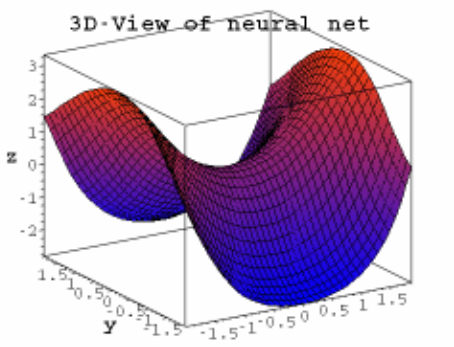
por Ralf Wieland
<rwielandQzalf.de>

Sobre el autor:

Mi interés se centra en la programación entornos de simulación,redes neuronales y sistemas difusos. El programa más reciente lo he desarrollado bajo Linux (desde 0.99pl12). Además, estoy interesado en la electrónica , en el hardware y la manera de relacionarlos relacionarlos con Linux.

El Linux en la Ciencia ó Cómo se ha desarrollado una aplicación de Redes Neuronales



Resumen:

En este artículo demuestra la idoneidad para la ciencia del software desarrollado bajo Linux . Se presentará el desarrollo de herramientas para la simulación de paisajes. Como ejemplo, se explicará un simulador de redes neuronales que se ha desarrollado con GPL.

Traducido al español por:
Alberto Pardo
<apardoyo/at/yahoo.es>

¿Cuál es el objetivo?

Trabajo en un instituto de investigación , el cual se centra en investigaciones sobre la tierra. Investigamos preguntas como las siguientes:

- ¿Podremos beber agua durante más de 50 años sin restricciones?
- ¿Qué plantas y animeles continuarán viviendo si hay un cambio climático? ¿Todavía habrá bosques? ¿Se podrán continuar sembrando las tierras de cultivo?
- ¿Hacia dónde vuela la arena del desierto? y ¿Cómo se expanden los desiertos?

Cada una de estas preguntas encierra una gran trabajo de investigación desarrollado por muchos científicos. Mi ocupación se centra en cómo linux puede utilizarse para contestar estas preguntas. Para aclarar todo esto,

tenemos que examinar la manera con la que se realizan las investigaciones y los análisis. Este tipo de retos tan desmoralizadores, como los mencionados anteriormente, consisten analizar muchos subproblemas. Si uno quiere clarificar si el agua permanece potable, se necesita obtener los datos del agua. Las fuentes de los datos son diversas. Sirva como ejemplo que la agricultura constribuye considerablemente en la polución del agua.¿Pero con que grado lo hace actualmente? y ¿Con qué intervalo de tiempo se contamina el agua?
Para contestar este tipo de preguntas, se ha de considerar cada dato aportado. Es un proceso elaborado y por desgracia, acompañado de errores serios. Quién sabe a ciencia cierta la contaminación que aporta el aire, através de los desechos industriales o de la agricultura. Los datos están asociados con las precipitaciones, su transporte y la evaporación. Además , estos factores están influenciados por los cambios climáticos. Para poder investigar este proceso, la simulación por ordenador es un requisito indispensable . Como paso previo a la simulación, se han de determinar un número de restricciones, parámetros y funciones. Las funciones y los parámetros se extraen de los test de laboratorio, test de campo y observaciones reales. Nos indican la absorción del fertilizante por las plantas, el proceso de degradación del suelo, etc.
La simulación se basa en dar por supuesto un escenario base y se ejecuta mediante Simulación Monte Carlo. Para hacer esto se varía estocasticamente los datos y se repite la simulación con diferentes condiciones iniciales. El resultado es una colección de datos posibles, pero con diferentes probabilidades. Este conjunto de datos han de ser analizados y editados para servir como base para la toma de decisiones. El logro del modelo regional es la preparación para cambios potenciales y desarrollar hoy técnicas para un uso sostenible de la tierra.
No podemos predecir el futuro, pero estamos preparandonos para él.
Para ser más precisos, tenemos que reconocer que el trabajo de modelación consiste en dos partes.En la primera, los modelos se han de adaptar, se necesita un conjunto de datos para analizar y se deben escribir artículos. La segunda parte consiste en el desarrollo de un software optimizado para hacer la investigación.

El trabajo detallado diario con Linux

La misión diaria, consiste en el análisis de datos,quejandonos de los datos de medidas incorrectos, transformando los datos a los diversos formatos, escribir informes,etc..., beneficiandonos de Linux. Si aún alguien cree que con Excel puede puede hacer todo, es que no ha probado la combinación de Perl, Emacs,octave [www.octave.org], R [www.r−project.org] etc. Juntos han demostrando ser un duro adversario en la batalla con los datos.*Perl* es muy versatil,no esta limitado por la conversión de datos y requiere basesde datos (MySQL), ejecuta cálculos, etc., todo esto de manera rápida y repetidamente. Especificamente, esto último es muy portante, ya que el trabajo manual lleva a manejar los datos de manera erronea. Esto es raro si se utlizan scripts ya comprobados. Escribir artículos con *LaTeX*, permite obtener un resultado convincente y de gran cálidad gráfica. Linux posee herramientas con las que hacen actractivo el trabajo a los científicos. No queremos ocultar una desventaja: el proceso de aprendizaje de estas herramientas es largo. No todo se puede hacer intuitivamente y no todo el mundo es un "FREAK" de la programación.

