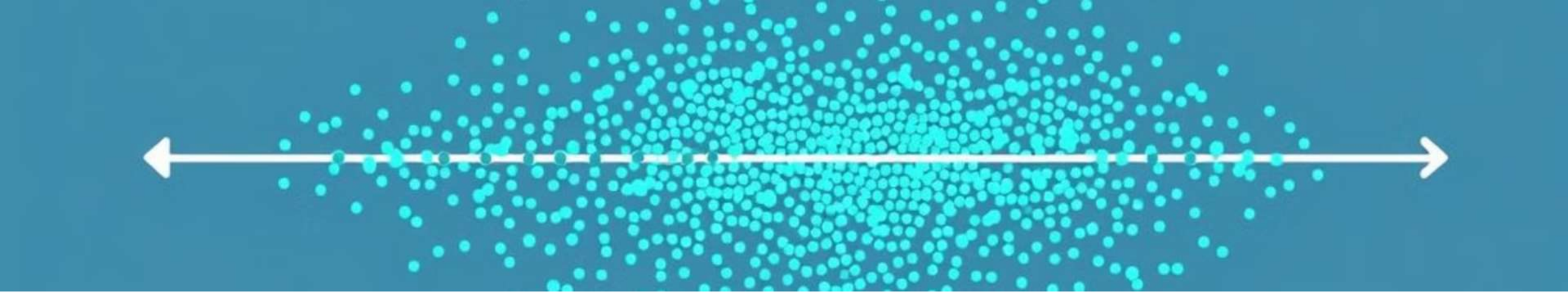
Aprendizaje Automático: Máquinas de Vector Soporte

Las Máquinas de Vector Soporte (SVM) son un poderoso algoritmo de aprendizaje automático utilizado para problemas de clasificación y regresión. Permiten encontrar un hiperplano óptimo que separa diferentes clases de datos con un margen máximo.



¿Qué son las Máquinas de Vector Soporte?

Algoritmo de Aprendizaje Supervisado

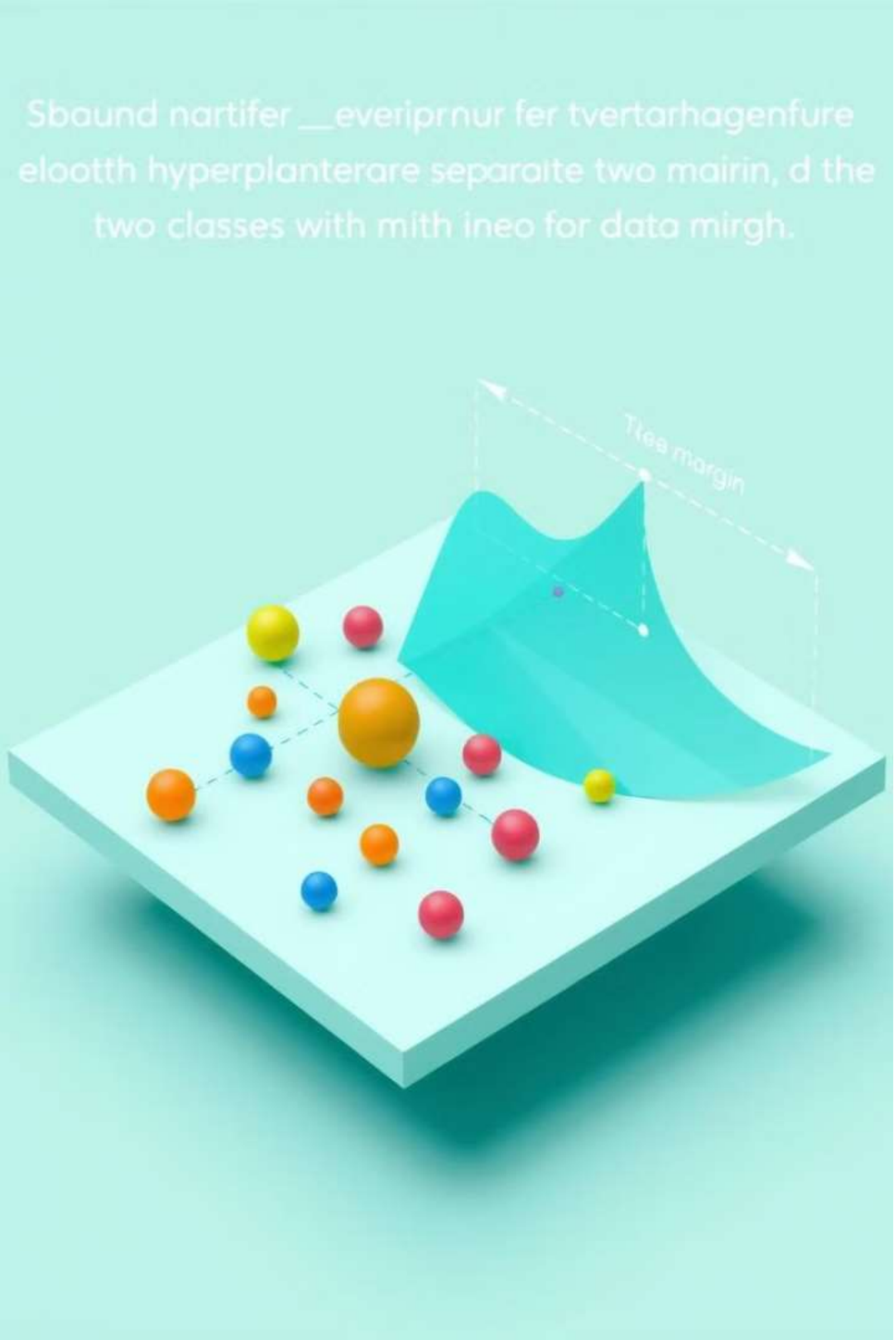
Las SVM aprenden a partir de datos etiquetados para construir modelos predictivos.

