Reconocimiento de dígitos escritos a mano mediante métodos de tratamiento de imagen y modelos de clasificación

Luis Miralles Pechuán, Dafne Rosso Pelayo, Jorge Brieva

Facultad de Ingeniería, Universidad Panamericana,
México

{lmiralles,drosso,jbrieva}@up.edu.mx

Resumen. El ROC (Reconocimiento Óptico de Caracteres) es una línea de investigación dentro del procesamiento de imágenes para la que se han desarrollado muchas técnicas y metodologías. Su objetivo principal consiste en identificar un carácter a partir de una imagen digitalizada que se representa como un conjunto de píxeles. En este trabajo realizamos para el ROC un proceso iterativo que consta de cinco fases. Para ello aplicamos varios métodos de tratamiento de imagen, dos métodos de selección de variables y exploramos diversos métodos supervisados de aprendizaje automatizado. Entre los modelos de clasificación destacamos los de Deep Learning por su novedad y su enorme potencial.

Palabras clave: reconocimiento óptico de caracteres (ROC), modelos predictivos y selección de variables, métodos de clasificación, deep learning, métodos supervisados de aprendizaje automatizado.

1. Introducción

El ROC (Reconocimiento Óptico de Caracteres) consiste en identificar un símbolo, que suele ser un número o una letra, a partir de la digitalización de una imagen [1], [2]. Actualmente existen múltiples procesos en los que se aplica el reconocimiento de caracteres.

Algunos ejemplos de ROC pueden ser: automatizar la redirección de cartas en el correo postal, el reconocimiento de las matrículas de los coches en los radares o la digitalización de los apuntes tomados en una clase mediante la escritura con lápiz óptico en una tablet.

Nuestra investigación consiste en efectuar la identificación de dígitos a partir de un conjunto de pixeles que representan números escritos a mano [3]. La aportación que realizamos es una metodología compuesta de 5 fases para la identificación de dichos números. Para ello, aplicamos la reducción de dimensionalidad, la extracción de características de la imagen original, la selección de variables y la evaluación de varios modelos supervisados de aprendizaje automatizado.

Para la selección de variables hemos incorporado técnicas de PCA (Análisis de componentes principales) [4] y RFE (Eliminación Recursiva de Características) [5] que permiten crear modelos más eficientes para el ROC. Con estas técnicas el tamaño del conjunto de datos decrece y el tiempo de clasificación de los caracteres se reduce. En nuestro caso existen algunas entradas que mantienen siempre el mismo valor o que presentan muy poca variación en sus datos. Estas variables serán descartadas.

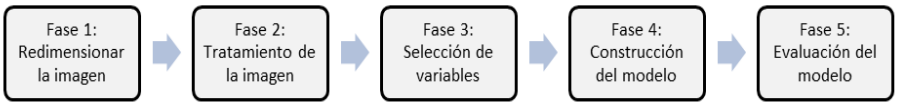

Fig. 1. Fases de la investigación.

El método conformado por 5 fases (Figura 1) incorpora el tratamiento de imágenes así como las fases iterativas correspondientes a la creación de modelos de aprendizaje automatizado.

Para construir un modelo adecuado es importante que las muestras de datos sean de calidad y tengan el suficiente número de muestras. Para la investigación emplearemos el nivel de *accuracy* o precisión. Este valor representa el porcentaje de aciertos en el rango [0,1] para evaluar la calidad de los modelos presentados. Para finalizar, presentaremos una tabla resumen con los resultados de la evaluación.

2. Descripción de la base de datos

Normalmente, el proceso de ROC inicia con la digitalización de las imágenes de escritura realizada a mano que contienen los dígitos, en nuestra investigación partimos de un data set público [3].

Este data set está compuesto por un total de 70,000 muestras [6] de las cuales elegimos 42,000 muestras de manera aleatoria. Cada muestra tiene 785 campos. Un campo que consideramos la salida del modelo representa el dígito y el resto de los campos son las entradas. Las entradas son los píxeles de la imagen que representa el dígito escrito a mano. Cada imagen tiene una resolución de 28x28, lo que hace un total de 784 píxeles. Cada pixel se representa con un número entre el intervalo de 0 y 255 que indica el nivel de gris. De forma que el valor 0 representa el color blanco y el valor 255 el color negro.

