Implementación de la función XOR mediante una red neuronal: nntool de MATLAB

La función XOR es una función lógica tal que cuando alguna de las entradas es 1, siendo la otra cero, nos produce una salida igual a 1. Si las dos entradas tienen el mismo valor, la salida será 0.
La tabla siguiente nos explica esto de una manera más visual:

XOR		B	
		0	1
A	0	0	1
	1	1	0

Lo primero que debemos tener para la creación de la red neuronal es el conjunto de datos « entrada → salida » obtenidos del sistema al que se va a implementar la red neuronal. En nuestro caso, el conjunto de datos los creamos nosotros mismos, pues sabemos de antemano el resultado que se va a obtener.

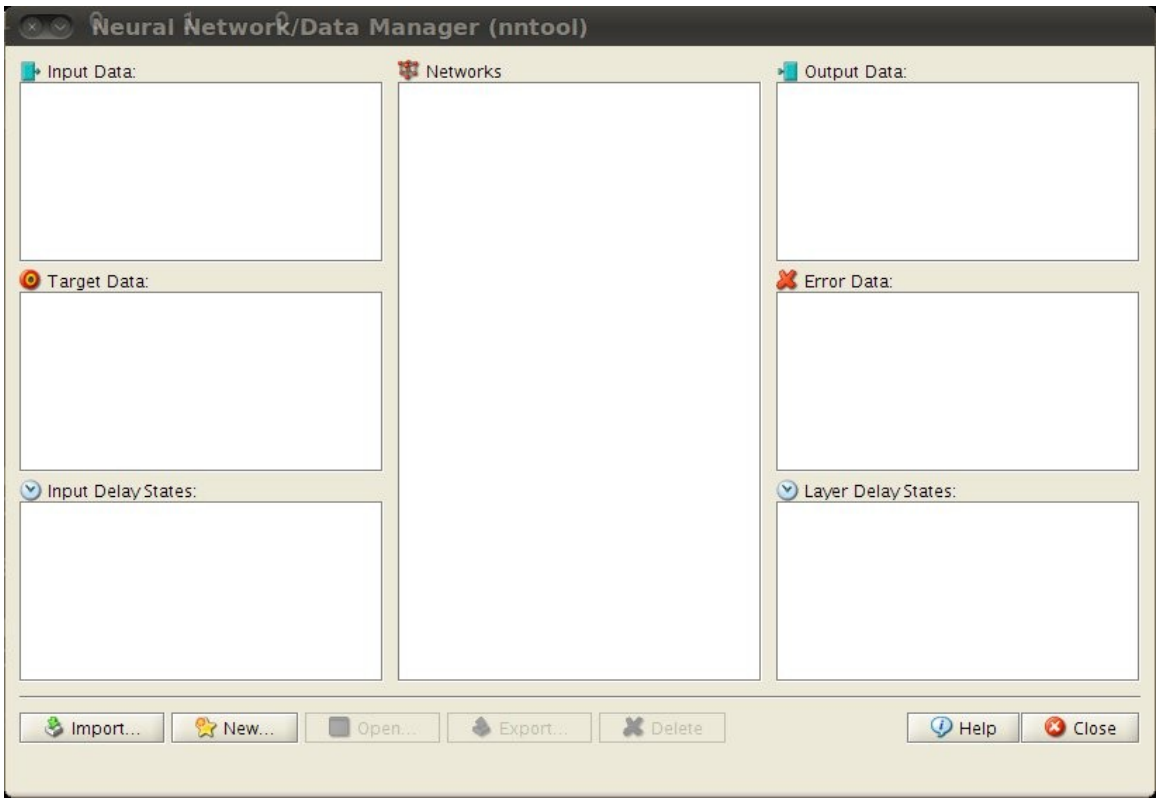
Nuestro conjunto de entrada tiene la forma : $input = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$

donde cada fila representa los posibles valores de cada entrada. Cada columna será pues un posible conjunto de entradas. Este detalle que puede obviarse es de suma importancia a la hora de usar la *nntool* de MATLAB, pues si los datos no se presentan de esta manera (datos de cada variable de entrada en una fila distinta), la creación de la red será erróneo cuando no imposible.
Nuestro conjunto de salida tiene la forma : $output = \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$

