Trabajo Final
Redes Neuronales

Silvina Moyano
Franco Rodríguez Fabregues

05 de agosto de 2011

1 . Introducción

El objetivo de este proyecto es desarrollar un detector de código CAPTCHA utilizando el algoritmo de Backpropagation con la posibilidad de incluir Momentum.

CAPTCHA es el acrónimo de Completely Automated Public Turing test to tell Computers and Humans Apart (Prueba de Turing pública y automática para diferenciar máquinas y humanos). Es decir, un CAPTCHA es un programa que protege sitios web contra bots. Generando y clasificando pruebas que los seres humanos pueden pasar pero los programas actuales no. Ver Figura 1.

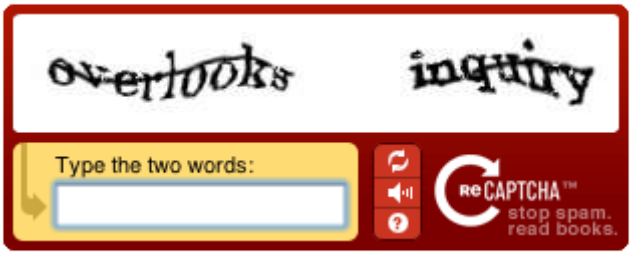

Figura 1: Ejemplo código CAPTCHA.

2 . Implementación

Para implementar el detector de código CAPTCHA se desarrolló una aplicación en lenguaje Python con una interfaz gráfica Qt. La llamamos Cabsha. Dividimos el informe en diferentes fases o etapas para explicar el funcionamiento de la aplicación.

2 .1. Fase de preprocesamiento

Se cuenta con una galería de imágenes CAPTCHA, en formato BMP de 155x40 píxeles. Ver Figura 2.

Este procedimiento consiste en realizar un filtrado de moda con una ventana de 3x3 para remover el ruido, ver Figura 3.

El paso siguiente es separar los caracteres de la imagen original en 5 nuevas imágenes y cambiar el tamaño de las imágenes obtenidas con el fin de que todas tengan tamaño 16x16 píxeles para facilitar el procesamiento.

Por último, estas imágenes son transformadas en 5 matrices de 0's y 1's. Los 0's representan los píxeles de la imagen que no forman parte de la letra. Los 1's por el contrario representan los píxeles que pertenecen al caracter. Ver Figura 4 De esta manera, se realiza una abstracción de los datos para que la red neuronal pueda interpretarlos y procesarlos.

2 .2. Fase de entrenamiento

El algoritmo de Backpropagation tiene una estructura de 16x16 neuronas de entrada, una capa oculta de 24 neuronas y 36 neuronas de salida (256x24x36). Cada neurona de entrada se corresponde

Figura 2: Códigos a procesar.

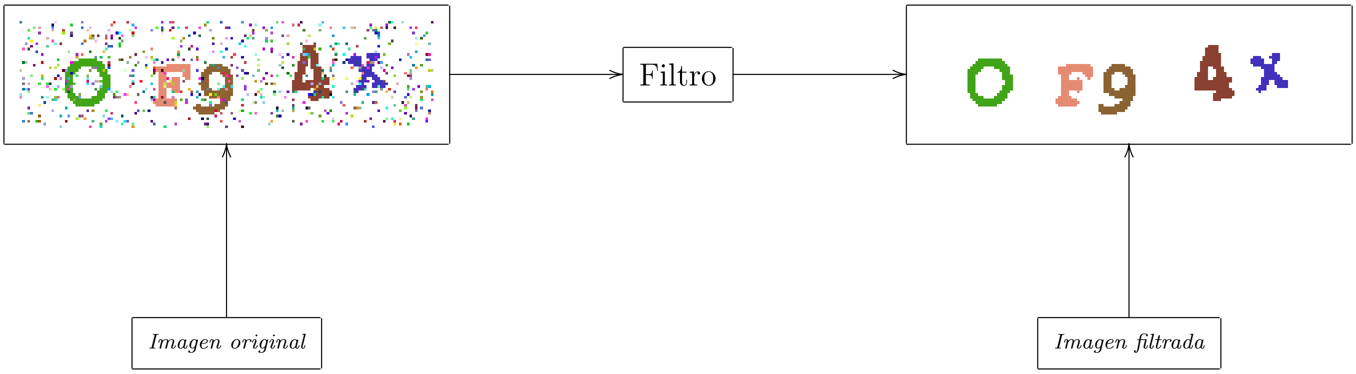

Figura 3: Filtrado de la imagen.

