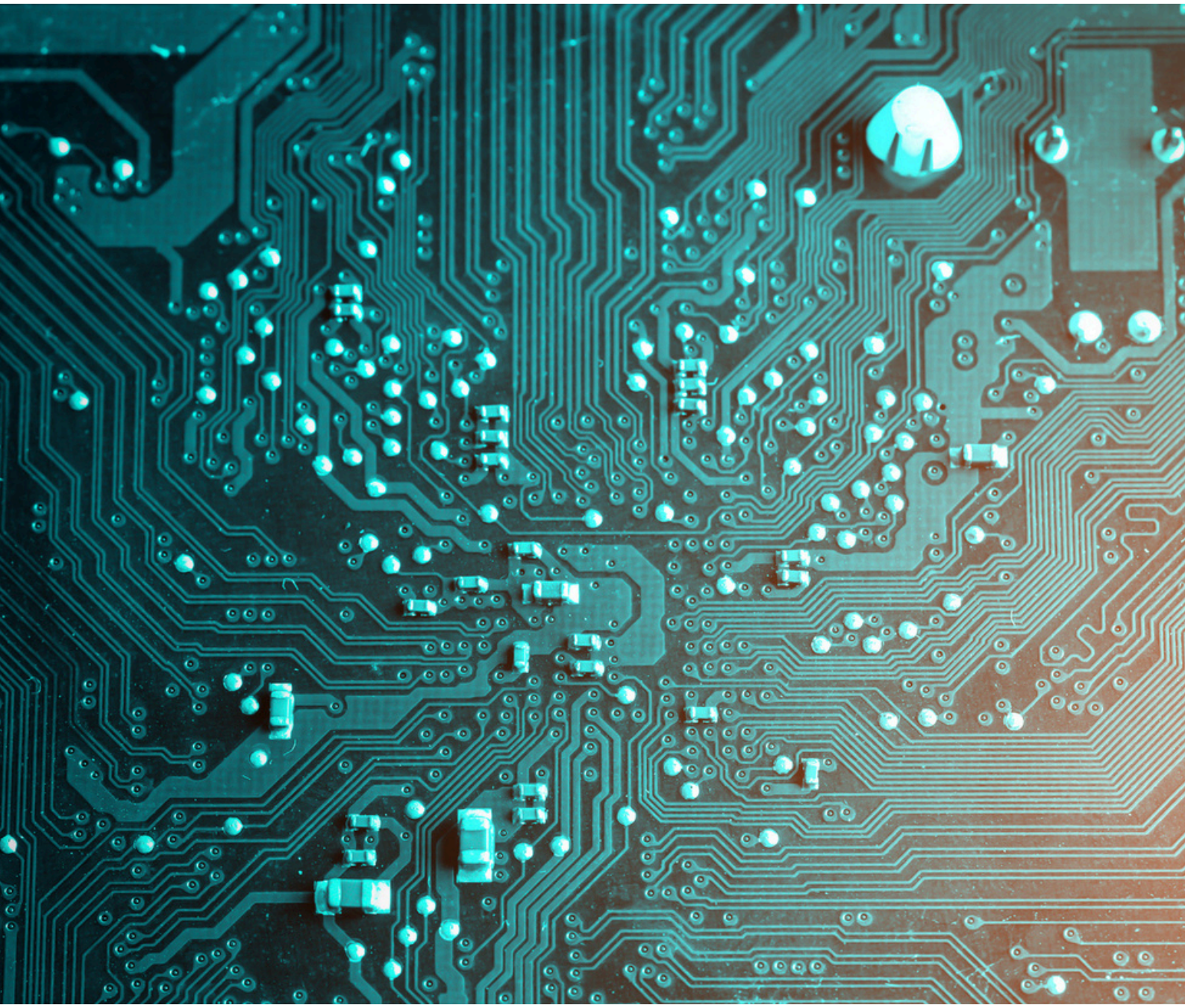
TIC



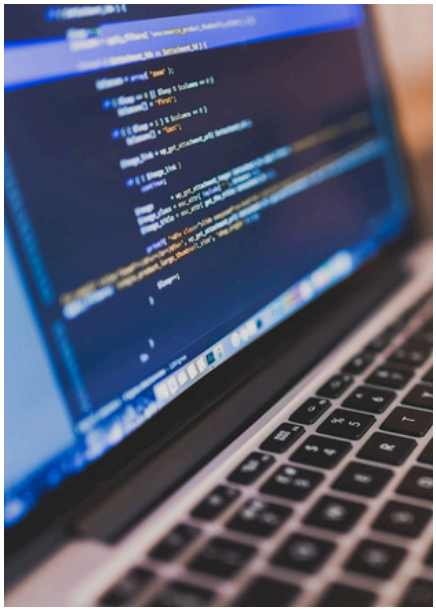
Máquinas de Soporte Vectorial

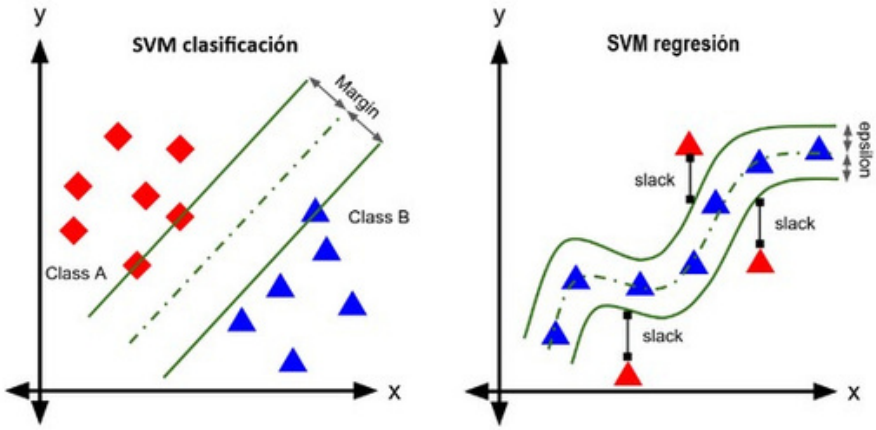
TALENTO TECH

UTP Universidad Tecnológica de Pereira | faceIT

Máquinas de Soporte Vectorial

Las SVM son un poderoso algoritmo de aprendizaje supervisado que se utiliza tanto para problemas de clasificación como de regresión. En la **clasificación**, SVM busca encontrar el hiperplano óptimo que separa las clases en el espacio de características, mientras que 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.

1 Máquinas de soporte vectorial en clasificación

Las Máquinas de Soporte Vectorial (SVM) 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

```python
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,
n_informative=2, n_redundant=0, n_clusters_per_class=1, 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)
```

2 Máquinas de soporte vectorial en 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

```
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)

# 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()
```