UNIVERSIDAD DE GRANADA

UNIVERSIDAD DE GRANADA E.T.S.I. INFORMÁTICA Y TELECOMUNICACIÓN - Course 2019/ 2019

Visión por Computador

SVM + HOG

Trabajo 5

Jaime cloquell Capp

Email:
jaumecloquell@correo.ugr.es

Índice

1. Introducción

Este trabajo trata sobre clasificar imagenes de si contiene o no peatones caminando. Para desarrollar dicho algoritmo, se utilizan los algoritmos HOG (histograma de gradientes orientados) y SVM (máquina de soporte vectorial). El HOG es un método que se basa en la orientación del gradiente en áreas locales de una imagen para encontrar los objetos de dicha imagen. Destaca por su robustez ante los cambios de iluminación, en los fondos y en los objetos. El SVM por su parte, es un clasificador, el cual se usará para realizar el entrenamiento del HOG. Para mejorar el rendimiento, se hará un estudio de los parámetros del sistema que permiten conseguir los mejores resultados, tanto en el tiempo computacional como en la eficiencia en la detección. Por último, se recurrirá también a las librerías de OpenCV, útiles para el procesamiento de imágenes.

2. Descriptores HOG

Los descriptores HOG (del inglés Histogram of Oriented Gradients) se basan en la orientación del gradiente en áreas locales de una imagen. El descriptor HOG permite aprovechar de forma eficiente la información del gradiente (Figura 1) a partir de combinar esta información en forma de histogramas orientados locales, que se calculan en celdas de pequeño tamaño, las cuales se distribuyen de forma uniforme por toda la imagen. Dichos histogramas nos proporcionan información de las orientaciones de los contornos que dominan en cada una de las posiciones de la imagen. Esta información nos va a permitir distinguir la forma de los objetos presentes en una imagen y es una buena base para detectar y reconocer dichos objetos. Gracias a esa información, podemos ver la frontera entre un objeto y otro.

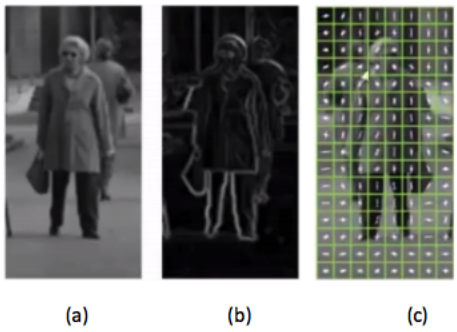

Figura 1: Gradientes e histogramas de una imagen: a) Imagen original, b) Gradientes de la imagen, c) Histograma de Gradientes Orientados

Una vez obtenido los histogramas, para conseguir mejores resultados, los histogramas calculados en cada celda se agrupan en bloques de un tamaño un poco mayor. Estos bloques sirven para normalizar la representación final y hacerla más invariante a los cambios de iluminación y a distorsiones en la imagen. Así, la representación final será la concatenación de la representación de todos estos bloques. La razón de hacer la normalización en bloques en lugar de hacerla en la imagen completa se debe a

que puede ocurrir que los cambios de iluminación no sean uniformes en la imagen, habiendo zonas de la imagen con mayor o menor iluminación que las otras.

3. SVM

Las Máquinas de Vectores Soporte son estructuras de aprendizaje basadas en la teoría estadística del aprendizaje. Su función es transformar el espacio de entrada en otro de dimensión superior (infinita) para que el problema pueda ser resuelto mediante un hiperplano óptimo (de máximo margen). Estos métodos están relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas como pertenecientes a una u otra clase. Dicho de otra forma, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Para conseguir una clasificación correcta será necesario que exista una buena separación entre las clases. La SVM busca un hiperplano que separe de forma óptima a los puntos de una clase de la de otra (Figura 2), que han podido ser previamente proyectados a un espacio de dimensionalidad superior.

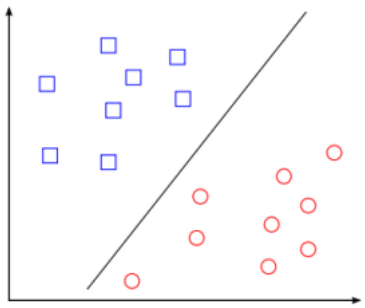

Figura 2: SVM

En ese concepto de "separación óptima" es donde reside la característica fundamental de las SVM: este tipo de algoritmos buscan el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. Por eso, también a veces se les conoce a las SVM como clasificadores de margen máximo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado. Para separar linealmente los datos se procede a realizar un cambio de espacio mediante una función que transforme los datos de manera que se puedan separar linealmente. Esta función recibe el nombre de Kernel. En nuestro caso, los conjuntos positivos y negativos corresponden con las clases "personas" y "no personas". Para ello, se necesita un entrenamiento previo de la máquina, introduciéndole ejemplos positivos (personas) y ejemplos negativos (no personas). Con todos los ejemplos de entrenamiento,

el algoritmo de clasificación SVM crea una curva M-dimensional que divide ambos conjuntos, obteniendo de esta forma el kernel de la máquina. Las dimensiones del espacio dependen del número de componentes de cada vector a clasificar.