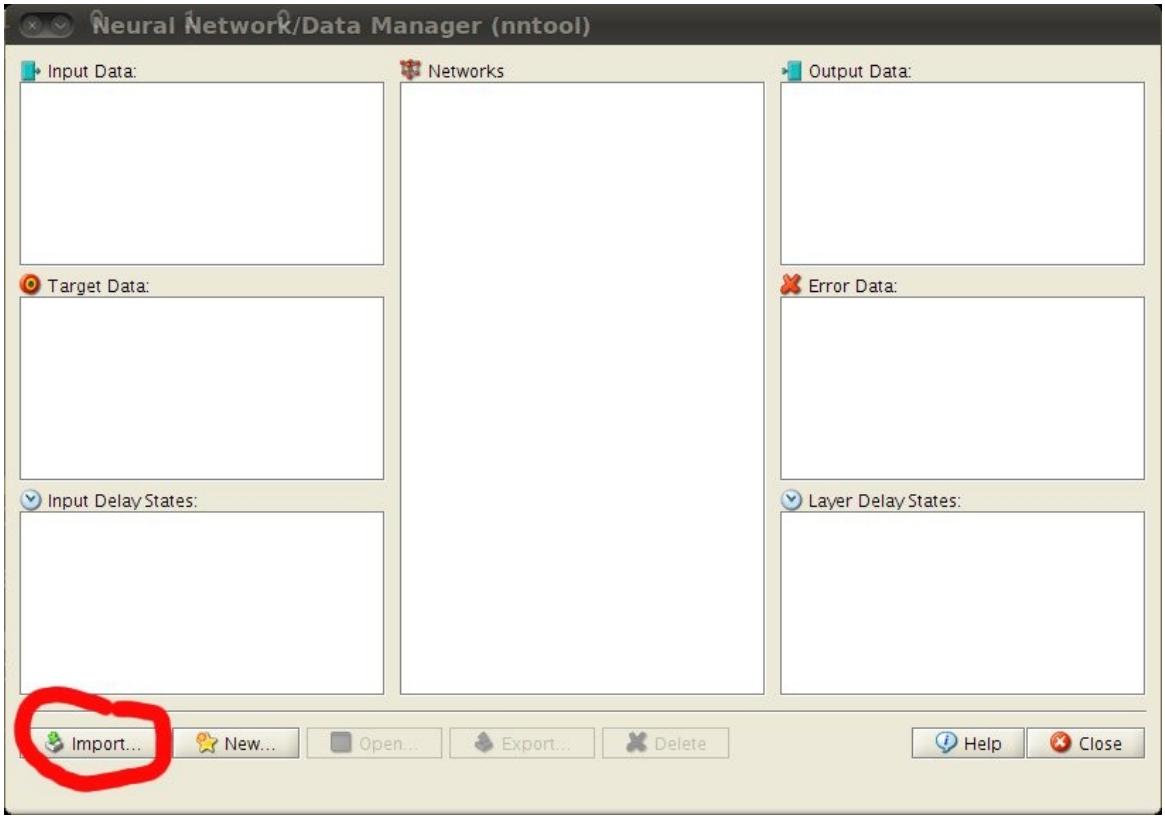
Un detalle a tener en cuenta a la hora de trabajar con redes neuronales es el entrenamiento de las mismas. Cuantos más datos tengamos/usemos para la fase de entrenamiento, mejor será el resultado obtenido. Por ello, se duplican las matrices de entrada/salida para ser usadas en el entrenamiento. Así, y usando notación MATLAB, las matrices de entrada/salida a usar para el entrenamiento serán :

input = [0 1 0 1 0 1 0 1 0 1 0 1 0 1;0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1];
output = [0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0];

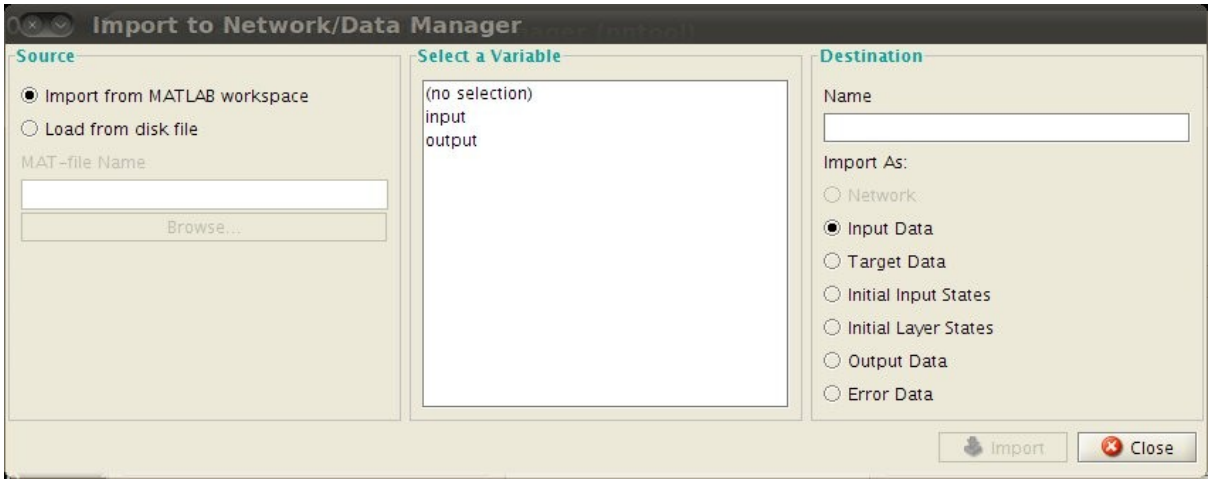
Ya estamos en disposición de empezar a usar la herramienta *nntool*.
Tras abrir la GUI con el comando *nntool*, nos aparecerá la siguiente ventana :



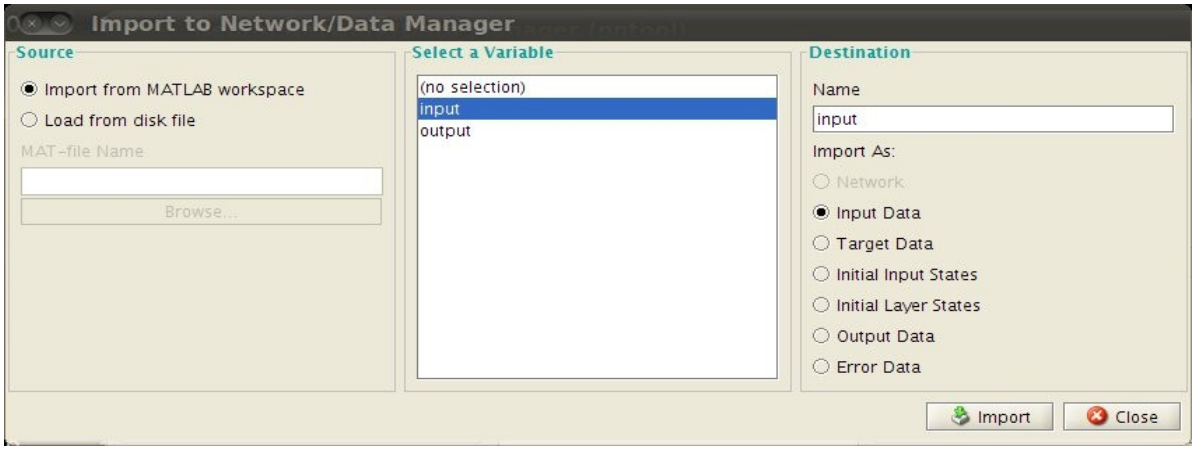
En esta ventana debemos seleccionar los valores que vamos a usar para el entrenamiento. Para ello pulsamos en *import* (ver figura siguiente) :



