**Máster de Visión Artificial**

**Asignatura: Aplicaciones Industriales**

**Práctica puntuable – Reconocimiento Manuscrito**

**Orión García Gallardo**

INTRODUCCIÓN

La escritura a mano ha persistido como medio de comunicación y almacenamiento de información durante los años incluso con la introducción de las nuevas tecnologías. Dado a la gran cantidad de transacciones que se hacen con la escritura a mano, el reconocimiento manuscrito de caracteres tiene vital importancia. Se han conseguido resultados notorios en los pasados años en el reconocimiento de textos tanto on-line [1–3] como off-line [4-6]. El reconocimiento de texto genérico parece todavía una tarea larga [7], sin embrago, se han llevado a cabo interesantes investigaciones en campos menos ambiciosos como reconocimiento de direcciones postales en sobres [8-9], o la cantidad legal en un cheque bancario[10-11].

La siguiente práctica implementa un sistema para el reconocimiento de números manuscritos. Para la realización de este trabajo se ha hecho uso de unos ficheros de ejemplo facilitados por la base de datos del NIST [12].

DESCRIPCIÓN DEL MÉTODO DESARROLLADO

Entre las múltiples opciones a la hora de implementar esta práctica se opta por diseñan una red neuronal [13]. Las redes neuronales son una tecnología usada en inteligencia artificial que simula la forma en la que funciona el cerebro humano. Una serie de celdas que computan cálculos trabajan en paralelo con el objetivo de producir un resultado. Estas celdas son capaces de aprender por ellas mismas y pueden reponerse relativamente bien si una o más celdas de la red fallan.

Para la implementación de esta red neuronal se hace uso de las funciones que aporta la librería Encog [12]. Encog es un framework de dominio público de redes neuronales e inteligencia artificial disponible para Java, .Net y Silverlight. Encog contiene clases para crear una amplia variedad de redes y para normalizar y procesar datos para estas redes. Encog permite entrenar las redes neuronales usando diferentes técnicas, entre las más usadas en esta práctica se usa la backpropagation. La retropropagación (backpropagation) consiste en propagar el error hacia atrás, es decir, de la capa de salida hacia la capa de entrada, pasando por las capas ocultas intermedias y ajustando los pesos de las conexiones con el fin de reducir dicho error.

Para esta práctica se elige una red neuronal sin capas ocultas, lo cual puede provocar que sea poco tolerante al ruido en el entrenamiento, pero permiten que la red sea mucho más sencilla y el cálculo por iteración sea muchos más rápido. De todas maneras el sistema es configurable y permitiría añadir fácilmente capas ocultas a la red neuronal tan solo añadiendo unos valores al xml de configuración.

Uno de los hándicaps a los que se hace frente al crear la red neuronal es la cantidad de pixeles que tienen las imágenes a procesar. Las muestras de la base de datos NIST tiene imágenes de 28x28 pixeles lo crea una red neuronal con 784 entradas. Con el objetivo de reducir este número de entradas y simplificar la red de se dividen estas imágenes en regiones para posteriormente codificarlas a un número real entre 0 y 1. Para simplificar el proceso se establece que estas regiones sean cuadradas, es decir, que tengan el mismo número de filas que de columnas. Por lo tanto, para reducir las imágenes a 7x7, o 49 entradas en la red neuronal, se dividen estas en regiones de 4x4. Para la codificación de estas regiones se emplea la siguiente fórmula:

Sea ajk el valor en escala de grises de la imagen en la fila j columna k.

El valor de la región i viene dada por:

Codei = a00 \* 2560 + a01 \* 2561 + …+a10 \* 2564 + …+ ajk \* 256 (4\*j + k)+ …+a33 \* 25615

Se normaliza el valor a un valor entre 0 y 1:

Regioni = Codei / (25616 -1)

De esta manera cada región viene determinada por un valor entre 0 y 1 generado a partir de los valores de los pixeles que lo forman, y la identificaría de manera inequívoca.

Igual que el número de capas y las neuronas por capas son configurables también los son las entradas que queremos para la red neuronal. En el xml de configuración se puede introducir el número de entradas de la red neuronal teniendo en cuenta que debería ser un número que se ajuste a una posible reducción de la resolución de las imágenes en regiones cudradas. Estas resoluciones son: 28x28 (784 entradas), 14x14 (196 entradas), 7x7 (49 entradas), 4x4 (16 entradas), 2x2 (4 entradas) y 1x1 (1 entrada). Sin embargo, a la hora de ejecutar el sistema con estas resoluciones sólo permitiría bajar hasta una resolución de 4x4 (16 entradas) porque con menores resoluciones los números Codei y Regioni se saldrían de rango.

