UNIDAD

7

DIPLOMATURA EN PYTHON APLICADO A LA CIENCIA DE DATOS

Máquinas de Soporte Vectorial

Unidad 7: Máquinas de Soporte Vectorial

- Contenido Teórico
- Instalación y uso del paquete
- Ejemplo en Python – Problema Concreto
- Principales parámetros de ajuste y control

Un poco de contenido teórico:

Maquinas de soporte vectorial

¿Qué es Máquina de Soporte Vectorial?

La máquina de Soporte Vectorial apareció en los años 60s y luego fue mejorada en los años 90s.

Pertenece a la categoría de algoritmos supervisados y se han vuelto muy reconocidas ya que se obtienen resultados muy eficientes.

Una máquina de soporte vectorial se implementa un poco diferente comparados con otros algoritmos de machine Learning.

Es capaz de hacer tareas de clasificación, regresión y detección de outliers.

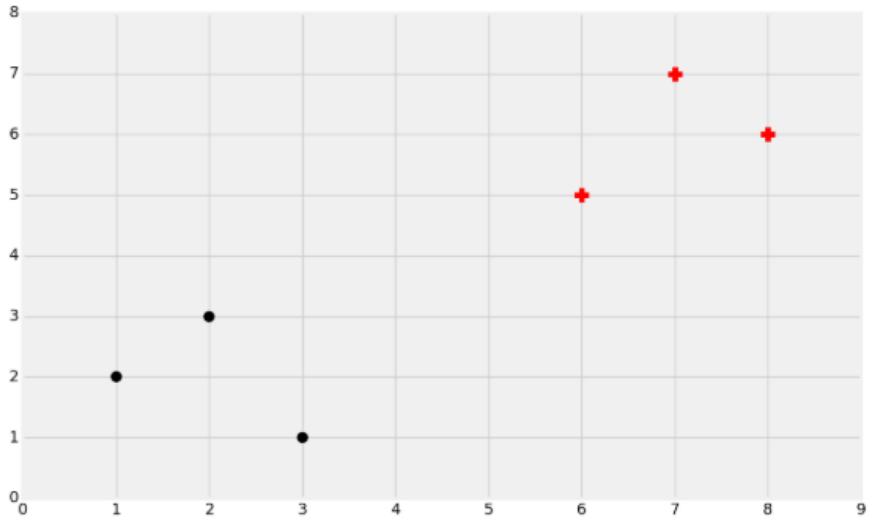
La máquina de Soporte Vectorial es un clasificador discriminativo diseñado formalmente por un hiperplano separativo.

Es una representación de ejemplos como puntos en el espacio que se asignan de manera que los puntos en diferentes categorías se separan por un espacio lo más amplio posible. Además de esto, un SVM también puede realizar una clasificación no lineal.

¿Cómo funciona una máquina de soporte vectorial?

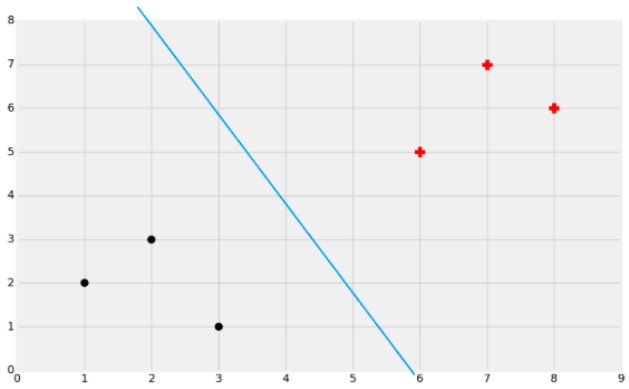
El objetivo principal de una máquina de soporte vectorial es segregar los datos dados de la mejor manera posible.

Cuando se realiza la segregación, la distancia entre los puntos más cercanos se conoce como el margen. El enfoque es seleccionar un hiperplano con el margen máximo posible entre los vectores de soporte en el dataset dado.

