TALENTO TECH

TIC

Lección 3

Máquinas de Soporte Vectorial (SVM)

Universidad Tecnológica de Pereira

faceIT

Contenido

1. Introducción a las Máquinas de Soporte Vectorial (SVM)
2. Máquinas de Soporte Vectorial en Clasificación
3. Máquinas de soporte vectorial en regresión
4. Aplicación de Máquinas de Soporte Vectorial en Sklearn

1. Introducción a las Máquinas de Soporte Vectorial (SVM)

Las Máquinas de Soporte Vectorial (SVM) son un poderoso algoritmo de aprendizaje supervisado que se utiliza tanto para problemas de clasificación como de regresión.

En la clasificación

Busca encontrar el hiperplano óptimo que separa las clases en el espacio de características.

En la regresión

Busca encontrar la función que mejor se ajusta a los datos.

SVM es particularmente efectivo en conjuntos de datos que tienen una separación clara entre las clases o patrones no lineales que pueden ser transformados en espacios de características de mayor dimensión.

• Concepto de márgenes y cómo SVM busca maximizarlos para mejorar la separación entre clases:

Los márgenes en SVM se refieren a la distancia perpendicular entre el hiperplano de separación y los puntos más cercanos de las clases. SVM busca maximizar estos márgenes para encontrar la mejor separación entre las clases. Esto se debe a que un margen grande indica una mayor confianza en la clasificación/regresión, mientras que un margen pequeño puede indicar sobreajuste.

Para maximizar los márgenes, SVM utiliza técnicas de optimización que implican la minimización de la norma del vector de pesos, sujeto a ciertas restricciones. En otras palabras, SVM busca el hiperplano que maximiza la distancia entre las observaciones de las clases más cercanas, lo que proporciona una clasificación/regresión más robusta y generalizable. Esta optimización se realiza utilizando multiplicadores de Lagrange para formular el problema de optimización de SVM, lo que garantiza la eficiencia y la precisión en la separación de las clases.

Clasificación con SVM utilizando Scikit-learn

```python
from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

# Datos ficticios para clasificación

X_classification = [[0, 0], [1, 1], [1, 0], [0, 1]]

y_classification = [0, 1, 1, 0]

# Dividir el conjunto de datos en entrenamiento y prueba

X_train, X_test, y_train, y_test = train_test_split(X_classification,

y_classification, test_size=0.3, random_state=42)
```

```python
# Crear y entrenar un modelo SVM para clasificación

svm_classifier = SVC(kernel='linear')

svm_classifier.fit(X_train, y_train)

# Predecir las etiquetas para el conjunto de prueba

y_pred = svm_classifier.predict(X_test)

# Calcular la precisión del modelo

accuracy = accuracy_score(y_test, y_pred)

print("Precisión del modelo SVM para clasificación:", accuracy)
```

Visualización de la Frontera de Decisión de SVM

```
import numpy as np

import matplotlib.pyplot as plt

# Datos ficticios para visualización de la frontera de decisión

np.random.seed(0)

X_visualization = np.random.randn(200, 2)

y_visualization = np.logical_xor(X_visualization[:, 0] > 0,

X_visualization[:, 1] > 0)

# Entrenar un modelo SVM

svm_classifier_visualization = SVC(kernel='linear')

svm_classifier_visualization.fit(X_visualization, y_visualization)
```

```
# Crear una malla para visualizar la frontera de decisión

xx, yy = np.meshgrid(np.linspace(-3, 3, 500),

                     np.linspace(-3, 3, 500))

Z = svm_classifier_visualization.predict(np.c_[xx.ravel(),

yy.ravel()])

# Visualizar la frontera de decisión

plt.contourf(xx, yy, Z.reshape(xx.shape), alpha=0.4)

plt.scatter(X_visualization[:, 0], X_visualization[:, 1],

c=y_visualization, s=20, edgecolors='k')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('SVM Decision Boundary')

plt.show()
```

SVM para Regresión

```
from sklearn.datasets import make_regression

from sklearn.svm import SVR

from sklearn.metrics import mean_squared_error

# Datos ficticios para regresión

X_regression, y_regression = make_regression(n_samples=100,

n_features=1, noise=10, random_state=42)

# Dividir el conjunto de datos en entrenamiento y prueba

X_train, X_test, y_train, y_test = train_test_split(X_regression,

y_regression, test_size=0.2, random_state=42)
```

```
# Crear y entrenar un modelo SVM para regresión

svm_regressor = SVR(kernel='linear')

svm_regressor.fit(X_train, y_train)

# Predecir valores para el conjunto de prueba

y_pred = svm_regressor.predict(X_test)

# Calcular el error cuadrático medio

mse = mean_squared_error(y_test, y_pred)

print("Error cuadrático medio del modelo SVM para regresión:", mse)
```

