Tarea 4 EL7008 – Primavera 2022
Detección de Personas usando Adaboost y características tipo Haar

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: 6 de Octubre de 2022
Fecha entrega: 19 de Octubre de 2022

El objetivo de esta tarea es diseñar y construir un sistema de detección de personas, que utilice características tipo Haar, también llamadas características rectangulares, y un clasificador Adaboost. El cálculo de las características, así como el clasificador Adaboost, deben ser implementados desde cero.

En esta tarea, se usará el mismo conjunto de datos de la tarea 3.

Preparación de Conjuntos de Entrenamiento y Prueba

Para las tareas de entrenamiento y prueba se debe utilizar las imágenes de la base de datos subida a U-Cursos (370 imágenes), la cual incluye 185 imágenes con personas y 185 imágenes sin personas (sillas y autos). Esta base de datos debe ser separada en 80% para entrenamiento y 20% para prueba.

Extracción de Características

Características tipo Haar[1]: Estas características consideran máscaras, formadas por rectángulos, los cuales pueden tener valores positivos o negativos, para calcular la suma de pixeles en regiones de la imagen cubiertas por estas máscaras. De esta manera, el valor de una característica, en una cierta posición de la imagen, está dado por la resta entre la suma de los pixeles que caen en cada zona (oscura y clara) de la máscara. En esta tarea se debe usar los cinco tipos de máscaras tipo Haar mostrados en la Figura 1. Adicionalmente, las máscaras pueden tener polaridad positiva o negativa. Las máscaras mostradas en la Figura 1 tienen todas polaridad positiva; su versión con polaridad negativa se construye sencillamente cambiando el signo de cada región, con lo cual las zonas amarillas se convierten en azules y las azules en amarillas. El cálculo de las características tipo Haar se puede acelerar usando imágenes integrales.

Clasificación

Se debe implementar desde cero un clasificador de tipo Adaboost, sin usar cascadas. El clasificador debe diferenciar entre personas y no-personas.

[1] Ver paper: (1) P. Viola, M. Jones."Rapid object detection using a boosted cascade of simple features", (2) Apuntes de la clase donde se vio en detalle el algoritmos Adaboost y las características Haar.

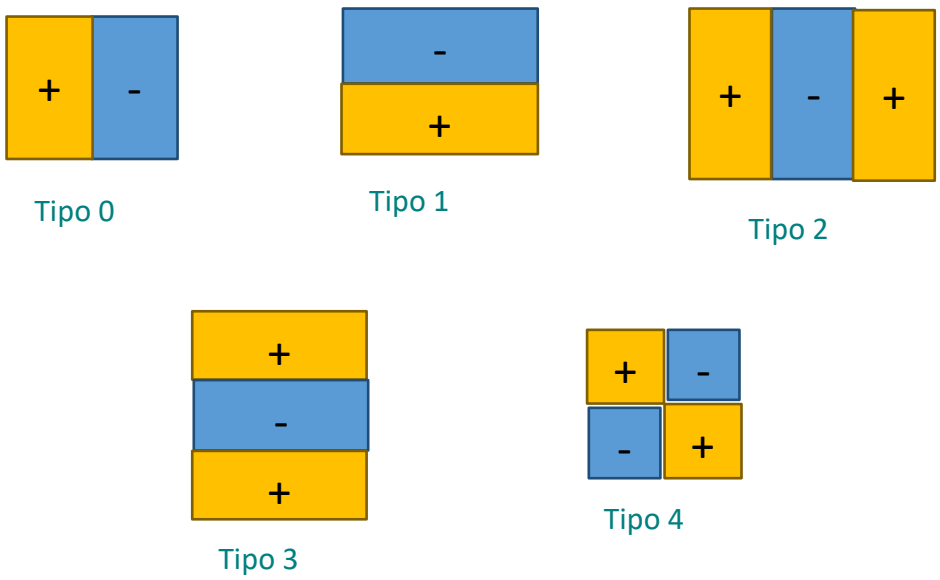

Figura 1. Máscaras Haar/rectangulares a ser usados en esta tarea. Las máscaras mostradas están asociadas a características con polaridad positiva. En el caso de la polaridad negativa, los signos asociados a las máscaras se invierten.

Se pide:

1. Leer las imágenes de la base de datos, dividiéndolas en 80% para entrenamiento, y 20% para prueba. Las imágenes se deben redimensionar a 24x24. Para dividir los datos en entrenamiento y prueba, se debe utilizar una semilla fija al usar train_test_split( ). De este modo, la división en entrenamiento y prueba va a ser siempre la misma, y esto permitirá que los resultados sean repetibles. Las etiquetas de cada imagen deben ser +1 para las imágenes con personas, y -1 para las imágenes sin personas.
2. Implementar una función en Cython que permita calcular la imagen integral, dada una imagen.
3. Implementar una función que genere parámetros que permitan determinar las máscaras de las características Haar. Las máscaras se deben parametrizar como: (*y1*, *x1*, *y2*, *x2*, *tipo*, *polaridad*), donde:
    a. (*y1*,*x1*) es la esquina superior izquierda de la máscara
    b. (*y2*,*x2*) es la esquina inferior derecha de la máscara
    c. *Tipo*: número entre 0 y 4, que permite determinar el tipo de máscara
    d. *Polaridad*: +1 (positiva) o –1 (negativa).

