Obligatorio: detección de objetos en imágenes

Machine Learning para IA - Universidad ORT

2024

1. Introducción

El objetivo del obligatorio es aplicar las técnicas de clasificación vistas en el curso al problema de *detección de objetos* en imágenes. Concretamente nos enfocaremos en la *detección de rostros*, ver Figura 1 (Derecha). Actualmente dichos problemas se abordan con redes neuronales profundas específicas para el tratamiento de imágenes, como lo son las redes convolucionales. Sin embargo, los primeros algoritmos construídos para este fin se basaron en técnicas más sencillas de ML, y gracias a su bajo costo computacional los mismos siguen siendo utilizados en dispositivos modernos. Ejemplo emblemático de estos últimos es el detector de rostros de Viola-Jones [1].

Figura 1: A la izquierda, una imagen monocolor es una matriz 2D. Las entradas de la matriz corresponden a los valores de los píxeles que van de 0 a 255. A la derecha, ejemplo de detección de rostros en imágenes. Extraído del artículo de Viola y Jones [1].

2. Imágenes y features

Nosotros trabajaremos con imágenes monocolor que suelen representarse en una escala de grises como se muestra en la Figura 1 (Izquierda). De este modo una imagen es una matriz $\boldsymbol{x} = [x_{ij}]_{i=0,j=0}^{i=H-1,j=W-1}$ con H (height) filas y W (width) columnas. Las entradas x_{ij} corresponden a los valores de los píxeles. En el caso monocolor es usual que el rango de valores consista en

enteros que van desde 0 (negro) hasta el 255 (blanco). Pero es común también que dichos valores estén normalizados para que su rango sea entre 0 (negro) y 1 (blanco).

No trabajaremos con los píxeles directamente, sino que nuestro enfoque se basará en la extracción de *features*. Una feature es, en términos generales, cualquier información relevante para resolver la tarea computacional relacionada con una determinada aplicación. Pero para nosotros una feature es una *función* $f : \mathbb{R}^{H,W} \to \mathbb{R}$. Por ejemplo, una feature f puede estar relacionada con la detección de bordes en una determinada región de la imagen, en cuyo caso valores extremos de $f(\boldsymbol{x})$ indican la presencia de dicha característica para $\boldsymbol{x}$.

3. Detectores de rostros basados en clasificadores

Construiremos detectores de rostros a partir de clasificadores. Si disponemos de un clasificador $h : \mathbb{R}^{H,W} \to \{0, 1\}$ para predecir si una imagen es un rostro o no, es decir $h(\boldsymbol{x}) = 1$ cuando $\boldsymbol{x}$ es un rostro, podemos construir un detector de rostros escaneando todas las sub-imágenes de la imagen original. Por esta razón nuestro *principal objetivo* será elaborar clasificadores livianos (desde el punto de vista computacional) que permitan detectar si una imagen es un rostro.

Para esto dispondremos de un conjunto de datos de entrenamiento $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ en donde $\boldsymbol{x}_i \in \mathbb{R}^{W,H}$ es una imagen e $y_i \in \{0, 1\}$ es la etiqueta que indica con 1 si $\boldsymbol{x}_i$ es un rostro. De las N imágenes, p de ellas son rostros (ejemplos positivos) y n son background (ejemplos negativos).

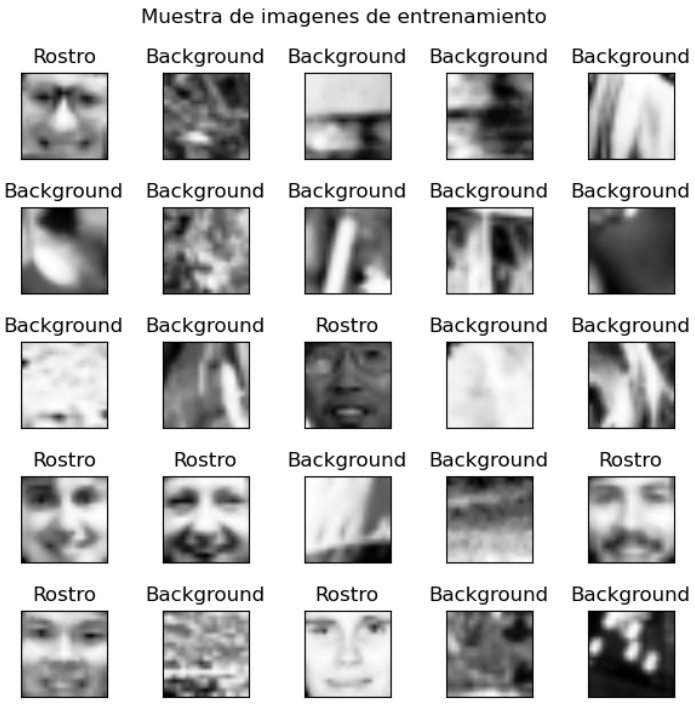

Figura 2: Muestra del conjunto de datos de entrenamiento $\mathcal{S}$.

