Clasificación de números escritos a mano mediante proyecciones

Alejandro Garcia Varela, Adrián Muñoz Perera, Daniel Perdices Burrero, Ana de Santos Martín

Resumen.

El proyecto que presentamos a continuación consiste en la clasificación de números a partir de las imágenes escaneadas de los mismos escritos a mano por los alumnos de Fundamentos de Aprendizaje Automático.

Para la solución del problema, presentamos un conjunto de atributos basado en proyecciones, que ha sido obtenido mediante un análisis exploratorio y comparativo de los resultados con varias iteraciones que nos han permitido escoger un modelo depurado. Tras la validación de cada modelo, el que nos ha mostrado mejores resultados en un contexto general ha sido *Random Forest* utilizando las proyecciones en los ejes X y Y como atributos.

1 Introducción

El problema que se pretende analizar es la identificación de números escritos a mano. Por tanto, los datos analizados serán matrices de enteros sin signo de 8 bits, por lo que el espacio de estudio es $[0, 255]^{wh}$ con w el ancho de la imagen y h la altura. Cada una de estas imágenes contiene, en escala de grises, un número escrito por un alumno. Nótese que estos números no tienen porque estar centrados en la imagen ni tener una escala dada.

Para esto, en este trabajo se pretende estudiar diferentes conjuntos de atributos para así poder seleccionar los que sean relevantes para el problema. Estos atributos tendrán que responder bien en conjunto a cambios de escala y desplazamientos en la medida de lo posible. Se observa que el sistema no ha de ser robusto a ningún tipo de rotación debido a que esta puede cambiar la percepción que tenemos del número.

Este problema tiene ya un estado del arte muy desarrollado, en este caso, el dataset MNIST[1] que incluye 60000 imágenes (en este caso centradas y normalizadas) ha sido estudiado de manera intensiva. Actualmente, diferentes modelos basados en redes neuronales convolucionales obtienen tasas de error menores al 1%, véase [2] y [3], lo que podría decir que el problema quedaría resuelto así. No obstante, en este trabajo queremos buscar un conjunto de atributos simple y natural que permite resumir la información de un número y que además sean robustos a cambios de escala y desplazamiento, alejándonos por tanto de las redes convolucionales.

Este documento se organiza de la siguiente manera: la sección 2 realiza un análisis exploratorio de los datos, probando diferentes técnicas que se consideran adecuadas.

Posteriormente, la sección 3 realiza un estudio de los atributos que se van a utilizar que se completa en la sección 4 en la que se aplican diferentes clasificadores. Para acabar, la sección 5 se reserva para comentar los resultados y dar unas conclusiones finales.

2 Análisis exploratorio de los datos

El dataset contiene 1200 imágenes de 302x402 píxeles (8 bits) en escala de grises. Estas imágenes no han sido tratadas y contienen ruido variado proveniente de las imperfecciones del papel, errores del escáner, etc.

Por ello, una de las primeras cosas que se hacen previo análisis es el preprocesamiento de las imágenes. Para ello se aplica un filtro de mediana [4] de kernel circular (disco) de 7 píxeles de radio que permite eliminar el ruido de sal y pimienta proveniente de la textura del folio. Además, para trabajar con una imagen más simple, se procede a realizar un proceso de umbralización [5] con un umbral que minimice la varianza inter-clase. Esto convierte la imagen en binaria. En la Tabla 1, se muestran diferentes agregaciones globales, es decir, de todos los píxeles, éstas agrupadas por clase:

Así se ven que estos atributos no son suficientes para clasificar la imágen ya que se pierde mucha información de la forma. Por ello, lo siguiente que hacemos es extraer las proyecciones X e Y, que van a resumir información de la forma y posición del número. Se muestra un ejemplo de estas proyecciones en la Fig. 1.

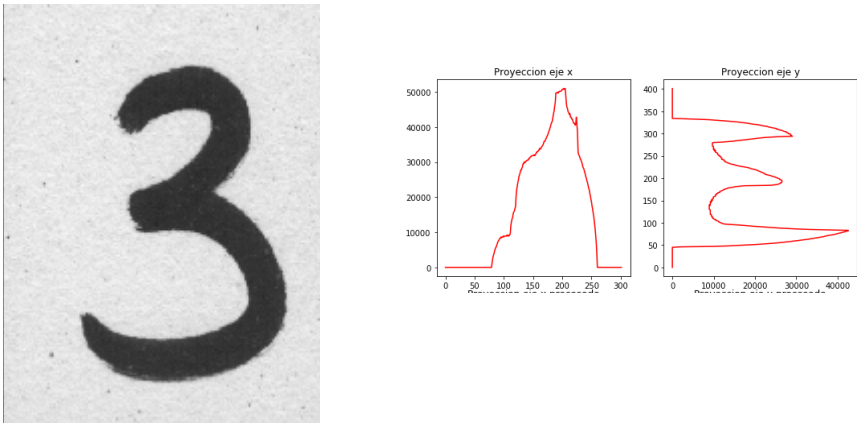

Fig. 1 Imagen de un tres con sus respectivas proyecciones, sobre el eje horizontal y sobre el eje vertical.

