Búsqueda de Parámetros Sistemática en Redes Neuronales

Autor: Christian Sengir Busquiel Sanz

Tutor del proyecto: Rita Korvodányi

Cotutor del proyecto: Julio Villena Román

1.- Objetivos

El objetivo principal de este proyecto consiste en implementar una aplicación que pueda simular una red neuronal utilizando un rango de valores para una serie de parámetros dados de forma automática. Es decir, ejecutar la red neuronal con cada combinación de valores de un rango de cada parámetro de forma que se obtengan los resultados de la red para cada una de esas combinaciones. Una vez el proyecto haya sido completado, los usuarios finales podrán probar sistemáticamente sus redes neuronales y representar los resultados de una forma que tenga fácil interpretación. Este trabajo será dividido en dos partes principales: Búsqueda de parámetros y representación de resultados.

1.1.- Búsqueda de parámetros

Los usuarios podrán elegir cualquier número de parámetros y, para cada uno de ellos, elegir un rango de valores para ejecutar la red neuronal en vez de elegir sólo un valor cada vez que se quiere lanzar la red. Para ello, los usuarios podrán elegir un valor mínimo, un valor máximo y el tamaño de salto entre cada dos ejecuciones de la red.
Entonces, la red se ejecutará una vez por cada combinación de valores y los resultados obtenidos se almacenarán juntos estructurados de forma que se pueda identificar fácilmente los valores de los parámetros que producen cada resultado.

1.2.- Representación de datos

Los datos obtenidos de ejecutar todas las combinaciones deben representarse gráficamente. La ejecución de la red neuronal produce una gran cantidad de medidas tales como el número de pasos que ha dado la red hasta obtener un error cero, el error cuadrático medio y muchas medidas más. Una vez este proyecto se complete, el usuario será capaz de elegir una de las medidas realizadas durante la ejecución de la red neuronal y mostrarla gráficamente. Esta representación gráfica se realizará mediante una cuadrícula coloreada que muestre los valores de salida de las medidas. En cada una de las casillas, se mostrará el valor de la medida tomada para una combinación de parámetros mediante una escala de color, de forma que dicho color represente el valor de la medida elegida con la combinación de los valores de los parámetros representados por la ubicación del color en esa casilla. De esta forma, es muy fácil y visual para el usuario encontrar la combinación o zona de combinación de valores de los parámetros que obtienen una mejor respuesta para la medida elegida.

2.- Neuronas y Redes Neuronales.

Una red neuronal artificial es la simulación del cerebro humano utilizando un ordenador, haciendo que esta red aprenda mediante ejemplos, como una persona normal haría.
Una red neuronal es un sistema de elementos simples interconectados llamados neuronas, que pueden calcular una información de salida utilizando el conocimiento aprendido utilizando la información recopilada previamente.
Las redes neuronales se utilizan en entornos en los que se necesita flexibilidad y generalización frente a nuevos datos de entrada. Las Máquinas de Von Neumann se basan en información exacta y devuelven resultados dependientes del programa diseñado para utilizar estas máquinas; es decir, no pueden adaptarse a diferentes tipos de entradas, pues siempre requieren una información completa de las mismas. Su configuración de proceso es serie, con un solo hilo; y separan debido a su configuración el procesado de información y la memoria.
Las redes neuronales se basan en el procesado en paralelo para obtener un mejor resultado.
Cada neurona trabaja en paralelo con las otras para calcular el resultado, y cada una de ellas tiene una pequeña cantidad de memoria. De esta forma, todo el trabajo está distribuido en estas unidades: Procesado y conocimiento.

Este tipo de trabajo es debido a la estructura de la red neuronal: Muchos elementos unitarios interconectados entre sí que reaccionan a un estímulo mediante una función interna para proporcionar la salida deseada.
Las ventajas que proporcionan las redes neuronales son:

- Rapidez (Procesado en paralelo)
- Alta capacidad para generalizar
- Alta potencia expresiva

Sus desventajas son:

- Dificultad para elegir la red adecuada para el problema planteado.
- El entrenamiento de la red tiene un alto coste
- Gran dificultad para calcular la dimensión
- Difíciles de interpretar, de entender el trabajo interno de la red

2.1.- Introducción a las neuronas

Una red neuronal está compuesta por muchas unidades adaptativas que están interconectadas y que envían información escalar a las otras unidades. Esas unidades adaptativas se denominan neuronas. Una neurona es la unidad más pequeña que procesa información, y es la base del conocimiento humano.
Cada neurona obtiene información de diferentes fuentes o entradas, condensa esa información en un valor escalar que representa la relación entre la información obtenida y la especialización en detección de la neurona, y envía dicho valor como salida. Las neuronas no sólo se especializan en una tarea, sino que contribuyen en una pequeña cantidad de tareas, aunque dichas tareas pueden cambiar dinámicamente. Por eso, una neurona tiene dos funciones en el sistema: memoria y procesado.

