**Práctica 5. Entrenamiento de redes neuronales**

**Descripción**

La red neuronal dispone de tres capas, la primera capa donde se introducen los datos, la capa oculta y la capa de salida. Los datos de entrada constan de pixeles de imágenes de dígitos. Como las imágenes tienen un tamaño de 20X20 se dispondrá de un tamaño de 400 en la capa de entrada. Por diseño se ha decidido tener 25 unidades en la capa oculta.

A diagram of a network

Description automatically generated

**Carga de datos**

Los datos que se van a utilizar en esta práctica vienen dados en el directorio *data* y se cargarán a través del método ***load\_data***

A computer screen shot of a code

Description automatically generated

**Identificación de dígitos. Implementación de *one\_hot***

La capa de salida tiene 10 unidades correspondientes a las 10 clases de dígitos. Mientras que las etiquetas originales (en la variable y) son 0, 1, . . ., 9, con el propósito de entrenar una red neuronal, necesitamos codificar las etiquetas como vectores que contienen solo valores 0 o 1 (codificación "one-hot"), de manera que

A number line with numbers and a number on it

Description automatically generated with medium confidence

Por ejemplo, si *x(i)* es una imagen con el dígito 5, entonces el correspondiente *y(i)* que se tendrá con la red neuronal será un array de tamaño 10 con *y(5)* = 1 y los demás elementos de ese array serán 0.

Para poder implementar esto se desarrollará una función denominada ***one\_hot***:

A computer screen shot of a code

Description automatically generated

De esta manera calculamos la posición en la que se encuentra el número del que debemos de hacer el intercambio de 0 a 1.

**Cálculo del coste**

La ecuación para determinar el coste de una red neuronal es la siguiente:

A math equation with black text

Description automatically generated with medium confidenceDonde hθ x(i) está computado como se ha mostrado en la figura 1.1 y *K* = 10 es el número total de posibles clases. Nótese que hθ(x(i))k es la activación (el valor de salida) para el valor k-ésimo del valor.

Se ha proporcionado un conjunto de redes de parámetros (Θ(1), Θ(2)) ya entrenadas. Estos parámetros poseen dimensiones que concuerdan con la red neuronal de 25 unidades en la segunda capa y 10 unidades de salida (correspondiendo con los 10 dígitos de clases).

Para implementar la función de coste se desarrolla el siguiente método:

A computer code with colorful text

Description automatically generated

Donde ***fordward\_propagation*** es un método auxiliar que calcula hθ(x(i))k

A screen shot of a computer code

Description automatically generated

A continuación, se explica un poco más en detalle la implementación de ***fordward\_propagation***:

1. Se agrega una columna de unos a la matriz de características de entrada. Esta modificación se hace para añadir el término de sesgo y permitir que la red neuronal aprenda un término de sesgo para cada neurona en las capas ocultas y de salida.
2. Se calcula la salida de la capa oculta aplicando la función de activación sigmoide a la suma ponderada de las entradas y los pesos entre la capa de entrada y la capa oculta. Esto se realiza mediante la multiplicación de la matriz de entrada extendida por la matriz de pesos ***theta1*** transpuesta.
3. Se vuelve a agregar una columna de unos a la matriz de la capa oculta. Esta modificación tiene el mismo fin que en el primero punto, añadir el término de sesgo y permitir que la red neuronal aprenda un término de sesgo para cada neurona en las capas ocultas y de salida.
4. Se calcula la probabilidad de que un valor de entrada se corresponda con un valor de salida. Esto se consigue gracias a la función de activación o **sigmoide**.

**Cálculo del coste regularizado**

Si se desea expandir la función coste de tal manera que devuelva el cálculo del coste regularizado de las redes neuronales, se añade el siguiente campo a la anterior función:

A math equations on a white background

Description automatically generated

En este caso concreto se puede especificar los campos de las capas y de las unidades:

A math equations and symbols

Description automatically generated with medium confidence

Por tanto, ampliamos el método anteriormente definido definiendo un nuevo método llamado ***cost\_reg***

A computer screen shot of a code

Description automatically generated with medium confidence

**Backpropagation**

Se implementará el algoritmo de **backpropagation** que calculará el gradiente del coste de la red neuronal. Se implementará inicialmente el algoritmo para que calcule los gradientes de los parámetros de una red neuronal no regularizada.

Teniendo en cuenta la explicación de las diapositivas

A math equations and formulas

Description automatically generated with medium confidence

**Backpropagation no regularizado**

Se realiza la implementación del algoritmo backpropagation no regularizado de la siguiente manera:

A computer screen shot of a code

Description automatically generated

Donde ***gradients*** es el método donde se calcula los diferentes gradientes no regularizados:

A computer screen shot of a program code

Description automatically generated

A continuación, se explica brevemente los pasos que sigue el método:

1. Se inicializan los deltas de las capas ocultas y de salida (Δ​hidden y Δout) con matrices de ceros del mismo tamaño que los parámetros θ1​ y θ2, respectivamente.
2. Se recorren los ejemplos de entrenamiento uno por uno.
3. Para cada ejemplo de entrenamiento, se realiza la propagación hacia adelante para calcular las activaciones en las capas ocultas y de salida (*h*). También se obtienen las activaciones de la capa oculta (A2​) y de la capa de salida (A3).
4. Se calculan los errores de la capa de salida (δout​) y de la capa oculta (δhidden) para el ejemplo de entrenamiento actual.
5. Se actualizan los deltas sumando el producto externo de δhidden ​y xk​ para Δ​hidden, y el producto externo de δout y A2 para Δout.
6. Se repiten los pasos **2-5** para todos los ejemplos de entrenamiento.
7. Se divide cada delta total por el número de ejemplos de entrenamiento (*m*) para obtener el gradiente promedio.

La función ***main*** ha sufrido ciertas modificaciones para conseguir implementar el algoritmo:

A screen shot of a computer code

Description automatically generated

**Backpropagation regularizado**

Para el caso de implementar el algoritmo de ***backpropagation*** a una red neuronal regularizada modificamos el cálculo del gradiente y del coste en el método anterior

Para el cálculo del gradiente:

A computer code with colorful text

Description automatically generated

Por lo que el algoritmo quedaría de la siguiente manera:

A computer screen shot of a code

Description automatically generated

**Aprendizaje de parámetros**

Después de implementar correctamente el cálculo del coste y del gradiente de la red neuronal el siguiente paso es usar el descendiente del gradiente para que aprenda un buen conjunto de parámetros. Por ello se inicializa las diferentes *thetas* de la siguiente manera:

A math equations with numbers

Description automatically generated with medium confidence

Antes del entrenamiento es necesario inicializar aleatoriamente los parámetros. Una estrategia eficaz para la inicialización aleatoria consiste en seleccionar aleatoriamente valores para Θ(l) de manera uniforme en el rango [ −εinit, εinit].

Se usará εinit= 0,12. Con este rango de valores se garantiza que los parámetros se mantengan pequeños y consigue un aprendizaje eficiente.

Se ha desarrollado la función ***learning\_parameters()*** para la ejecución de esta parte de la práctica:

A computer screen shot of a program code

Description automatically generated

Donde ***train\_neural\_network()*** es el método descrito a continuación:

A computer screen shot of code

Description automatically generated

Se realiza la llamada a ***initialize\_weights()*** para la creación de los Θ iniciales. Posteriormente estos Θ se verán afectados por los gradientes obtenidos en el bucle.

A computer code with red and white text

Description automatically generated