El sistema se puede ejecutar en modo entrenamiento o en modo test. En el primer modo lo primero que se realiza es pasar la base de datos de NIST de entrenamiento a la configuración de entradas seleccionada usando el algoritmo anteriormente explicado. A continuación, haciendo uso de las librerías de Encog, se crea la red neuronal. Esta red se puede crear a partir de una existente desde un fichero xml pero si este no existe, se crea desde cero con las entradas que se seleccionaron en el archivo de configuración. Cuando la red es creada desde cero se construyen también las capas intermedias con el número de neuronas que se indiquen en el archivo de configuración. Seguidamente se crea un entrenamiento de tipo retropropragación con el conjunto de datos de entrenamiento transformados a las entradas seleccionadas, el error de aprendizaje (learning error) y el momento (momentum) que se van a usar en el entrenamiento de retropropagación. El

PRUEBAS REALIZADAS SOBRE LA MUESTRA

Las pruebas realizadas se han hecho variando tanto el número de entradas en la red neuronal como el aprendizaje del error en cada iteración. Como muestra la tabla (Tabla 1) para las pruebas se ha ejecutado el sistema durante unos 300 segundos aproximados y se ha registrados los resultados obtenidos.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Error Learn | Time (secs) | Iterations | Error (%) | | 784 | 0.00001 | 300 | 89 | 7,1973 | | 196 | 0.00001 | 264 | 300 | 4,2693 | | 49 | 0.00001 | 274 | 1000 | 4,7734 | | 196 | 0.0001 | 300 | 348 | 2,1137 | | 196 | 0.001 | 300 | 323 | 2,1820 | | 196 | 0.0005 | 300 | 339 | 1,9428 | | 196 | 0.00075 | 300 | 319 | 2,1719 | | 196 | 0.00025 | 300 | 318 | 1,9286 | |
| Tabla 1. Entrenamiento de la red neuronal |

Con la configuración que da mejores resultados se ha hecho un entrenamiento más largo para observar cual sería un óptimo relativo del sistema. Seguidamente se han ejecutado los datos de test para ver cuál es la precisión de la aplicación al intentar reconocer un número de una imagen (Tabla 2)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Error Learn | Training Time (secs) | Iterations | Error (%) | | 196 | 0.00025 |  | 6000 |  | |
| Tabla 2. Resultados de la muestra de test sobre la red entrenada. |

CONCLUSIONES

Mejoras:

Mas capas

+ entradas:

* densidad normalizada.- Un valor

entre 0 y 1 que mide la proporción de

píxeles activos entre píxeles totales de

la caja que contiene al carácter.

* min(nº objetos / 3,1).- Un valor entre 0 y

1 proporcional al número de elementos

conexos de la imagen si esté numero es

menor a tres.

Modificar error learning dinámicamente.

REFERENCIAS

[1] Bellegarda EJ, Bellegarda JR, Nahamoo D, Nathan K. A probabilistic framework for on-line handwriting recognition. Proc 3rd International Workshop on Frontiers in Handwriting Recognition, Buffalo, NY, 1993; 225–234

[2] Connell S. Online Handwriting Recognition Using Multiple Pattern Class Models. PhD thesis, Michigan State University, East Lansing, MI, May 2000

[3] Jaeger S, Manke S, Reichert J, Waibel A. Online handwriting recognition: The NPEN\_\_ recognizer. Int J Document Analysis and Recognition 2001; 3:169–180

[4] Bozinovic RM, Srihari SN. Off-line cursive script word recognition. IEEE Trans Pattern Analysis and Machine Intelligence, 1989; 22(1):63–84

[5] Bunke H, Roth M, Schukat-Talamazzini EG. Off-line cursive handwriting recognition using hidden markov models. Pattern Recognition 1995; 28(9):1399–1413

[6] Chen MY, Kundu A, Zhou J. Off-line handwritten word recognition

using a hidden Markov model type stochastic network. IEEE Trans Pattern

Analysis and Machine Intelligence 1994; 16(5):481–496

[7] Kim G, Govindaraju V, Srihari SN. An architecture for handwriting text recognition systems. Int J Document Analysis and Recognition 1999; 2:37–44

[8] Chen MY, Kundu A, Zhou J. Off-line handwritten word recognition using a hidden Markov model type stochastic network. IEEE Trans Pattern Analysis and Machine Intelligence 1994; 16(5):481–496

[9] El-Yacoubi A, Gilloux M, Sabourin R, Suen CY. Unconstrained handwritten word recognition using hidden markov models. IEEE Trans Pattern Analysis and Machine Intelligence 1999; 21(8):752–760

[10] Dimauro G, Impedovo S, Pirlo G, Salzo A. Automatic bankcheck processing: A new engineered system. In: Impedovo S, Wang PSP, Bunke H, eds, Int J Pattern Recognition and Artificial Intelligence, World Scientific, 1997; 467–503

[11] Guillevic D, Suen CY. HMM-KNN word recognition engine for bank cheque processing. Proc International Conference on Pattern Recognition, Brisbane, Australia, 1998; 1526–1529

[12] <http://yann.lecun.com/exdb/mnist/>

[13] C.G. Looney, Pattern Recognition using Neural Networks: Theory and Algorithms for Engineers and Scientists, Oxford University Press, 1997.

[14] http://www.heatonresearch.com/encog