Manual de Usuario de NN

Versión 1

Francisco J. Sánchez

Septiembre 2020

Índice

Sobre este manual

En los ejemplos y explicaciones de código de este manual se usará la elipsis indicada por “...” para indicar que falta algo en esa parte para que el programa sea válido, pero que no afecta a la explicación a la que ayude el fragmento de código en el que se encuentre.

En este manual no se explica qué es ni cómo funciona una red neuronal.

1. Estructura del programa

Un programa de NN es una colección de redes neuronales e instrucciones para darle valores de entrada a las redes, seleccionar salidas y, en base a todo lo anterior, calcular los resultados. En princípio, está almacenado en un fichero de texto.

El orden y la posición de las redes en el programa no importa, tienen alcance global. Las instrucciones se ejecutan en el orden que aparecen.

En este manual se intenta dar un formato al código para que sea fácilmente legible y entendible, pero cualquier sangrado, espaciado o saltos de línea son opcionales y no tienen ningún efecto en el programa.

2. Elementos

Los elementos son las redes, las capas y las neuronas. Combinándolos se pueden crear redes neuronales.

Cada elemento tiene un nombre único y que por tanto no puede repetirse en todo el programa. Los nombres serán un conjunto de letras de la 'a' a la 'z' (sin tildes ni 'ñ') y números siendo el primer carácter una letra mayúscula.

2.1. Redes

Las redes conectan capas u otras redes una tras otra, en el orden en el que se definan en su interior. Pueden ser definidas directamente en el programa, dentro de otra red o dentro de una capa. Conectan o capas o redes, no se puede mezclar. Han de contener un mínimo de dos elementos.

La manera de definir una red es con su nombre seguido de las definiciones de los elementos que conecta entre paréntesis separados por comas.

Las salidas de la red serán las salidas de último elemento definido en ella.

```
R (
  Elemento1 ...,
  Elemento2 ...,
  ...
  ElementoN ...
)
```

En esta red se ejecutará el "Elemento1", sus salidas se conectan al "Elemento2" (como se explica en 2.3) que se ejecuta después y continúa de esta manera, en el orden que estén definidos, hasta el "ElementoN". En este ejemplo, las salidas de la red son las salidas de "ElementoN".

2.2. Capas

Las capas conectan redes o neuronas dentro de otras redes. Han de ser definidas dentro de una red. Entre los propios elementos de una capa no puede haber conexiones, estas han de estar con los elementos de la capa anterior (como entradas) o la capa siguiente (como salidas). Han de contener un mínimo de un elemento.

La manera de definir una capa es con su nombre seguido de las definiciones de los elementos que contiene entre corchetes separados por comas.

Las salidas de la capa serán la unión de todas las salidas de los elementos definidos en ella.

```
...
C [
  Elemento1 ...,
  Elemento2 ...,
  ...
  ElementoN ...
]
...
```

En esta capa se ejecutarán todos los elementos, desde "Elemento1" hasta "ElementoN", de manera "paralela"[1]. El conjunto de salidas de la capa es la unión todas las salidas de "Elemento1", "Elemento2", hasta "ElementoN".

2.3. Neuronas

Las neuronas son las que realizan los cálculos en la red neuronal. Han de ser definidas dentro de una capa. Constan de una o más entradas, una o más salidas (en el sentido de a dónde va el valor resultante), un valor de *bias* y una función de activación.

[1] La ejecución técnicamente es secuencia uno tras otro en el orden indicado, pero como no pueden tener conexiones las salidas de ninguno de los elementos de la capa con la entrada de ninguno de los elementos de la capa, de manera práctica para el usuario, es como si se ejecutara en paralelo.

La manera de definir una neurona es con su nombre seguido de las propiedades entre llaves separadas por comas. Las propiedades se definen con el nombre de la propiedad (siempre en minúscula), dos puntos y el valor de la propiedad. El orden en el que se indiquen las propiedades no afecta.

Como se ha dicho existen cuatro propiedades, que son:

- `input`: Es obligatoria. Define las entradas de la neurona. Su valor con las conexiones entre corchetes separadas por comas, al menos una. Una conexión es el nombre de la neurona de la que tomar la salida, dos putos y el peso por el que multiplicar esa salida. El peso es un número real con notación científica si se desea.
- `output`: Es obligatoria. Define las neuronas a las que irá el valor de salida de la neurona. Su valor son los nombres de las neuronas objetivo entre corchetes separados por comas, al menos uno.
- `bias`: Es opcional. Indica el valor de *bias* de la neurona. El valor es un número real con notación científica si se desea. Si no se indica esta propiedad, tomará como valor 0.
- `activation`: Es opcional. Indica la función de activación que se aplicará al cálculo de la salida de la neurona. Ha de ser el nombre de una de las funciones que implemente el lenguaje, actualmente:
  - `identity`
  - `sigmoid`
  - `binary`

  Si no se indica, tomará como valor `identity`.

Las conexiones que se definen mediante `input` y `output` han de ser lógicas y válidas. Si una neurona A tiene en su `input` a otra neurona B, B ha de tener en `output` a A. A y B han de estar "adyacentes" una a la otra, no puede haber capas entre las dos. No pueden estar en la misma capa. A tiene que aparecer después de B en la red.

```
...
N {
  input: [Entrada1: 1, Entrada2: 1.3e-1],
  output: [Salida1, Salida2, Salida3],
  bias: -0.14,
  activation: sigmoid
}
...
```

En ese ejemplo, se define la neurona "N" con entrada desde la neurona "Entrada1" con peso 1 y desde "Entrada2" con peso $1{,}3e-1$ (=0,13); cuyo valor de salida irá a las neuronas "Salida1", "Salida2" y "Salida3"; bias de -0.14 y función de activación sigmoidal.