En la siguiente sección, se realizará un estudio más extenso de los atributos, así como de los tratamientos necesarios para que estos sean más robustos.

Clase	Rectas	Intensidades medias	Altos	Anchos
0	19.67 ± 9.43	219.50 ± 9.79	204.65 ± 40.51	165.01 ± 32.29
1	2.18 ± 1.54	230.94 ± 7.89	212.10 ± 41.93	107.98 ± 43.77
2	4.85 ± 2.61	221.92 ± 8.71	218.08 ± 40.84	152.83 ± 30.87
3	10.83 ± 6.60	220.82 ± 8.80	226.82 ± 39.83	148.82 ± 26.55
4	4.55 ± 2.19	221.96 ± 9.24	236.45 ± 47.66	148.61 ± 27.95
5	6.97 ± 5.57	219.76 ± 8.72	226.09 ± 35.14	159.25 ± 30.00
6	10.82 ± 6.96	219.92 ± 8.78	223.68 ± 36.85	146.88 ± 29.02
7	4.26 ± 2.23	222.07 ± 8.80	226.83 ± 41.32	153.32 ± 33.82
8	10.14 ± 7.24	214.45 ± 10.29	229.26 ± 38.41	147.93 ± 29.25
9	3.23 ± 2.07	219.81 ± 9.14	244.90 ± 48.48	144.09 ± 31.25

Tabla 1: Medias y desviaciones típicas de los atributos observados en el análisis exploratorio realizado previo a un análisis más exhaustivo.

3 Atributos

i. Proyecciones

Basándose en el apartado anterior, tomamos como base las proyecciones. Estas proyecciones dependen de la posición y escala del número y además contienen ruido en pequeña cantidad proveniente de trazos finos o bordes. Para tratar solucionar esto, se pretende transformar la proyección de manera que tenga las siguientes propiedades:

- Robusto frente a traslaciones: es decir, que no dependa de la posición del número dentro de la imagen.

- Robusto frente a escala: es decir, que no dependa de la escala del número dentro de la imagen.

- Longitud fija: esto es, que las proyecciones tengan siempre la misma longitud así son atributos comparables.

Para ello se propone el siguiente tratamiento: primero, se recorta la proyección, recorriendo desde el principio hasta el primer punto cuya proyección sea distinto de cero y desde el final hasta el primer punto cuya proyección sea distinto de cero, recortando ambos trozos de las proyecciones que son completamente cero y que nos indican la posición del número. Segundo, se ve que dependiendo de la imagen, en ciertos casos las proyecciones contienen cierto ruido, por lo que se realiza una convolución con un kernel media de longitud 10. Por último, para lograr la robustez frente a escala y la longitud fija, remuestreamos (o interpolamos) la proyección a 128 puntos de manera que sea robusto ante estos aspectos. Nótese que, de no realizar la convolución del segundo paso, la proyección contendría pequeño ruido y al remuestrear, se podría extraer el ruido en vez de un valor fiel de la proyección. Se muestra en la Fig. 2 el resultado visual.