3. Fase 1: Redimensionar la imagen

Al número de pixeles que tiene una imagen le llamamos resolución. Cuantos más pixeles usemos para representar una imagen mayor será la resolución. Cada imagen viene representada por un conjunto de pixeles en forma de matriz. Si la imagen es en color tendremos tres valores RGB (Red-Green-Blue) para representar a un píxel. En nuestro caso la imagen está en escala de grises por lo que cada pixel está entre 0 y 255.

Como las imágenes son un insumo de los modelos y estos tienen un número de entradas fijo, debemos homogeneizar las imágenes para que presenten el mismo tamaño. En algunos casos deberemos disminuir el tamaño de la imagen con lo que perderemos información y en otros casos agrandaremos la imagen con lo que deberemos de interpolar los píxeles.

A partir de la imagen tratada, se pueden extraer algunas características que pueden resultar de gran utilidad para aumentar el grado de acierto en las predicciones del modelo. Podemos medir algunas características como: la media de los píxeles, el número de píxeles mayores a un umbral o el número de píxeles que forman una región. También podemos aplicar algoritmos para detectar el tipo de textura o el número de agujeros que tiene cierto símbolo.

En nuestro conjunto de datos, cada imagen está representada por un arreglo bidimensional de píxeles en escala de grises de 28 x 28.
Por lo tanto cada imagen está representada por 785 atributos:

- Digito en la imagen, su valor es entero desde 0 hasta 9.
- 784 valores en los píxeles (resultado de los 28x28 píxeles), toman valores enteros de 0 hasta 255, siendo el valor 255 la intensidad más oscura (negro) y el 0 la intensidad más clara (blanco).

Trabajar con un conjunto de datos del tamaño 42,000 (número de imágenes) por 197 (número de atributos) es complicado porque cuanto más grande es la dimensión de los datos la creación de los modelos de aprendizaje es más compleja y más lenta. Por ello, es conveniente reducir la dimensión de los datos.

El primer paso para reducir la dimensión de los datos es reducir el tamaño de las imágenes. Esto es, en lugar de que cada imagen sea de 28 x 28 píxeles, trabajaremos con una imagen de 14 x 14 píxeles. Para lograr esto, los valores de cuatro píxeles continuos en la imagen original se promedian para obtener un píxel en la imagen de menor escala (Figura 2).

125	136	252	253
109	95	126	52
125	236	123	145
59	87	38	74

116	171
127	95

Fig 2. Reducción del tamaño de la imagen

Después de realizar este proceso, cada imagen está representada por 197 atributos:

- Digito en la imagen, su valor es entero desde 0 hasta 9.
- 196 valores en los píxeles (resultado de los 14x14 píxeles), valores enteros de 0 hasta 255.

4. Fase 2: Tratamiento de la imagen

El tratamiento de imágenes es un conjunto de técnicas que se utiliza para mejorar la calidad de la imagen o para facilitar la búsqueda de información en ellas. Algunos ejemplos podrían ser eliminar el ruido, localizar bordes o suavizar la imagen.

Las técnicas más utilizadas y que podemos aplicar a nuestro problema y que aplicaremos en la fase dos son:

- **Binarización:** Consiste en convertir todos los píxeles de una imagen originalmente tengan varios valores a solamente dos tonos: blanco y negro. Para ello se establece un umbral por encima del cual convertiremos en negro y por debajo del cual será blanco. Esta operación es muy frecuente puesto que algunos algoritmos trabajan en esta escala. Las imágenes ocupan un tamaño en memoria muy pequeño puesto que cada píxel se puede representar con un píxel.
- **Fragmentación o segmentación:** Esta técnica consiste en seleccionar una parte de la imagen original. Para ello se desarrollan algoritmos que se basan en la detección de bordes que son pequeños cambios en la intensidad de los colores y otras propiedades. Existen muchos métodos genéricos de fragmentación aunque para cada tipo de imagen se suele aplicar un algoritmo más específico. Por ejemplo, para detectar un cáncer de mama primeramente se aplica un filtro sobre las regiones sospechosas y posteriormente se aplica un algoritmo que evalúa ese conjunto de píxeles.
- **Adelgazamiento de componentes:** Uno de las técnicas más comunes consiste en borrar de manera iterativa los puntos de los contornos. Esta técnica se debe aplicar con cautela para que las imágenes no pierdan su forma original y de manera iterativa. Este método simplifica la identificación de caracteres y permite la extracción de características como la altura de la imagen según los pixeles o el ancho de los píxeles que forman la imagen.
- **Calcular promedio de píxeles:** Este método de tratamiento de imágenes consiste en tomar todas las muestras de un determinado dígito y crear una nueva imagen. Donde cada píxel de esa imagen sea el promedio de todas las imágenes de que representen ese dígito. Por ejemplo, el pixel [1,1] de la imagen promedio de ocho, es el promedio del píxel [1,1] de todas las muestras del número ocho.

