Extracción de Características en Imágenes (ECI)
Universidad de Granada

Extracción de Rasgos

Marta Verona Almeida

Índice

1. Introducción

El objetivo de esta práctica es extraer distintos tipos de rasgos de una imagen y usarlos para entrenar y evaluar clasificadores, mediante el uso de distintas técnicas.

El problema de clasificación consiste en aprender a distinguir entre personas y fondo, es decir, es un problema de clasificación binaria. En esta práctica, además de medir la bondad del clasificador y encontrar los mejores parámetros para el mismo, se utilizan distintos descriptores para obtener información sobre la imagen.

La implementación ha sido realizada en java, partiendo del ejemplo visto en clase. La pantalla principal resultante puede verse en la figura 1.1. Se ha implementado una función para el cálculo de la bondad del clasificador utilizando validación cruzada. Para ctivar esta función, basta pulsar el primer botón incorporado a la barra superior. El siguiente botón activa una función que calcula la bondad del clasificador SVM utilizando distintos parámetros. Por último, se permite cambiar de descriptor con el último selector de la barra superior.

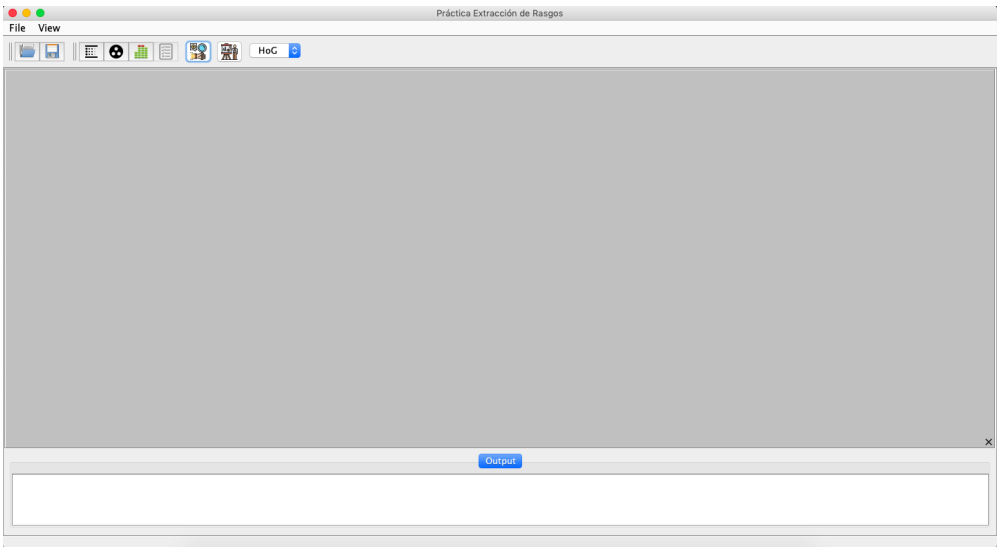

Figura 1.1: Pantalla principal

Todas estas tareas serán descriptas con mayor detalle a lo largo de esta memoria. Además, se proporcionarán y comentarán los resultados obtenidos en las distintas pruebas.

2. Evaluación de la clasificación

2.1. Validación Cruzada

En primer lugar, se ha implementado la validación cruzada. De esta manera, se pretende obtener una evaluación más fiable del modelo.

Esta técnica consiste en dividir el conjunto de entrenamiento disponible en k conjuntos sobre los que se itera. De manera que, en cada iteración, se utilizará uno de ellos como conjunto test y el resto como conjunto de entrenamiento. Este proceso puede verse con mucha claridad en la imagen 2.1.