También dispondremos de un conjunto de datos de test para evaluar los clasificadores construidos.

Como mencionamos antes, los clasificadores deberán trabajar con features extrídas de las imágenes, y no con los datos brutos representados por la matriz de píxeles. Para esto construiremos una *matriz de features* $\boldsymbol{X}$ cuyas filas representan imágenes y columnas features. Debemos elegir una cantidad M de features $\{f_j\}_{j=1}^{M}$ de forma tal que la entrada $i,\ j$ de la matriz $\boldsymbol{X}$ es $f_j(\boldsymbol{x}_i)$.

El par $(\boldsymbol{X}, \boldsymbol{y})$, en donde $\boldsymbol{y}$ es el vector de etiquetas, servirá de input a los clasificadores, ver la Figura 3.

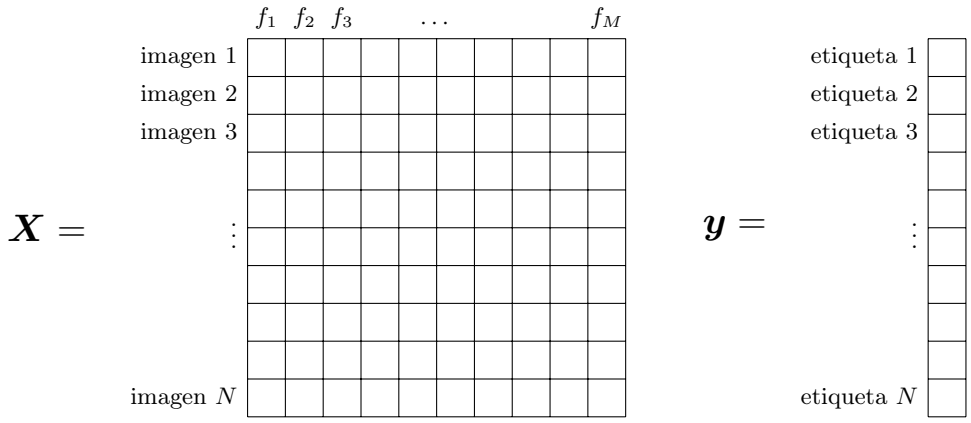

Figura 3: El par $(\boldsymbol{X}, \boldsymbol{y})$ en donde $\boldsymbol{X}$ es la matriz de features e $\boldsymbol{y}$ es el vector de etiquetas.

4. Un ejemplo: las Haar features[1]

Para ilustrar mejor el concepto de feature, veamos un conjunto estándar de features, llamadas *Haar features*, cuya gran ventaja radica en su velocidad de cómputo. Estas features no utilizan la multiplicación, que es más costosa en cómputos, si no que simplemente suman y restan valores de píxeles en la imagen.

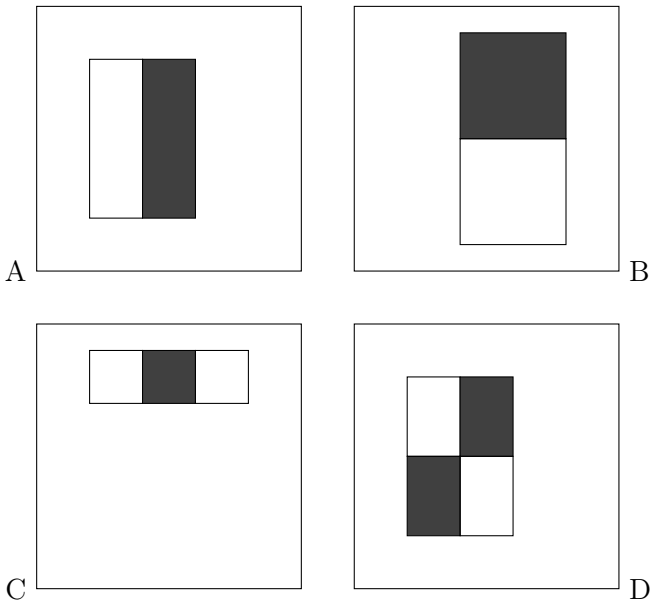

Figura 4: Ejemplo de features rectangulares. La suma de los píxeles que se encuentran dentro de los rectángulos blancos se resta de la suma de los píxeles en los rectángulos negros. Las 2-rectangles features se muestran en (A) y (B). La figura (C) muestra una 3-rectangle feature y (D) una 4-rectangle feature.