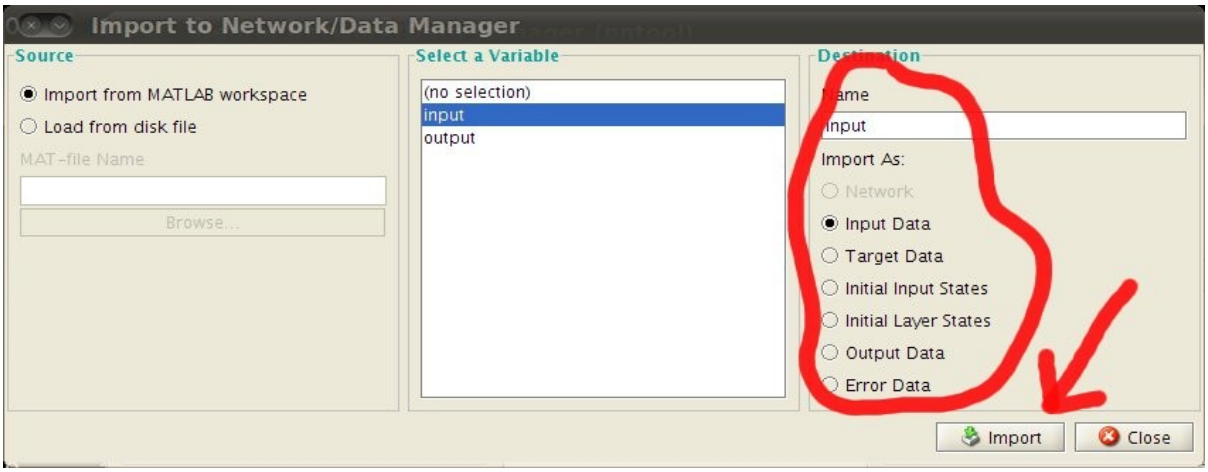
Este botón nos abrirá una nueva ventana :



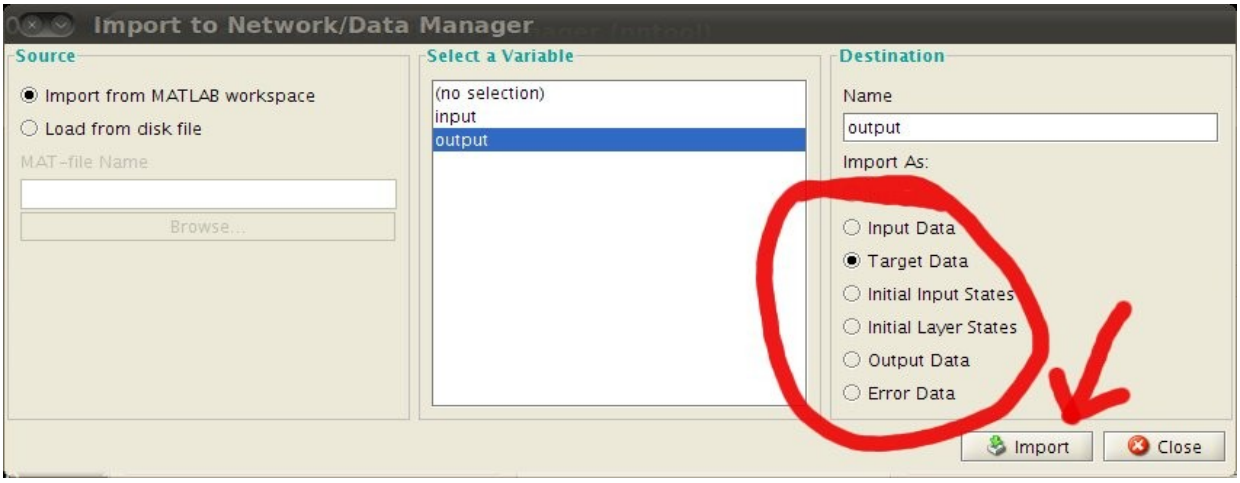
Ésta nos presentará las variables presentes en el *workspace* de MATLAB. En nuestro caso teníamos creadas sólo los datos de entrada y salida que vamos a usar en este ejemplo. Si clickeamos sobre la varible *input :*



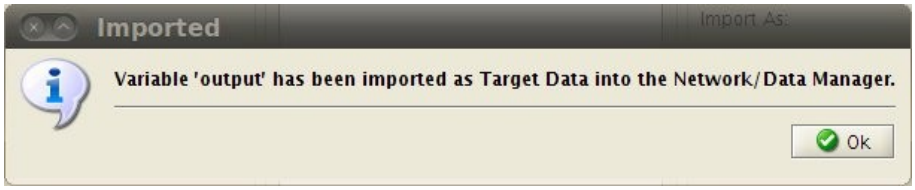
vemos que se nos activa el botón para importar la selección. Previamente debemos decidir que categoría tendrá la variable seleccionada. En nuestro caso parece trivial, pues los nombres dados a las variables coinciden con la categoría que tienen y tendrán. Esto se elige en el cuadro derecho de esta ventana :