2.2.- Modelo de Neurona

Una neurona biológica es muy compleja de simular. Debido a ello, se utiliza un modelo simplificado en su lugar. Este modelo se parece a un detector, como un detector de humo, por ejemplo.
Cuando la neurona recibe suficiente señal de entrada, se activa, se dispara, o proporciona una salida relacionada a la señal de entrada entregada a la neurona.
La función de activación es la función que suma todas las entradas y calcula la señal de salida de la neurona que será enviada a las siguientes unidades. Dicha señal de salida se conoce como valor de activación. Cada señal de entrada tiene un peso relacionado a ese valor de activación; de manera que cuando más peso tenga una señal de entrada a la neurona, más importante será para la activación. Las neuronas pueden detectar patrones de actividad debido a estos pesos, no a señales de entrada aisladas.
La neurona artificial utiliza las señales de entrada, utilizando pesos y una función de transferencia, para decidir la señal de salida o la activación que enviará a las neuronas artificiales a las que está conectada, es decir, a la siguiente capa de neuronas. El proceso de aprendizaje se basa en los cambios que se realizan en los pesos para obtener un mejor resultado en la salida global de la red.
Un tipo interesante de neurona es el perceptrón simple. El perceptrón simple es un tipo de neurona que cambia su peso proporcionalmente a la diferencia entre la salida real y la salida deseada siguiendo la siguiente ecuación.

2.3.- Tipos de redes neuronales

Las redes neuronales están compuestas por capas de neuronas. Hay tres tipos de capas: la capa de entrada, la capa de salida y la capa oculta. Una red puede tener cualquier número de capas de

cada tipo. Cada capa conecta sus neuronas con las neuronas de otra capa. Estas conexiones crean diferentes tipos de redes:
- Redes neuronales de alimentación progresiva: En estas redes, las neuronas siempre se conectan con neuronas de la siguiente capa sin crear bucles (No se conectan con neuronas de la misma o de anteriores capas). Se utilizan en el reconocimiento de patrones.
- Redes neuronales con retroalimentación: Las neuronas de estas redes se pueden conectar a neuronas en la misma capa o a neuronas de capas anteriores. Estas redes son muy potentes debido al alto número de conexiones y de bucles que contienen. Su estado cambia hasta que consiguen un equilibrio. Una vez se llega a este punto, no cambian hasta que las entradas son diferentes y necesitan encontrar otro punto de equilibrio.
- Perceptrón multicapa (MLP): Las neuronas de estas redes son perceptrones. Se utilizan para el reconocimiento de patrones y para aprendizaje supervisado. Aunque una capa de perceptrones no es capaz de realizar un reconocimiento de patrones básico, como fue demostrado por Minsky y Papert (1969), muchas capas de dichos perceptrones pueden realizar esta tarea de una manera muy eficiente. Los MLPs se utilizan para aproximaciones globales, intentando simular el comportamiento de una neurona biológica.
- Redes con función base radial (RBFN): Estas redes se utilizan para aproximaciones locales, como aproximaciones de funciones.

2.4.- Funciones de activación:

Hay dos tipos básicos de funciones de activación para redes neuronales: La función escalón (También llamadas funciones de decisión dura), y la función sigmoide (Denominada también función de decisión blanda). Dichas funciones aparecen representadas en las figuras 2.3 y 2.4 de la memoria en inglés.