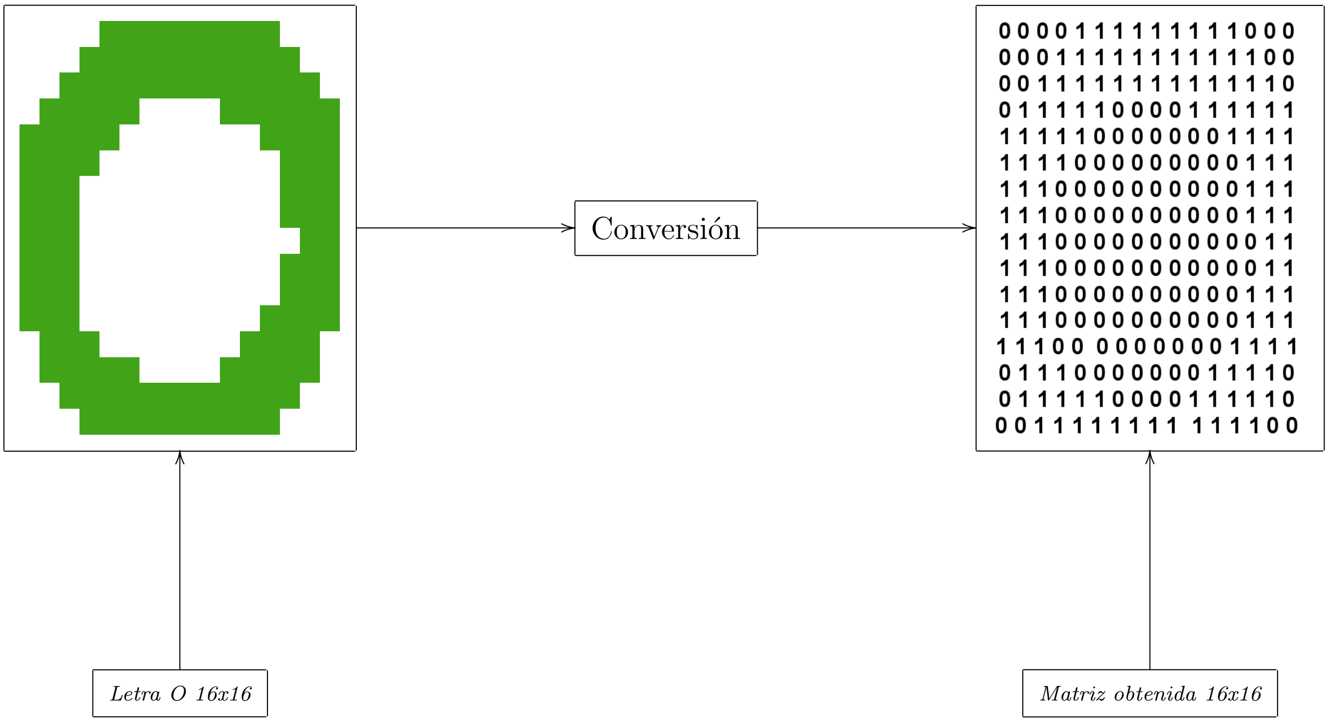

Figura 4: Conversión a la matriz de entrada.

con los píxeles de la imagen de entrada. Las neuronas de salida representan cada letra del alfabeto (sin contar la letra ñ) y los 10 dígitos.

El entrenamiento de una carpeta se realiza hasta que el error promedio de las imágenes de dicha carpeta sea menor a un cierto valor (en nuestro caso, 0.05).

El algoritmo de Backpropagation tiene la posibilidad de incluir Momentum a la hora de entrenar la red en cuestión.

2 .3. Generalización e intefaz gráfica

Se incluyó una interfaz gráfica para generalizar entrenamiento de la red y para hacer pruebas sobre la misma. La primer sección de la interfaz consiste en un campo de texto donde se indica la carpeta que contiene las imagenes CAPTCHA a descifrar y el botón Entrenar. Es necesario que la carpeta a elegir tenga un archivo hash.txt que esté estructurado de la siguiente manera: una columna con el nombre de los archivos de las imágenes CAPTCHA a descifrar, otra columna con las letras correspondientes a la decodificación deseada para cada código. Como es bastante engorroso construir el archivo "hash.txt" para

el entrenemiento se puede optar por utilizar la red que viene entrenada por defecto. Para ello, los pesos se hallan en el archivo "pesos.txt". A la hora de rellenar la matriz de pesos W, el programa carga los pesos directamente del archivo. En caso de que no se encontrase el mismo, se eligen los pesos de manera aleatoria.