[1]Esta sección es simplemente un ejemplo introductorio al concepto de feature que nos ayuda a entender su uso práctico. En el curso veremos más en detalle este tipo de extracción de features.

Más precisamente, una Haar feature se define como

$$f(\boldsymbol{x}) = \sum_{i=0,j=0}^{H-1,W-1} p_b(i,j) - p_n(i,j)$$

en donde $p_b(i,j)$ es 1 si el pixel (i,j) es blanco y 0 si no, y análogamente para $p_n(i,j)$.

Estas features tienen una interpretación intuitiva. Por ejemplo, en la Figura 5 se muestran dos Haar features utilizadas para detectar la nariz y los ojos respectivamente.

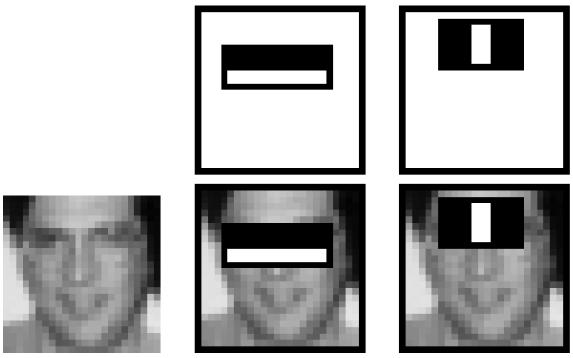

Figura 5: Detección de nariz y ojos mediante Haar features.

La primera feature mide la diferencia de intensidad entre la región de los ojos y la región de la parte superior de las mejillas. La misma se basa en que la región de los ojos suele ser más oscura que las mejillas. La segunda feature compara las intensidades en las regiones de los ojos con la intensidad en la nariz.

Las rectangle features se pueden calcular muy rápidamente usando una representación intermedia para la imagen que se llama *integral image*. La *integral image* $I(\boldsymbol{x})$ de la imagen original $\boldsymbol{x}$ en la posición (i,j) es igual a la suma de los píxeles arriba y a la izquierda de (i,j), inclusive:

$$I(\boldsymbol{x})_{ij} = \sum_{i' \leq i, j' \leq j} x_{i',j'}$$

Se puede calcular de forma recursiva:

$$\begin{cases} s(\boldsymbol{x})_{i,j} = s(\boldsymbol{x})_{i,j-1} + x_{ij} \\ I(\boldsymbol{x})_{i,j} = I(\boldsymbol{x})_{i-1,j} + s(\boldsymbol{x})_{i,j} \end{cases}$$

en donde $s(\boldsymbol{x})$ es la suma acumulada de filas. De este modo la integral image puede calcularse rápidamente en una sola pasada de la imagen.

Una vez a disposición la inetgral image, la suma de los píxeles en un rectángulo puede calcularse con una cantidad constante de operaciones. A modo de ejemplo, la suma de los píxeles dentro del rectángulo D en la Figura 6 se puede calcular con cuatro operaciones. El valor de la integral image en la posición 1 es la suma de los píxeles en el rectángulo A. El valor en la posición 2 es $A+B$, en la posición 3 es $A+C$ y en la posición 4 es $A+B+C+D$. La suma dentro de D se puede calcular como $4+1-(2+3)$.

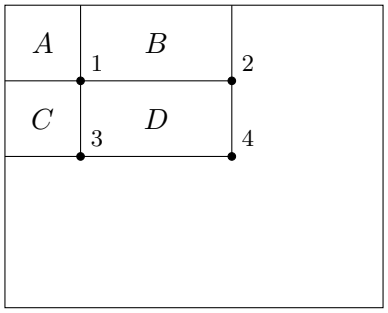

Figura 6: Sumar los píxeles en un rectángulo con la integral image.

5. Un ejemplo clásico de detector liviano: la cascada

En la detección de objetos poco frecuentes, como es el caso de los rostros, existe un mecanismo llamado *attentional cascade* para construir una cascada de clasificadores que logra un mayor rendimiento de detección mientras reduce radicalmente el tiempo de ejecución. En el curso veremos técnicas como esta que permiten combinar varios clasificadores mejorando así su desempeño global. Dichas técnicas de combinación de clasificadores se conocen bajo el nombre de *ensembles*. La cascada es un caso particular que veremos en detalle más adelante en el curso.