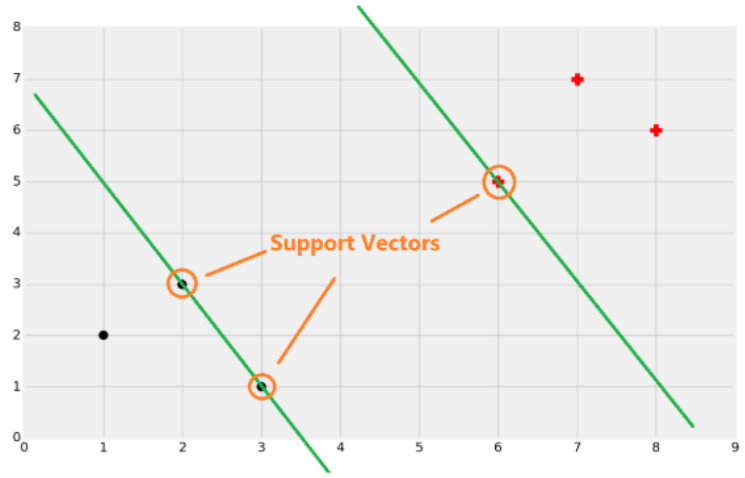


Comenzaremos con los datos anteriores. Notamos en el pasado que la intuición más común es que clasificaría un nuevo punto de datos en función de lo que está más cerca o

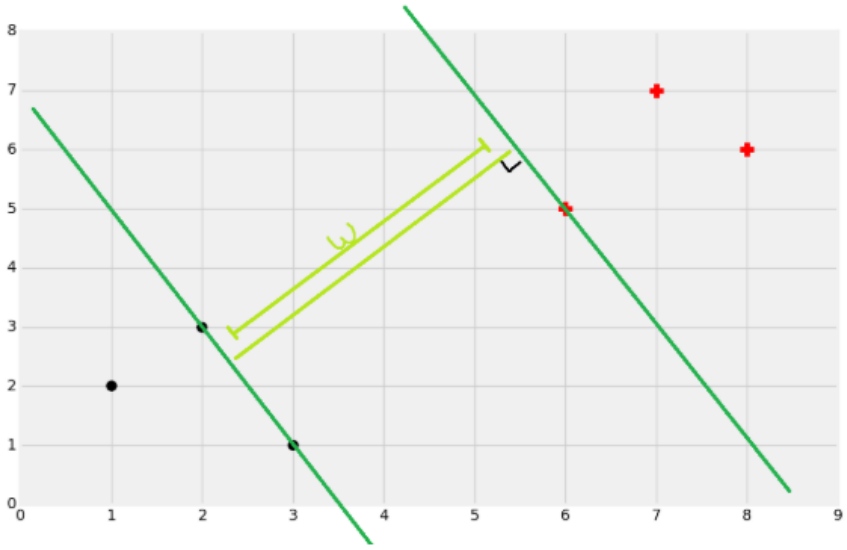
de la proximidad, que es lo que el algoritmo K Nearest Neighbours (Vecinos Cernanos) hace por nosotros.
El problema principal con este objetivo es que, por punto de datos, debe compararlo con cada punto de datos para obtener las distancias, por lo que el algoritmo simplemente no escala bien, a pesar de ser bastante confiable en cuanto a precisión. Lo que pretende hacer la máquina de vectores de soporte es, una vez, generar la línea de "mejor ajuste", pero en realidad un plano, y más específicamente un hiperplano, que divide mejor los datos.
Una vez que se descubre este hiperplano, nos referimos a él como un límite de decisión. Hacemos esto, porque este es el límite entre ser una clase u otra.
Una vez que calculamos este límite de decisión, nunca necesitamos volver a hacerlo, a menos que, por supuesto, volvamos a entrenar el conjunto de datos.
Por lo tanto, este algoritmo va a ser más rápido, a diferencia del clasificador Vecinos Cercanos.
La curiosidad es, por supuesto, ¿cómo descubrimos realmente el mejor hiperplano divisor? Bueno, continuamos con el gráfico.



