

Procesamiento de texto manuscrito usando conjuntos de clasificadores

Emilio Samuel Aced Fuentes
Roberto Alcober Couso
Arturo Blázquez Pérez
Nicolás Trejo Moya

5 de enero de 2019

1. Introducción

En este proyecto vamos a clasificar imágenes de letras manuscritas intentando predecir de forma correcta el símbolo que representan. Utilizaremos varios algoritmos de clasificación y los compararemos para encontrar los mejores resultados posibles, viendo diferencias de tiempos y tasa de acierto.

2. Análisis de los datos

En el dataset provisto, están representadas las 10 primeras letras del alfabeto (A-J), por tanto 10 clases las cuales hemos etiquetado en nuestra base de datos etiquetadas de 0 a 10 respectivamente. Las imágenes nos vienen en un tamaño de 206×150 en escala de grises.

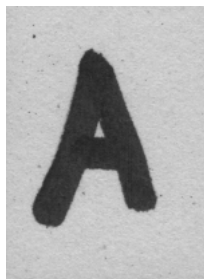


Figura 1: Ejemplo de una A

Como podemos observar las imágenes contienen ruido, el cual podría dificultar la tarea de clasificación gravemente. Este problema lo hemos solucionado sometiendo a la imagen a diferentes tratamientos.

2.1. Tratamientos

Para todos los experimentos de este proyecto hemos umbralizado la imagen mediante otsu y posteriormente hemos realizado un filtro de mediana con un kernel cuadrado de tamaño 3. Esto nos ha dado el siguiente resultado:

A continuación explicaremos los tratamientos que le hemos aplicado a las imágenes para reducir la complejidad del problema



Figura 2: Ejemplo de una A tratada

2.2. Atributos

Teniendo en cuenta que las imágenes son matrices 206×150 lo cual nos da un total de 30900 atributos, lo cual es una cantidad de atributos desorbitada, por ese motivo hemos decidido ver como afecta reducir la dimensión. Las principales modificaciones que realizaremos a las imágenes para reducir su dimensionalidad son las siguientes:

Eliminar filas y columnas poco importantes Esto lo realizaremos mediante la selección de un umbral, con el cual consideraremos que las filas y columnas que tengan una cantidad menor de zona pintada que el umbral especificado no son de relevancia y por tanto las despreciaremos del problema

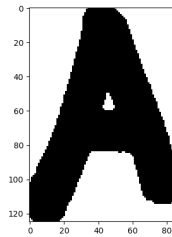


Figura 3: Ejemplo de una A recortada

Redimensionar la imagen interpolando Con el fin de evitar el aliasing (meter bordes que no existían al reducir una imagen) hemos redimensionado la imagen siempre aplicando un kernel gaussiano previamente.

A continuación mostraremos distintos valores de los tamaños de las imágenes que compararemos en nuestro estudio:

3. Modelos

Usaremos los siguientes clasificadores en este trabajo:

1. Random Forest
2. SVC
3. KNN

en los cuales variaremos los tamaños de las imágenes y compararemos tiempos y tasa de aciertos para poder decidirnos por uno.

Para todos ellos destinaremos la mitad de los datos para train y la otra mitad para test.

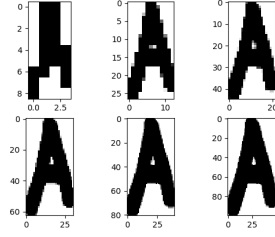


Figura 4: Varios tamaños de una A que usaremos

3.1. Random Forest

Con random forest hemos encontrado que funciona muy bien incluso reduciendo mucho la dimension de los datos encontrando que la mejor tasa de acierto comparandola con el tiempo es reducir las imagenes a cuadrados 9×4 con 500 arboles de profundidad máxima 10 consiguiendo los siguientes resultados:

1. Tasa de acierto: 92.57 %
2. Matriz de confusion:

Clase	Precisión	Sensibilidad
A	0.93	0.99
B	0.86	0.75
C	0.97	0.99
D	0.91	0.92
E	0.95	0.92
F	0.88	0.98
G	0.96	0.97
H	1	0.93
I	0.88	0.87
J	0.90	0.93

$$M = \begin{bmatrix} 69 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 48 & 0 & 5 & 2 & 3 & 0 & 0 & 4 & 0 \\ 0 & 0 & 73 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 59 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 54 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 61 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 66 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 & 67 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 58 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 56 \end{bmatrix}$$

3. Ejemplos de clasificacion

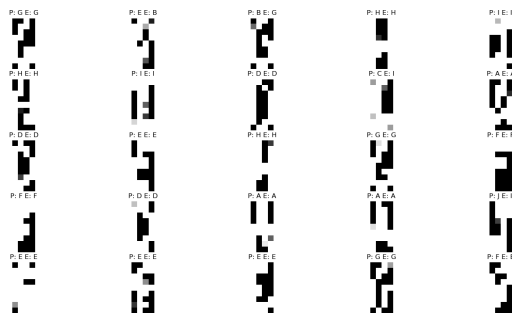


Figura 5: Ejemplos de clasificacion con random forest

3.2. Clasificador SVC

Es el algoritmo que más tarda de los 3 con los que hemos experimentado, pero a su vez es el que consigue la mejor tasa de acierto. Para el análisis de resultados de este clasificador con estos datos decidimos reducir las imágenes a 100×50 con una gamma de 0.001 obteniendo los siguientes resultados:

1. Tasa de acierto: 96.21 %
2. Matriz de confusion:

Clase	Precisión	Sensibilidad
A	0.96	0.96
B	0.92	0.89
C	0.99	1
D	0.94	0.97
E	0.97	0.97
F	0.93	1
G	1	0.97
H	1	0.97
I	0.97	0.90
J	0.92	0.97

3. Ejemplos de clasificacion

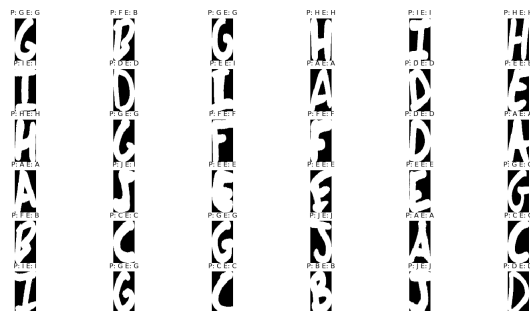


Figura 6: Ejemplos de clasificacion con SVC

3.3. Clasificador KNN

Como podremos ver más adelante este es un clasificador con el que conseguimos unos resultados medios, no tarda tanto como SVC ni tiene una tasa de acierto tan grande con imágenes tan reducidas como random forest pero aún así funcionó nos pareció suficientemente bueno como para incluirlo en la memoria.

1. **Tasa de acierto:** 94.39 %

2. Matriz de confusion:

Clase	Precisión	Sensibilidad
A	0.97	0.99
B	0.95	0.83
C	0.91	1.00
D	0.97	0.95
E	0.98	0.88
F	0.91	1.00
G	0.98	0.91
H	1	0.99
I	0.85	0.91
J	0.93	0.97

$$M = \begin{bmatrix} 69 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 53 & 0 & 2 & 0 & 3 & 0 & 0 & 5 & 0 \\ 0 & 0 & 74 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 61 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 52 & 2 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 62 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 1 & 62 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 71 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 62 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 57 \end{bmatrix}$$