2. Máquinas de Soporte Vectorial en Clasificación

Son un algoritmo de aprendizaje supervisado que se utiliza ampliamente en problemas de clasificación. La idea central detrás de SVM es encontrar el hiperplano de separación óptimo que maximiza el margen entre las clases en un espacio multidimensional. El margen se define como la **distancia perpendicular entre el hiperplano y los puntos de datos más cercanos de cada clase**. SVM busca encontrar el hiperplano que maximiza este margen, lo que se traduce en una mejor generalización y capacidad de clasificación.

- **Uso de hiperplanos para separar clases en un espacio multidimensional:**

En un problema de clasificación binaria, un hiperplano es una superficie (n-1)-dimensional que divide el espacio en dos partes, una para cada clase. Para casos más complejos con más de dos clases, SVM utiliza múltiples hiperplanos para separar las clases en un espacio multidimensional. Estos hiperplanos se eligen de tal manera que maximizan el margen entre las clases y minimizan el error de clasificación.

• Entrenamiento de un modelo SVM para clasificación

```
from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

# Generar datos de clasificación ficticios

X, y = make_classification(n_samples=100, n_features=2, n_classes=2,
random_state=42)

# Dividir el conjunto de datos en entrenamiento y prueba

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
# Crear y entrenar un modelo SVM para clasificación

svm_classifier = SVC(kernel='linear')

svm_classifier.fit(X_train, y_train)

# Predecir las etiquetas para el conjunto de prueba

y_pred = svm_classifier.predict(X_test)

# Calcular la precisión del modelo

accuracy = accuracy_score(y_test, y_pred)

print("Precisión del modelo SVM para clasificación:", accuracy)
```

Visualización de la Frontera de Decisión

```
import numpy as np

import matplotlib.pyplot as plt

# Generar datos ficticios para visualización

np.random.seed(0)

X_visualization = np.random.randn(200, 2)

y_visualization = np.logical_xor(X_visualization[:, 0] > 0,
X_visualization[:, 1] > 0)

# Entrenar un modelo SVM

svm_classifier_visualization = SVC(kernel='linear')

svm_classifier_visualization.fit(X_visualization, y_visualization)
```

```
# Crear una malla para visualizar la frontera de decisión

xx, yy = np.meshgrid(np.linspace(-3, 3, 500),

                     np.linspace(-3, 3, 500))

Z = svm_classifier_visualization.predict(np.c_[xx.ravel(),
yy.ravel()])

# Visualizar la frontera de decisión

plt.contourf(xx, yy, Z.reshape(xx.shape), alpha=0.4)

plt.scatter(X_visualization[:, 0], X_visualization[:, 1],
c=y_visualization, s=20, edgecolors='k')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('SVM Decision Boundary')

plt.show()
```

Experimentación con diferentes kernels

```
# Crear y entrenar modelos SVM con diferentes kernels
svm_classifier_rbf = SVC(kernel='rbf')
svm_classifier_poly = SVC(kernel='poly')
svm_classifier_rbf.fit(X_train, y_train)
svm_classifier_poly.fit(X_train, y_train)


# Predecir etiquetas para el conjunto de prueba
y_pred_rbf = svm_classifier_rbf.predict(X_test)
y_pred_poly = svm_classifier_poly.predict(X_test)
```

```
# Calcular la precisión de los modelos
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
accuracy_poly = accuracy_score(y_test, y_pred_poly)


print("Precisión del modelo SVM con kernel RBF:", accuracy_rbf)
print("Precisión del modelo SVM con kernel Polinomial:",
accuracy_poly)
```

3. Máquinas de soporte vectorial en regresión

Son conocidas principalmente por su aplicación en problemas de clasificación, pero también pueden ser utilizadas eficazmente en problemas de regresión. En la regresión, SVM se utiliza para predecir valores numéricos en lugar de clasificar muestras en diferentes categorías. A diferencia de la clasificación, donde SVM busca encontrar el hiperplano que mejor separa las clases, en la regresión, SVM **busca ajustar una función que se aproxime a los datos con el menor error posible mientras mantiene un margen aceptable**.

