UNIVERSIDAD
DE GRANADA

Inteligencia Computacional

Práctica de redes neuronales

Reconocimiento óptico de caracteres
MNIST

Curso 2020-2021

Máster en Ingeniería Informática

Departamento de Ciencias de la Computación e Inteligencia Artificial

Práctica 1

MNIST

El objetivo de esta práctica es resolver un problema de reconocimiento de patrones utilizando redes neuronales artificiales. Deberá evaluar el uso de varios tipos de redes neuronales para resolver un problema de OCR: el reconocimiento de dígitos manuscritos de la base de datos MNIST (`http://yann.lecun.com/exdb/mnist/`).

1 Implementación

Para realizar esta práctica puede optar por implementar usted mismo los algoritmos de entrenamiento que considere oportunos o bien utilizar como base alguna de las muchas bibliotecas que implementan dichos algoritmos y están disponibles públicamente, si bien las decisiones que tome al respecto pueden influir en su calificación final.

Aunque la implementación secuencial de la mayoría de los algoritmos de entrenamiento de redes neuronales no es especialmente compleja, se pueden encontrar en Internet múltiples bibliotecas que implementan algunos de los algoritmos descritos en clase de forma eficiente. Dichas implementaciones, disponibles para múltiples lenguajes de programación, suelen estar diseñadas para aprovechar los distintos núcleos de un microprocesador actual o la capacidad de cómputo de una GPU, en caso de tener una disponible. Si, en vez de utilizar un lenguaje de programación de propósito general, prefiere emplear una herramienta matemática, también puede encontrar *toolboxes* de Matlab o paquetes en R prácticamente para cualquier tipo de red neuronal que desee evaluar.

No obstante, el uso de bibliotecas proporcionadas por terceros también tiene algunos inconvenientes que hay que tener en cuenta. En primer lugar, nadie nos garantiza que la implementación disponible esté completamente libre de errores y, aunque así sea, es posible que resulte difícil adaptar la implementación disponible a nuestras necesidades concretas cuando deseemos realizar un experimento particular.

IMPORTANTE: Utilizar implementaciones de terceros sin realizar las atribuciones correspondientes es constitutivo de plagio, un delito contra la propiedad intelectual que conlleva un suspenso automático en la asignatura.

Sobre el cálculo del gradiente

A la hora realizar su implementación, se recomienda utilizar una estrategia de tipo TDD [*test-driven development*] o, al menos, realizar pruebas de unidad que garanticen que el cálculo del gradiente se realiza correctamente. Para realizar esta comprobación, recuerde la definición de la derivada:

$$\frac{d}{d\theta}J(\theta) = \lim_{\epsilon \to 0} \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}. \tag{1.1}$$

Para cualquier valor de θ, se puede aproximar la derivada utilizando un valor de epsilon (ϵ) pequeño, p.ej. 10^{-4} (un valor demasiado pequeño podría ocasionar errores de redondeo, por lo que no conviene apurar demasiado en este sentido):

$$g(\theta) \approx \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}. \tag{1.2}$$

Como θ es un vector de parámetros, no un simple número real, tendremos que comprobar que el gradiente se calcula correctamente para cada θ_i. Para cada vector de parámetros θ, evaluaremos $g_i(\theta)$ como una aproximación de $\frac{\partial}{\partial \theta_i}J(\theta)$. Podemos definir $\theta^{(i+)} = \theta + \epsilon\vec{e}_i$, donde $\vec{e}_i$ es un vector con un uno en la posición i-ésima y ceros en las demás:

$$\vec{e}_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \tag{1.3}$$

Entonces $\theta^{(i+)}$ es similar a θ, salvo que su i-ésima componente se ha incrementado en ϵ. De la misma forma, podemos obtener $\theta^{(i-)} = \theta - \epsilon\vec{e}_i$, el vector θ con su i-ésima componente reducida en ϵ. Con esto podemos verificar numéricamente el gradiente para cada parámetro:

$$g_i(\theta) \approx \frac{J(\theta^{(i+)}) - J(\theta^{(i-)})}{2\epsilon}. \tag{1.4}$$

Sólo tenemos que repetir el cálculo **para distintos valores de** θ, con el objetivo de comprobar que la diferencia entre los valores calculados por nuestra implementación y los valores aproximados numéricamente no difieren en exceso.

Otra alternativa interesante consiste en la utilización de técnicas de diferenciación automática (`https://en.wikipedia.org/wiki/Automatic_differentiation`). No obstante, la implementación de este tipo de técnicas puede que sólo compense en proyectos de mayor envergadura. Básicamente, la diferenciación automática se encarga de calcular el gradiente de una función por nosotros, de forma que se evitan posibles errores de implementación que, de otra forma, podrían llegar a pasar desapercibidos.

2 Análisis de resultados

A la hora de resolver un problema de clasificación, el clasificador se entrena con el conjunto de entrenamiento. En este caso, el conjunto de entrenamiento contiene 60000 ejemplos etiquetados. Los ejemplos de entrenamiento son imágenes normalizadas de 28x28 píxeles y se encuentran en el fichero `train-images-idx3-ubyte`, mientras que las etiquetas correspondientes a los ejemplos se pueden encontrar en el fichero `train-labels-idx1-ubyte`.