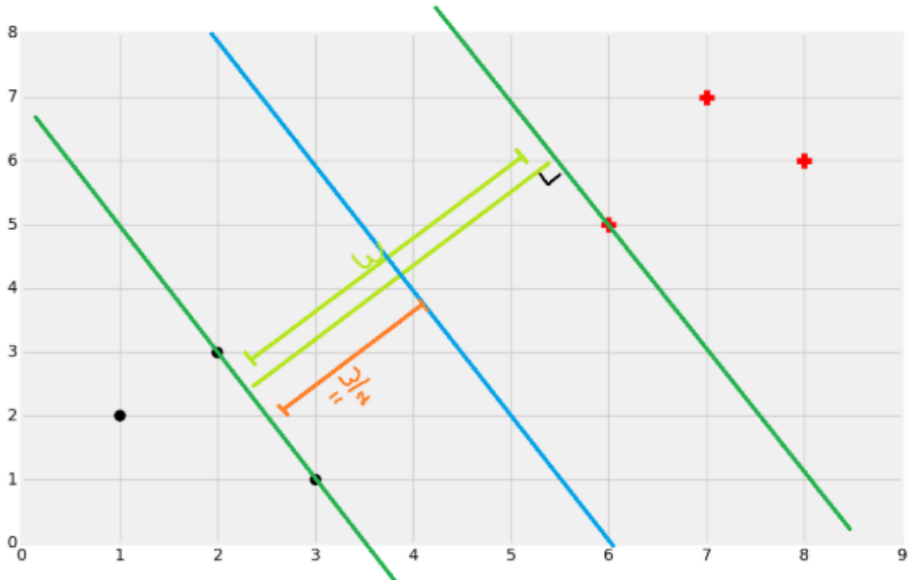
Probablemente sea lo correcto, pero, ¿cómo encontramos eso?
Bueno, primero se encuentran los vectores de soporte (Los vectores de soporte son los puntos que definen el margen máximo de separación del hiperplano que separa las clases). Se llaman vectores, en lugar de puntos, porque estos «puntos» tienen tantos elementos como dimensiones tenga nuestro espacio de entrada. Es decir, estos puntos multi-dimensionales se representan con un vector de *n* dimensiones.



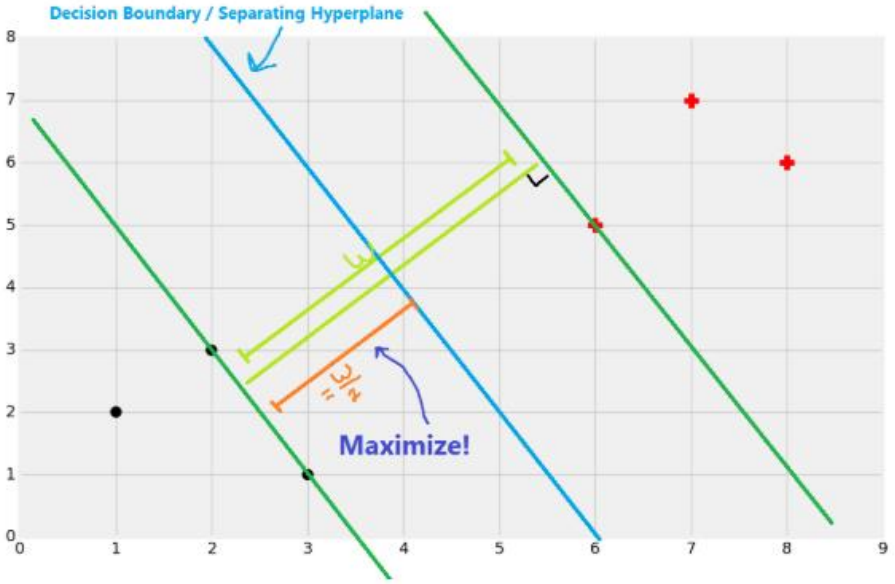
Una vez que encuentre los vectores de soporte, desea crear líneas que estén separadas entre sí al máximo. Desde aquí, podemos encontrar fácilmente el límite de decisión tomando el ancho total:



Dividiendo por 2:



Acá se encuentra el límite



Ahora, si un punto está a la izquierda del límite de decisión / hiperplano de separación, entonces decimos que es una clase de punto negro. Si está a la derecha, entonces es una clase de signo más rojo.

Vale la pena señalar, por supuesto, que este método de aprendizaje solo funcionará de forma nativa en datos separables linealmente.

En el caso que tengamos datos con el siguiente comportamiento:

¿Se puede crear un hiperplano de separación aquí? No.