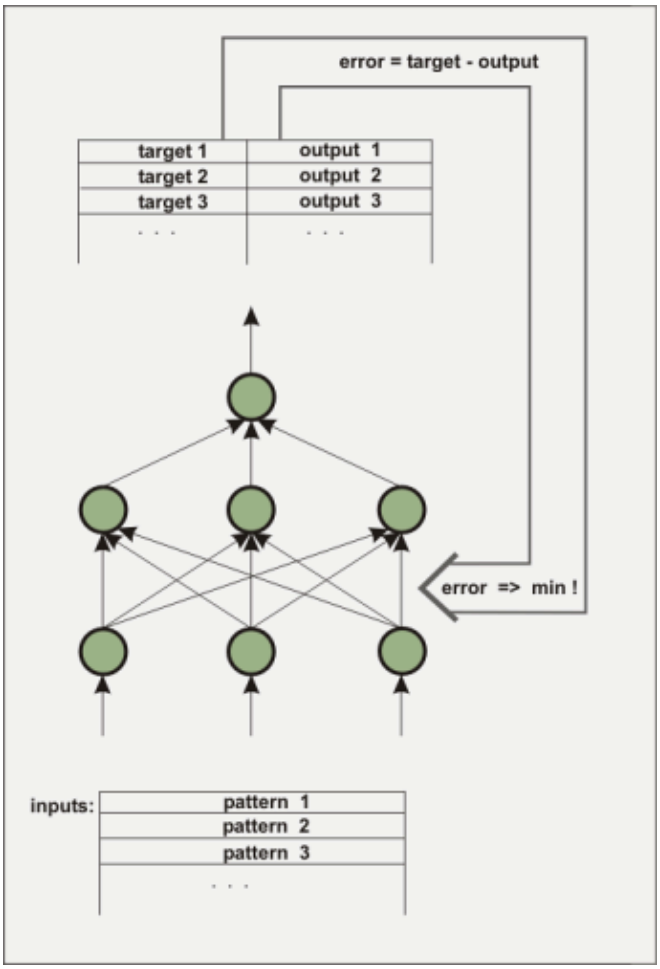
El Siguiente Paso – El Desarrollo de las Herraminetas

¿Por qué se ha de desarrollar uno mismo las herramientas?, ¿No esta todo disponible? Existen herramientas de alto rendiemiento para la simulación, como Matlab [www.mathworks.com]. Existen herramientas para procesar datos geográficos del Sistema de Información Geografica (GIS) como ARCGIS [www.esri.com/software/arcgis] o sotware libre Grass [grass.itc.it] . Y también hay programas para estadística . Así, ¿Por qué continuar desarrollando?
El problema no está en la cálidad de los componentes, sino en su interacción en el sistema. En la simulación, las subtareas las realizan diferentes programas que se comunican de una manera , que se le podría llamar pesada, atraves de interfaces creados por el propio usuario. Este hecho queda agravado por el hecho que los datos disponibles en grandes cantidades (datos espaciales) contienen un gran porcentaje de errores. Las simulaciones básicas se han de ajustar para esta carácteristica. Un algoritmo ha de conseguir resultados

validos, y si los datos entrados no están totalmente emparejados, ha de indicarlo mediante un mensaje de aviso . En el procesado de grandes cantidades de datos ,es normal tener matrices con más de un millon de elementos y por lo tanto, se requieren algoritmos rápidos. Sólo se obtienen algoritmos robustos y rápidos, si se los desarrolla uno mismo.
La principal desventaja de los sistemas comerciales es que su código fuente es secreto . ¿Cuantos científicos desarrollan e intercambian modelos si las fuentes no son libres? Con esta conclusión decidí desarrollar una Herramienta de análisis espacial y de modelación ("Spatial Analysis and Modeling Tool" (SAMT)) como software de código abierto.
. Es una herramienta de simulación que incorpora un gestor para datos espaciales, interfaces con la base de datos MySQL y a GIS. Contiene las funciones elementales para gestionar datos basados en "raster" y tiene capacidad para manipularlos (homogeneizar, distancias, interpolar, etc.) y capaz de generar presentaciones de los datos en dos o tres dimensiones.
Nota: Los datos del tipo "raster" se basan en dividir un mapa en una rejilla de cuadrados pequeños. La información se almacena en varias capas de datos raster.Un modelo accede a la informació de las capas.Además de accededer a la información cada vez más profunda, también es fundamental la información envolvente del mismo nivel. Esto es la base para el módelado de flujos laterales, como ocure en la erosión del suelo, causado por el aire y el viento.
SAMT genera la estructura en la cual las herraminetas – como interpretador difuso (muy rápida) y la herramiente de red neuronal (*nnqt*) – pueden adaptarse. Los modelos difusos sirven para integrar sistemas expertos en la simulación. Un sistema experto con frecuencia describe un proceso o lo controla, principalmente si no hay módelo matemático. Las redes neuronales tienen procesos los cuales nos permiten derivar correlaciones funcionales de los datos de medida. A continuación introduciremos el desarrollo de una herramienta de redes neuronales.

¿Qué es una red neuronal?

Una red neuronal consiste en varios estratos o capas. El primer estrato se cargará con los datos iniciales a probar,en números de punto flotante. La capa entre la entrada y la salida no es visible desde fuera, por lo que se le llama 'estrato oculto'. En algunos casos se pueden tener varios estrato ocultos. El estrato de salida, por ejemplo, únicamente tiene un elemento. Este tipo de arquitectura se usa para construir funciones con varias entradas y una salida. Los estratos ocultos son necesarios para analizar el comportamiento no lineal, por ejemplo el de la función x^2−y^2. ¿Cómo sabe la red neuronal cual es la función buscada? Inicialmente,la red no conoce la función. Los conectores (pesos) entre los elementos (nodos) son atribuidos con valores estocasticos. Durante el proceso de entrenamiento, el algoritmo que esta aprendiendo trata de cambiar los pesos de manera que el error cuadrático medio entre el ordenador y el valor predeterminado de salida sea mínimo. Hay una variedad de algoritmos que ya se encargan de ello y no será necesario elaborar ninguno en este artículo. Tres algoritmos fuerón implementados en *nnqt*. El proceso de 'Enseñanza supervisada consiste en diseñar la salida dependiendo de la entrada.