3. Ejemplos de clasificacion

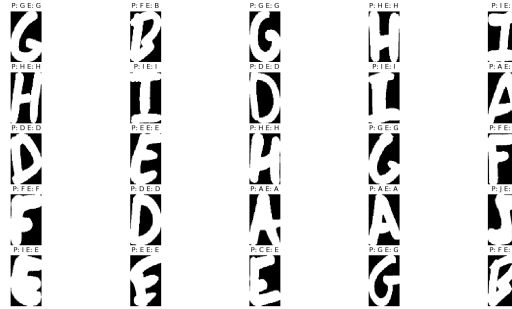


Figura 7: Ejemplos de clasificacion con KNN

3.4. Comparar los clasificadores por su tasa de acierto

Hemos decidido ir variando el número de parámetros que consideramos para ver como esto afecta tanto al tiempo de clasificacion como a la tasa de aciertos:

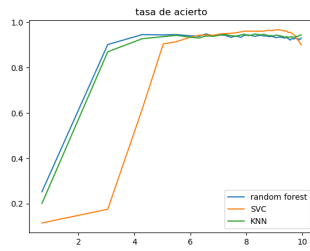


Figura 8: Tiempos de ejecucion

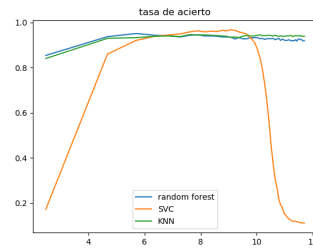


Figura 9: Tiempos de ejecucion

En esta figura muestra en el eje de las x la dimensión de los datos que usamos para clasificar en escala logarítmica y en el eje y la tasa de acierto. Como podemos ver a medida que aumentamos la cantidad de datos a usar SVC aumenta su tasa de acierto alcanzando un máximo con imágenes de tamaño 100×50 , pero al contrario que con KNN y random forest, vemos que cuando empezamos a interpolar y a hacer las imágenes más grandes que el tamaño original su tasa baja. Miremos aumentando más aún el tamaño de las imágenes:

Como podemos ver su rendimiento baja notablemente, por eso concluimos en que SVC es menos robusto que KNN y Random Forest

3.5. Comparar los clasificadores por su tiempo de ejecución

En este apartado compararemos los tiempos de construir el clasificador y clasificar de KNN, SVC y Random Forest.

Como podemos ver en la figura 10 la diferencia entre los tiempos de ejecución en el limite son bastante notables encontrando que SVC es el más lento seguido de KNN y Random Forest el más rápido.

A continuación compararemos el tiempo que tardan con su tasa de acierto, con el objetivo de poder comparar para un limite de tiempo dado cual es mejor:

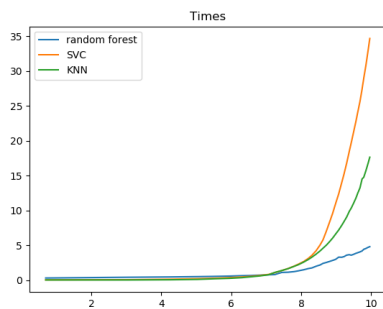


Figura 10: Tiempos clasificación

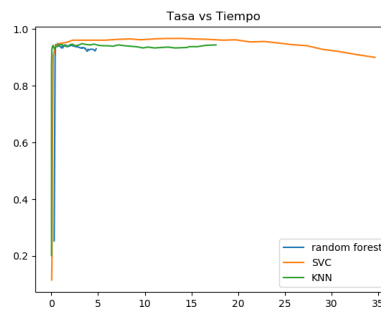


Figura 11: Tasa acierto vs tiempos

Finalmente compararemos solamente el tiempo de clasificación sin contar el tiempo de construir el clasificador.

3.6. Conclusiones

A la vista de los resultados obtenidos, dependiendo de nuestras necesidades, elegiremos distintos modelos. Si nuestra prioridad es la correcta clasificación de los datos y no tenemos una cantidad de tiempo limitada usaremos SVC, sin embargo, si necesitamos una rápida clasificación usaremos Random Forest. También es relevante dónde se aprecian los errores en la clasificación. Por ejemplo, los clasificadores suelen confundir las jotras con las ies.

3.7. Referencias

Scikit-learn