  El algoritmo procesa las imágenes de todos los dígitos y como resultado obtenemos 10 imágenes con el promedio para todos los píxeles. Esas diez imágenes nos servirán como entradas para crear un nuevo modelo. Pero su tasa de acierto es muy baja por lo que deberemos encontrar un método mejor.
- **Eliminación de filas y columnas en blanco:** Uno de los procedimientos que vamos a realizar en el tratamiento de imágenes es eliminar aquellas filas y columnas que estén blancas y casi blancas. Es decir vamos a eliminar los contornos de todas las imágenes tanto del data set ya que aportan poca información.

  Para realizar esto aplicamos un método que suma los píxeles, como el blanco se representa con cero si la suma no es mayor que un determinado umbral eliminaremos esa fila. Lo mismo para las columnas.
- **Selección de Características del objeto:** En esta etapa se extraen características del objeto presente en la imagen después de una binarización simple (Se tomaron los pixeles mayores que cero). El objeto en la imagen es caracterizado mediante un vector con las siguientes componentes: Área, Centroide (componente x,y), Longitud del eje mayor, Longitud del eje menor, Excentricidad, Número de Euler, Diámetro equivalente, Solidez,

Perímetro, Centroide proporcional al nivel de gris (Componentes x, y), Grado, Intensidad Máxima, Intensidad Mínima, e Intensidad promedio. Este vector será la entrada al clasificador en los siguientes procesos.

5. Fase 3: Selección de variables

Una vez completadas las fases anteriores utilizaremos PCA y RFE para seleccionar las características más importantes y descartar las menos relevantes.

El PCA es una técnica estadística empleada en el descubrimiento de patrones en conjuntos de datos con una alta dimensionalidad. Consiste en la selección de componentes que otorgan una mayor aportación de información para el modelo.

- Información mutua y PCA: La entropía [4] es una medida de incertidumbre en una variable aleatoria discreta, esto es, mide que tan uniforme es una variable. En este caso, si la variable aleatoria tiene una distribución uniforme, la entropía es máxima. Para calcular la entropía se utiliza la siguiente función:

$$\mathrm{H(X)} = -\sum_{\mathrm{x}} \mathrm{P(x)}\ln\big(\mathrm{P(x)}\big)$$

La información mutua es una medida de dependencia entre dos variables aleatorias discretas, esto es porque mide la reducción de la incertidumbre (entropía) de una variable, de acuerdo al conocimiento de otra variable. Para el cálculo de la información mutua se utilizan las siguientes ecuaciones:

$$\mathrm{H(X,Y)} = -\sum_{\mathrm{x}}\sum_{\mathrm{y}} \mathrm{P(x,y)}\ln\big(\mathrm{P(x,y)}\big)$$

$$\mathrm{IM(X,Y)} = \mathrm{H(X)} - \mathrm{H(Y)} - \mathrm{H(X,Y)}$$

En el caso de la clasificación de los dígitos manuscritos, las variables más importantes para generar los modelos de clasificación deben coincidir con los píxeles que reducen la incertidumbre de la clase. Esto es, las variables de los píxeles que tengan mayor información mutua. Por lo tanto se evalúa la información mutua de 196 variables y se seleccionan las 150 variables con mayor información mutua. Al analizar este proceso se observa que algunas de las variables descartadas coinciden con los píxeles de los bordes de la imagen, esto es lógico debido a que los bordes generalmente no contienen información importante.