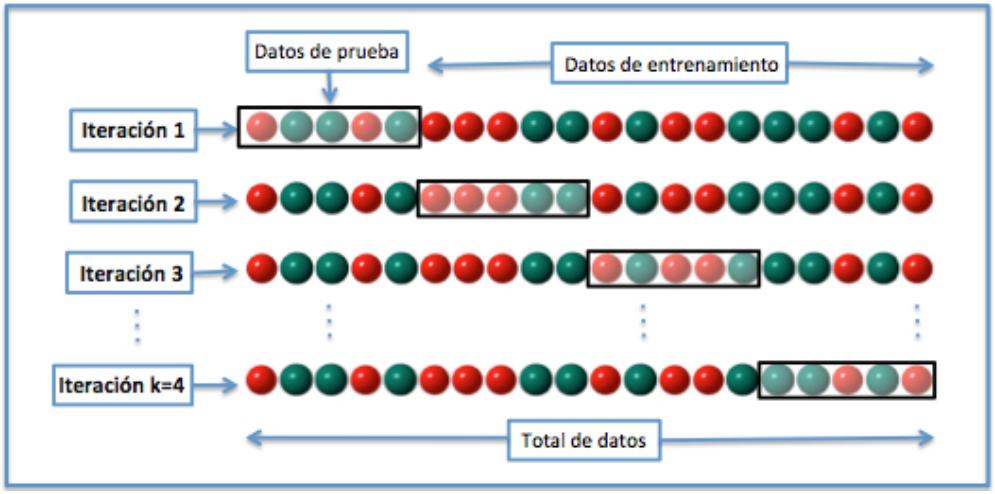

Figura 2.1: Validación Cruzada

En cuanto la implementación llevada a cabo en este apartado, se divide el conjunto de entrenamiento inicial en 6 conjuntos, sobre los que se iterará cambiando el conjunto de test. Dado que las clases se encuentran algo desbalanceadas, he decidido procurar, sin perder aleatoriedad, que el número de elementos de cada clase sea proporcionado en los distintos subconjuntos.

En primer lugar se han obtenido los índices correspondientes a elementos cada clase y se han *barajado*, por separado. A continuación, se almacenan los índices de forma proporcionada, es decir, procurando que el número de elementos de cada clase sea el mismo en los distintos subconjuntos que se utilizarán como test. Finalmente, esta lista de índices se utilizará para generar los conjuntos de entrenamiento y evaluación en cada iteración, tal y como se muestra en la figura anterior.

2.2. Medidas de bondad

Para medir la bondad de nuestro modelo se obtiene tres medidas, que serán descritas a continuación.

- **Accuracy:** precisión del modelo en el sentido general, es decir, la proporción de elementos clasificados correctamente. Matemáticamente, se expresa como sigue

$$\frac{TotalAciertos}{TotalEjemplos}.$$

- **Tasa de Aciertos Positivos (TPR):** precisión del modelo en los casos positivos, en este caso, indica la proporción de *fondos* detectados correctamente. Se calcula como el número de elementos de la clase positiva clasificados correctamente entre el número total de elementos de dicha clase. La fórmula que calcula esa medida es la siguiente

$$\frac{PositivosCorrectos}{TotalPositivos}.$$

- **Tasa de Aciertos Negativos (TNR):** indica lo mismo que TPR pero con las clases negativas, en este caso, el número de *personas* clasificados correctamente. Matemáticamente

$$\frac{NegativosCorrectos}{TotalNegativos}.$$

Estas técncias serán utilizadas más adelante, junto con la validación cruzada, para decidir qué parámetros constituyen el mejor clasificador para este problema. Así pues, para medir la bondad del clasificador, se calculará la media del accuracy, TPR y TNR en las distintas iteraciones.

3. LBP-básico

LBP es un descriptor de texturas que *etiqueta* los píxeles teniendo en cuenta su vecindario, concretamente, sus 8 vecinos colindantes.

Para explicar su funcionamiento, nos fijaremos en la imagen 3.1. Como puede observarse, en este caso se estudia el vecindario del pixel con valor 111. A cada vecino se la asigna un 0 si su valor es menor que el que se está estudiando y 1 en caso contrario. Este proceso se realiza ral y como se muestra en la figura. El siguiente paso es obtener el número binario que se acaba de generar. Esto se hace de forma circular, tal y como muestra la flecha roja de la imagen. Por tanto, en este ejemplo, el número binario obtenido será el 0001110. El último es pasar este número a decimal y asignar este valor al pixel objeto de estudio.
Una vez se ha realizado este proceso sobre todos los pixeles de la imagen, obteniendo una lista de enteros comprendidos en $[0, 255]$, el descriptor resulta de su histograma.