    Dado que en cada imagen las características de pueden calcular en muchas posibles posiciones, se recomienda aplicarlas usando el concepto de ventana deslizante usando un paso de tamaño 3. Además, se recomienda usar sólo 3 valores para los anchos de las máscaras y 3 valores para los altos. En consecuencia, hay 9 posibles combinaciones de anchos/altos. En el caso de las máscaras de tipo 0, 1 y 4, los anchos deben ser múltiplos de 4,

mientras que para las características 2 y 3, el ancho debe ser múltiplo de 3. Además, se debe almacenar la polaridad de cada máscara, la cual puede ser +1 o -1.

4. Implementar una función que, dado un conjunto de imágenes y parámetros de máscaras, calcule vectores de características. Dichos vectores deben contener todas las características tipo Haar determinadas en el punto anterior, para cada imagen.
5. Implementar el clasificador Adaboost. Para esto, se debe seguir los siguientes pasos:
    a. Implementar una función $h(x,u)$, Su salida debe ser: +1 cuando $x > u$, -1 cuando $x < u$. En el caso en que *x* sea un arreglo, su salida debe tener la misma dimensionalidad que *x*, y debe contener +1 o -1 en cada elemento. Esta función se usará como base para construir clasificadores débiles $h(x_i,u)$ asociados a la característica número *i*.
    b. Implementar una función que permita elegir el mejor *u* para cada clasificador débil, dada una matriz de características *X*, un vector de etiquetas *y*, y un vector de pesos *w*. Para realizar este paso, se debe dividir el rango de cada característica *i* en 10 valores, y se debe encontrar el valor de *u* en el cual el clasificador débil asociado a esa característica predice las etiquetas con el menor valor *r*, donde
    $$r = \sum_{k=0}^{N-1} w_k * y_k * h(x_{k,i},u)$$
    con *N* el número de ejemplos de entrenamiento, *k* el índice de estos ejemplos.
    c. Implementar el entrenamiento del clasificador Adaboost, eligiendo *T* clasificadores débiles. Para esto, se debe inicializar el vector de pesos de cada ejemplo como 1/*N*, con *N* el número de muestras de entrenamiento. Luego, se debe encontrar el clasificador débil número *i* que genera el menor valor de *r*, con su *u* correspondiente. Posteriormente se debe calcular el α asociado y luego se deben actualizar los pesos. Estos pasos se deben iterar *T* veces. Los valores α_t, i_t, y u_t se deben almacenar en cada iteración t, para poder construir y evaluar posteriormente el clasificador fuerte.
    d. Implementar la clasificación. Para esto, se debe calcular una suma ponderada de los T clasificadores débiles elegidos, ponderados por los α correspondientes. El clasificador final es:
    $$H(x) = signo\left(\sum_{t=0}^{T-1} \alpha_t * h_t(x)\right)$$
    Donde
    $$h_t(x) = h(x_{i_t}, u_t)$$
6. Para cada una de las imágenes de la base de datos, calcular las imágenes integrales y, a partir de ellas, calcular sus características tipo Haar
7. Entrenar el clasificador Adaboost usando el conjunto de entrenamiento, considerando *T* = 10.
    a. Evaluar el clasificador Adaboost sobre el conjunto de entrenamiento. Indique el accuracy, y muestre la matriz de confusión, usando ConfusionMatrixDisplay( )

   b. Evaluar el clasificador Adaboost sobre el conjunto de prueba. Indique el accuracy, y muestre la matriz de confusión, usando ConfusionMatrixDisplay( )
8. Repetir el paso anterior (paso 7), considerando $T = 5$ y $T = 20$
9. Analice los resultados para cada uno de los T, tanto de forma individual como comparativa
10. Amplíe 3 de las imágenes conteniendo personas a 192x192, y dibuje sobre ella las 5 mejores máscaras elegidas.
11. Documente cada uno de los pasos anteriores en el informe, describiendo cada punto y agregando los trozos de código correspondientes.

El código entregado debe ejecutar cada uno de los pasos anteriores. En el caso de que el código no ejecute todo lo pedido al ejecutarlo celda por celda, se debe adjuntar un readme. El notebook entregado debe poder correr en colaboratory.

Los informes, los códigos (en formato .ipynb) y el archivo README.txt deben ser subidos a U-Cursos hasta las 23:59 horas del día miércoles 19 de Octubre.

Importante:

- La evaluación de esta tarea considerará el correcto funcionamiento del código, la calidad de los experimentos realizados y de su análisis, las conclusiones, así como la prolijidad y calidad del informe entregado.

- Para realizar divisiones enteras, se debe usar el operador // en vez del operador /

- Se debe calcular r usando np.sum( ) en vez de usar un ciclo *for*. Además, la función $h(x,u)$ no debe usar instrucciones *if* ni ciclos *for*. Se pide esto por eficiencia del código.

- En colaboratory, se recomienda subir el .zip con las imágenes y ejecutar:
  `!unzip imagenes_tarea_4_2022.zip`

- El informe de la tarea en formato PDF debe ser subido a turnitin, con el código agregado en un anexo.