Enfoque de Margen Máximo

Buscan el hiperplano que maximiza la distancia a los puntos más cercanos de cada clase.

Flexibilidad y Versatilidad

Pueden manejar tanto problemas lineales como no lineales mediante el uso de kernels.



Principio de funcionamiento: Hiperplano Óptimo

1

Separación de Clases

Las SVM encuentran un hiperplano que separa los datos de diferentes clases con el mayor margen posible.

2

Puntos de Apoyo

Los vectores de soporte son los puntos más cercanos al hiperplano y definen el margen óptimo.

3

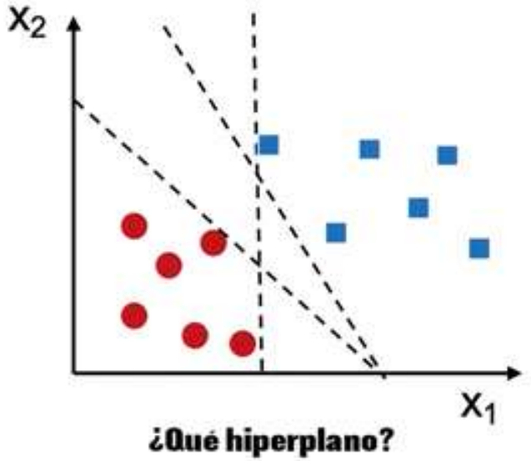
Optimización del Margen

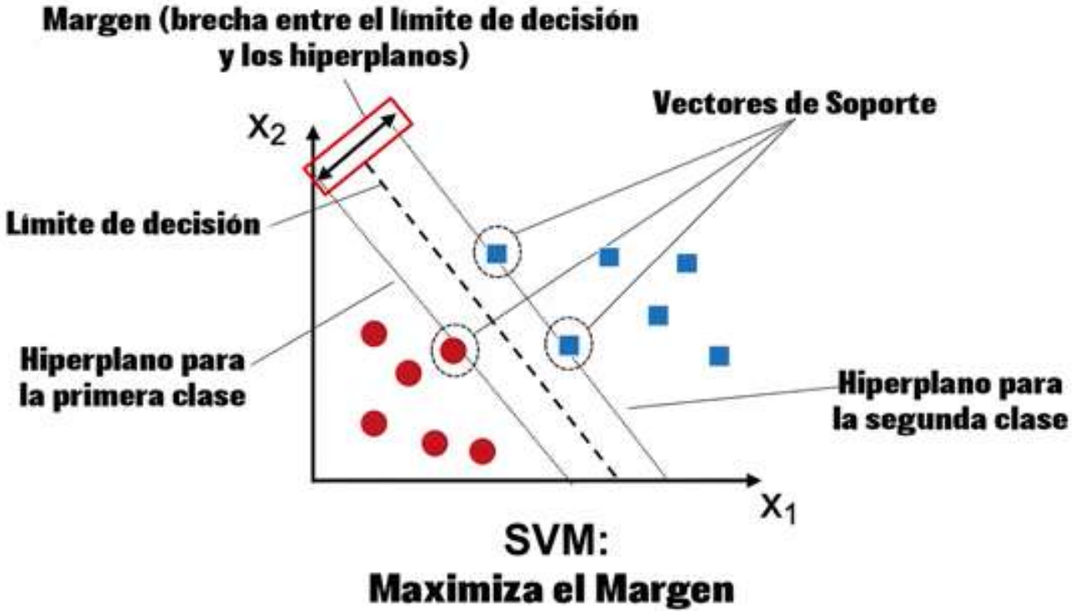
El algoritmo ajusta iterativamente el hiperplano para maximizar la distancia a los vectores de soporte.

¿Cómo funciona?

Separación de datos: SVM busca un hiperplano (en 2D es una línea, en 3D es un plano) que divida las clases de datos con el mayor margen posible.

Margen óptimo:
El margen es la distancia entre el hiperplano y los puntos más cercanos de cada clase (llamados vectores de soporte).
El objetivo es maximizar este margen para lograr una buena generalización en los datos.





Transformación no lineal (Kernel Trick):
Si los datos no son linealmente separables, SVM utiliza funciones kernel (como RBF o polinómicas) para transformar los datos a un espacio de mayor dimensión donde sí puedan ser separados linealmente.

Regularización:
Un parámetro llamado C controla el equilibrio entre maximizar el margen y permitir errores de clasificación en los datos.

Parámetros principales de SVC (Clasificación SVM):

1. C (float, default=1.0)
   Controla el margen de separación y el nivel de regularización.