Pasos que se recomiendan seguir para implementar SMV:

1. Cargar los datos
2. Explorar los datos
3. División de datos
4. Generar el modelo
5. Evaluación del modelo

Cómo lidiar con planos inseparables y no lineales

En algunos casos, los hiperplanos no pueden ser muy eficientes. En esos casos, la máquina de vectores de soporte utiliza un truco del núcleo para transformar la entrada en un espacio de dimensiones superiores.
Con esto, se hace más fácil segregar los puntos.

Núcleo SVM

Un núcleo SVM básicamente agrega más dimensiones a un espacio dimensional bajo para que sea más fácil segregar los datos. Convierte el problema inseparable en problemas separables al agregar más dimensiones utilizando el truco del kernel. Una máquina de soporte vectorial es implementada en la práctica por un núcleo.

El truco del núcleo ayuda a hacer un clasificador más preciso. A continuación los diferentes núcleos en la SVM.

Núcleo lineal: un núcleo lineal se puede usar como un producto de punto normal entre dos observaciones dadas. El producto entre los dos vectores es la suma de la multiplicación de cada par de valores de entrada. La siguiente es la ecuación lineal del núcleo.

$$f(x) = B(0) + sum(ai * (x,xi))$$

Núcleo polinómico: es una forma bastante generalizada del núcleo lineal. Puede distinguir el espacio de entrada curvo o no lineal. A continuación se muestra la ecuación del núcleo polinomial.

$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

b = Grado del Núcleo a = Constante

Kernel de función de base radial: el kernel de función de base radial se usa comúnmente en la clasificación SVM, puede mapear el espacio en dimensiones infinitas. La siguiente es la ecuación del núcleo RBF.

$$K(X_1, X_2) = exponent(-\gamma \|X_1 - X_2\|^2)$$

$||X_1-X_2||^2$ = Distancia Euclidiana

Ventajas de SVM

• Efectivo en espacios de altas dimensiones.
• Sigue siendo efectivo en casos donde el número de dimensiones es mayor que el número de muestras
• Utiliza un subconjunto de puntos de entrenamiento en la función de decisión que hace que la memoria sea eficiente
• Se pueden especificar diferentes funciones del kernel para la función de decisión que también la hace versátil

Desventajas de SVM

• Si el número de características es mucho mayor que el número de muestras, evite un ajuste excesivo al elegir las funciones del núcleo y el término de regularización es crucial.

• Los SVM no proporcionan directamente estimaciones de probabilidad, estas se calculan utilizando una validación cruzada de cinco veces.

¿Cómo invoco a este algoritmo en Python? – Uso del Paquete

from sklearn import svm

Ejemplo en Python: - Problema Concreto

Dada los datos de cáncer de mamas se determina si el cáncer es benigno p maligno.
0 es benigno y 1 es maligno. También se determina la precisión del modelo.

```
from sklearn import datasets

cancer_data = datasets.load_breast_cancer()
print(cancer_data.data[5])
print(cancer_data.data.shape)
#target set
print(cancer_data.target)
from sklearn.model_selection import train_test_split

cancer_data = datasets.load_breast_cancer()

X_train, X_test, y_train, y_test = train_test_split(cancer_data.data, cancer_data.target,
test_size=0.4,random_state=109)
from sklearn import svm
#create a classifier
cls = svm.SVC(kernel="linear")
#train the model
cls.fit(X_train,y_train)
#predict the response
pred = cls.predict(X_test)
print(pred)
from sklearn import metrics
#accuracy
print("acuracy:", metrics.accuracy_score(y_test,y_pred=pred))
#precision score
print("precision:", metrics.precision_score(y_test,y_pred=pred))
#recall score
print("recall" , metrics.recall_score(y_test,y_pred=pred))
print(metrics.classification_report(y_test, y_pred=pred))
```