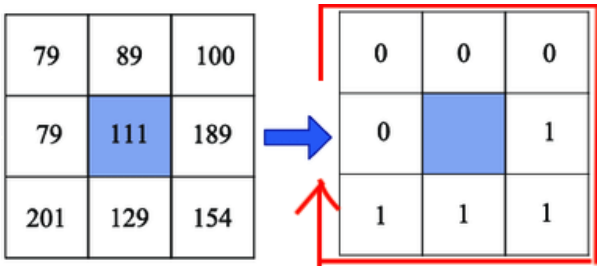

Figura 3.1: LBP

Hasta ahora se ha explicado el funcionamiento LBP aplicado a toda la imagen, sin embargo, este proceso suele realizarse por bloques con solapamiento para obtener una mayor robustez. Para ello, en lugar de aplicar directamente el algoritmo sobre la imagen completa, se aplica por bloques de 16×16 pixeles. El hecho de que haya solapamiento entre los bloques significa que, cada 8 pixeles se comienza un nuevo bloque, hasta completar la imagen, obteniendo un total de 105 bloques.Ésta es la versión que se ha implementado en esta práctica.
Finalmente, el resultado es la concatenación de los histogramas resultantes para cada bloque.

4. Modificación de parámetros.

Con el objetivo de encontrar el mejor modelo posible, he variado los parámetros del algoritmo utilizado, **SVM**. Las SVM (Máquinas de Soporte Vectorial) son algoritmos de aprendizaje supervisado que tratan de encontrar un hiperplano que separe los puntos de ambas clases. En caso de que el problema no sea lineal, es posible utilizar kernels para trasladar los datos a un espacio donde el hiperplano solución sea lineal. Una vez obtenida la solución, se vuelve a transformar al estado original.
Los parámetros que se han variado han sido el kernel y el tipo algoritmo empleado para la clasificación.

4.1. Parámetros modificados

Kernel

Como ya se ha dicho, el kernel ayuda en aquellos casos donde el problema no es lineal. Por ello, además del kernel lineal se ha probado el polinómico.

- **Lineal:** Funciona correctamente cuando los datos son linealmente separables.

$$K(x_i, x_j) = x_i^T x_j$$

- **Polinomial:** En este caso se ha utilizado d igual 2,3 y 4. Es un kernel popular en el procesamiento de imágenes.

$$K(x_i, x_j) = (\gamma x_i^T x_j + c)^d$$

Tipo de clasificador

El tipo de clasificador utilizado en SVM puede ser modificado. En esta práctica se han utilizado dos algoritmos muy simulares que se recomiendan para clasificación binaria.

- **C-Support Vector Classification (C-SVC):** Recomendado para clasificación binaria
- **NU-Support Vector Classification (nu-SVC):** Se añade un parámetro, μ, comprendido en el intervalo $[0, 1]$. Éste indica el límite superior de errores en el entrenamiento y el límite inferior de la fracción de vectores de soporte. Es decir, un valor de μ más alto, hace a la SVM más permisiva con los outliers, mientras que un valor más vajo, la hace menos permisiva con ejemplos extraños. Este algoritmo es recomendado para problemas de clasificación binaria. En esta práctica se ha variado μ, tomando los valores 0.1,0.2,0.3 y 0.4.

4.2. Resultados

4.2.1. HoG

En este adartado, se calculan los mejores parámetros para el algoritmo SVM utilizando el descriptor HoG.
En primer lugar, en la tabla 4.1 podemos observar los resultados obtenidos utilizando C-SVC. En ella, odemos observar que el Kernel polinómico ofrece mejores resultados que el lineal, llegando a proporcionar un 0.974 de accuracy si utilizamos grado 3 o 4. La diferencia en estos dos casos reside en el TPR y en el TNR, que varía ligeramente entre ellos. En este problema, como la clase positiva es la minoritaria, podríamos darle preferencia a una configuración que la clasifique ligeramente mejor. Por tanto, decidimos que el mejor kernel para C-SVC es el polinómico de grado 3, ya que tiene un TPR ligeramente más alto.