Una vez entrenada la red, se incluye la posibilidad de probar su funcionamiento. Para ello se presenta un campo de texto donde se coloca la ruta de la imagen a descifrar y el botón descifrar. Luego la aplicación muestra la decodificación encontrada. En caso de que la decodificación no sea la deseada, lo cual será analizado más adelante, se ofrece la posibilidad de corregir el resultado y al presionar el botón Modificar se realiza un nuevo entrenamiento a la red.

3 . Resultados

3 .1. Aprendizaje del algoritmo

En la Figura 5 se presentan los gráficos correspondientes a la curva de aprendizaje del algoritmo Backpropagation para diferentes momentum M:

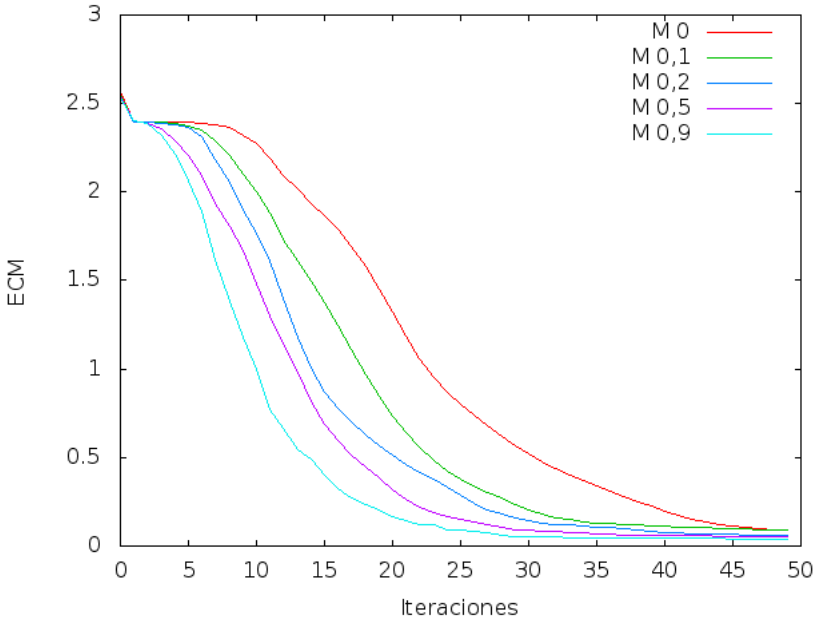

Figura 5: ECM por iteraciones.

3 .2. Decodificando Captchas

Una vez entrenada la red, probamos con diferentes códigos. En general, los caracteres incluídas dentro del entrenamiento son reconocidos perfectamente. Sin embargo, no ocurre lo mismo con códigos que tienen caracteres superpuestos o códigos que contienen caracteres que no fueron aprendidos. En el caso de los caracteres superpuestos, es posible separarlos teniendo en cuenta que en este tipo de código, 2 caracteres consecutivos tienen diferente color de fondo. Para el caso de los caracteres que no fueron aprendidos, por ejemplo si aprendió la 'V' y luego se pide descifrar la 'Y', es probable que confunda la Y por la V.

Para evitar esto es recomendable entrenar la red con varias muestras de un mismo caracter. Si bien esta solución no contempla el caso de letras cortadas debido a la superposición de las mismas, si la muestra de entrenamiento es realmente buena se obtienen resultados realmente muy buenos.

Otro detalle a destacar es que el algoritmo tiende a asociar con las últimas entradas aprendidas. Este inconveniente también se sortea utilizando una muestra variada.

4 . Conclusiones

El algoritmo de Backpropagation es muy útil para el reconocimiento de caracteres y la inclusión de Momentum posibilita una convergencia aún más rápida, como se pudo observar en la Figura ref{Figura 5. También posee una implementación relativamente simple y standard.

Una de las carencias del algoritmo es que le es difícil adaptarse a los cambios, ya que tiende a sobreaprender o hasta casi "memorizar"los caracteres. Por lo que si un caracter es presentado ligeramente en forma diferente, puede llegar a confundirlo con otro.