Ejemplos donde se usa Máquina de Soporte Vectorial

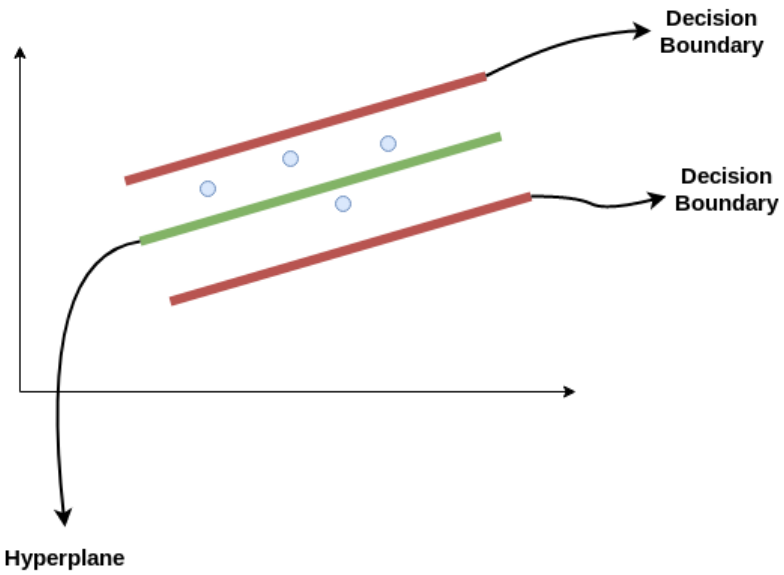
- Detección de rostro
- Categorización de texto e hipertexto
- Clasificación de imágenes
- bioinformática
- Pliegue de proteínas y detección remota de homología
- Reconocimiento de escritura a mano
- Control predictivo generalizado

Link en Librería de Python:

https://scikit-learn.org/stable/modules/svm.html

Máquina de Soporte Vectorial Regresión:

Support Vector Regression (SVR) utiliza el mismo principio que SVM, pero para problemas de regresión.
El problema de la regresión es encontrar una función que se aproxime al mapeo de un dominio de entrada a números reales sobre la base de una muestra de entrenamiento. Así que ahora profundicemos y comprendamos cómo funciona SVR en realidad.



Considere estas dos líneas rojas como el límite de decisión y la línea verde como el hiperplano. Nuestro objetivo, cuando avanzamos con SVR, es básicamente considerar los puntos que están dentro de la línea límite de decisión. Nuestra mejor línea de ajuste es el hiperplano que tiene un número máximo de puntos.

Suponiendo que la ecuación del hiperplano es la siguiente:

$$Y = wx+b \text{ (Ecuación del Hiperplano)}$$

Entonces las ecuaciones del límite de decisión se convierten en:

$$wx+b= +a$$

$$wx+b= -a$$

Con lo cual podemos decir lo siguiente:

-a < Y- wx+b < +a

Repasemos: Nuestro objetivo principal aquí es decidir un límite de decisión a una distancia "a" del hiperplano original de modo que los puntos de datos más cercanos al hiperplano o los vectores de soporte estén dentro de esa línea de límite.

Por lo tanto, vamos a tomar solo aquellos puntos que están dentro del límite de decisión y tienen la menor tasa de error, o están dentro del Margen de Tolerancia. Esto nos da un mejor modelo de ajuste.

¿Cómo invoco a este algoritmo en Python? – Uso del Paquete

```
from sklearn.svm import SVR
```

Link in libería de Python
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html?highlight=svm#sklearn.svm.SVR

Máquina de Soporte Vectorial Outliers:

Los problemas de aprendizaje desequilibrados a menudo obstaculizan a los nuevos para tratar con ellos.
Cuando la proporción entre clases en sus datos es 1: 100 o mayor, los primeros intentos de modelar el problema son recompensados con una precisión muy alta pero una especificidad muy baja.

- Se puede resolver el problema de especificidad en el aprendizaje desequilibrado de diferentes maneras:
- Se puede sopesar ingenuamente las clases, haciendo que su modelo sea preferencial a la clase minoritaria.
- Se puede usar submuestreo, sobremuestreo o una combinación de ambos.
- Se puede cambiar su objetivo de tratar de equilibrar el conjunto de datos, a tratar de predecir la clase minoritaria utilizando técnicas de detección de valores atípicos.

Un ejemplo de estos casos puede ser detección de Fraude.

¿Cómo invoco a este algoritmo en Python? – Uso del Paquete

```
from sklearn.svm import OneClassSVM
```

Link en Libería de Python:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html