Método recursivo de selección: El método RFE *(Recursive Feature Selection)* o eliminación recursiva de características es un método orientado a la selección de variables. Este método es de tipo *wrapper*, es decir, se van construyendo modelos a partir de una combinación de entradas. [5].

En primer lugar el algoritmo clasifica las entradas del modelo de mayor a menor relevancia. Para ello, construye modelos con cada una de las variables de forma independiente y evalúa los resultados.

Posteriormente va construyendo modelos de la siguiente manera: En primera instancia con la primera variable, en segunda instancia con la primera y segunda variable, en tercera instancia con la primera, la segunda y la tercera y así

sucesivamente para todas las variables. La salida del algoritmo es una gráfica con todas las agrupaciones pero señalando la mejor combinación.

A partir de los resultados obtenidos en la evaluación de modelos se van descartando aquellas variables que son menos importantes o que generan ruido. El objetivo es doble: quedarnos con un número de variables más reducido y elegir la combinación que nos dé mejores resultado. Esto nos permitirá construir modelos más sencillos y más precisos. La aplicación de RFE redujo el número de variables de 196 a 148.

6. Fase 4: Construcción del modelo

6.1. Metodología en la construcción de los modelos

Para la construcción y evaluación de modelos utilizaremos una librería llamada Caret. Caret responde a las siglas: *"Classification And REgression Training"* [7], [8] . Esta librería permite construir modelos en pocas líneas de código y de forma muy sencilla.

A continuación mostramos el código necesario para la construcción de un modelo empleando *Random Forest*:

```
1) control = trainControl (method = "cv" , number = 10, classProbs = T)
2) grid = expand.grid ( .mtry = c ( 5, 10, 25, 50, 75, 100 ) )
3) modeloRForest = train ( Entradas, Salida, method = "rf", tuneGrid =
grid ,ntree = 1000 ,trControl = control )
```

En la primera instrucción trainControl sirve para indicar el número de modelos que ejecutaremos por cada configuración. En la segunda instrucción "expand_grid" indica el número de árboles de tipo Random Forest que se van a utilizar. La tercera instrucción sirve para crear y evaluar el modelo. Esto lo hacemos con la función "train" que tiene como parámetros: una matriz con las entradas, un arreglo con las salidas y el método que se usará en este caso el Random Forest. El número de árboles que emplearemos serán 1000. Las variables "control" y "grid" las hemos explicado en las dos primeras instrucciones.

A continuación presentamos el algoritmo genérico para la construcción de los modelos:

```
Algoritmo de construcción de métodos
- Leer el Data Set
- Seleccionar variables más significativas mediante Random forest o PCA
- Desde modelo: 1 hasta N
               - Crear conjuntos de muestras diferentes mediante CV
               - Desde configuración 1 hasta configuración N
                       - Crear varios modelos con una parte Training Set
                       - Probar el modelo con la partición restante
               - Fin Desde
               - Elegimos la media de los modelos y el mejor resultado
- Fin Desde
- Clasificar los métodos según los mejores resultados
Fin del Algoritmo
```

7. Métodos de clasificación utilizados

Para la construcción de modelos seleccionamos algunos de los clasificadores más conocidos como son: Random Forest, SVM, KNN, Redes Neuronales y experimentamos con métodos de aprendizaje profundo Deep Learning.

Deep Learning: Es un grupo de algoritmos que implementan una metodología basada en crear modelos abstractos a partir de las entradas [9]. Estos algoritmos utilizan una serie de técnicas de aprendizaje orientadas a extraer características de los datos de manera jerárquica. Es decir, las características de nivel superior se forman a partir de las características de los niveles inferiores. Esta técnica permite construir modelos predictivos más precisos y con mejores resultados para un amplio número de problemas [10].

Por ejemplo, una imagen en Deep Learning se puede representar como un conjunto de bordes, una serie de formas o un número de huecos. En lugar de simplemente una combinación de píxeles como hacen la mayoría de los algoritmos.

Estas técnicas de aprendizaje aplican transformaciones no lineales para hacer representaciones de los datos. Para ello utilizan un conjunto de capas concatenadas donde la salida de una capa es la entrada de la siguiente capa. Cuanto más grandes sean las capas y mayor sea su número más aumenta la capacidad de abstracción.

