En este trabajo vamos a enfrentarnos a un problema de clasificación supervisada donde las entradas al problema van a ser imágenes y el objetivo es determinar a qué clase pertenece cada una de ellas.

En la actualidad, existen muchos modelos que son capaces de aprender rápidamente a distinguir entre las distintas clases con una tasa de acierto muy elevada. Sin embargo, con los conocimientos que tenemos, vamos a retrotraernos unos años atrás y vamos a aplicar las técnicas más básicas que se utilizaban (y se siguen utilizando) antes de la llegada del deep learning.

Cuando queremos aprender un clasificador que, a partir de una imagen, nos prediga la clase a la que pertenece, debemos distinguir dos fases bien diferenciadas: fase de extracción de características y fase de aprendizaje de los modelos.

Cuando las imágenes a clasificar son sencillas y pequeñas, normalmente solemos hacer que los clasificadores aprendan directamente de los píxeles de las imágenes. Es decir, la imagen, o, mejor dicho, sus píxeles, son la entrada al modelo. La salida, simplemente será una etiqueta para cada una de las clases. Sin embargo, cuando las imágenes son grandes, la variabilidad de formas/colores/texturas es elevada, no podemos alimentar directamente los modelos con píxeles. Para eso, necesitamos un paso intermedio que permita transformar las imágenes en un conjunto de características que, de alguna manera, sean más útiles para el clasificador en la tarea de discriminar entre clases. A este proceso le llamamos extracción de características.

Una vez que hemos realizado la extracción de características y, por tanto, hemos transformado una imagen en un vector de números (características), la segunda fase sería la del aprendizaje de los modelos. En este caso, estaríamos ante un problema clásico de clasificación supervisada, donde para cada imagen de entrenamiento tendríamos su vector de características junto con su clase. El objetivo del modelo es, por tanto, aprender un mapeo que nos permita asignar a un vector de características una etiqueta concreta.

El objetivo de este trabajo es, por tanto, doble. En primer lugar, debéis implementar dos formas diferentes de extracción de características que os proponemos. Evidentemente, las técnicas que os presentamos tienen parámetros diferentes que podéis modelar para generar así muchos extractores de características diferentes. Como segundo objetivo, debéis utilizar los conocimientos que tenéis de aprendizaje automático para aprender modelos de clasificación que, a partir del conjunto de características programado, sean capaces de clasificar correctamente entre las diferentes clases del problema.

Nota: en primer lugar, asumiremos que las imágenes las vamos a tratar en escala de grises. La utilización de imágenes en color no es estrictamente necesaria y dependerá de cuanto queráis probar.

Extracción de características 1

A continuación, os mostramos la primera forma de extracción de características que debéis programar. El algoritmo de extracción de características tiene los siguientes pasos:

- Dividir la imagen en bloques disjuntos de NxN píxeles (por ejemplo, N = 16). A estos bloques les llamaremos celdas.
- Para cada píxel de la celda, lo vamos a comparar con cada uno de sus 8 vecinos, empezando, por ejemplo, por el de arriba a la izquierda y siguiendo la dirección de las agujas del reloj.
- Los valores de los 8 vecinos los vamos a sustituir por un 0 o un 1 en función de si el vecino es menor (0) o mayor (1) que el píxel central.
- Una vez realizado este cambio, si tomamos el valor de los 8 vecinos siguiendo la misma distribución que en el paso anterior, nos generará un número binario que, al transformarlo a decimal, resultará en un número entre 0 y 255
- Modificaremos el valor del píxel por el del número decimal obtenido.

Una vez realizado esto para cada píxel de la celda, vamos a construir un histograma de los NxN píxeles obtenidos. El valor del histograma será el vector de características asociados a la celda.

Por último, si concatenamos todos los histogramas de la imagen siguiendo el orden de las celdas, obtendremos el vector de características de la imagen.

Ejemplo: dada una imagen de 128x128 y tomando N=16, esto nos generará 64 celdas distintas y el vector de características finales tendrá 16.384 elementos.

¿Puedo generar modificaciones sobre esta extracción de características? Claro que sí, tantas como se os ocurran. Por ejemplo:

- Realizar lo mismo para las imágenes en RGB y concatenar los histogramas.
- Aplicación de técnicas de reducción de la dimensionalidad (PCA u otros)
- Utilizar el concepto de transición uniforme. En este caso, todos aquellos números binarios generados que contengan más de dos saltos de 0 a 1 o de 1 a 0 se considerará una transición no-uniforme. Todas las transiciones no-uniformes se codificarán de la misma manera y, por tanto, irán a parar al mismo valor del histograma. Con esto conseguimos reducir el valor del histograma a 59.