La red está entrenda para detectar si ha alcanzado un error mínimo con los datos de aprendizaje tanto como con los datos de control (es aconsejable separar los datos previos del entrenamiento de los usados como datos para verificar la función de aprendizaje). Los pesos determinan la conducta de la red y son almacenados para este proposito. ¿Qué se espera conseguir con esta red? Aparte de las aplicaciones para la ciencia, tambíen hay aplicaciones más o menos serias. Hay intentos de predecir las tendencias de los mercados financieros. No he conseguido nada en este campo, pero quizas alguien lo haga.
Otra aplicación interesante podría se la aplicación de las redes neuronales para la predicción del tiempo a corto plazo. Las estaciones meteorológicas electrónicas pueden usarse para instruir a la red neuronal. La utilidad podría ser analizar la presión atmosférica y sus cambios,así como las precipitaciones. Las estaciones meteorologicas ya siguen estos patrones. ¿Puede una red neuronal mejorarlo?. Para dar soporte a la propia experimentación esta disponible el software *nnqt* esta disponible como software GPL.

La Herramienta de Red Neuronal *nnqt*

Los científicos reclamaban el desarrollo de herramientas de redes neuronales , así que analicé sus requerimientos. Querian una herramienta lo más simple posible, la cual pueda ser utilizada para aplicaciones espaciales,es decir, deseaban ver como se relacionaban los datos con su situación espacial. De acuerdo, existen excelentes herramientas comercializadas de redes neuronales. Por suerte, se encuentran disponibles herramientas libres como SNNS [www−ra.informatik.uni−tuebingen.de/SNNS/] o librerias como fann [fann.sourceforge.net]. SNNS es estupenda , pero no es fácil de utilizar por alguien que no sepa programar, ya que entrega los resultados de salida en código C. En su campo puede resultar apabullante para un usuario ocasional.*nnqt* necesita conocer algunos requerimientos:

- Integración hacia SAMT (usa datos "raster" como conjunto de entrenamiento y aprovecha redes almacenadas en modelos espaciales)
- Manipulación interactiva, integración de técnicas de análisis
- Utilización como herramienta fuera del SMAT

El desarrollo de nnqt

El desarrollo consta de los siguientes pasos:

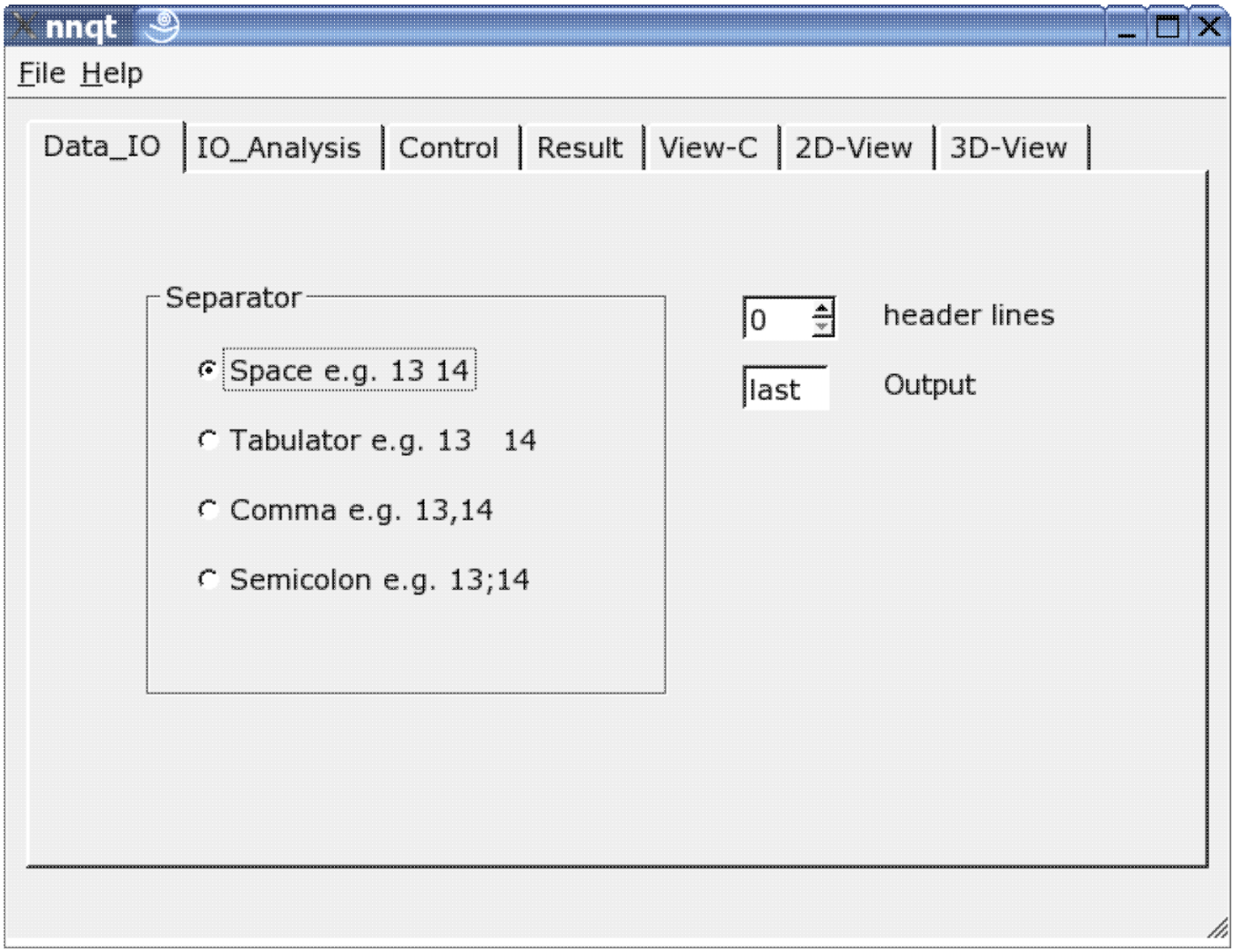
1. Desarrollo y test del algoritmo
2. Implementacion como aplicación qt
3. Integración con SAMT