La función de decisión dura se elige cuando la red neuronal se utiliza para tomar una decisión (por ejemplo, verdadero o falso, encender o apagar una alarma), mientras que la decisión blanda (sigmoide) se utiliza cuando la salida esperada está entre un rango de valores (Por ejemplo, la cantidad de humedad en el aire para una red de predicción meteorológica).
Las redes del tipo RBFN tienen diferentes tipos de activación, asemejándose a campanas gaussianas.
Los perceptrones multicapa (MLP) se utilizan para aproximaciones globales, intentando simular el comportamiento de las neuronas biológicas, mientras que las redes RBFT se utilizan para aproximaciones locales.
Como se puede ver, la salida de cada unidad se encuentra entre los valores 0 y 1, pero se puede escalar o cambiar el valor central dependiendo del rango deseado para los datos de salida.

3.1.- Búsqueda de parámetros en redes neuronales

Los sistemas, las ecuaciones y las redes neuronales se basan en información que se combina para obtener un resultado sobre algo desconocido o para demostrar que un resultado dado es correcto.
Esos procesos se componen de muchas partes pequeñas que se combinan para formar una herramienta compleja, el proceso en sí. Normalmente, esas partes no encajan correctamente y necesitan ajustarse para que todo el sistema funcione correctamente. Dichas partes son los parámetros de los sistemas, de las ecuaciones y de las redes neuronales.
Cuanto mayor es un sistema, más parámetros necesita. Cuantos más parámetros hay, se debe reajustar el sistema un mayor número de veces por el usuario para obtener mejores resultados.
En este caso, se puede utilizar una búsqueda de parámetros, que consiste en manipular los parámetros automáticamente de manera que el usuario no necesita supervisar todo el proceso intentando averiguar cuál es la mejor combinación de dichos parámetros.

3.2.- Representación de parámetros

El problema de la combinación de parámetros reside en el número de parámetros que se utilizan en un sistema. Es demasiado complicado para el cerebro humano manejar todos los parámetros a la vez, más aún si hay una gran cantidad de valores diferentes para cada parámetro que necesitan tenerse en cuenta. Representarlos de una forma apropiada puede ayudar a realizar las elecciones correctas entre los valores de los parámetros que se necesitan para el sistema del usuario.
Cada parámetro tiene un rango de valores entre los que el sistema funciona correctamente. Por ejemplo, el rango de valores de un cuenta-kilómetros de un automóvil debe ir entre 0 y 200 Km/h, y no entre 0 y 2000 Km/h, porque no es posible que un coche vaya a una velocidad tan grande. Se necesita un rango apropiado para cada parámetro.
También es importante el tamaño de cada paso entre los valores de los parámetros que se van a probar en el sistema. Utilizando el mismo ejemplo, los pasos del cuenta-kilómetros deberían ser más grandes que 5 Km/h y menores que 25 Km/h para representarlos apropiadamente (Los valores que se utilizan son valores discretos, pues los valores que se utilizan como parámetros y porque los ordenadores no pueden trabajar con valores continuos).
Sorel Bosan y Thomas R.Harris (1996) propusieron una representación para una matriz de datos de más de tres dimensiones en “A Visualization-Based Análisis Method for Multiparameter Models of Vapillary Tissue-Exchange”. Su proceso se puede dividir en varios pasos. Cada paso mejora la representación del paso anterior añadiendo una nueva dimensión de forma que el usuario puede ver la relación entre cada uno de los parámetros.
Como ejemplo, se puede tomar el mismo que aparece en la memoria en inglés:
El primer paso, es tener una representación en 3D de los datos, como aparece en la figura 3.1. En esa figura, aparece una función de dos parámetros y sus valores en una escala de colores.
Dicha función puede representarse en dos dimensiones utilizando sólo colores, tal y como aparece en la figura 3.2.
Si se añade un tercer parámetro, no se puede representar de esta forma, porque involucraría una cuarta dimensión y no podemos representarla gráficamente.
Bosan y Harris (1996) proponen repetir estas imágenes en 2D para cada uno de los valores de la nueva dimensión que quiere añadirse y representarlas una al lado de la otra en otra escala con los valores de la nueva dimensión añadida, como se puede ver en la figura 3.3, en forma de columnas.
Al añadir una quinta dimensión, debe representarse las columnas anteriores con los valores del nuevo parámetro en filas consecutivas como aparece en la figura 3.4.

