Práctica 5

Redes Neuronales en Keras

Javier Herrer Torres (NIP: 776609)

Inteligencia Artificial
Grado en Ingeniería Informática

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
Curso 2020/2021

Tipo de red	Técnicas	Accuracy train	Accuracy test
Perceptrón	SGD + sigmoid	90.00 %	90.68 %
	RMSprop + softmax	93.21 %	92.70 %
	Adam + softmax	93.26 %	92.72 %
Perceptrón multi-nivel con una capa oculta Capa oculta: 512	epochs: 20 earlystop + dropout	99.75 %	98.23 %
	epochs: 40 earlystop + dropout sin validation_split	99.99 %	98.47 %
Perceptrón multi-nivel con dos capas ocultas Capas ocultas: 512	dropout	99.73 %	98.50 %
Red Convolucional	Test loss: 0.033728806659879595 % Test accuracy: 0.9907000064849854 %		

Cuadro 1: Resultados obtenidos

1. Introducción

El objetivo es resolver mediante redes neuronales un problema real de clasificación: el reconocimiento de dígitos manuscritos del *dataset* MNIST. La figura muestra un ejemplo de los mismos. Cada muestra es una imagen de 28x28 píxeles. Como atributos para la clasificación utilizaremos directamente los niveles de intensidad de los 28x28 píxeles. Utilizaremos Keras, una API sencilla para Redes Neuronales, que permite entrenar y ejecutar redes neuronales utilizando Tensorflow como *back-end*.

2. Perceptrón

Se han empleado los siguientes algoritmos de entrenamiento:

- **SGD** (*Stochastic Gradient descent*) es un método iterativo de optimización de una función de pérdida.
- **RMSprop** (*Root Mean Square Propagation*) tiene un factor de entrenamiento diferente para cada dimensión, pero en este caso el escalado del factor de entrenamiento se realiza dividiéndolo por la media del declive exponencial del cuadrado de los gradientes.
- **Adam** (*Adaptive moment estimation*) mantiene un factor de entrenamiento por parámetro y además de calcular RMSProp, cada factor de entrenamiento también se ve afectado por la media del *momentum* del gradiente.

Se ha comprobado que **el algoritmo Adam tiene un rendimiento superior** al ser el más reciente y estar construido en base a sus predecesores. También se han probado las siguientes funciones de activación:

- **sigmoidal** para valores pequeños (< -5), devuelve un valor cercano a 0, y para valores altos (> 5) devuelve valor cercano a 1. Es ideal para clasificación binaria. Podemos interpretar la salida de la red como $P(y = 1|x)$. Puede funcionar con coste cuadrático, pero va mejor con coste *cross-entropy*.
- **softmax** convierte un vector de reales en un vector de probabilidades categóricas. Es ideal para clasificación multi-clase. La neurona i representa $P(y = i|x) softmax(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$. Las probabilidades están normalizadas (a diferencia de las sigmoidales), y además va bien con coste *cross-entropy*.

La **función softmax tiene un rendimiento superior** al ser la indicada para clasificaciones multiclase. Mientras que la función sigmoidal es usada para clasificaciones binarias.

3. Perceptrón multi-nivel con una capa oculta

Se han empleado las siguientes funciones de activación:

1. **relu** para la capa oculta. Es la más usada en la actualidad para las capas ocultas. Funciona muy bien en redes profundas. Es muy sencilla, y su derivada también. Que no tenga derivada en $x = 0$ no es problema en la práctica. Tiene cierta base biológica, las neuronas reales parecen tener un umbral a partir del cual se activan.
2. **softmax** para la capa de salida siguiendo el razonamiento del apartado anterior.

El **tamaño de la red** es el parámetro de diseño más importante. En general, cuanta mayor profundidad y anchura puede funcionar mejor pero la red es más costosa de entrenar (GPUs) y más propensa al sobreajuste. Suele ser mejor dos capas con menos neuronas (100), que una capa con un número de neuronas enorme (10.000). Para realizar el cálculo del número de neuronas en la capa oculta se ha seguido el siguiente proceso:

1. El número de neuronas debe estar entre el tamaño de la capa de entrada y la de salida.
2. Tras **experimentar con diferentes valores**, observar si la diferencia obtenida tras aumentar el número de neuronas oculta merece la pena. Es decir, si la diferencia en el resultado es realmente significante como para aumentar el tamaño de la red.

Tamaño capa oculta	Accuracy train	Accuracy test
10	94.29 %	93.47 %
512	99.93 %	98.27 %
784	99.75 %	98.00 %

Cuadro 2: Experimentación del tamaño de la capa oculta

En el cuadro 2 se muestra la experimentación realizada con diferentes tamaños de capa oculta. Se ha probado con un tamaño igual al de las entradas, igual al de las salidas y, finalmente, el mejor resultado se ha obtenido en un valor intermedio.

Una vez se ha obtenido que el número óptimo de neuronas en la capa oculta es de 512, se han empleado las siguientes técnicas de reducción del sobreajuste: early stopping, dropout y aumentar las épocas sin validation_split. Los resultados se encuentran en el cuadro 1.

4. Perceptrón multi-nivel con dos capas ocultas

Se ha continuado empleando las mismas funciones de activación para las capas ocultas de salida. Además, las dos capas ocultas tienen el **mismo tamaño**, y se ha mantenido únicamente la técnica de **dropout**. Los resultados obtenidos mejoran en **0.03** décimas a la red anterior (ver cuadro 1). Al no haberse obtenido una mejora significante, no tendría sentido aumentar el tamaño de la red.

5. Técnicas de reducción de sobreajuste

En esta práctica se han empleado las siguientes técnicas de sobreajuste (ver cuadro 1):

- **Early stopping** de las 60.000 muestras de `x_train` el 90 % es usado para calcular pesos, y el 10 % para ver el error de validación y parar si no baja (iterar demasiado provoca sobreajuste). El inconveniente es que desaprovecha el 10 % de los datos.
- **Dropout** consiste en apagar aleatoriamente algunas neuronas (una fracción de ellas) al entrenar. Suele funcionar muy bien en redes complejas.
- **Regularización** añade a J una penalización por los pesos.

6. Red Convolucional

La red convolucional descrita en el fichero `mnist_cnn.py` cuenta con un diseño más inteligente y obtiene unos resultados significativamente mejores. Las dos redes planteadas anteriormente manejaban unos resultados en torno al 98 % en *accuracy test* mientras que **la red convolucional llega al 99 %**.