Desarrollo y test del algoritmo

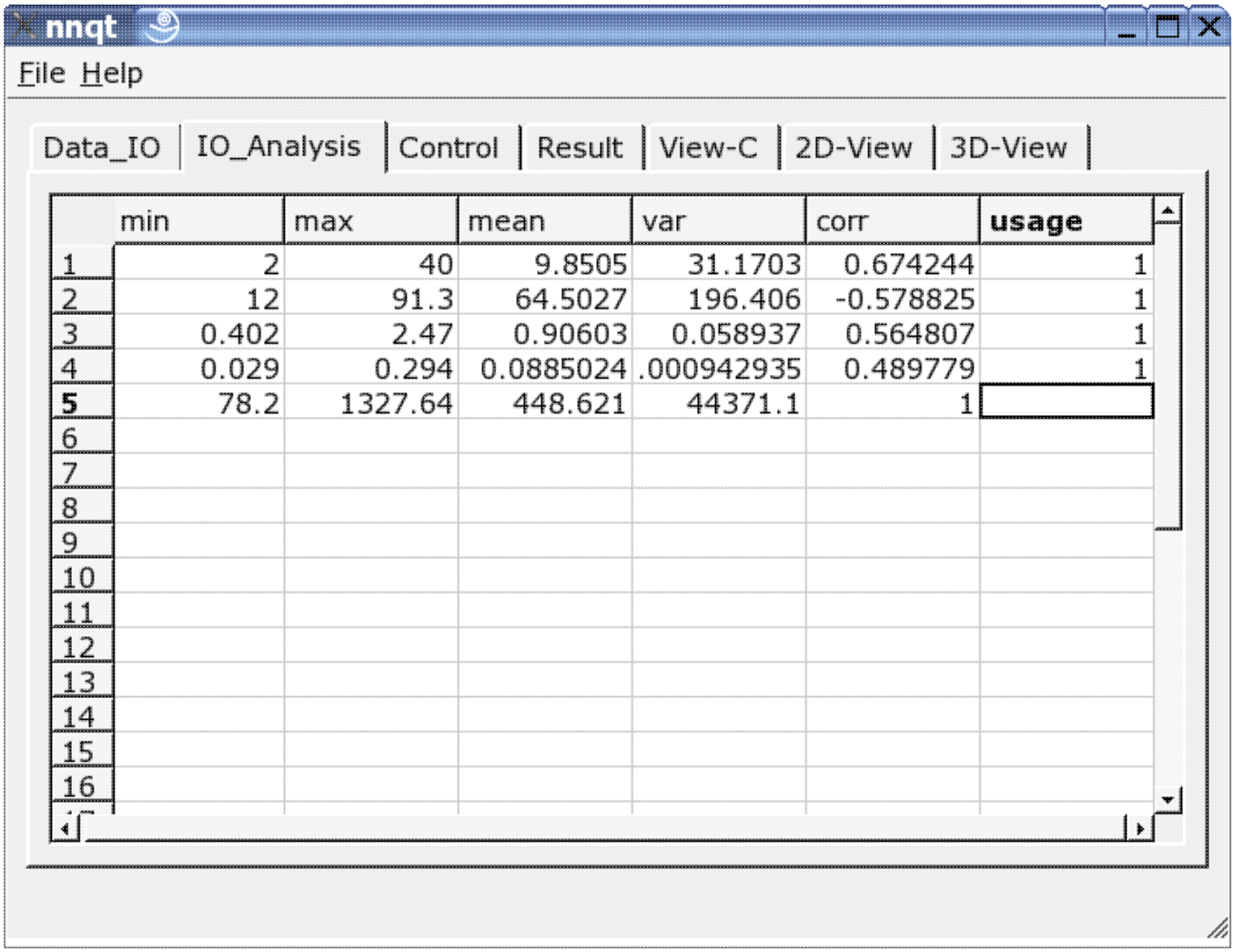
Existe una buena y abundante literatura sobre redes neuronales. Como representación citaremos el siguiente libro. Sin embargo, algunas veces hay agujeros que tiene que ser cerrados mediante la propia experiencia o por intercambios con otros.Me gustaba el trabajo rápido con Matlab aplicando el algoritmo Levenberg−Marquardt−algorithm. Sólo después de una intensiva busqueda en Internet encontré articel [www.eng.auburn.edu/~wilambm/pap/2001/FastConv_IJCNN01.PDF][local copy, 105533 bytes] la cual describe el uso de este algoritmo con redes neuronales."Lo único" que hice fue integrar mis funciones favoritas *tanh* (tangens hyperbolicus) en el algoritmo. También, use el software de Linux: el sistema de algebra computacional Maxima [maxima.sourceforge.net]. Es posible con este tipo de sistema la manipulación de ecuaciones complicadas, to diffentiate and so on, operaciones que no son simples de resolver con papel y lápiz. Maxima hizó posible realizar las manipulaciones necesarias e implementar la primera versión del algoritmo en una semana. La implementación en C utilizó para probar los parámetros. Utilicé el sistema de simulación de "open source" desire [members.aol.com/gatmkorn] (¡Muchas gracias a su desarrollador el Prof. Korn!) como herramienta para la comparación y poder ejecutar el primer modelo.La nueva versión del algoritmo no fue tan mal. El tiempo de entreno para el problema *xor*, se vió favorecido con el ejemplo para redes neuronales, alcanzando una media de 70ms con un ordenador Pentium a 3GHz. (La gran parte de este tiempo esta siendo usado por el proceso de lectura del disco duro. En un viejo Athlon a 750MHz, este tiempo es ligeramente superior) .
Como alternativa , se implementó y analizó el conocido algoritmo "back propagation". Despues de estos preparativos, que erán las bases para fomentar mejoras en el algoritmo, se continuo con la puesta en marcha de la "toolbox".