   Valores altos de C: Penalizan más los errores de clasificación, priorizando un ajuste más estricto a los datos de entrenamiento. Esto puede llevar a *overfitting*.
   Valores bajos de C: Permiten más errores en los datos de entrenamiento, lo que resulta en márgenes más amplios y mejor generalización.
2. kernel (str, default='rbf')
   Especifica el tipo de función kernel para transformar los datos en un espacio de mayor dimensión si no son linealmente separables.
   Opciones:
      'linear': Hiperplano lineal.
      'poly': Kernel polinómico.
      'rbf': Kernel de función de base radial (Gaussian Kernel). Es el más común.
      'sigmoid': Función sigmoide.
      'precomputed': Utiliza una matriz de kernel precomputada.
   También puedes definir un kernel personalizado pasando una función.
3. degree (int, default=3)
      Grado del polinomio si el kernel es 'poly'.
4. gamma (str or float, default='scale')
   Controla la influencia de un único punto de dato en el modelo para kernels no lineales como 'rbf', 'poly' y 'sigmoid'. Opciones:
      'scale': 1 / (n_features * X.var()), ajusta gamma según el tamaño de las características.
      'auto': 1 / n_features, ajusta según el número de características.
      Valor flotante: Define el gamma manualmente.

Parámetros principales de SVC (Clasificación SVM):

5. coef0 (float, default=0.0)
Término independiente en kernels 'poly' y 'sigmoid'.

6. probability (bool, default=False)
Habilita la estimación de probabilidades (con model.predict_proba), pero puede ser más lento y consumir más memoria

7. shrinking (bool, default=True)
Si se utiliza heurística para acelerar el entrenamiento (generalmente no se cambia).