En el caso de la cascada, la idea es construir clasificadores eficientes en el rechazo de subventanas negativas (que no contienen rostros), es decir, clasificadores para los cuales la tasa de verdaderos negativos sea cercana a uno. Los primeros clasificadores de la cadena, más simples, se utilizan para rechazar la mayoría de las subventanas antes de recurrir a clasificadores más complejos (posteriores en la cadena) para lograr tasas bajas de falsos positivos.

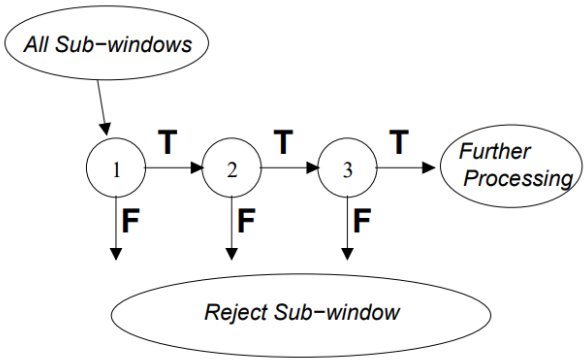

Figura 7: Attentional cascade. Diagrama extraído del artículo de Viola y Jones [1].

La forma general del proceso de detección es la de un árbol de decisión degenerado, lo que se llama una *cascada* (ver Figura 7). Un resultado positivo del primer clasificador desencadena la evaluación de un segundo clasificador que también se ha ajustado para lograr tasas de detección muy altas. Un resultado positivo del segundo clasificador desencadena un tercer clasificador y así sucesivamente. Un resultado negativo en cualquier punto conduce al rechazo inmediato de la subventana.

6. Entrega y tareas a desarrollar

Se deberá entregar un documento que contenga un **informe/reporte** de los resultados obtenidos detallando las diferentes experimentaciones realizadas. El **código** implementado deberá ser entregado también, en una Jupyter notebook (Python).

Se deberán desarrollar las siguientes tareas que deberán estar incuídas en el informe:

- **Preprocesamiento**: definir y explicar detalladamente el tratamiento inicial de las imágenes, como pueden ser la normalización, el rescalado y la aplicación de transformaciones en general.
- **Extracción de features**: definir y explicar detalladamente los procesos de extracción de features utilizados y la construcción de la matriz de features.
- **Clasificadores**: implementar los algoritmos de clasificación vistos en el curso. A modo de ejemplo:
  - Árboles de decisión
  - Regresión logística
  - Ensembles (Random Forest, Boosting, Cascada)
  - Redes neuronales
- **Evaluación y selección de modelos**:
  - Detallar el proceso de evaluación (validación) y selección de los diversos clasificadores (búsqueda de hiperparámetros) utilizando las técnicas vistas en el curso (Holdout, Repeated Holdout y/o Cross-Validation).
  - Utilizar las diferentes métricas relevantes al problema de clasificación (accuracy, precision, recall, TPR, FPR, curvas ROC, AUC, balanced accuracy, etc).
  - Resumir los resultados de los experimentos en tablas que permitan recopilar la información de cada modelo evaluado y sus respectivas métricas. Se recomienda incluir una tabla por cada algoritmo que resuma el desempeño del mismo para distintos valores de los hiperparámetros, así como una tabla final que resuma el desempeño de los mejores algoritmos.
- **Modelo final**: se deberá elegir un modelo final. El informe deberá contener la justificación de la elección de dicho modelo. El mismo deberá adjuntarse a la entrega.

7. Información importante

Recordar:

- El obligatorio tiene un máximo de 40 puntos.
- Se puede hacer en grupo con un máximo de 3 integrantes por grupo.
- La inscripción es individual y **obligatoria** a través del sitio de Gestión.
- La entrega es online a través del sitio de Gestión.
- **Fecha de entrega**: ver sitio de Gestión.

Referencias

[1] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001.* Vol. 1. Ieee, 2001.

[2] Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014. Chapter 10.4.

[3] Schapire, Robert E., and Yoav Freund. "Boosting: Foundations and algorithms." *Kybernetes 42.1 (2013)*: 164-166. Chapter 3.4.3.

[4] Baumann, Florian, et al. "Cascaded random forest for fast object detection." *Image Analysis: 18th Scandinavian Conference, SCIA 2013, Espoo, Finland, June 17-20, 2013. Proceedings 18.* Springer Berlin Heidelberg, 2013.

[5] Detecting Faces (Viola Jones Algorithm) - Computerphile