Algunos algoritmos de Deep Learning están basados en la neurociencia y simulan el comportamiento de las neuronas en el cerebro y en el sistema nervioso. El algoritmo trata de establecer una relación entre un estímulo y sus respuestas.

A diferencia de los algoritmos tradicionales que se basan en desarrollar métodos muy específicos para resolver problemas concretos. El Deep Learning trata de crear técnicas útiles orientadas a problemas genéricos.

Las aplicaciones de estos algoritmos pueden ir desde la detección de fraude en las tarjetas de crédito a la predicción de resultados bursátiles. Estos algoritmos dan buenos resultados en el reconocimiento de imágenes y en el procesamiento del lenguaje natural.

Deep Learning H20: El H20 Deep Learning es un paquete de R Studio que se basa en redes neuronales multicapa con alimentación hacia adelante. Estas redes se entrenan con descenso de gradiente estocástico utilizando retropropagación.

Este modelo descarga a través de Internet una serie de datos empleados en la configuración de los nodos. Estos datos son el resultado de la aplicación del algoritmo a muchos problemas a nivel mundial. Es posible descargar bajo pago ciertas librerías que mejoran sensiblemente tanto los resultados como la rapidez en la ejecución de estos algoritmos.

Random forest: Son una combinación de árboles predictivos donde para cada árbol *k* se genera un vector aleatorio γ_k, creado de manera independiente a los $\gamma 1$ ... γ_{k-1} vectores y con la misma distribución empelada en cada árbol *k* del conjunto (*forest*). El árbol se construye usando el conjunto de entrenamiento y γ_k, dando como resultado un clasificador $h(x, \gamma_k)$ donde x es el vector de entrada. Se selecciona la clase más popular para x, después de que fueron generados un numero suficientemente grande de arboles. [11]

KNN (K-Nearest neighbors): Es un algoritmo de clasificación basado en el cálculo de la distancia más cercana, normalmente distancia Euclidiana, entre los

puntos del conjunto de datos a un punto dado, donde *K* es el número de vecinos cercanos seleccionados. [12]

SVN (Support Vector Machines): es un clasificador que separa los datos por medio de hiperplanos. En él se busca por el hiperplano óptimo, el cual, es el hiperplano que brinda la máxima distancia desde los patrones de entrenamiento más cercanos, los SV son entonces los vectores más cercanos a estos patrones [13].

8. Fase 5: Evaluación del modelo y muestra de resultados

La evaluación consiste en medir el grado de precisión de cada modelo generado. Esta fase informa si la dimensionalidad elegida es adecuada para la construcción del modelo. Para medir la calidad de los modelos utilizamos el *accuracy* de cada método. Para calcular la exactitud sumamos los elementos de la matriz que forman parte de la diagonal principal y los dividiremos entre el total de elementos.

Podemos apreciar en la Tabla 1, los resultados de aplicar el método SVM con 10.000 muestras de entrenamiento y 15.000 para el test.

Tabla 1. Matriz de confusión con el método SVM

		Valores predichos									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Valores Reales	**0**	1425	0	6	2	2	4	7	1	5	5
	1	0	1677	3	2	3	2	0	6	6	1
	2	1	7	1421	17	14	3	0	11	5	2
	3	2	0	9	1503	2	29	0	7	11	8
	4	2	2	3	1	1413	6	3	8	9	25
	5	7	0	0	18	2	1279	7	0	10	5
	6	4	3	2	2	9	12	1454	0	6	0
	7	0	6	15	9	7	1	0	1507	3	18
	8	6	1	11	29	2	10	3	3	1385	14
	9	1	2	4	5	28	3	0	19	10	1398

Tabla 2 Fases en el procesamiento de imágenes

Fase 1: Reducción imagen	**Fase 2: Tratamiento de imagen**	**Fase 3: Selección de variables**	**Fase 4: Construcción de modelos**	**Fase 5: Resultados**
No hacer nada	Ancho de píxeles, Altura píxeles	No extraer carac-terísticas	KNN	**0,9425**
Imagen 14x14	Quitar filas en blanco	PCA	Random forest	**0,9315**
Imagen 28x28	Binarizar	RFE	Neuronal Network	**0,9625**