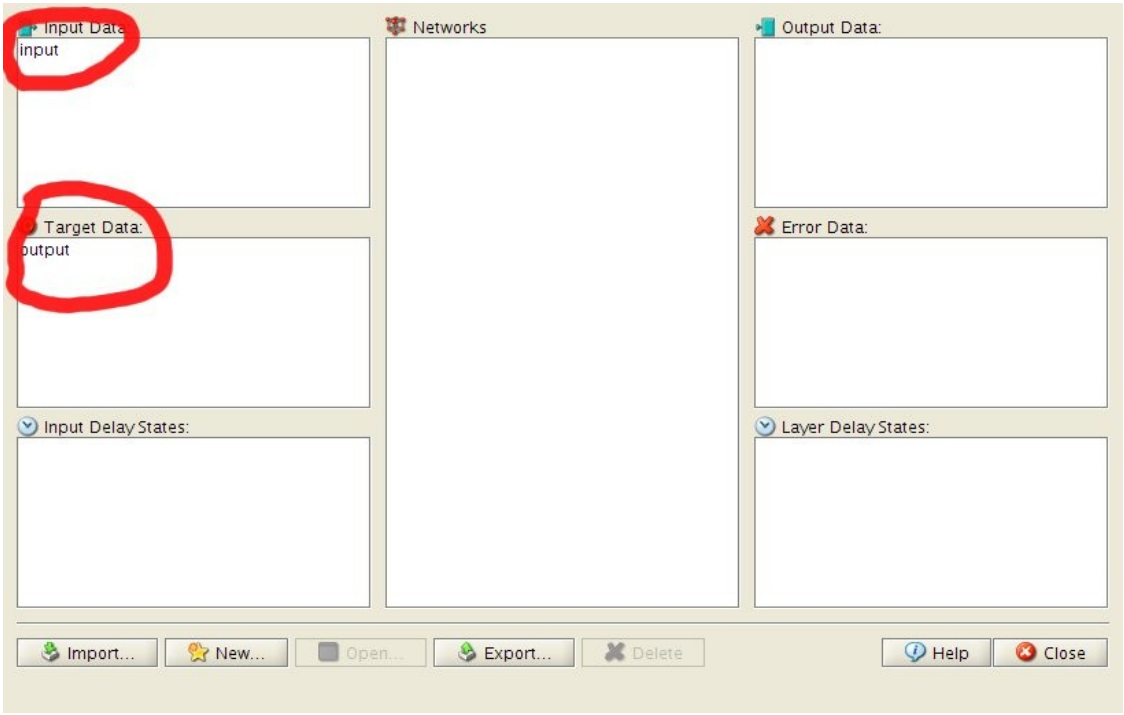
Tras importar la variable de entrada, hacemos lo propio con la de salida :



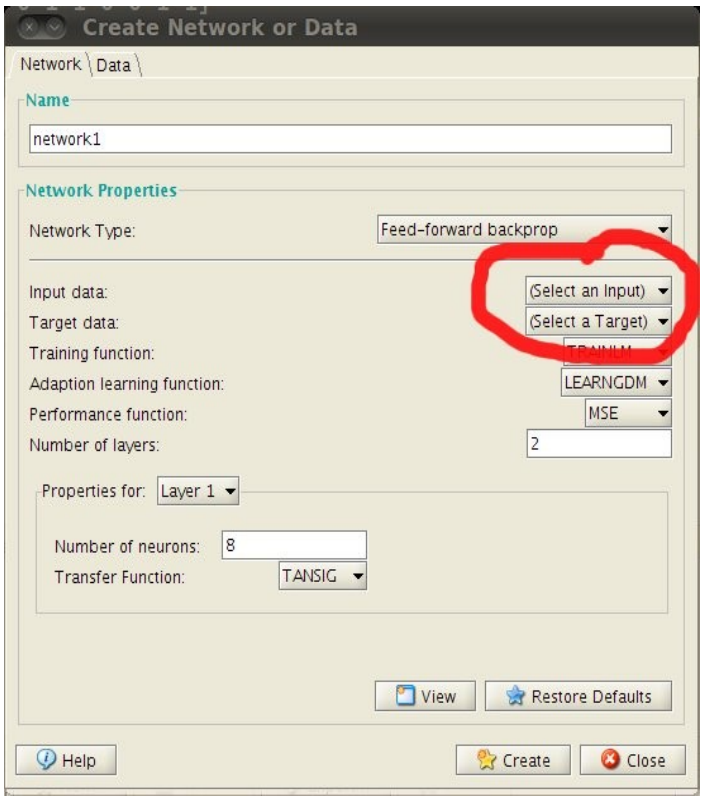
Después de importar alguna variable, la interfaz nos pide aceptar el paso, ni que decir tiene que si no se acepta, el proceso no puede continuarse :



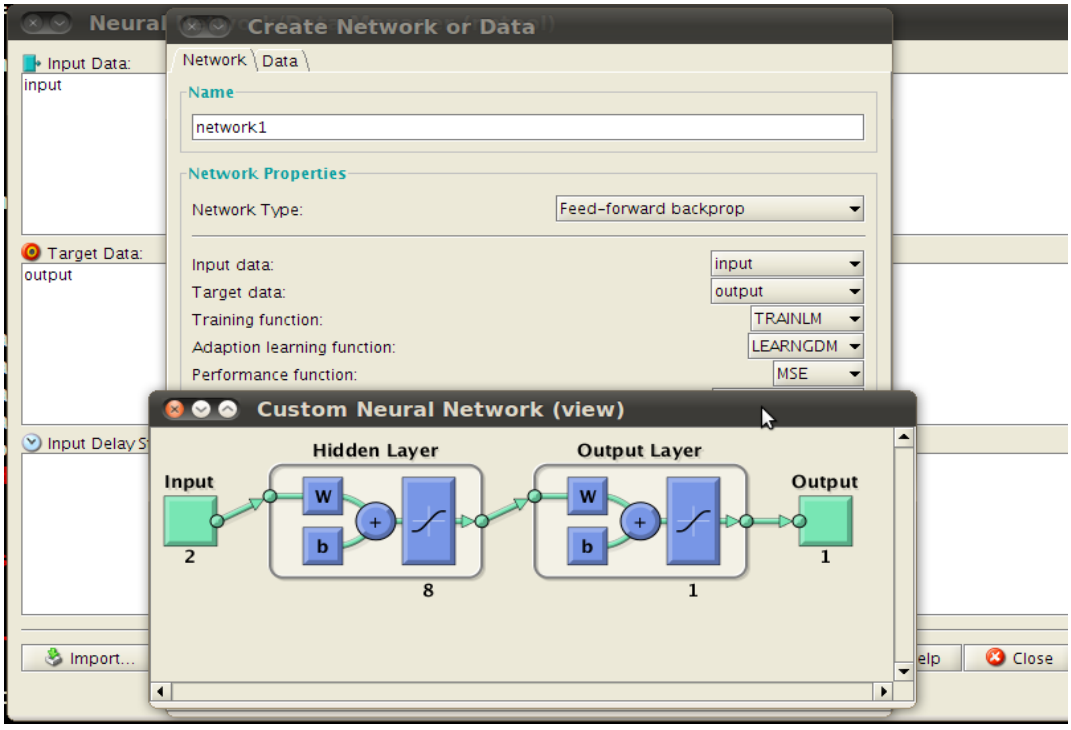
La interfaz inicial debe quedar como sigue, donde ahora aparecen los datos de entrada y los datos que se deben obtener en su lugar correspondiente :