Puesta en marcha y resultados

Como entorno de desarrollo priviligié a *qt*, esta bien documentada y puedo usarla con mi editor Emacs. El planificador de *qt* ayuda con el diseño de la superficie. Por contra, no hay sufucientes opciones para el desarrollo de *nnqt*. Necesitaba cosas como diagramas, escalas , etc. Con todo esto la comunidad de desarrolladores ganaban ayuda. Las librerias de qwt [qwt.sourceforge.net] y qwt3d [qwtplot3d.sourceforge.net] podrían ser utuluzadas, esta ayuda reducuriá el tiempo de desarrollo drásticamente. Equipado con estas fuentes , en pocas semanas se creo *nnqt*. Cuando estuve los sufucientemente satisfecho con los resultados , lo dirigí a los usuarios. ¡Tuvo mucha demanda!. El conjunto de datos debía ser dividido automaticamente hacia el grupo de aprendizaje y el grupo de comprobación,querían asignar nombres para mejorar la organización, más análisis como gráficos con curvas con parametros, etc. Bien, algunas cosas las podía integrar inmediatamente, otras características me llevarán más tiempo. A continuación se presentan algunas pantallas:

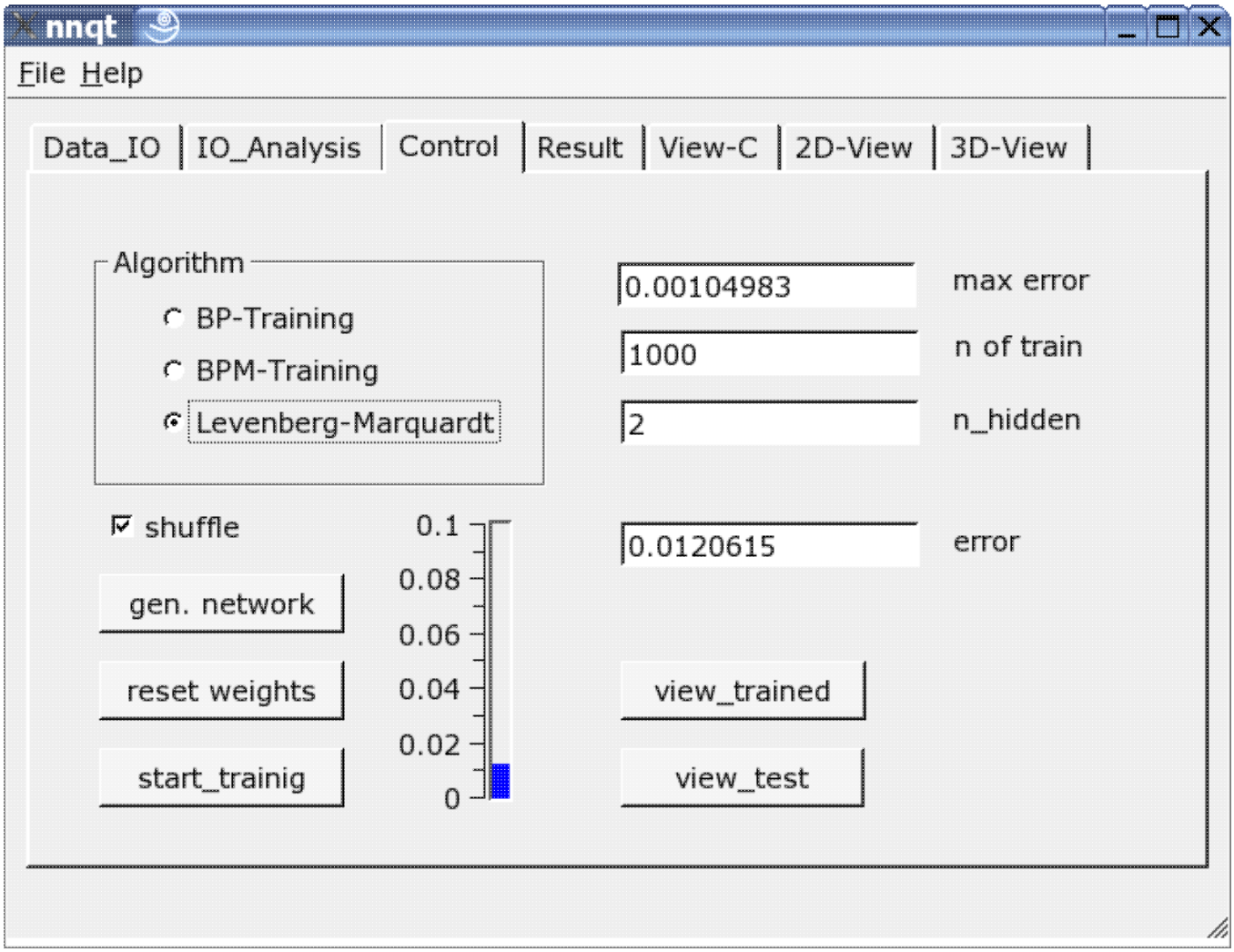


Aquí el se puede ajustar el *lector* para adaptare al formato de los datos de entrada. Pueden usarse varios separadores,se pueden ocultar algunas lineas de cabecera o escoger libremente el objetivo del conjunto de datos . Nota: el formato de los datos debe ser bien conocido, puesto que *nnqt* depende de la entrada que le indique el usuario.