Evaluar la calidad de un clasificador utilizando los mismos datos con los que se entrena puede resultar engañoso, motivo por el que se dispone de un conjunto de datos aparte, que NO se utiliza durante el entrenamiento. Dicho conjunto de prueba, almacenado en el mismo formato que el conjunto de entrenamiento, puede encontrarlo en los ficheros `t10k-images-idx3-ubyte` (imágenes) y `t10k-labels-idx1-ubyte` (etiquetas).

Si bien casi cualquier técnica de aprendizaje automático puede conseguir resultados excepcionales sobre el conjunto de entrenamiento, no todas las técnicas son igual de buenas cuando se trata del conjunto de prueba. Algunas técnicas de aprendizaje sobreaprenden (se adaptan en exceso al conjunto de entrenamiento con el que se entrena el modelo) y luego no generalizan correctamente, por lo que obtienen resultados pobres sobre conjuntos de datos diferentes al conjunto de entrenamiento.

Para evaluar los resultados de la práctica utilizaremos la tasa de error sobre el conjunto de prueba (el número de instancias del conjunto de prueba que nuestra red neuronal no clasifica correctamente). A título orientativo, estos son los resultados que debería obtener con algunos modelos concretos de redes neuronales artificiales:

- Red neuronal simple, con una capa de entrada y una capa de salida de tipo softmax: 7.8 % de error sobre el conjunto de prueba y 5.6 % de error sobre el conjunto de entrenamiento (tiempo de entrenamiento necesario: sobre un minuto usando una implementación en un lenguaje interpretado como Matlab).

- Red neuronal multicapa, con una capa oculta de 256 unidades logísticas y una capa de salida de tipo softmax: 3.0 % de error sobre el conjunto de prueba y 0.0 % de error sobre el conjunto de entrenamiento (tiempo de entrenamiento necesario: unos cuatro minutos usando una implementación en un lenguaje interpretado como Matlab).

- Red neuronal convolutiva entrenada con gradiente descendente estocástico: 2.7 % de error sobre el conjunto de prueba (tiempo de entrenamiento necesario: unos 13 minutos usando una implementación en un lenguaje interpretado como Matlab).

- "Deep learning" usando pre-entrenamiento de autoencoders para extraer características de las imágenes usando técnicas no supervisadas y una red neuronal simple con una capa de tipo softmax: del 1.8 % al 2.2 % de error sobre el conjunto de prueba (tiempo de entrenamiento necesario: unos veinte minutos usando una implementación en un lenguaje interpretado como Matlab).

La red propuesta por Yann LeCun, de tipo convolutivo y con múltiples capas ocultas, consigue reducir la tasa de error al 0.82 % (82 errores de los 10000 ejemplos del conjunto

de prueba). Usando un algoritmo de "deep learning" más sofisticado, el error se puede reducir al 0.35 % (35 errores sobre 10000). Combinando múltiples redes neuronales se puede reducir aún más el error, hasta el 0.23 % (sólo 23 errores sobre 10000). En la página web de MNIST (`http://yann.lecun.com/exdb/mnist/`) se pueden encontrar los resultados que se han obtenido con multitud de técnicas de clasificación, incluyendo varios tipos de redes neuronales artificiales.

3 Documentación y entrega de la práctica

- Entrene distintas redes neuronales artificiales sobre el conjunto de entrenamiento de MNIST y evalúe los resultados obtenidos utilizando el conjunto de prueba. Envíe los resultados que vaya obteniendo a través de la página web habilitada al efecto (`https://goo.gl/xiXVSK`). Puede enviar sus resultados tantas veces como desee, sólo se tendrá en cuenta el mejor resultado que haya obtenido.

  NOTA: El envío de resultados debe realizarlo utilizando la misma dirección de correo con la que aparezca registrado en la asignatura y ha de incluir todos los parámetros necesarios para replicar el experimento, incluyendo la topología de la red neuronal utilizada (número de capas, neuronas por capa, tipo de neuronas...) y el algoritmo de entrenamiento empleado para ajustar los pesos de la red (con la combinación particular de parámetros que haya utilizado).

- Elabore una memoria en la que se recoja la experimentación realizada durante la elaboración de esta práctica y los gráficos que considere oportunos para ilustrar los resultados obtenidos. La memoria deberá entregarse en formato PDF e irá acompañada de todos los ficheros correspondientes a las distintas implementaciones efectuadas (y los enlaces correspondientes a las bibliotecas externas que haya empleado). La entrega de la memoria en PDF y de los ficheros de código en un ZIP deberá realizarse, a través de la plataforma docente de la asignatura, antes del **13 de diciembre de 2020 a las 23:59**.

Evaluación de la práctica

- 30 % por el trabajo de implementación realizado:
  - 10 % como máximo si utiliza una implementación externa de los algoritmos de aprendizaje de redes neuronales (0 % si se limita a replicar los experimentos de algún tutorial ya disponible en Internet).
  - 30 % como máximo si desarrolla su propia implementación de los algoritmos de aprendizaje de redes neuronales.
- 40 % por la memoria de la práctica (documentación de los experimentos y pruebas realizadas junto con su análisis de resultados):
  - 20 % por la descripción de su implementación y de los algoritmos utilizados durante la realización de la práctica.
  - 20 % por la descripción de los experimentos que haya realizado para establecer los parámetros de su red neuronal y su análisis de los resultados obtenidos.
- 30 % por los resultados conseguidos (usando un ranking lineal definido sobre las tasas de error obtenidas sobre el conjunto de prueba, con la puntuación máxima del 30 % para el que llegue al 0.3 % de error y una puntuación mínima del 10 % para el que consiga bajar del 2 % de error).