El siguiente paso es la creación de la red neuronal en sí. Para ello, clickeamos en *New*, lo que nos abrirá una ventana :

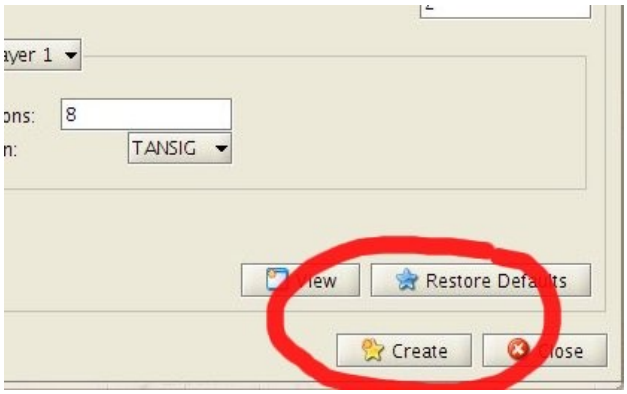


En esta ventana se seleccion el tipo de red que se quiere crear y las características de las mismas, que diferirán de una a otra. En nuestro caso, nos vamos a quedar con una red tipo *Feed-forward backdrop*, con dos capas y 8 neuronas en la capa inicial. Importante seleccionar los datos de entrada y los datos objetivo para poder crear la red. Una vez seleccionados éstos, podemos ver la estructura de la red gracias al botón *View* :

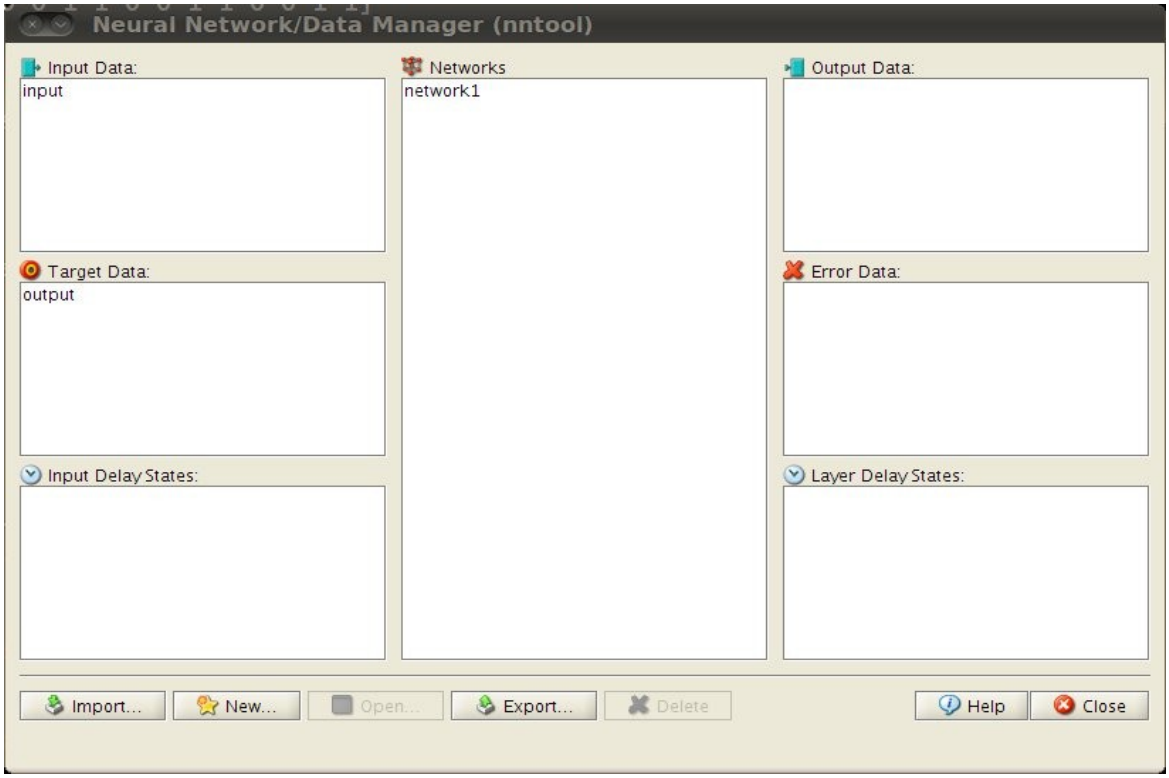


Esta representación lo que nos viene a decir es que nuestra red tendrá dos entrados, las cuales pasan sus valores a las neuronas de la primera capa (ocho neuraonas). Esta capa computa estos valores y pasa el resultado a la siguiente capa (con una sola neurona), la cual hace el cómputo final que pasa a la única salida de nuestra red.