En resumen; para los dos primeros parámetros de la función, se ha utilizado un patrón de colores para representar la función en una gráfica. El siguiente parámetro mostrará una gráfica para cada uno de los valores de dicho parámetro en una fila, usando una columna para cada valor del nuevo parámetro; de esta forma, la gráfica resultante mostrará una fila de gráficas de dos parámetros que varían dependiendo del tercer parámetro. El siguiente parámetro que se añada, será representado por una fila de las anteriores por cada valor que tome dicho nuevo parámetro creando una columna de filas de gráficas. Si se quiere añadir más parámetros, se añadirán columnas o filas de las gráficas anteriores creando un grafo de grafos que parece una matriz de grafos, representando los valores de cada parámetro en las escalas de dicha representación.
Si sólo se representa un valor escalar usando colores en vez de un número o una gráfica como se usó en el ejemplo anterior, el comportamiento del sistema utilizando una combinación de parámetros se puede mostrar como la figura 3.5.
Con esta representación, el usuario puede encontrar la mejor combinación de parámetros que el sistema necesita para cumplir su objetivo. Tiene una sencilla visualización y el usuario podrá disponer de toda la información que necesita en un solo grafo.
Este modelo proporciona una representación del espacio de valores de los parámetros para la función de salida. Es útil para el usuario porque se puede manejar de una forma sencilla y proporciona un mapa amplio del comportamiento de los parámetros utilizados en el sistema. A veces, el usuario sólo quiere averiguar la región de la combinación de parámetros que ajusta al

sistema para el objetivo para el que fue designado, en vez de optimizar directamente el valor de salida.
Esta representación es útil también para encontrar relaciones entre parámetros, como la sensibilidad o identificabilidad explicadas por Bosan y Harris. Éstas son herramientas para determinar rápida y fácilmente la influencia en la función de salida y la independencia entre los parámetros, respectivamente.
Beck y Arnold (1989) demostraron que los parámetros pueden ser estimados fácilmente y pueden ser diferenciados entre sí si no son linealmente dependientes en el rango de los valores de los parámetros que se están estudiando. Esta aproximación puede hacerse comparando sólo dos parámetros cada vez. Si se comparan más de dos parámetros, el coste computacional se dispararía y no se obtendrían resultados visibles. También debe tener en cuenta que la influencia de parámetros en una función también se modifica por el valor de otros parámetros; por ejemplo, el parámetro P1 puede tener mucha influencia en la función si el parámetro P2 tiene un valor bajo y el parámetro P3 tiene un valor alto (Relativamente para el rango de valores usado en la búsqueda de parámetros); y puede tener poca influencia si P2 tiene un valor alto y P3 tiene un valor bajo.

4.- Método

Este capítulo describe el programa utilizado para el desarrollo del proyecto y enlaza los objetivos generales que fueron explicados en el primer capítulo con dicho proyecto. Este capítulo también explica el proceso de implementación del proyecto: decisiones tomadas, alternativas posibles y resultados obtenidos.

4.1.- Estado inicial

Para poder desarrollar un motor de búsqueda de parámetros sistemático, se necesita un programa que gestione redes neuronales. Emergent es un simulador de redes neuronales que permite la creación y la gestión de modelos complejos de cerebro, es decir, de redes neuronales. Emergent es un programa de código abierto y es utilizado por muchos investigadores de redes neuronales, con lo cual, es una muy buena herramienta para utilizar en el desarrollo de este proyecto: el hecho de que sea un programa de código abierto nos permite acceder fácilmente al código fuente y modificarlo, pero también significa que los propietarios del programa están dispuestos a aceptar nuevas ideas y nuevas contribuciones de otras personas para mejorar Emergent. El hecho de que sea utilizado por muchos investigadores de redes neuronales nos alentó a realizar un buen trabajo en este proyecto ya que una gran cantidad de gente podría beneficiarse de esta nueva funcionalidad.
La última versión que estaba disponible cuando se empezó el proyecto era la versión 5.0.1.

La pantalla del programa Emergent se divide en tres marcos principales como se puede ver en la figura 4.1.
El marco de la izquierda permite al usuario acceder a todos los datos relacionados con el proyecto en el que se esté trabajando. Las secciones más importantes de este marco son la red neuronal, los datos de entrada y los datos de salida, y los programas modificables por el usuario que ejecutan y entrenan la red.
El marco central muestra el objeto seleccionado en el marco izquierdo. En este marco es donde los usuarios pueden leer y editar los parámetros. En la figura 4.1 los datos que aparecen con el fondo verde están relacionados con el objeto que especifica las conexiones entre las capas de la red (Llamado LeabraConSpec_0 en el proyecto de ejemplo)
El marco de la derecha muestra una representación gráfica de la red. En este marco, los usuarios también pueden crear tablas o gráficos en tres dimensiones que representan los resultados obtenidos del entrenamiento y la ejecución de una red neuronal.
La figura 4.1 también muestra que el objeto de especificación de conexiones entre capas tiene más de veinte parámetros que el usuario puede modificar. En toda la red, el número de