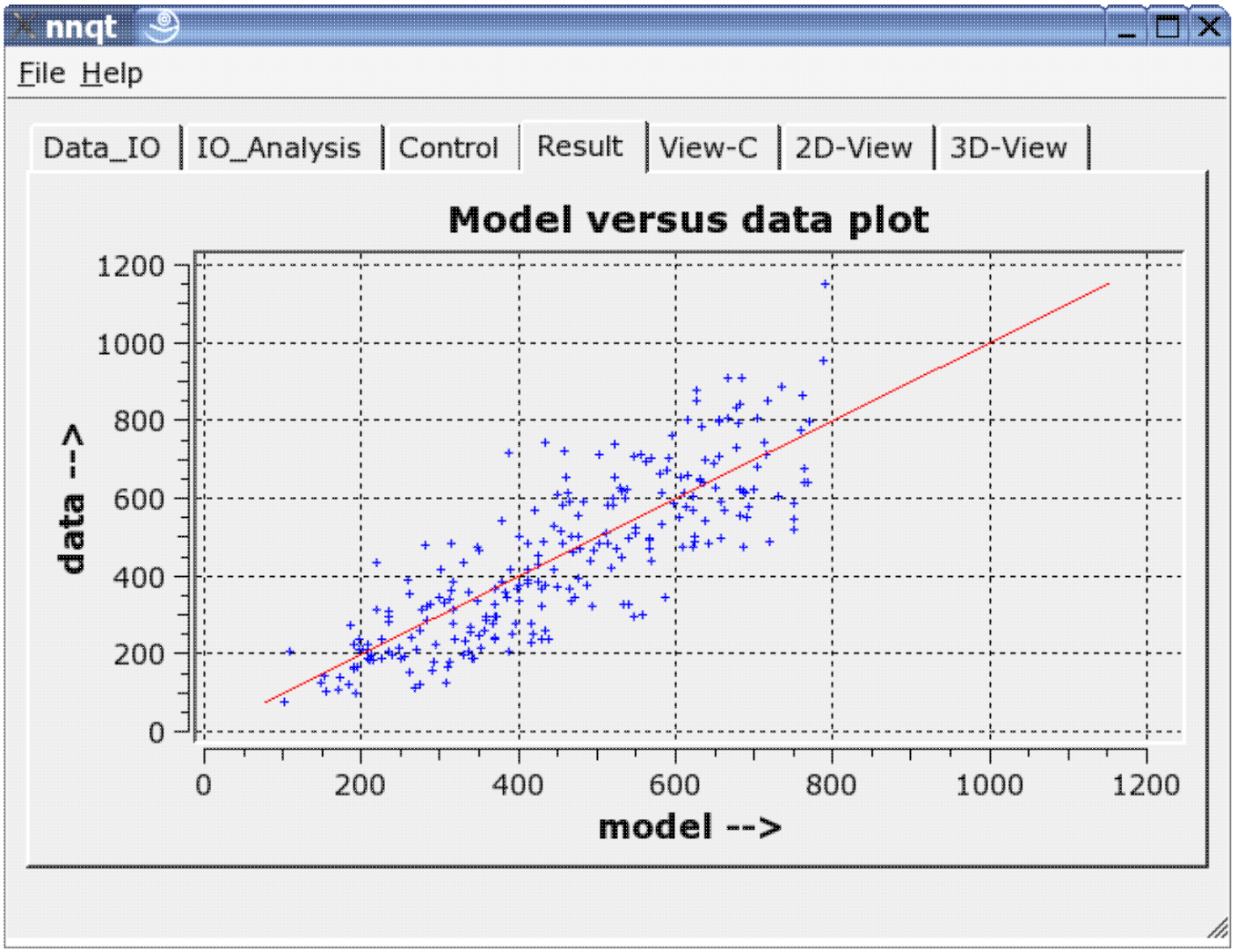


	min	max	mean	var	corr	usage
1	2	40	9.8505	31.1703	0.674244	1
2	12	91.3	64.5027	196.406	-0.578825	1
3	0.402	2.47	0.90603	0.058937	0.564807	1
4	0.029	0.294	0.0885024	.000942935	0.489779	1
5	78.2	1327.64	448.621	44371.1	1	
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

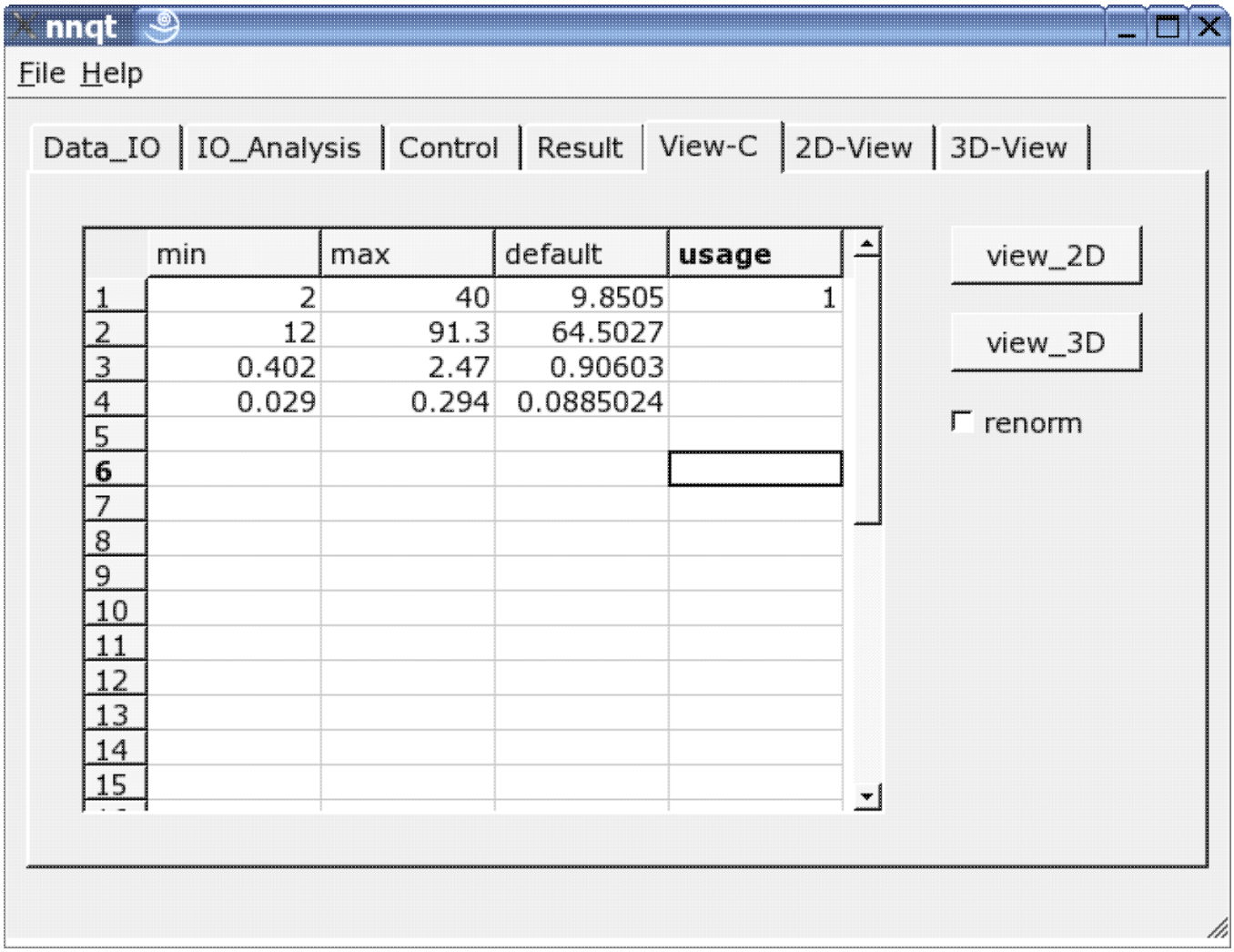
Después de una correcta introducción de los datos, nos situamos en la página de análisis de datos. Encontramos alguna información sobre los datos y se selecciona los datos , de entre todas las columnas,que serviran para el aprendizaje . Un '*1*' en la última columna marca la entrada como un valor de enseñanza. (Se utilizarán más de 29 valores)