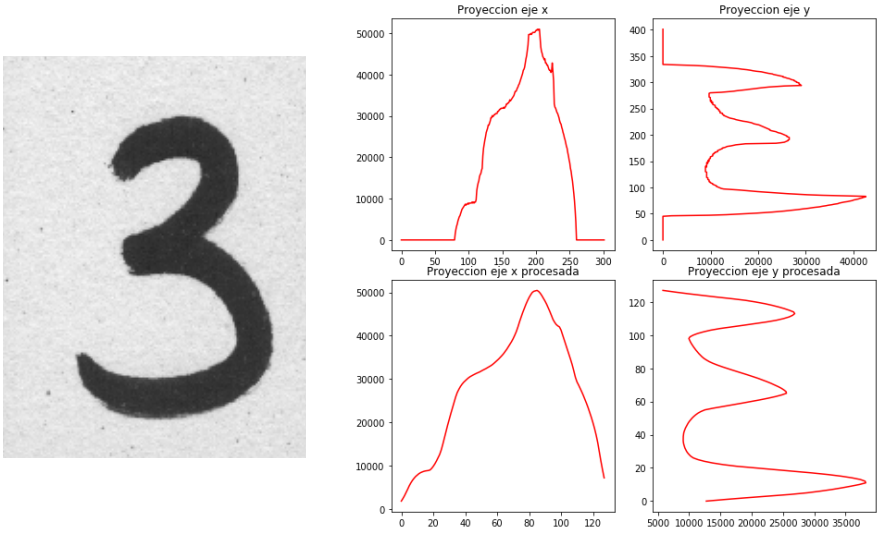

Fig. 2: Imagen del número 3 y proyecciones antes y después del procesado.

ii. Contar rectas

En vistas de extraer un atributo un poco más elaborado, se pretende contar las rectas que conforman los números. Para ello, se coge la imagen (filtrada y umbralizada) y se realiza un proceso de esqueletización morfológica, que se ha utilizado previamente para la detección de formas [6]. Esto permite reducir la imagen a trazos de grosor un pixel. Posteriormente, sobre esta, se realiza un algoritmo de detección de rectas basado en la utilización de la transformada de Hough [7]. Se muestran en la Fig. 3, las rectas superpuestas. Guardaremos como atributo el número de rectas detectadas por el algoritmo. Se muestra también los resultados por clases en la Tabla 1.

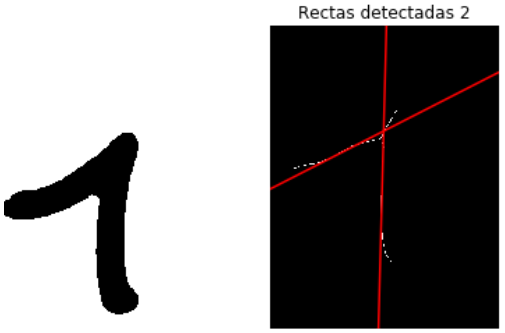

Fig. 3: Imagen del número 1 y rectas superpuestas

iii. Otros atributos

- Anchos y altos

Con las proyecciones, previo al tratamiento, se pueden guardar también los extremos del número dentro de la imagen, lo que permite tener una idea del alto o el ancho en píxeles del número dentro de la imagen. Este atributo podría ser de gran utilidad para distinguir números "estrechos" como el uno, de números más anchos como el cero o el ocho, por ejemplo. Lo mismo ocurre con el alto, los números uno y siete, normalmente, son más altos que el nueve y el seis por ejemplo. Por ello, esta técnica nos permite distinguir ciertos números de otros, pero no nos permite establecer un número concreto, sino disminuir las posibilidades de elección, facilitando la tarea.

- Luminosidad media

Previa a la umbralización, se guarda la media del valor de todos los píxeles de la imagen, conocido esto como luminosidad media o intensidad media de la imagen.

Esto permite tener una idea de la cantidad de píxeles que ocupa el número dentro de la imagen.

Modelos de clasificación y resultados

Una vez tratadas las imágenes usando un conjunto de funciones creadas por nosotros y varias de la librería scikit-image [8], ya que los equipos no disponían de OpenCV [9] correctamente instalado. Posteriormente, usamos scikit-learn [10] para tratar los datos ya en forma de matrices numéricas.

Para la validación se ha utilizado el mecanismo de validación cruzada estratificada con 5 pliegues/iteraciones (*folds*) en todos los casos.

iv. k vecinos próximos (*k Nearest Neighbours, kNN*)

Se utilizan varios k con resultados similares en todos los casos. Además, se utilizan más métricas aparte de la euclídea. Para este clasificador, solo se han usado los atributos de las proyecciones, ya que los otros atributos frente a los 256 puntos de estas proyecciones eran despreciables y solo añadían ruido. Se muestra en la Tabla 2 estos valores de score entre 0 y 1 según la métrica y el k.

Número de vecinos (k)	Score	
	Métrica Hellinger	Métrica Minkowsi
1	(0.86 ∓ 0.01)	(0.83 ∓ 0.01)
3	(0.85 ∓ 0.02)	(0.82 ∓ 0.01)
5	(0.85 ∓ 0.02)	(0.82 ∓ 0.02)
7	(0.85 ∓ 0.02)	(0.83 ∓ 0.03)
11	(0.85 ∓ 0.02)	(0.82 ∓ 0.03)
13	(0.85 ∓ 0.02)	(0.82 ∓ 0.03)
15	(0.85 ∓ 0.03)	(0.81 ∓ 0.02)

Tabla 2: Comparación entre los vecinos y las métricas para el clasificador kNN. Se dan medias y desviaciones típicas del *Score* obtenidos

v. Naive-Bayes

Se prueba el clasificador Naive-Bayes normal (*GaussianNB* en scikit). Este resulta en alrededor de una media del 85% de tasa de acierto con una desviación típica del 1%.