La exactitud la podemos calcular con la siguiente fórmula:

$$Accuracy = \frac{\sum_{i=1}^{K} F_{ii}}{N}$$

Tabla 3. Comparativo de los resultados de los modelos para imagen de 28x28

Reducción Imagen	Tratamiento imagen	Selección de Variables	Modelo Predictivo	Parámetros modelo	Resultado
Imagen 28x28	Aplicación de múltiples métodos como densidad, anchura, tamaño, …	RFE	Random Forest	Árboles: 1000 - mtry: 75	**0.7373**
	Promedio de píxeles	No hacer selección	Random Forest	Árboles: 5	**0.171**
				Árboles: 20	**0.1615**
				Árboles: 50	**0.3011**

Tabla 4. Comparativo de los resultados de los modelos para imagen de 14x14

Reducción Imagen	Tratamiento imagen	Selección Variables	Modelo Predictivo	Parámetros modelo	Resultado
Imagen 14x14	Sin tratamiento	PCA	Random Forest	Árboles: 1000 - mtry: 75	**0.9284**
		RFE	Random Forest	Árboles: 2000 - mtry: 75	**0.9650**
				Árboles: 1000 - mtry: 2	**0.9541**
				Árboles: 1000 - mtry: 75	**0.9653**
				Árboles: 1000 - mtry: 148	**0.9516**
			SVM con núcleo polinómico	degree = 3, scale = 0.1 and C = 0.25.	**0.9722**
			Neural Networks con extracción de características	size = 5 and decay = 0.1	**0.7518**
	Reducción de bordes	Todas	Random Forest	Árboles: 1000 - mtry: 75	**0.9330**
			Deep Learning	TanhWithDropout	**0.4363**
				Tanh	**1**
				RectifierWithDropout	**0.9998**

En la matriz de confusión cada columna representa el número de predicciones de cada clase y cada fila representa las instancias de la clase real. La precisión o *accuracy* del modelo es: 0.9641.

Los resultados obtenidos en el segundo problema los representamos en la tabla 2. El experimento fue realizado utilizando un total de 42,000 muestras con un CV de 10 particiones. Los resultados se muestran en tabla 2.

En la tabla 3 mostramos el nombre del modelo, los parámetros que lo configuran y el promedio de la precisión de los 10 modelos creados.

En la tabla 4 mostramos los resultados de algunos métodos tradicionales y de los métodos Deep Learning. Estos métodos son los que mejor se adaptan a estas imágenes. Posiblemente por su capacidad de abstracción.

En la tabla se muestra en cada columna los métodos aplicados en cada una de las fases. Como se aprecia en los resultados los métodos Deep Learning son los que tienen mayor precisión.

9. Conclusiones

Después de haber trabajado con R Studio y la librería Caret lo calificamos como un entorno muy adecuado para resolver problemas relacionados con la construcción y evaluación de modelos predictivos.

Las principales ventajas de esta herramienta son que permite crear modelos con distintas configuraciones, la selección automática de la mejor combinación y la muestra de resultados en distintas métricas. Además, contempla la opción de *cross-validation*, que crea diferentes particiones para crear varios modelos y así hacer cálculos más robustos.

El hecho de que sea software gratuito representa una gran ventaja porque permite a estudiantes y a personas de la comunidad científica disfrutar de una herramienta con una amplia gama de funciones. Un inconveniente de R es que no tienen ningún elemento que indique la estimación del tiempo restante. Esto dificulta hacer una planificación eficiente para probar los modelos. Tampoco da garantías en caso de que algún error ocurra como indica claramente al inicializar el intérprete de comandos.

La documentación de R no recoge todos los detalles de las funciones por lo que en algunos casos resulta laborioso encontrar la configuración de algunos parámetros. Esto nos ha sucedido en el caso de las funciones RFE. Por último, vemos que es posible implementar esta herramienta con otros lenguajes como Java.

Consideramos que el redimensionamiento de la matriz para trabajar con imágenes más reducidas ha permitido ahorro de tiempo y complejidad en la creación de los modelos.