Extracción de características 2

A continuación, os mostramos la segunda forma de extracción de características que debéis programar. El algoritmo de extracción de características tiene los siguientes pasos:

- Aplicar un filtro espacial de aproximación de la derivada parcial en ambas direcciones mediante las máscaras de convolución −1 0 1 y −1 0 1 .
- Calcular la magnitud y orientación de gradiente. Para la orientación utilizar la función np.arctan2 junto con np.rad2deg para obtener la orientación en grados. Además, realizar la operación orientación % 180 para obtener todas las orientaciones en positivo y entre 0 y 180 grados.
- Dividir la imagen en bloques disjuntos de NxN píxeles (por ejemplo, N = 16). A estos bloques les llamaremos celdas
- Construiremos un histograma para cada una de las celdas. El histograma estará formado por 9 valores o bins (hasta ahora hemos construido siempre histogramas de 256 valores o bins, uno para cada intensidad): el primer bin servirá para las orientaciones entre 0 y 20 grados; el segundo para las orientaciones entre 20 y 40; y así sucesivamente hasta las orientaciones entre 160 y 180.
- Para rellenar el histograma, no vamos a hacer un conteo de cuántas orientaciones hay en cada bin, sino que rellenaremos el histograma mediante el valor de la magnitud del gradiente. Por ejemplo, si el primer píxel de la celda tiene orientación 45º, entonces en el bin asociado a 45º (tercer bin) sumaremos el valor de la magnitud en dicho punto. Una vez sumadas las magnitudes en cada correspondiente bin, tendremos un vector de 9 valores para cada celda.
- Vamos a normalizar el histograma, dividiéndolo entre el valor de la norma euclídea de los 9 valores (raíz cuadrada de la suma al cuadrado de los valores).

Por último, si concatenamos todos los histogramas de la imagen siguiendo el orden de las celdas, obtendremos el vector de características de la imagen.

Ejemplo: dada una imagen de 128x128 y tomando N=16, esto nos generará 64 celdas distintas y el vector de características finales tendrá 576 elementos.

¿Puedo generar modificaciones sobre esta extracción de características? Claro que sí, tantas como se os ocurran. Una de las que mejor resultado os puede dar es la siguiente:

- Normalizar el histograma de las celdas teniendo en cuenta los histogramas vecinos. Para ello, vamos a definir el concepto de bloque como un conjunto de MxM celdas (por ejemplo, M = 2). De esta manera, tomaremos los $M2$ histogramas ($9M2$ valores), y los normalizaremos por la norma de dichos valores. - Además de lo anterior, podemos hacer que los bloques de MxM celdas se vayan moviendo de 1 en 1 (como en un filtro espacial), generando así cierto solapamiento a la hora de calcular las normalizaciones. Evidentemente, esto hará que el vector de características sea más grande (por el solapamiento), pero puede obtener características de mayor calidad. Siguiente con el ejemplo, una imagen de 128x128 y tomando N=16, M=2, y con un paso de 1 en 1 entre bloques, generaría 64 celdas distintas, que agruparemos en 49 bloques distintos (7x7), generando así un vector de características de 1764 elementos.

Segunda fase: entrenamiento de los modelos

A partir de lo programando en los apartados anteriores, estás en disposición de tener una (o varias) función (o funciones) que, a partir de una imagen, genera un vector de características. Con esto, ya puedes construir tu propio dataset que contendrá tantas filas como imágenes y tantas columnas como elementos contenga el vector de características (¿tendrán que tener entonces todas las imágenes el mismo tamaño?).

El objetivo es que probéis con algún clasificador que os proporcione la librería scikit-learn (o pytorch) y que seáis capaces de evaluar la calidad de las extracciones de características que hayáis generado. Se entiende que, cuanta mayor precisión tenga el clasificador en el conjunto de test, mejor será la extracción de características.

Para realizar todas estas pruebas, os proporcionamos dos conjuntos de datos que están separados en carpetas “train” y “test” y, a su vez, cada carpeta está dividida en tantas carpetas como clases. La construcción de los datasets la debéis hacer de manera automática. Para ello, es importante que utilicéis librerías como os o funciones del módulo glob. El primer conjunto está formado por 100 imágenes por clase y el segundo por 500 imágenes por clase. El conjunto de test está formado por otras 100 imágenes por clase.

Características del trabajo

¿Qué debéis entregar?

- Uno o varios notebooks con vuestro trabajo (ojo, revisaremos códigos en busca de copias y plagios)
- - Un informe (no más de 15 hojas) en el que hagáis un resumen de lo que habéis hecho, pruebas, resultados, conclusiones, etc.
- Una presentación de unos 5 minutos, que subiréis a alguna plataforma tipo Youtube, vimeo, etc. donde hagáis una explicación clara y concisa del trabajo realizado. De nuevo, pruebas, resultados, conclusiones, etc.

¿Qué se valora?

- Que los algoritmos estén correctamente implementados
- Que se hayan hecho uso de algoritmos vistos en clase para mejorar la calidad de las imágenes
- Que se hayan probado diferentes formas de extracción de características
- Que se hayan entrenado diferentes modelos de entrenamiento
- Que la metodología de entrenamiento, validación y test sea correcta
- Que se hayan evaluado diferentes alternativas y se obtengan conclusiones demostradas. ¿Qué método de extracción es mejor? ¿Cómo afecta el preprocesamiento de las imágenes al resultado del clasificador?
- Que la memoria esté bien escrita, organizada, etc.

- Que la presentación esté bien estructurada, el discurso sea correcto y coherente
- Etc…
-