Cómo SVM puede utilizarse para predecir valores numéricos en lugar de clases.

En problemas de regresión, SVM busca encontrar una función que minimice el error entre las predicciones y los valores reales de las muestras. Para lograr esto, SVM utiliza un enfoque similar al de la clasificación, pero en lugar de encontrar un hiperplano de separación, busca una función que pase cerca de la mayoría de los puntos de datos, mientras mantiene un margen definido. Esto se logra mediante el uso de vectores de soporte, que son los puntos de datos más cercanos a la función de regresión.

- **SVM para regresión:**

```python
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
import numpy as np


# Generar datos de regresión ficticios
X_regression, y_regression = make_regression(n_samples=100,
n_features=1, noise=10, random_state=42)


# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_regression,
y_regression, test_size=0.2, random_state=42)
```

```python
# Crear y entrenar un modelo SVM para regresión
svm_regressor = SVR(kernel='linear')
svm_regressor.fit(X_train, y_train)

# Predecir valores para el conjunto de prueba
y_pred = svm_regressor.predict(X_test)
# Calcular el error cuadrático medio
mse = mean_squared_error(y_test, y_pred)
print("Error cuadrático medio del modelo SVM para regresión:", mse)

# Visualizar resultados
plt.scatter(X_test, y_test, color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.xlabel('Feature')
plt.ylabel('Target')
plt.title('SVM Regression')
plt.show()
```

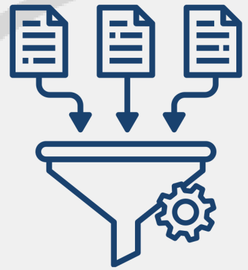
4. Aplicación de Máquinas de Soporte Vectorial (SVM) en Sklearn

Guía paso a paso para que aplicar Máquinas de Soporte Vectorial (SVM) en Scikit-learn a una base de datos tipo CSV descargada de plataformas como Kaggle o UCI Machine Learning Repository.

- **Procedimiento:**

1 Descargar y explorar los datos

Descarga el archivo CSV de la plataforma deseada y examina su contenido para comprender la estructura de los datos, las características disponibles y la variable objetivo que se desea predecir.



2 Preprocesamiento de datos

- Limpia los datos eliminando filas con valores faltantes o duplicados.
- Divide los datos en características (X) y la variable objetivo (y).

COLOMBIA POTENCIA DE LA VIDA

TIC

3 División de

Utiliza la función **train_test_split** de Scikit-learn para dividir los datos en conjuntos de entrenamiento y prueba. Esto permitirá evaluar el rendimiento del modelo en datos no vistos.

4 Escalamiento de

- Escala las características para asegurarte de que todas estén en la misma escala. Esto es
- especialmente importante para SVM. Utiliza el escalador **StandardScaler** de Scikit-learn para estandarizar las características.

TECH AZ PROYECTOS EDUCATIVOS UTP Universidad Tecnológica de Pereira

5 Selección de modelo

- Elige el tipo de SVM adecuado para tu problema.
  Para problemas de clasificación, puedes usar **SVC** (Support Vector Classifier) y para problemas de regresión, puedes usar
  SVR (Support Vector Regressor).

- Importa el modelo correspondiente de Scikit-learn.

6 Entrenamiento del modelo

- Ajusta el modelo a los datos de entrenamiento utilizando el método fit.

- Puedes ajustar diferentes hiperparámetros del modelo, como el tipo de kernel (lineal, polinómico, radial, etc.) y el parámetro de regularización (C), utilizando la validación cruzada u otras técnicas de optimización de hiperparámetros.

7 Evaluación del modelo

Utiliza métricas de evaluación adecuadas para medir el rendimiento del modelo en los datos de prueba. Para problemas de clasificación, puedes utilizar métricas como precisión, recall y F1-score. Para problemas de regresión, puedes utilizar el error cuadrático medio (MSE) o el coeficiente de determinación (R^2).

Visualiza los resultados del modelo utilizando gráficos como la matriz de confusión (para clasificación) o diagramas de dispersión de las predicciones vs. los valores reales (para regresión).

Siguiendo estos pasos, un estudiante puede aplicar con éxito Máquinas de Soporte Vectorial (SVM) en Scikit-learn a una base de datos tipo CSV descargada de plataformas de aprendizaje automático como Kaggle o UCI Machine Learning Repository.