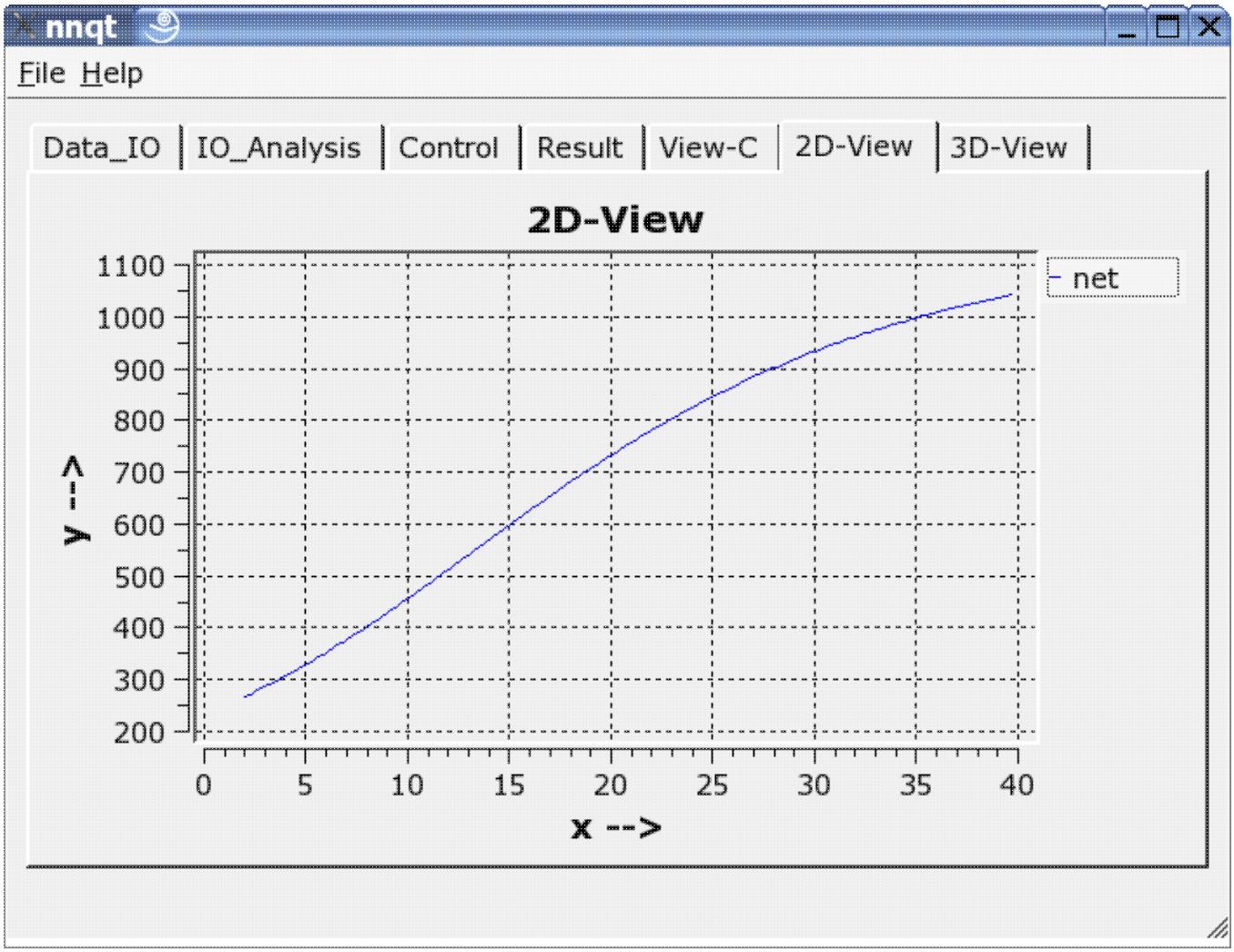
La más importante es la página de control. Aquí se definen el número de elementos ocultos, el número de pasos de aprendizaje y el algoritmo de adiestramiento. El adiestramiento puede visualizarse en una escala vertical, ya sea como barra o valor. El adiestramiento se repite hasta queel resultado es una función directa del parámetro inicial que fue estocásticamente escogido. Seleccionando la opción "*shuffle*" genera una selección aleatoria – en lugar de una selección secuencial – de los datos de adiestramiento, lo cual a veces es una ventaja. Si hubieramos tenido suerte con un error cuadratico medio lo suficientemente bajo, podemos obtener el primer gráfico seleccionando el botón "view_trained" :