2.3.1. Excepciones en las propiedades de input y output

Las primeras neuronas de la red principal (una que esté indicada directamente en el programa, no dentro de ningún elemento), en la capa de entrada de la red, obviamente no existen otras neuronas de las que tomar la entrada. Por tanto, en la propiedad `input`, en lugar del nombre de otra neurona se indica el nombre de una variable de entrada. Además en esta capa las neuronas solo pueden tener una entrada (uno de los valores de entrada). Estas variables de entrada son indicadas en la instrucción adecuada (3.1) y sólo pueden usarse en estas neuronas. Si se usa una variable de entrada que no esté indicada por una instrucción será indefinida y por tanto, inválida.

Las últimas neuronas de la red principal, en la capa de salida, obviamente no existen otras neuronas a las que llevar la salida, por tanto tenemos que guardarla en una variable (explicado en 2.3.2), que en cierto modo, se puede entender como darle nombre a la salida de la red.

2.3.2. Variables

En las propiedades de `input` y `output` de las neuronas, además de otras neuronas también pueden aparecer variables. Las variables tienen las mismas reglas para los nombres que los elementos (2).

En la propiedad de `output` si aparece un nombre que no sea de ningún elemento se asignará la salida de esa neurona a la variable. Esta variable ahora puede utilizarse como entrada de otra neurona (cumpliendo las mismas reglas de conexión lógica y válida que hay entre neuronas) o una salida interna (3.2.1). Es en cierto modo, una manera de renombrar la salida de la neurona.

Hay que tener en cuenta que la neurona sigue necesitando que su salida vaya a algún sitio, así que si se decide utilizar variables como (únicas) salidas de neuronas, estas variables han de ser utilizadas para que el resultado de la neurona vaya a, al menos, otra neurona (o una salida de la capa de salida).

Se pueden utilizar las variables más de una vez reasignándoles más valores, siempre y cuando estos valores no sean sobrescritos en la misma capa invalidando la conexión de dos neuronas.

A continuación se muestra un ejemplo básico del uso de variables para conectar "N1" y "N2".

```
...
C1 [
  N1 { input: [...], output: [SalidaDeN1] },
  ...
],
C2 [
  N2 { input: [SalidaDeN1: 0.5, ...], output: [...] },
  ...
]
...
```

3. Instrucciones

3.1. Variables de entrada

Para declarar una variable de entrada y asignarle un valor se pone el nombre de la variable, un símbolo igual y el valor.

```
Entrada=3.14
```

En ese fragmento de código se ha asignado 3,14 a la variable de entrada "Entrada".

Si en algún punto del programa se desea cambiar el valor de la variable de entrada, se puede asignar de la misma manera.

3.2. Selección de salidas

Antes de ejecutar una red neuronas tenemos la opción de seleccionar salidas que se quieran devolver en la siguiente ejecución.

Si no se selecciona ninguna salida, se seleccionarán automáticamente todas las salidas de la red (capa de salida) con los nombres que se le hayan dado.

Para seleccionar como salida una variable se escribe un símbolo 'mayor que' seguido del nombre de la variable.

```
> Salida
```

En este fragmento de código se selecciona la salida "Salida" para que en la siguiente ejecución que se realice se devuelva su valor.

Se pueden seleccionar tantas salidas como se quiera. Si un nombre no está definido, se devolverá `None`.

3.2.1. Salidas internas y elementos como salidas

Se pueden seleccionar variables internas de la red como salida también, esto incluye tanto variables que se utilicen para conectar neuronas como variables que símplemente se declaren en el `output` para asignarles el resultado.

También se puede seleccionar cualquier elemento para salida utilizando el nombre de un elemento en lugar de una variable. La diferencia es que el seleccionar elementos se muestra una básica explicación del cálculo que se realiza. Si es una neurona se muestra el cálculo de la neurona. Si es una red o una capa se muestran los cálculos de todas las neuronas de salida del elemento.

Estas "salidas internas" para depurar una red neuronal que sospechemos que esté dando algún valor erróneo porque nos hemos equivocado al definir la red.

3.2.2. Salida especial "all"

Se puede seleccionar también "all" como salida. Al seleccionarla se irán imprimiendo los resultados de todas las neuronas según se vayan calculando. Esto no influye en el resto de salidas seleccionadas.

3.3. Ejecución de una red

Para ejecutar una red se escribe una flecha con un guión y un símbolo 'mayor que' (`->`) seguido de el nombre de la red a ejecutar.

```
 ...
-> R
```

En ese fragmento se ejecuta la red "R" definida en el mismo programa.

La red tomará las variables de entrada que previamente hayan sido asignadas y imprimirá por pantalla las salidas seleccionadas. Una vez termine se desmarcarán todas las salidas.

4. Ejecutar un programa

Un programa de NN se ejecuta a través del script `nn` de la siguiente manera:

```
nn [-h] [-i input_file] [-I input] [-a] file
```

Siendo `file` la ruta hasta el fichero que se desea ejecutar. Además de la opción de ayuda (`-h`) dispone de otras 3 opciones que pueden resultar útiles a la hora de ejecutar los programas.

- `-i`: Junto con un fichero en el que solo se definan variables, se pueden pasar variables de entrada para el programa y separarlas por tanto de la definición de la red neuronal.
- `-I`: Junto a una declaración de variable de entrada, permite pasar variables de entrada al programa directamente como opción al ejecutarse.
- `-a`: Si está presente, la salida especial "all" (3.2.2) estará activada durante todo el programa.

Las variables que se pasen por esta vía, ya sea con un fichero o con por la línea de comandos, sobrescribirán a las del programa (durante toda la ejecución) en el caso de que tengan el mismo nombre.