))
print("\nReporte de clasificación:\n", classification_report(y_test,
y_pred))


x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

Z = svm_rbf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure(figsize=(8,6))
plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
plt.xlabel('Largo del sépalo')
plt.ylabel('Ancho del sépalo')
plt.title('Clasificación SVM con Kernel RBF')
plt.show()
```

4. Parámetros Importantes en el Código

```
SVC(kernel='rbf', gamma=0.7, C=1.0)
```

C (Regularización)
Controla el balance entre un margen ancho y pocos errores de clasificación. Un valor alto busca clasificar todo correctamente, arriesgando sobreajuste.

gamma (Parámetro del Kernel)
Define la influencia de un solo punto. Un valor alto hace que la influencia sea cercana, ajustándose más a los datos de entrenamiento.

5. Análisis del Código de Ejemplo

Desglosamos el script de Python paso a paso.

Paso 1 y 2: Cargar y Dividir Datos

```
# Usamos el dataset "Iris"
iris = datasets.load_iris()
# Tomamos 2 características para poder visualizar
X = iris.data[:, :2]
y = iris.target

# 70% para entrenar, 30% para probar
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

5. Análisis del Código (cont.)

Paso 3 y 4: Crear y Entrenar el Modelo

```
# Creamos el clasificador SVM con kernel radial
svm_rbf = SVC(kernel='rbf', gamma=0.7, C=1.0)

# El modelo aprende a separar las clases
svm_rbf.fit(X_train, y_train)
```

5. Análisis del Código (cont.)

Paso 5 y 6: Predecir y Evaluar

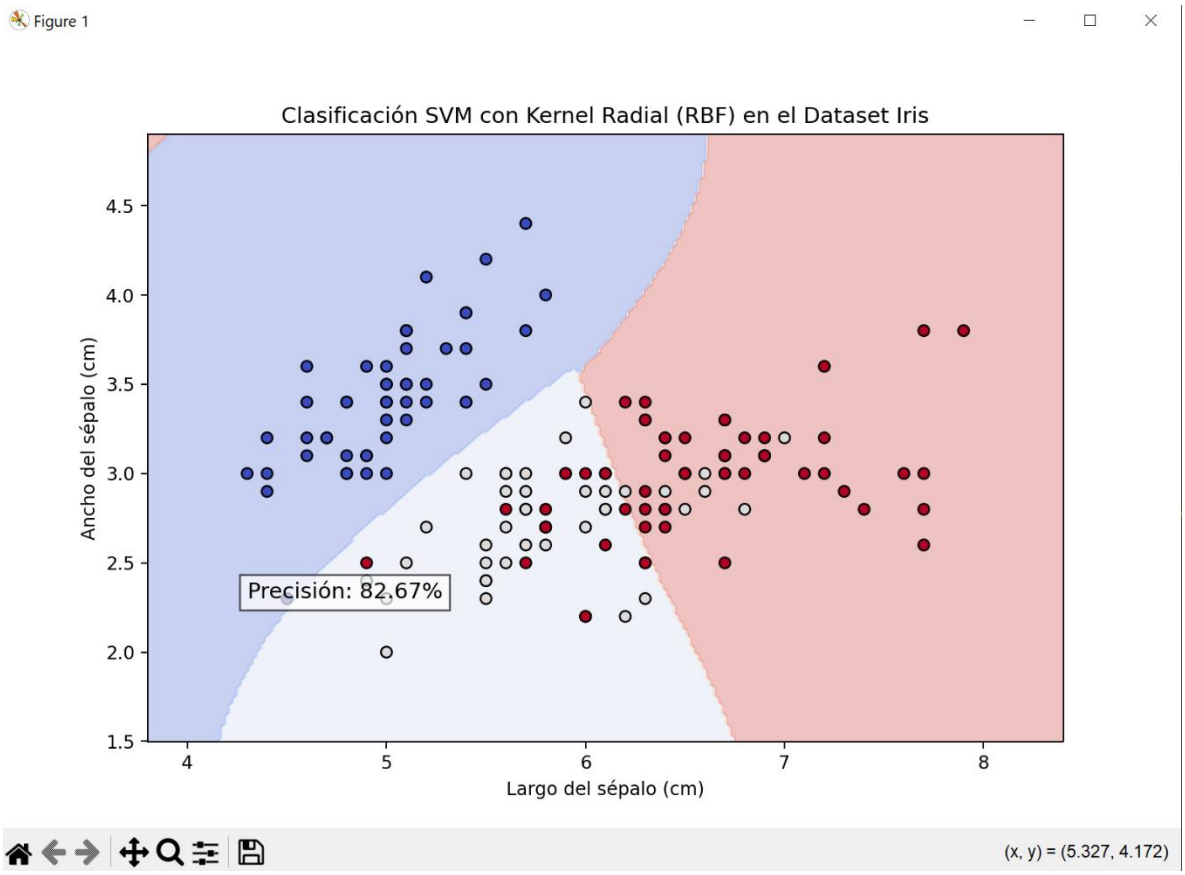
```
# Predecimos sobre los datos de prueba
y_pred = svm_rbf.predict(X_test)

# Comparamos predicciones con la realidad
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

5. Análisis del Código (cont.)

Paso 7: Visualizar Fronteras de Decisión

Esta es la gráfica resultante. Las áreas coloreadas muestran cómo el modelo clasificaría cada punto. Vemos que la clase 0 (morada) se separa bien, pero hay solapamiento entre la 1 (turquesa) y 2 (amarilla).



6. Interpretando los Resultados

Reporte de Clasificación

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.70      0.54      0.61        13
           2       0.62      0.77      0.69        13

    accuracy                           0.80        45
   macro avg       0.78      0.77      0.77        45
weighted avg       0.81      0.80      0.80        45
```

- **Accuracy (Exactitud) Total: 80%.** El modelo clasificó correctamente el 80% de las flores de prueba.
- **Clase 0:** Perfecta (100% en todas las métricas). Esto concuerda con la gráfica.
- **Clase 1 y 2:** Hay confusión entre ellas. Por ejemplo, para la clase 1, el `recall` es 0.54, lo que significa que el modelo solo encontró al 54% de las flores que realmente eran de esa clase.

7. Conclusión

- **SVM es un clasificador muy eficaz**, especialmente en espacios de alta dimensión.
- Funciona bien incluso si el número de dimensiones es mayor que el número de muestras.
- El **truco del kernel** lo hace increíblemente flexible para datos no lineales.
- La elección correcta de los parámetros **C** y **gamma** es crucial para un buen rendimiento.