Kernel	Accuracy	TPR	TNR
Lineal	0.963	0.942	0.979
Polinomial, d = 2	0.971	0.953	0.985
Polinomial, d = 3	0.974	0.955	0.990
Polinomial, d = 4	0.974	0.953	0.991

Tabla 4.1: C-SVC

En segundo lugar, se estudia la mejor configuración NU-SVC. En este caso, se varía el parámetro μ entre los valores 0.1,0.2,0.3 y 0.4. Además, se prueban distintos kernels tal y como se hizo para C-SVC. Los resultados obtenidos se muestran en la gráfica 4.2.

Kernel	Nu	Accuracy	TPR	TNR
Lineal	0.1	0.970	0.951	0.985
Lineal	0.2	0.967	0.942	0.986
Lineal	0.3	0.956	0.934	0.973
Lineal	0.4	0.941	0.933	0.948
Polinomial, d = 2	0.1	0.914	0.907	0.919
Polinomial, d = 2	0.2	0.914	0.907	0.919
Polinomial, d = 2	0.3	0.914	0.907	0.919
Polinomial, d = 2	0.4	0.914	0.909	0.919
Polinomial, d = 3	0.1	0.941	0.925	0.954
Polinomial, d = 3	0.2	0.941	0.925	0.954
Polinomial, d = 3	0.3	0.941	0.925	0.954
Polinomial, d = 3	0.4	0.941	0.925	0.954
Polinomial, d = 4	0.1	0.939	0.929	0.946
Polinomial, d = 4	0.2	0.939	0.929	0.946
Polinomial, d = 4	0.3	0.939	0.929	0.946
Polinomial, d = 4	0.4	0.939	0.929	0.946

Tabla 4.2: nu-SVC

En primer lugar, destaca el hecho de que en este caso, la variación del parámetro μ no afecte, prácticamente, cuando se utiliza el kernel polinómico de un cierto grado. De hecho, para cada grado tomado, con tres decimales no se aprecian diferencias en los resultados. Además, se observa que el kernel polinómico no funciona bien si se utiliza NU-SVC, ya que el mejor resultado obtenido es similar al peor obtenido utilizando el kernel lineal.

En cuanto a los resultados obtenidos variando μ con el kernel lineal, observamos que cuanto menor es éste, mejores resultados se obtienen. De esta forma, con $\mu = 0{,}1$ y utilizando el kernel lineal se obtiene un accuracy de 0.97.

Concluimos por tanto que, entre las configuraciones probadas, la que mejor funciona es un kernel lineal con un μ bajo.

Por último, observamos los mejores resultados obtenidos utilizando estos dos tipos de clasificados, esta información puede verse en la tabla 4.3. Podemos observar que la mejor configuración encontrada surge de utilizar C-SVC con un kernel polinómico de grado tres.

Tipo	Configuración	Accuracy	TPR	TNR
C-SVC	Polinomial, $d = 3$	0.974	0.955	0.990
NU-SVC	Lineal - $\mu = 0{,}1$	0.970	0.951	0.985

Tabla 4.3: Comparativa - HoG

4.2.2. LBP

En este apartado se realiza el mismo experimento que en el apartado anterior pero utilizando el descriptor LBP, explicado anteriormente.
En primer lugar se obtienen los resultados utlizando C-SVC, que pueden verse en la tabla 4.4. En este caso podemos observar que, los resultados obtenidos utilizando el kernel polinómico, son peores a medida que aumenta el grado. De hecho, el mejor resultado con este kernel se obtiene para grado 2, obteniendo un accuracy de 0.972. Esta precisión es sólo 0.001 mayor que la obtenida mediante un kernel lineal. Observamos así la primera diferencia obtenida por el uso de distintos descriptores.

Teniendo en cuenta el desbalanceo del conjunto de entrenamiento, decidimos cual es la mejor configuación, no sólo observando el accuracy, sino el TPR. En este caso sí existe una notable diferencia entre utilizar un kernel lineal y un kernel polinómico de grado dos, ya que se obtiene un TPR de 0.973 y 0.986 respectivamente. Por tanto, concluimos que para C-SVC, la mejor configuración viene dada por un kernel polinómico de grado dos.