8. tol (float, default=1e-3)
Criterio de tolerancia para la convergencia.

9. class_weight (dict or 'balanced', default=None)
Asigna pesos a las clases para manejar problemas de clases desbalanceadas.
'balanced': Ajusta automáticamente los pesos según la frecuencia de las clases.

10. max_iter (int, default=-1)
Número máximo de iteraciones. Si se establece en -1, no hay límite.

```
# Importar las librerías necesarias
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
import numpy as np

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

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Crear un modelo SVM con kernel lineal
model = SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)

# Predecir en el conjunto de prueba
y_pred = model.predict(X_test)

# Evaluar el modelo
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nReporte de clasificación:\n", classification_report(y_test, y_pred))

# Visualización del hiperplano para las dos primeras clases
def plot_svm_decision_boundary(X, y, model):
    h = 0.02  # Paso en la malla
    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 = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8)

    # Dibujar los puntos
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
    plt.xlabel('Característica 1')
    plt.ylabel('Característica 2')
    plt.title('Frontera de decisión SVM')
    plt.show()

# Visualizar el resultado (Nota: funciona mejor si hay dos clases en y)
plot_svm_decision_boundary(X_train, y_train, model)
```

Ejemplo de ajuste de parámetros

Supongamos que tienes un conjunto de datos desbalanceado y necesitas usar un kernel 'rbf' con regularización personalizada. Puedes configurar así:

```python
from sklearn.svm import SVC

# Crear el modelo con parámetros ajustados
model = SVC(
    C=10,                # Penalización alta
    kernel='rbf',        # Kernel no lineal
    gamma='scale',       # Configuración automática
    class_weight='balanced'  # Ajustar pesos según el desbalance
)

# Entrenar el modelo
model.fit(X_train, y_train)
```

Nota: Puedes optimizar estos parámetros utilizando métodos como **búsqueda en cuadrícula** (GridSearchCV) o búsqueda aleatoria (RandomizedSearchCV) para encontrar la mejor combinación para tus datos.

Ventajas de las Máquinas de Vector Soporte

Alta Precisión

Las SVM son capaces de lograr altos niveles de precisión en problemas de clasificación.

Eficiencia Computacional

Su complejidad computacional es relativamente baja en comparación con otros métodos.

Manejo de Datos de Alta Dimensión

Pueden manejar eficazmente conjuntos de datos de alta dimensionalidad.

Aplicaciones en Clasificación y Regresión

1 Clasificación de Imágenes

Las SVM son ampliamente utilizadas en problemas de visión artificial, como el reconocimiento de objetos y dígitos.