En este caso, tanto el conjunto de atributos completos (proyecciones y el resto) o solo las proyecciones, no era significativo en el resultado del clasificador

vi. Random Forest

Para el caso de Random Forest, aplicamos los atributos, tanto de las proyecciones como los mencionados anteriormente. Con esto obtuvimos un score del 86%, pero, comparando con los atributos de proyecciones, con el que obtuvimos un 88%, nos dimos cuenta de que, para este modelo de clasificación, los atributos empleados para contar rectas o la luminosidad no aportaban nuevos datos al modelo y "perjudicaban" su decisión de clasificación.

vii. Árboles de Decisión

Los árboles de decisión no parecen a priori el mejor clasificador para probar esto, ya que las proyecciones no se van a comportar bien dentro de un árbol de decisión, pero nos parecía una manera útil de ver cómo de significativos eran los atributos extra más allá de las proyecciones. En el caso de todo el conjunto de atributos, se obtuvo una media 76% de score con una desviación típica del 1%. Para el conjunto formado solo por las proyecciones, se obtuvo un 77% con una desviación típica del 1%. Esto nos viene aún más confirmar la hipótesis de que la proyecciones por sí solas son suficientes para clasificar bien y los atributos adicionales no aportan un valor adicional significativo.

4 Conclusiones

En este trabajo hemos utilizado diferentes técnicas de visión asistida por ordenador (*Computer Vision)* y de aprendizaje automático. Con esto hemos logrado obtener atributos sencillos que van más allá del actual estado del arte basado en redes convolucionales.

Principalmente, el atributo que ha dado mejores resultados es la proyección (debidamente tratada) en X e Y. Este atributo ha resultado resumir adecuadamente la forma del número. Otros atributos realizan un papel menos importante pero sin embargo permiten distinguir bien ciertos números de otros. Al comparar los atributos y ver la significancia, observamos que estos atributos adicionales no aportan mucha más información y teniendo en cuenta su coste computacional, no recomendamos su uso salvo en escenarios donde el coste computacional no sea un problema.

Como se ha mostrado en la sección anterior, el resultado es bastante satisfactorio para los distintos clasificadores empleados sin llegar a sobreaprendizaje. Creemos que no es buena idea dar un método de clasificación universal sino dar situaciones en las que cada uno de los métodos responde mejor. Por un lado, si queremos un método que reconozca la escritura de un grupo reducido de personas que van a proveer previamente muestras para el entrenamiento, se recomienda k vecinos próximos con k pequeño, ya que las proyecciones han de estar cerca de los de entrenados. Para

conjuntos de entrenamiento más grandes, generales y heterogéneos, k vecinos próximos puede resultar muy costoso en clasificación como para funcionar con aplicaciones en tiempo real. Por otro lado, Random Forest presenta una opción más costosa en entrenamiento pero fácilmente paralelizable en clasificación.

Mencionar además que de las técnicas estudiadas, se han utilizado Random Forest como clasificador y búsqueda en rejilla (Grid search) para la estimación de hiperparámetros. También se probó el clasificador de regresión logística con la estrategia *One vs All*, sin embargo, su tasa de acierto rondaba el 20% y fue eliminado del documento por no considerarse suficiente.

References:

[1] Y. LeCun and C. Cortes, "MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[2] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.

[3] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," *arXiv ePrints*, Feb. 2012.

[4] M.-H. Hsieh, F.-C. Cheng, M.-C. Shie, and S.-J. Ruan, "Fast and efficient median filter for removing 1–99% levels of salt-and-pepper noise in images," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, Apr. 2013.

[5] N. Otsu, "Threshold Selection Method From Gray-Level Histograms," *IEEE T Syst Man Cyb*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[6] P. . Trahanias, "Binary shape recognition using the morphological skeleton transform," *Pattern Recognition*, vol. 25, no. 11, Nov. 1992.

[7] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, Jan. 1972.

[8] S. van der Walt *et al.*, "scikit-image: image processing in Python.," *PeerJ*, vol. 2, p. e453, Jun. 2014.

[9] Intel, "OpenCV - Open Source Computer Vision Library 1.0." 2006.

[10] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.