SVM

Una máquina de vectores de soporte (SVM) es un clasificador discriminativo definido formalmente por un hiperplano de separación. En otras palabras, dados los datos de entrenamiento etiquetados ( aprendizaje supervisado ), el algoritmo genera un hiperplano óptimo que categoriza nuevos ejemplos.

El funcionamiento del algoritmo SVM se basa en encontrar el hiperplano que dé la mayor distancia mínima a los ejemplos de entrenamiento. Esta distancia recibe dos veces el importante nombre de margen dentro de la teoría de SVM. Por lo tanto, el hiperplano de separación óptimo maximiza el margen de los datos de entrenamiento.

import cv2 as cv  
import numpy as np  
  
# Los datos de entrenamiento de este ejercicio están formados por un conjunto de puntos 2D   
#etiquetados que pertenecen a una de dos clases diferentes; una de las clases consta de un punto y la otra de tres puntos.  
labels = np.array([1, -1, -1, -1])  
trainingData = np.matrix([[501, 10], [255, 10], [501, 255], [10, 501]], dtype=np.float32)  
  
# Train the SVM  
  
#cuando los ejemplos de entrenamiento se dividen en dos clases que son linealmente separables.   
#Sin embargo, las SVM se pueden utilizar en una amplia variedad de problemas   
#(por ejemplo, problemas con datos separables no linealmente, una SVM que utiliza una función del núcleo para aumentar la dimensionalidad de los ejemplos, etc.).  
#Como consecuencia de esto, tenemos que definir algunos parámetros antes de entrenar el SVM. Estos parámetros se almacenan en un objeto de la clase cv.ml.SVM() .  
svm = cv.ml.SVM\_create()  
  
#Tipo de SVM . Elegimos aquí el tipo C\_SVC que se puede utilizar para la clasificación de n clases (n≥2).   
#La característica importante de este tipo es que se ocupa de la separación imperfecta de clases  
svm.setType(cv.ml.SVM\_C\_SVC)  
  
#El núcleo realiza un mapeo realizado a los datos de entrenamiento para mejorar su parecido con un conjunto de datos linealmente separable.   
#Este mapeo consiste en aumentar la dimensionalidad de los datos y se realiza de manera eficiente utilizando una función del kernel.   
#Elegimos aquí el tipo LINEAR lo que significa que no se realiza ningún mapeo. Este parámetro se define usando setKernel .  
svm.setKernel(cv.ml.SVM\_LINEAR)  
  
#Procedimiento de entrenamiento SVM se implementa resolviendo un problema de optimización cuadrática restringida de forma iterativa .  
#Aquí especificamos un número máximo de iteraciones y un error de tolerancia para permitir que el algoritmo finalice en una menor cantidad de pasos incluso si aún no se ha calculado el hiperplano óptimo.  
#Este parámetro se define en una estructura cv.TERM\_CRITERIA\_MAX\_ITER .  
svm.setTermCriteria((cv.TERM\_CRITERIA\_MAX\_ITER, 100, 1e-6))  
  
#La función svm.train() que se utilizará posteriormente requiere que los datos de entrenamiento se almacenen como objetoscv.Mat() de flotadores.  
#Por lo tanto, creamos estos objetos a partir de las matrices definidas anteriormente:  
  
#Metodo de entrenamiento del modelo  
svm.train(trainingData, cv.ml.ROW\_SAMPLE, labels)  
  
# Data for visual representation  
width = 512  
height = 512  
image = np.zeros((height, width, 3), dtype=np.uint8)  
  
# Show the decision regions given by the SVM  
green = (0,255,0)  
blue = (255,0,0)  
for i in range(image.shape[0]):  
 for j in range(image.shape[1]):  
 sampleMat = np.matrix([[j,i]], dtype=np.float32)  
 #El método cv.ml.SVM.predict() se utiliza para clasificar una muestra de entrada utilizando un SVM entrenado.  
 response = svm.predict(sampleMat)[1]  
 if response == 1:  
 image[i,j] = green  
 elif response == -1:  
 image[i,j] = blue  
# Show the training data  
thickness = -1  
cv.circle(image, (501, 10), 5, ( 0, 0, 0), thickness)  
cv.circle(image, (255, 10), 5, (255, 255, 255), thickness)  
cv.circle(image, (501, 255), 5, (255, 255, 255), thickness)  
cv.circle(image, ( 10, 501), 5, (255, 255, 255), thickness)  
  
#Usamos aquí un par de métodos para obtener información sobre los vectores de soporte. El método cv.ml.SVM.getSupportVectors() obtiene todos los vectores de soporte.  
#Hemos utilizado estos métodos aquí para encontrar los ejemplos de entrenamiento que son vectores de soporte y resaltarlos  
# Show support vectors  
thickness = 2  
sv = svm.getUncompressedSupportVectors()  
for i in range(sv.shape[0]):  
 cv.circle(image, (int(sv[i,0]), int(sv[i,1])), 6, (128, 128, 128), thickness)  
cv.imwrite('resources/result.png', image) # save the image  
cv.imshow('SVM Simple Example', image) # show it to the user  
cv.waitKey()

27

The Kernel crashed while executing code in the current cell or a previous cell.   
  
Please review the code in the cell(s) to identify a possible cause of the failure.   
  
Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more info.   
  
View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further details.