2 Análisis de Texto

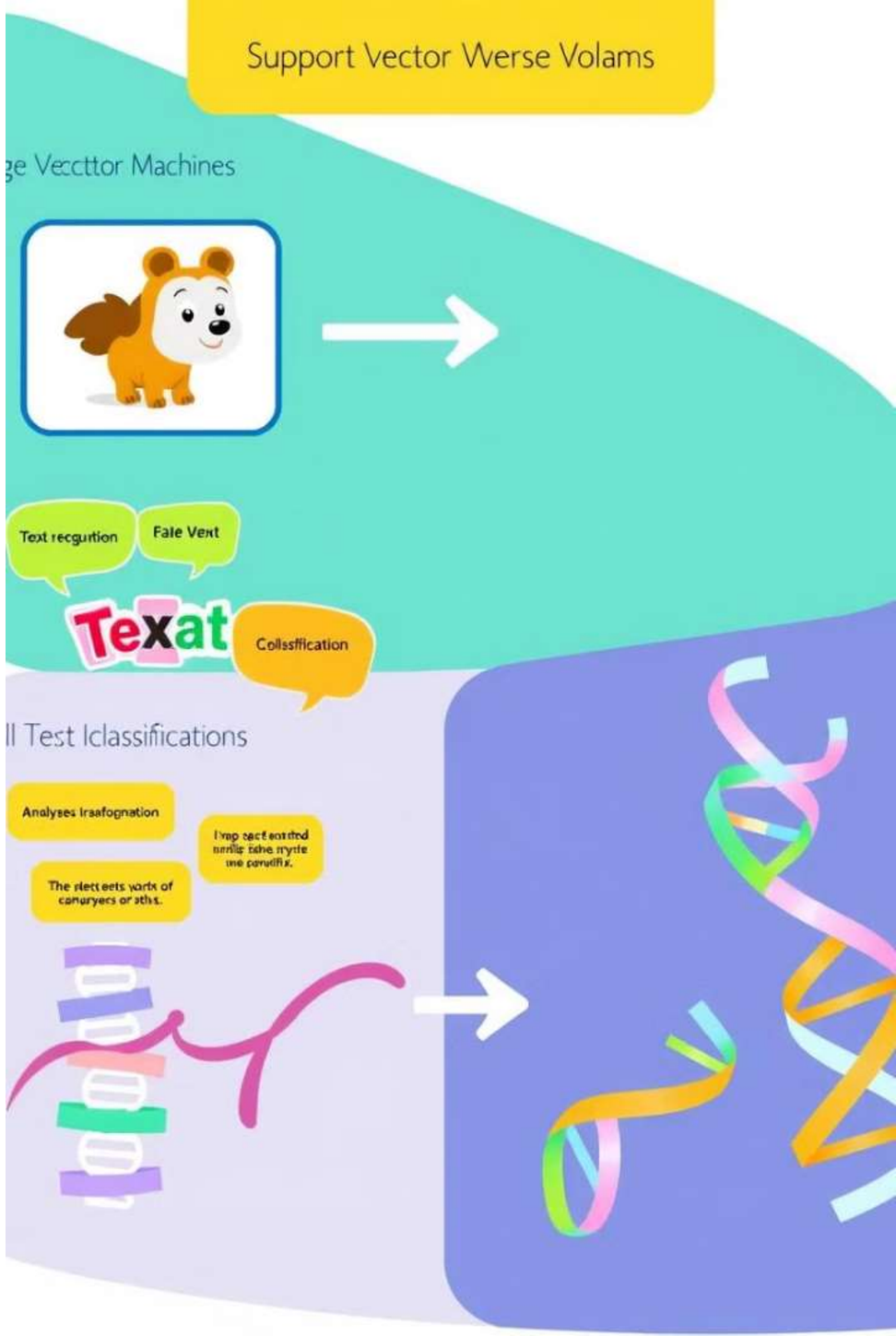
Se aplican en tareas de clasificación de texto, como detección de spam y análisis de sentimientos.

3 Bioinformática

Son útiles en el análisis de secuencias de ADN y la predicción de la estructura de proteínas.

4 Regresión

Las SVM también se utilizan en problemas de regresión, como la predicción de ventas o el pronóstico de series temporales.



Ejemplo: Clasificación de Imágenes de Dígitos

1. **Definir el Problema**

   Clasificar imágenes de dígitos manuscritos en 10 clases (0-9).

