Reconocimiento de caracteres con Redes neuronales

La identificación visual de la palabra escrita es una de las fronteras de la Inteligencia Artificial en nuestros días. De hecho, muchas empresas utilizan la identificación de palabras escritas como CAPTCHA (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) para comprobar que el interlocutor es un humano y no una máquina.

1 Objetivo

La finalidad de este trabajo es el reconocimiento de letras manuscritas. Cada letra vendrá dada por una imagen en blanco y negro, de un tamaño fijo. Se deberá desarrollar o adaptar una herramienta de Aprendizaje Automático basado en redes neuronales en Python, de modo que, tras un proceso de entrenamiento de la red, sea capaz de tomar una imagen nueva correspondiente a una letra e identificar de cuál se trata.

Como parte del trabajo se deberá responder a los siguientes objetivos más específicos:

- Implementación, o bien adaptación y mejora, de una implementación en Python del algoritmo de retropropagación sobre redes neuronales de tipo perceptrón multicapa.
- Preparación de un conjunto de entrenamiento a partir de ejemplos de letras en formato PGM.
- Representación de la red neuronal que haga uso de la implementación del objetivo 1 para representar la red de nuestro problema, incluyendo las distintas capas, neuronas, pesos, etc.

2 Implementación del algoritmo de retropropagación

- Implementar en Python desde cero una versión básica del algoritmo de retropropagación sobre redes neuronales de tipo perceptrón multicapa, o bien adaptar una implementación ya existente. De optar por la segunda opción, se deberá indicar la fuente de la que se ha partido y estudiar e implementar mejoras sobre la implementación de partida.
- En cualquiera de los casos, la implementación deberá ser genérica, no *ad-hoc* para el problema particular, de manera que acepte el número de capas que se le pase, así como el número de neuronas de la capa de entrada, de salida, de cada capa oculta, y los pesos iniciales correspondientes.

3 Preparación del conjunto de entrenamiento

- Generar un conjunto de entrenamiento. Cada elemento del conjunto de entrenamiento debe ser una imagen en dos colores (blanco y negro) de 30×30 píxeles en formato PGM (**P**ortable **G**raymap **F**ormat). Un fichero PGM contiene texto plano que puede ser modificado por un procesador de texto, pero también interpretado por un visor de imágenes (por ejemplo, Inkscape o GIMP). Por ejemplo, la figura 1 representa la letra a escrita a mano en una imagen de 30×30 píxeles. El fichero PGM de esa imagen es letra `a.pgm`, que se proporciona con este enunciado.

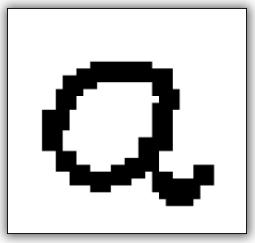

Figura 1: Letra *a* manuscrita en imagen de 30×30 píxeles

- El alumno debe proporcionar un conjunto de ficheros con extensión pgm y nombre un número de 1 hasta N, esto es, `1.pgm`, `2.pgm`, `3.pgm`,$\cdots$, `N.pgm`. Cada uno de estos ejemplos debe contener una letra escrita a mano. El número de ejemplos queda a elección del alumno, pero debe ser suficiente para realizar un proceso de aprendizaje.

- Representar en Python los ejemplos, almacenando para cada uno de ellos:

  - El identificador: Los ejemplos estarán numerados y el identificador será el número asociado con ese ejemplo. Cada ejemplo debe tener como identificador el nombre del fichero pgm que contiene la imagen.
  - La clasificación: podrán estudiarse distintas alternativas para representar las salidas (una neurona de salida por cada posible letra, una codificación que minimice el número de neuronas de salida, etc.), y en función de ellas se tendrá que dar la representación en los ejemplos.
  - Los píxeles: tendrán que venir de la sucesión de 0's y 1's tomada del correspondiente fichero PGM. Es decir, el fichero podrá tener datos entre 0 y 255, pero al procesarlo debemos quedarnos con los dígitos 0 y 1 únicamente (se deja al alumno la decisión de tomar como 1 todo aquello que sea distinto de cero, o bien considerar algún umbral para la binarización, si bien es deseable que este dato sea configurable en la implementación realizada).

  .

- Preparación de un conjunto de prueba: junto con el conjuto de entrenamiento, se deberá dotar también de un conjunto de prueba, para poder testear la red del apartado 4 una vez entrenada.

4 Desarrollo de la red

- Emplear la implementación del algoritmo de retropropagación del apartado 2 para entrenar la red correspondiente al problema planteado.
- La arquitectura de la red queda a elección del alumno, pero debe tener las neuronas necesarias en la capa de entrada para poder recibir como entrada la codificación de una imagen de 30×30 píxeles.
- La capa de salida debe tener las neuronas .
- Es recomendable emplear dentro del código la arquitectura de la red y del sistema de representación utilizado.
- Los pesos iniciales no deben generarse aleatoriamente, sino que deben ser proporcionados como entrada y poder ser modificados para estudiar distintos comportamientos del sistema bajo distintos conjuntos de pesos iniciales.
- Del mismo modo, el factor de aprendizaje debe ser configurable.

5 Experimentación y conclusiones

- Una vez desarrollados los apartados anteriores, se podrá proceder a analizar los resultados obtenidos bajo distintas condiciones iniciales, probando distintas arquitecturas de red (número de capas ocultas, neuronas en cada una de ellas, pesos iniciales, factor de aprendizaje, momento, posibles mejoras sobre el algoritmo de retropropagación, etc.)
- Se deberá documentar cada caso estudiado, proporcionando información acerca de la configuración de partida, los pesos obtenidos tras el entrenamiento, y los resultados correpondientes a la clasificación de los ejemplos del conjunto de prueba.
- Finalmente, se deberán aportar las principales conclusiones, a la luz de los resultados mencionados, acerca de las arquitecturas y parámetros que han resultado más prometedores para la resolución del problema estudiado.