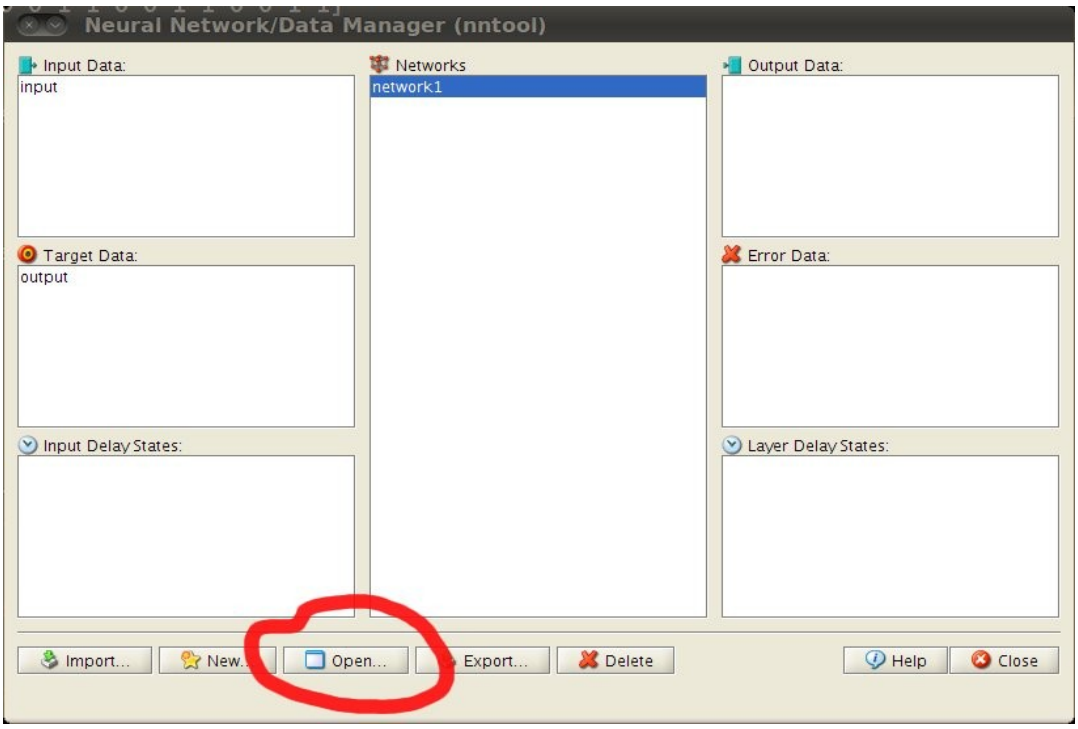
Podemos crear ya nuestra red :



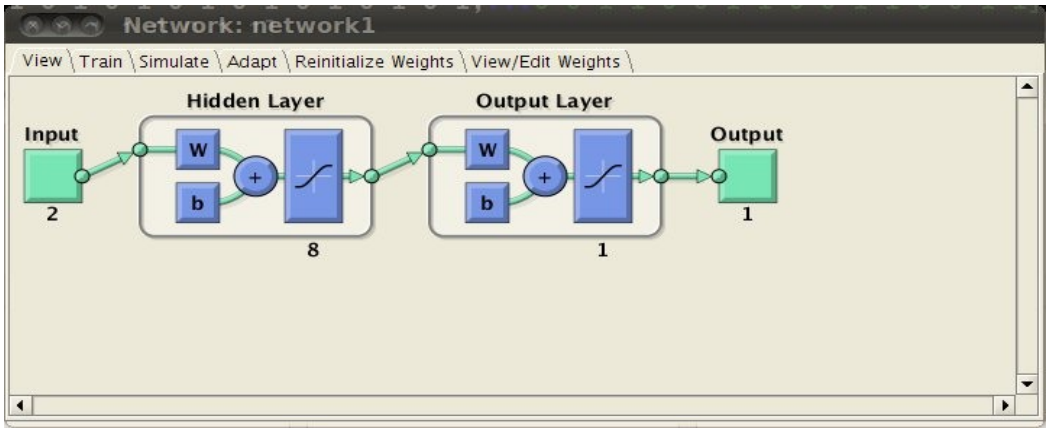
Tras crear nuestra red, ésta aparecerá en la interfaz inicial :



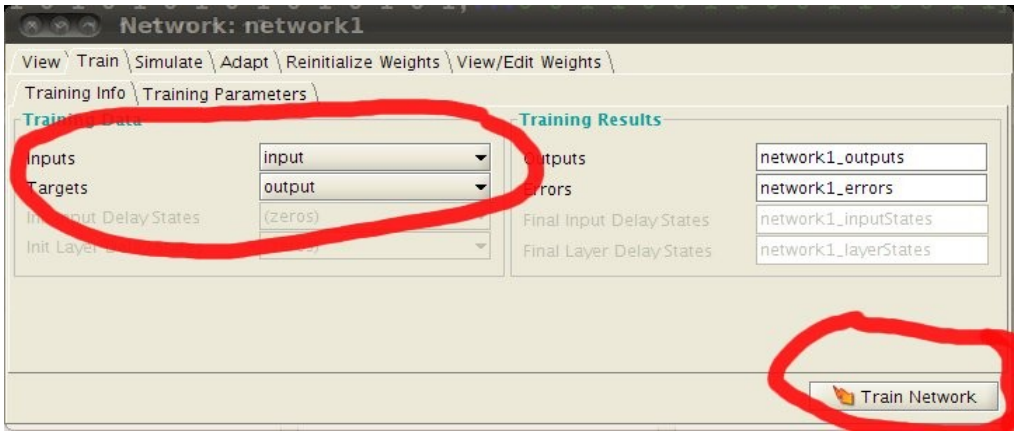
Si selecionamos la red y la abrimos :



nos aparecerá otra nueva ventana que nos ayudará a entrenar y establecer los valores de los pesos de cada neurona en nuestra red :