2. **Preprocesar los Datos**

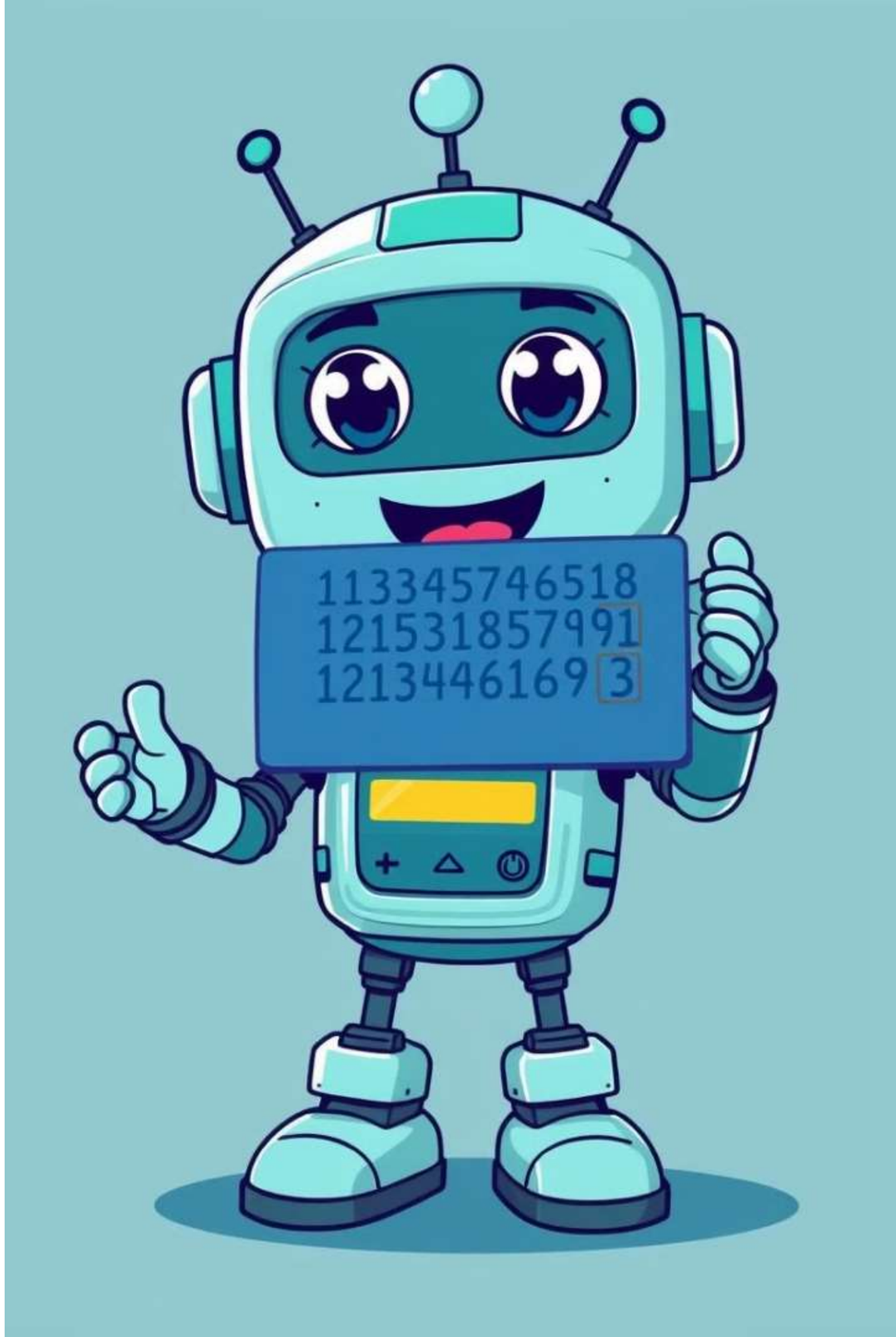
   Normalizar y escalar las imágenes de 28x28 píxeles.

3. **Entrenar el Modelo**

   Utilizar un kernel lineal y ajustar los hiperparámetros de regularización.

4. **Evaluar el Desempeño**

   Alcanzar una precisión superior al 97% en el conjunto de prueba.