Podemos comprobar en la tabla 4, que nuestro método propuesto basado en la reducción de dimensionalidad y la selección de características permitieron obtener un acierto del 100% en el reconocimiento de los dígitos escritos a mano. Aplicando *Random Forest* en el conjunto original sin reducción de la imagen se obtiene una exactitud de 0.7373 mientras que aplicando los elementos descritos en las fases del método se logra mejorar a una precisión de 0.9653, mientras que los algoritmos de Deep Learning lograron alcanzar un 0.9998 y un 1 de exactitud. Por lo que concluimos que esto método tiene un gran futuro en los modelos predictivos.

10. Trabajos futuros

El sistema que hemos creado parte de una fase en la que se localizan los dígitos y se centran en una imagen. De forma que partimos de imágenes ya recortadas que facilitan en gran medida el reconocimiento. Pero en la vida real los caracteres no están aislados sino que se encuentran dentro de textos. Una manera de segmentar este problema es descomponer los textos en pequeñas imágenes que reconozcan solamente un dígito.

Un siguiente paso en la investigación sería aplicar nuestro modelo a imágenes en color. Una forma sencilla sería pasar las imágenes a escala de grises y aplicar nuestra metodología para evaluar los resultados.

De la misma forma que hemos conseguido crear un modelo con una gran precisión para el reconocimiento de números se podría evaluar la misma metodología para el reconocimiento de letras. El estudio sería interesante, ya que si hemos alcanzado un nivel de precisión alto en el reconocimiento de dígitos esperamos buenos resultados para el reconocimiento de letras.

Una línea de investigación interesante para tener en cuenta la estabilidad de los algoritmos sería incrementar el ruido en las imágenes puesto que sabemos que en la vida real las imágenes están sujetas a todo tipo de distorsión. Entendemos por esto que el papel tenga otro color de fondo, que los números puedan ser diferentes, etc. Para ello se aplicaría un algoritmo de generación de ruido sobre las imágenes del conjunto de datos y se volvería a crear un modelo para evaluarlo.

Teniendo en cuenta los resultados obtenidos con los métodos de Deep Learning nos parece interesante seguir profundizando en este tipo de algoritmos. Pensamos que puede ser de interés general un estudio comparativo de los métodos Deep Learning Para ello, utilizaremos varios conjuntos de datos públicos y compararíamos tanto la precisión de los resultados y como el tiempo necesario para la construcción de modelos.

Bibliografía

1. Mori, S., Nishida, H., Yamada, H.: Optical character recognition. John Wiley & Sons, Inc (1999)
2. Vithlani, P., Kumbharana, C. K.: A Study of Optical Character Patterns identified by the different OCR Algorithms. Department of computer science, Saurashtra University, Rajkot, India (2012)
3. Mohiuddin, K., Mao, J.: A comparative study of different classifiers for handprinted character recognition. Pattern Recognition in Practice, Vol. IV, pp. 437–448 (2014)
4. Bro, R., Smilde, A. K.: Principal component analysis. Analytical Methods, Vol. 6, no. 9, pp. 2812–2831 (2014)
5. Zhang, X., Lu, X., Shi, Q., Xu, X. Q., Hon-chiu, E. L., Harris, L. N., Wong, W. H.: Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data. BMC bioinformatics, Vol. 7, no. 1, p. 197 (2006)
6. Le Cun, Y., Cortes, C.: MNIST handwritten digit database. AT&T Labs. En línea: http://yann. lecun. com/exdb/mnist [Último acceso: 15 Enero 2015]
7. Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A.: Caret: classification and regression training. R package, Vol. v515 (2012)

8. Kuhn, M.: Building predictive models in R using the caret package. Journal of Statistical Software, Vol. 28, no. 5, pp. 1–26 (2008)
9. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks, Vol. 61, pp. 85–117 (2015)
10. Weigel, V. B.: Deep Learning for a Digital Age: Technology's Untapped Potential to Enrich Higher Education. Jossey-Bass (2002)
11. Breiman, L.: Random Forests. In: Machine Learning. Kluwer Academic Publishers, no. 45, pp. 5–32 (2001)
12. Yu, X., Pu, K. Q., Koudas, N.: Monitoring k-Nearest Neighbor Queries Over Moving Objects. In: Proceedings of the 21st International Conference on Data Engineering (2005)
13. Duda, R. O., Hart, P. E., Stork, D. G.: Pattern Classification, USA: John Wiley & Sons (2012)