INSTITUTO TECNOLOGICO DE CULIACAN

CARRERA INGENIERIA EN SISTEMAS COMPUTACIONALES

TECNOLÓGICO
NACIONAL DE MÉXICO

Materia: Inteligencia Artificial

Hora:18-19

Nombre de Alumno: ABEL ALEJANDRO MARTÍNEZ LIZÁRRAGA

Nombre del docente: JOSE MARIO RIOS FELIX

Tarea 5- Clasificación de SVM, función radial

Culiacán Sinaloa,2 de Octubre de 2025.

Máquina de Vectores de Soporte

son algoritmos de aprendizaje supervisado que clasifican datos encontrando el hiperplano óptimo para separar las clases, maximizando la distancia entre ellas

Características

Hiperplano óptimo:

El objetivo es encontrar el hiperplano que maximice el margen, la distancia entre el hiperplano y los vectores de soporte.

Vectores de soporte:

Son los puntos de datos más cercanos al hiperplano de decisión y son los que definen la frontera.

Truco del kernel:

Para datos no linealmente separables, se utiliza una técnica de kernel para transformar los datos a un espacio de mayor dimensión donde sí puedan ser linealmente separables

```
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
h=0.02
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

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
```

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

Kernel defiene como el modelo separa los datos

- Linear: frontera recta
- Poly: frontera polinomial
- Rbf: frontera curva,flexible
- Sigmoid: tipo red neuronal

El kernel rbf permite encontrar frontera de decisión muy flexibles, útil cuando los datos no son linealmente separables.

Basicamente al SVM se le dice que use una frontera curva y adaptable

Gamma

Controla que tan lejos llega la influencia de un solo punto,

- Gamma alto: el modelo de ajusta mucho a cada punto
- Gamma bajo: el modelo es mas ligth y generaliza mas

```
gamma=0.7
Significa un influencia moderada. No muy ligth ni tampoco
exageradamente sensible
```

C=1.0: Controla:

- Magen ancho df sdf
- Clasificar bien cada punto
- C alto: el modelo intenta clasificar TODO perfecto(riesgo de sobreajuste)
- C bajo: permite mas errores pero crea un margen mas amplio

Analisis del código

1. Importaciones

```
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
```

Aquí solo estás trayendo:

- el dataset Iris,
- funciones para dividir datos,
- el modelo SVM,
- métricas para evaluar,
- herramientas para graficar y manejar arrays.

2. Cargar el dataset Iris

```
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
```

- iris.data contiene 4 características por flor.
- Al poner [:, :2] nos quedamos solo con las dos primeras:

1. largo del sépalo
2. ancho del sépalo

3. Dividir en entrenamiento y prueba

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

- 70%: entrenamiento
- 30%: prueba
- random_state=42 para que todos tengan el mismo resultado siempre

4. **Crear el modelo SVM**

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

- **kernel RBF**: frontera curva
- **gamma=0.7**: sensibilidad moderada
- **C=1.0**: buen equilibrio entre exactitud y margen

5. **Entrenar modelo**

```
svm_rbf.fit(X_train, y_train)
```

Aquí el SVM aprende dónde están las clases usando solo los datos de entrenamiento.

6. **Predicciones**

```
y_pred = svm_rbf.predict(X_test)
```

El modelo intenta adivinar las clases de los datos de prueba.

7. **Crear la malla para graficar la frontera**

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h=0.02
```

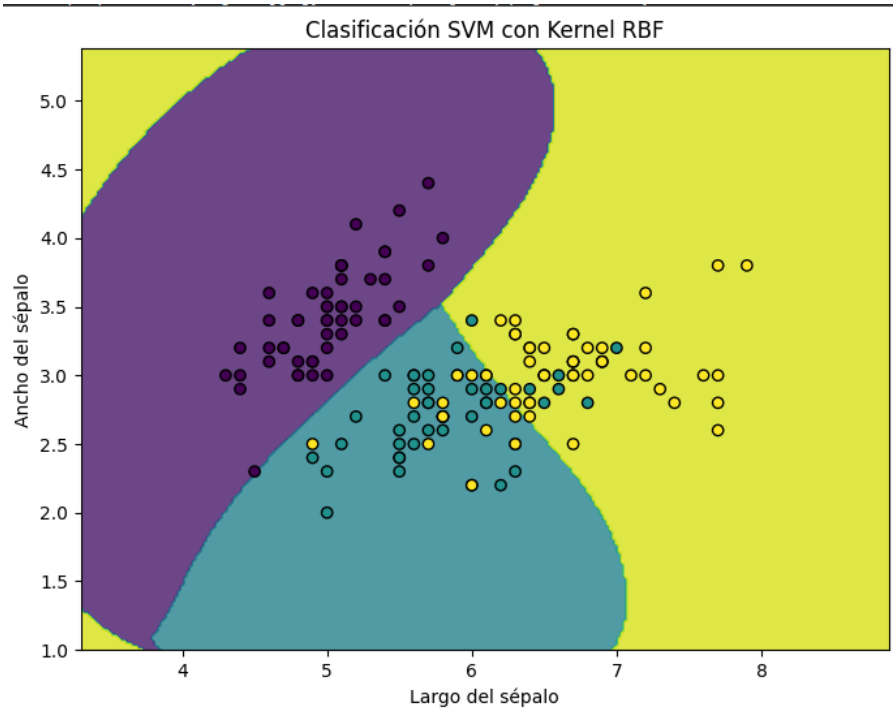
Defines los límites del plano y la resolución (h) de la malla, esto genera un "tablero" de puntos donde el modelo va a predecir la clase

8. Predecir sobre toda la malla

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

Aquí el modelo clasifica cada punto de ese tablero.
Luego se reacomoda para poder graficarlo.

8. **Grafica**



Esta gráfica muestra cómo el modelo SVM divide el espacio según las dos características usadas (largo y ancho del sépalo).
Cada color de fondo representa la zona donde el modelo clasificaría una flor:

- **Morado:** Clase 0
- **Turquesa:** Clase 1
- **Amarillo:** Clase 2

Los puntos encima son las flores reales del dataset.

Se observa que:

- **La clase 0 (morada) está muy bien separada**. No se mezcla con las otras clases.
- **Las clases 1 y 2 sí se solapan**, especialmente en la parte central, donde sus puntos caen en zonas contrarias al color de fondo.
- La forma curva de las regiones indica que el kernel RBF está intentando "ajustarse" a la forma real de los datos, pero con solo dos características la separación no es perfecta.

Esto explica visualmente por qué el modelo no alcanza el 100% de acierto.

10. Resultados

```
Exactitud del modelo: 0.8

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
```

Exactitud del modelo: 0.80 El modelo acierta el 80% de las veces.

Clase 0:

- Tiene valores perfectos porque está completamente separada de las otras.
- El modelo nunca se equivoca al clasificar estas flores.

Clase 1:

- El modelo falla más en esta clase.
- Detecta solo un poco más de la mitad de los ejemplos reales.
- Se confunde con la clase 2 debido al solapamiento en la gráfica.

Clase 2:

- Tiene mejor detección que la clase 1 (mayor recall),
  pero también toma puntos de la clase 1 como si fueran de la 2 (menor precision).
- También está afectada por el solapamiento central.

Resumen mental rápido:
El modelo funciona bien para la clase 0,
se defiende entre la 1 y la 2,
y la gráfica confirma por que sus áreas están mezcladas y esa mezcla se refleja en los números.