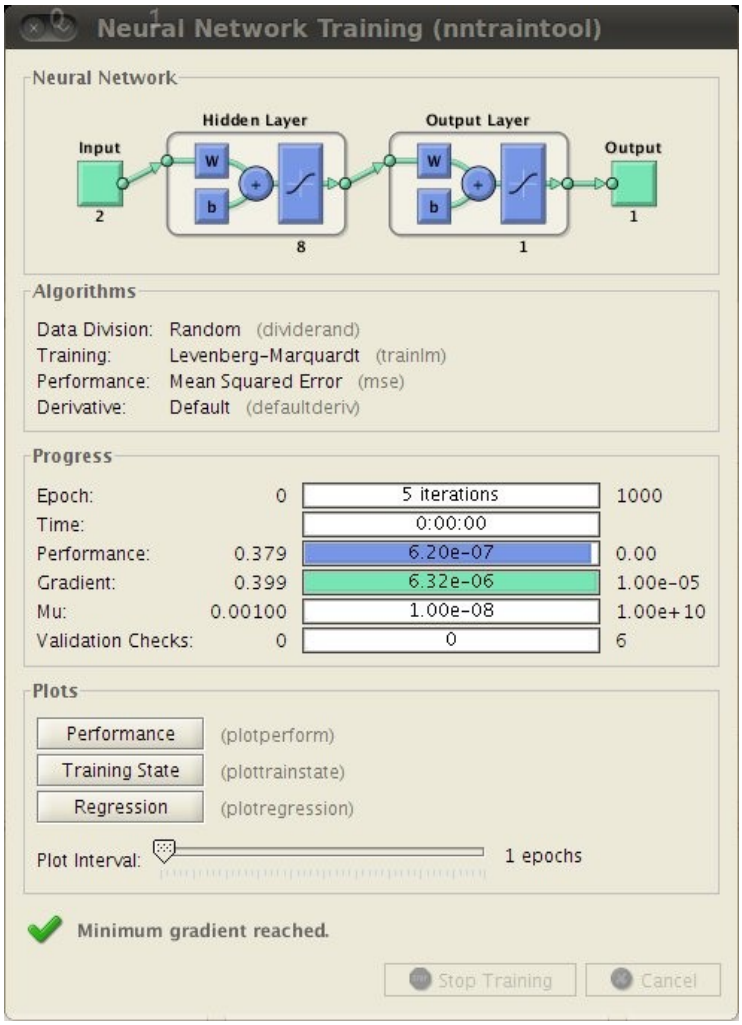


La primera imagen que nos presenta es la estructura de la red neuronal creada. Pero si clickeamos en la pestaña *Train*, nos aparecerá lo siguiente *:*



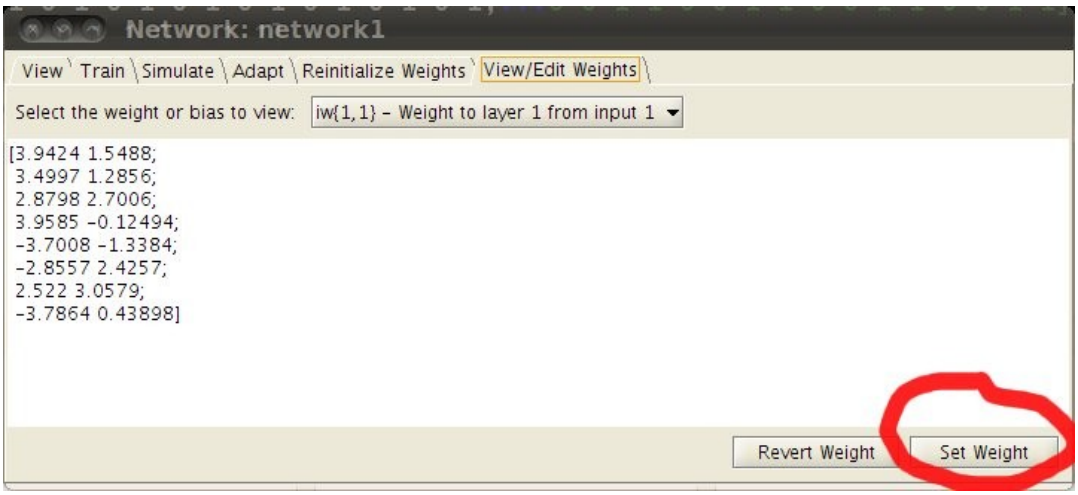
Aquí volvemos a seleccionar de nuevo los datos de entrada y objetivo (las quejas ante tanta insistencia en la selección de esto datos deben dirigirse a los creadores de la toolbox, oseasé, Mathworks Ltd.).

Con los datos seleccionados, entrenamos la red clickeando el botón *Train Network*, lo que nos presentará otra ventanita con la información del entrenamiento de la red.