Parámetros Clave y Ajuste del Modelo

Selección de Kernel

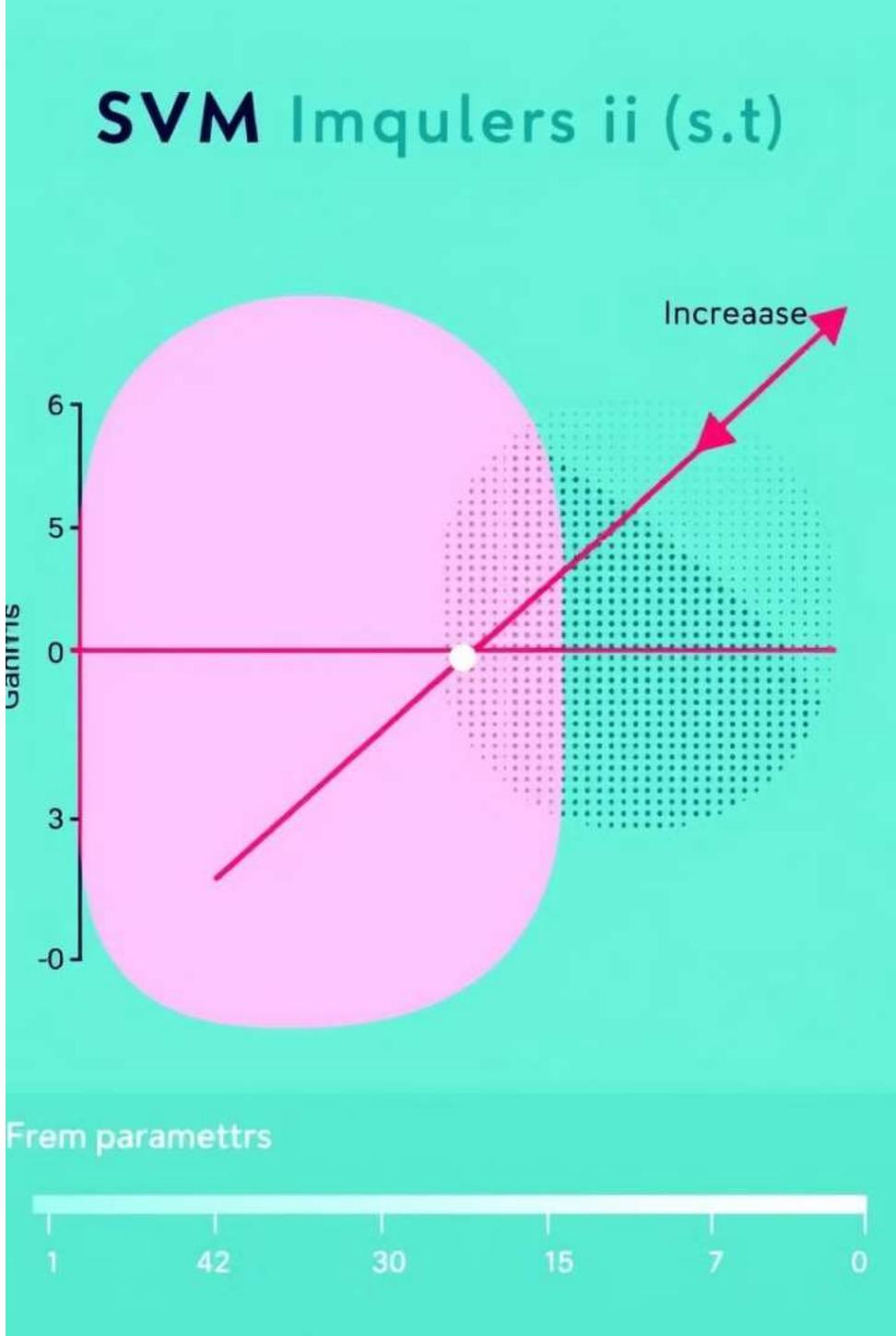
Elegir el kernel apropiado (lineal, polinomial, RBF, etc.) según la complejidad de los datos.

Parámetro de Regularización

Ajustar el parámetro C para balancear el margen y el error de entrenamiento.

Ajuste de Hiperparámetros

Optimizar los hiperparámetros del kernel (como gamma en RBF) mediante validación cruzada.



Conclusiones y Consideraciones Finales

1

Flexibilidad y Versatilidad

Las SVM pueden abordar una amplia gama de problemas de clasificación y regresión.

2

Precisión y Eficiencia

Ofrecen altos niveles de precisión con una complejidad computacional relativamente baja.

3

Ajuste de Hiperparámetros

El ajuste cuidadoso de los hiperparámetros es crucial para optimizar el rendimiento del modelo.

En resumen, las Máquinas de Vector Soporte son una poderosa herramienta del aprendizaje automático que han demostrado su efectividad en una amplia variedad de aplicaciones. Su enfoque de margen máximo y flexibilidad para manejar datos de alta dimensionalidad los convierten en una opción destacada para problemas de clasificación y regresión.