Kernel	Accuracy	TPR	TNR
Lineal	0.971	0.973	0.969
Polinomial, d = 2	0.972	0.986	0.960
Polinomial, d = 3	0.921	0.969	0.882
Polinomial, d = 4	0.629	0.718	0.561

Tabla 4.4: C-SVC

En segundo lugar, en la tabla 4.5 se pueden observar los resultados obtenidos utilizando NU-SVC para $\mu = 0{,}1, 0{,}2, 0{,}3, 0{,}4$ y cambiando el kernel utilizado. Al igual que con HoG, vemos que cuando utilizamos el kernel polinomial no afecta el valor μ tomado. Además, ánalogo a lo visto en el apartado previo, el mejor resultado se obtiene tomando el kernel lineal con el menor μ, 0.1.

Kernel	Nu	Accuracy	TPR	TNR
Lineal	0.1	0.977	0.978	0.976
Lineal	0.2	0.968	0.971	0.966
Lineal	0.3	0.960	0.967	0.954
Lineal	0.4	0.948	0.959	0.939
Polinomial, d = 2	0.1	0.960	0.964	0.957
Polinomial, d = 2	0.2	0.960	0.964	0.957
Polinomial, d = 2	0.3	0.960	0.964	0.957
Polinomial, d = 2	0.4	0.960	0.964	0.957
Polinomial, d = 3	0.1	0.954	0.961	0.948
Polinomial, d = 3	0.2	0.954	0.961	0.948
Polinomial, d = 3	0.3	0.954	0.961	0.948
Polinomial, d = 3	0.4	0.954	0.961	0.948
Polinomial, d = 4	0.1	0.946	0.956	0.937
Polinomial, d = 4	0.2	0.946	0.956	0.937
Polinomial, d = 4	0.3	0.946	0.956	0.937
Polinomial, d = 4	0.4	0.946	0.956	0.937

Tabla 4.5: nu-SVC - LBP

Por último, en la tabla 4.6 comparamos los resultados obtenidos utilizando los dos tipos propuestos: C-SVC y NU-SVC. Podemos observar que el mejor resultado se recoge para NU-SVC utilizando un kernel lineal y un μ de 0.1.

Tipo	Configuración	Accuracy	TPR	TNR
C-SVC	Polinomial, $d = 2$	0.972	0.986	0.960
NU-SVC	Lineal, $\mu = 0{,}1$	0.977	0.978	0.976

Tabla 4.6: Comparativa - HoG

4.3. Conclusión

A modo de conclusión, compararemos los mejores resultados obtenidos para cada descriptor utilizado: Hog y LBP. Estos, pueden verse en la tabla 4.7.

Como podemos observar, no existe una conclusión clara sobre qué descriptor es mejor para este problena. De hecho, un descriptor funciona mejor con un tipo de clasificador y el otro descriptor con el otro tipo. Concretamente, el descriptor HoG ofrece mejores resultados si se utiliza C-SVM con un kernel polinomial de tercer grado, que LBP con C-SVM utilizando kernel polinomial de segundo grado. Siendo ambos los mejores resultados obtenidos en esta categoría para cada descriptor.

Por otra parte, al utilizar NU-SVC sí se obtienen los mismo parámetros como mejor configuración utilizando los dos descriptores, por lo que la comparación puede llegar

a tener más sentido. En este caso, el hecho de aplicar el descriptor LBP supone una mejora con respecto al uso de HoG, utilizando un kernel lineal y $\mu = 0{,}1$.

Descriptor	Tipo	Configuración	Accuracy	TPR	TNR
HoG	C-SVC	Polinomial, $d = 3$	0.974	0.955	0.990
LBP	C-SVC	Polinomial, $d = 2$	0.972	0.986	0.960
HoG	NU-SVC	Lineal, $\mu = 0{,}1$	0.970	0.951	0.985
LBP	NU-SVC	Lineal, $\mu = 0{,}1$	0.977	0.978	0.976

Tabla 4.7: Comparativa - HoG vs LBP