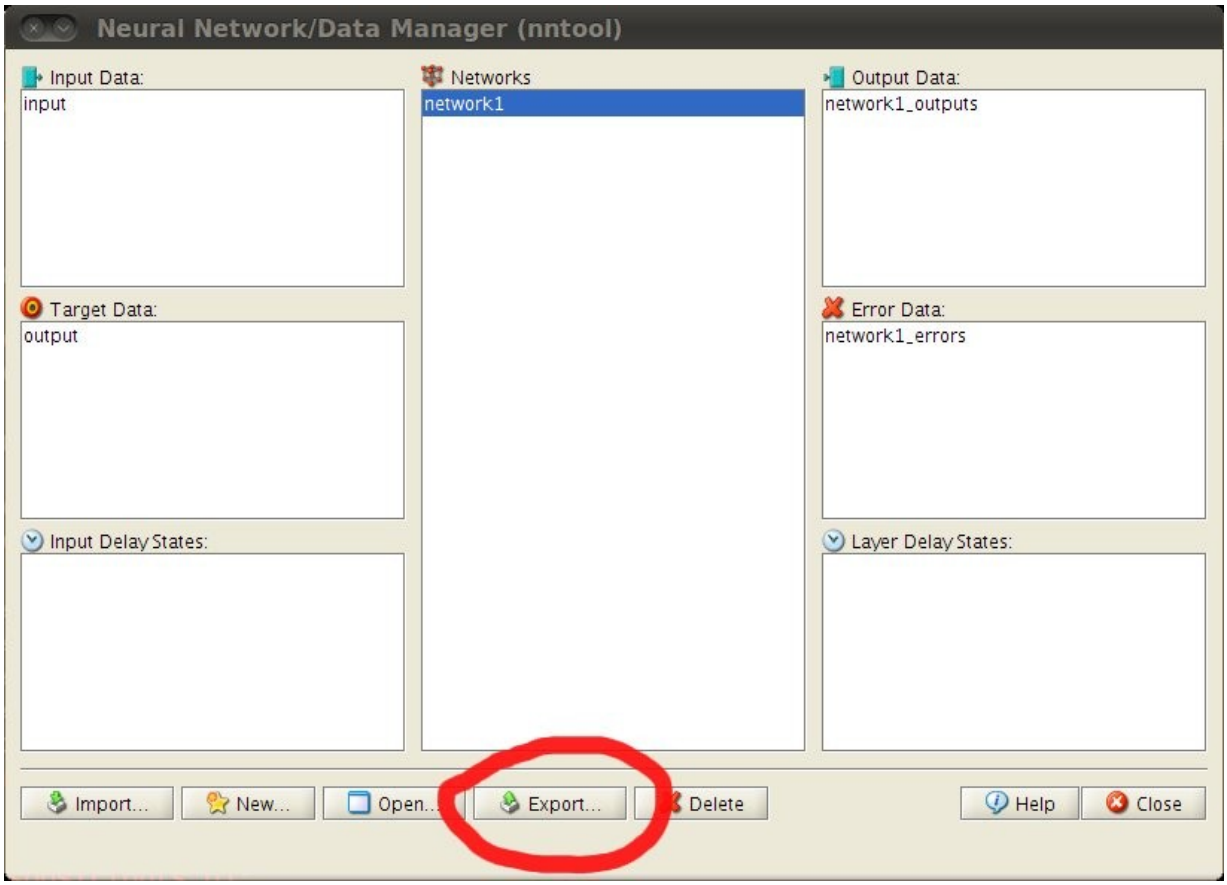


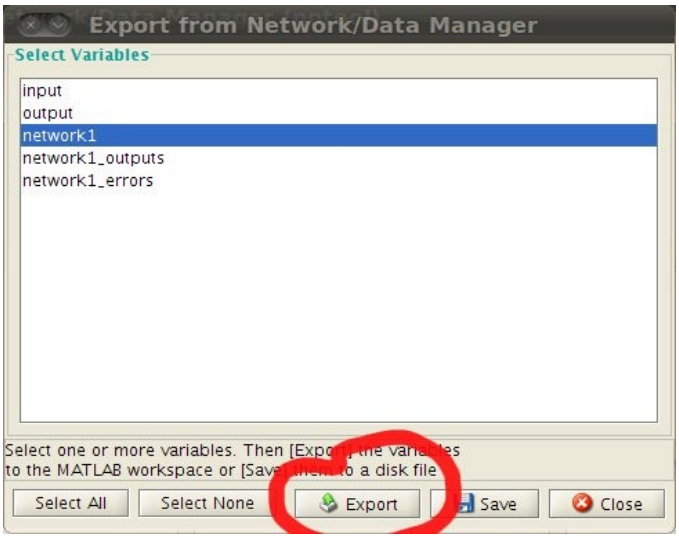
Aunque no lo parezca, esta ventana nos da mucha información de la acontecido en el entrenamiento, y puede servirnos para juzgar si debemos re-entrenar a nuestra red o quedarnos con el primer entrenamiento. Nosotros nos quedamos con el resultado obtenido.

Si cerramos esta ventana y volvemos a la anterior y clickeamos sobre *View/Edit Wieghts*, nos aparecerán unos números que parecen misteriosos, pero no lo son. Éstos números son los valores de los pesos de las distintas neuronas de nuestra red, y si nos gusta la pinta que tienen, los aceptamos tal como son mediante el botón *Set Weight*.



Una vez aceptados/establecidos estos valores, podemos cerrar esta ventana y volver a la inicial, en la que aparentemente nada debe haber cambiado. Y se dice aparentemente porque la realidad es bien distinta. Esa red neuronal creada hace rato, ahora tiene sus neuronas sintonizadas con los datos tantas veces selecionados. Podemos importar esta red al *workspace* para disfrutar de ella.





Con la red neuronal en nuestro *workspace*, podemos ver cómo funciona la misma. Ésta habrá sido exportada como una estructura con infinidad de propiedades. Lo que nos atañe en este caso es usar la red para que, según los datos de entrada, nos produzca una salida adecuada.

Desde la terminal de MATLAB, podemos probar lo siguiente :

```
>> salida = network1([0;0])
salida =
    0.0013
>> salida = network1([0;1])
salida =
    0.9996
>> salida = network1([1;1])
salida =
   1.0406e-04
>> salida = network1([1;0])
salida =
    0.9998
```

Vemos lo extraño de los valores de salida. Si queremos algo más.....'entero', podemos probar redondeando la salida con *round* :

```
>> salida = round(network1([0;0]))
salida =
     0
>> salida = round(network1([0;1]))
salida =
     1
```

```
>> salida = round(network1([1;0]))
salida =
    1
>> salida = round(network1([1;1]))
salida =
    0
```

Parece ser que nuestra red hace lo que se espera de ella, ¿no ?

Es interesante hacer lo mismo pero con conjunto de entrenamiento menores, por ejemplo usar

input = [0 1 0 1; 0 1 1 0];
output = [0 0 1 0] ;
como datos de entrada y objetivo. Se verá como cambia el resultado obtenido con la red.
También es interesante probar distinto numéro de neuronas (en lugar de 8, probar con 6 ó 4 ó 2 ó 20, por ejemplo). Muchas veces es sorprendente ver el buen resultado que se obtiene con un número de neuronas pequeño.