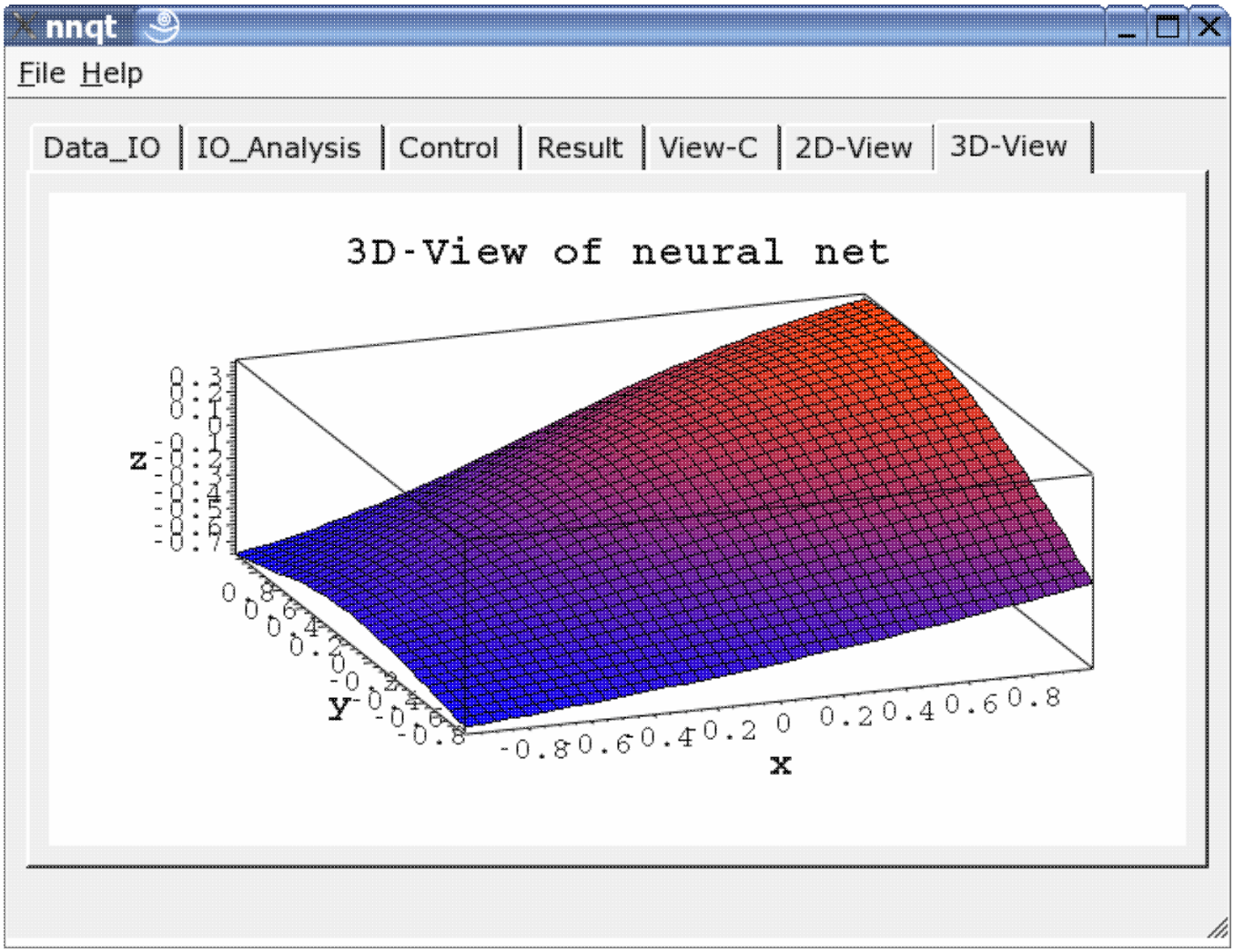


Se muestra la comparación entre los datos de adiestramiento con los datos generados por la red neuronal. Idealmente los datos deben esar en la diagonal. ¡Pero el ideal no se puede conseguir! No obstante, los resultados parecen bastante decentes. (Los datos de control – son los datos que no se pueden enseñar en el proceso de adiestramiento– están en rojo). El siguiente paso permite el análisis de la función de progresión. Los valores por defecto han de ser números significativos. Con esto, hemos de ser cuidadosos, puesto que el trabajo original de la red sólo cuadra con los datos de adiestramiento.



Se puede escoger la representación en dos o tres dimensiones.


nnqt
File Help
Data_IO
IO_Analysis
Control
Result
View-C
2D-View
3D-View
2D-View
net
1100
1000
900
800
700
600
500
400
300
200
y -->
0
5
10
15
20
25
30
35
40
x -->



Usando la *nnqt*

nnqt es software "open source", fue publicado bajo GPL. Cualquiera puede usarlo libremente y mejorarlo. Esto último es lo mejor. Sólo hay que instalar las librerias *qwt* y *qt*. nnqt.tgz es sencillo de desempaquetar.(*tar−zxvf nnqt.tgz*). Se creeará un nievo directorio llamado nnqt. Continuando con *cd nnqt*, un *qmake* . Si todo se interpreta correctamentese activara una variable shell ejecutando :
export NN_HOME=/pfad_zu_nnqt
Si *nnqt* se abre en otro terminal, nnqt deverá detectar los daros y los modelos. Espero que te diviertas con todo esto. Se incluye un conjunto de datos con dos entradas para las pruebas. ¿Alguien reconoce la función que ha aprendido? (se trata de x^2−y^2 en el rango de [−2..2].)

¿Qué podemos crear con todo esto? – Estoy ansioso por ver vuestras ideas.

Gracias a la comunidad

Se ha demostrado , que Linux posee un excelente entorno de desarrollo para resolver problemas científicos. He podido hacer el desarrollo gracias a un excelente software, sin el que no hubiera sido posible crear una

herramienta en un corto periodo de 6 semanas. Siempre es agradable usar softwre libre. Por ello, muchas gracias a todos los desarrolladores que con su trabajos hacen posible todas las cosas maravillosas que podemos hacer bajo Linux.

Referencias

James A. Freeman:
"Simulating Neural Networks with Mathematica", Addison−Wesley 1994

Download

El sotware nnqt y sus actualizaciones:descarga de nnqt

Contactar con el equipo de LinuFocus © Ralf Wieland "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Información sobre la traducción: de −−> −− : Ralf Wieland <rwielandQzalf.de> de −−> en: Jürgen Pohl <sept.sapinsQverizon.net> en −−> es: Alberto Pardo <apardoyo/at/yahoo.es>

2005−01−31, generated by lfparser_pdf version 2.51