4. Desarrollo del algoritmo

Para llevar a cabo la detección de peatones mediante el HOG y el SVM, primero se deben de preparar el las imagenes para su posterior procesamiento. En nuestro caso, el conjunto de imágenes positivas y negativas ha sido creado por nuestra profesora. Cabe destacar que son imágenes variadas, es decir, los peatones presentan distintas alturas, anchuras, vestimentas y posiciones. Además, los objetos de las imágenes negativas también son distintos. A continuación puede verse un ejemplo de los conjuntos de dichas imágenes (figuras 3 y 4):

Figura 3: Imagenes positivas

Figura 4: Imagenes negativas

Estos conjuntos de imágenes se utilizaran para entrenar un detector mediante la Maquina de Soporte Vectorial (SVM). La información que obtendremos de las imágenes dará lugar a un descriptor HOG, el cual se utilizará después para detectar a los peatones en las imágenes que se desee.

4.1. Implementación

Las partes del código más importantes son las relacionadas con el HOG y el SVM puesto que ambos algoritmos son la base del proyecto. Por ello, este capítulo se centrará en explicar las partes relacionadas con el entrenamiento y el test. Además, se hará también una pequeña descripción del código del etiquetador realizado por su utilidad en el proyecto.

4.1.1. Acceso a las carpetas de imágenes

La tarea del entrenamiento será encontrar y almacenar las características de todas las imágenes de las que disponemos para que después éstas puedan ser comparadas y se pueda realizar el test. En nuestro caso, las imágenes de las que disponemos se almacenan en distintas carpetas, a las cuales nuestro algoritmo tendrá que acceder para poder sacar las características mencionadas anteriormente.

```
% dividos los datos en training y test en un 80 %
[imdsTrain,imdsTest] = splitEachLabel(imds,0.8,'randomize');
```

Como se puede ver en la figura anterior, los dos directorios corresponden a las carpetas de imágenes positivas y negativas

4.1.2. Redimensionamiento

Como se ha indicado en capítulos anteriores, para llevar a cabo el proceso de extracción de características, las imágenes con las que se trabaja en el entrenamiento son de dimensiones 64 x 128 pixeles. Esto se debe a que la ventana de detección del HOG seleccionada tiene dichas dimensiones, puesto que son las que mejores resultados nos proporcionan.
Que las imágenes tengan las mismas dimensiones que la ventana de detección del HOG es fundamental, puesto que sino no se podrían extraer las características. Si se diese el caso de que alguna de las imágenes de entrenamiento (ya sean positivas o negativas) no tuviesen las dimensiones apropiadas, el código nos devolvería un error.

```
for i = 1:numImages
    imageTrain = readimage(imdsTrain,i);
    imageTrain = imresize(imageTrain,imageSize);
    featuresTrain(i,:) = extractHOGFeatures(imageTrain);
end
```

4.1.3. SVM

En el programa principal, lo último que hacemos es pasar las características que hemos sacado con los métodos anteriores para realizar el entrenamiento. A partir de este punto, el algoritmo se centra en la detección de los objetos de la imagen y sus clasificaciones posteriores. En nuestro caso utilizamos un suave entrenador SVM lineal.

```
classifer = fitcsvm(featuresTrain,trainLabels,'Standardize',true,'KernelFunction','RBF',
    'KernelScale','auto');
```

Este modelo será el que genere el vector descriptor que deberá ser cargado al hog para que pueda hacer distinción entre peatones y no peatones.

4.1.4. Código de Test

Una vez realizado el entrenamiento de las imágenes, se procede a la parte del test. En este apartado se analizarán las imágenes en las que se desean detectar personas y asi poder ver el grado de fiabilidad del entrenamiento generado.

```
k = zeros(numTest,1) ;
j = zeros(numTest,1) ;
for i = 1:numTest
    testImage = readimage(imdsTest,i);
    scaleTestImage = imresize(testImage,imageSize);
    featureTest = extractHOGFeatures(scaleTestImage);
    [predictIndex,score] = predict( classifer ,featureTest);

    k(i) = predictIndex == 'pos';
    j(i) = imdsTest.Labels(i) == 'pos';

    % figure;imshow(testImage);
    % title(['predictImage: ',char(predictIndex)]);
end
C = confusionmat(k,j);
```

Como se puede ver en la imagen, se itera encima e las imagenes de test para con el clasificador previamente entrenado predececir los valores de si existe o no peatones en la imagen. Para calcular la matriz de confusión, crearemos dos vectores donde guardaremos el valor predecido y el real.

5. Resultados

En este capítulo se muestra los resultados obtenidos del trabajo realizado. Como podemos observar, hemos obtenido un 100 % de acierto. sin ningún caso de falso positivo.

	POS	NEG
POS	10	0
NEG	0	185