parámetros que Emergent permite modificar varía dependiendo del número de capas que tiene dicha red y la complejidad de la misma, pero incluso la red más pequeña tiene cientos de parámetros. Esto nos da una idea de lo complicado que es intentar optimizar una red, incluso si es una red muy pequeña y muy sencilla. Incluso encontrar, elegir y modificar los parámetros deseados es una tarea tediosa, pues los usuarios tienen que invertir una gran cantidad de tiempo a través de la estructura de datos de la red. Emergent ofrece una herramienta para evitar dicha tarea manual: se llama SelectEdit, y su función consiste en permitir a los usuarios agrupar parámetros que quieren modificar, y mostrarlos en una única vista. SelectEdit puede verse en la figura 4.2.

En el panel de SelectEdit denominado ControlPanel de la figura 4.2, hay tres parámetros. Dichos parámetros están ubicados en diferentes objetos en la red, pero es muy fácil agruparlos aquí. La información visible es la sección a la que pertenece cada parámetro, su nombre y su valor. Dicho valor puede ser modificado por el usuario y, cuando se cambia el valor en el SelecEdit, el cambio es aplicado directamente a la red.
Utilizar un panel SelectEdit proporciona una forma de modificar los parámetros deseados sin tener que buscarlos en la estructura de datos cada vez que se quiere cambiar su valor. Aun así, esta funcionalidad no es suficiente como para realizar una búsqueda de parámetros exhaustiva.
Es lógico que un usuario que haya creado o vaya a trabajar con una red neuronal tenga que, en algún momento de su investigación, probar una serie de combinaciones de valores de algunos parámetros para averiguar con qué valores obtiene mejores resultados. La única forma de hacerlo es cambiar los valores de los parámetros manualmente y ejecutar los programas de entrenamiento y de prueba de la red una vez por combinación de valores. Después, debería salvar los resultados manualmente y comparar todos los resultados obtenidos. Este proceso es excesivamente costoso en tiempo, y muy poco práctico.

4.2.- Ampliación de Emergent

En esta sección, se explican todos los cambios añadidos a Emergent. En esta sección, se mostrarán las soluciones propuestas para el proyecto. También existen algunas alternativas a la solución implementada, pero se explicarán las razones para descartarlas o darlas por válidas. Se ha añadido una serie de diagramas en UML estándar para ver la evolución de la solución. Se ha elegido UML porque es un lenguaje representativo sencillo y usado en muchos ámbitos del modelado de software.
El modelo elegido para el desarrollo de este software ha sido el modelo en espiral, puesto que consiste en ir iterando procesos para desarrollar el software. Con este modelo, se va añadiendo funcionalidades lentamente a la vez que se realizan las pruebas de dichas funcionalidades, como se puede ver en la figura 4.3.
De esta forma, se podrá mostrar el proceso de diseño de cada parte y su implementación a la vez. Es decir, una vez explicado un proceso, se mostrarán los resultados obtenidos al mismo tiempo. De esta forma, el lector entenderá el estado del proyecto en cada momento en el que se tome cada decisión para el desarrollo del mismo. Así es como se ha desarrollado el proyecto, de acuerdo con el modelo de diseño que se ha decidido utilizar.
La codificación en Emergent sigue un estándar en el que las funciones deben empezar con mayúscula y las variables deben empezar con minúscula. Los diagramas utilizados también siguen ese estándar.

4.3.- Programa interno a Emergent 5.0.1

Emergent es capaz de utilizar programas creados por el usuario de forma interna, de forma que es posible crear programas propios para que Emergent los utilice. La forma de crear y modificar esos programas es arrastrando bloques de código desde un menú al programa que quiere crearse o modificarse. Esta forma de programar es diferente y puede parecer difícil al principio, pero se

tarda poco en aprender a manejarlo pues es bastante intuitivo. El lenguaje de programación utilizado es una adaptación de C++ para el entorno del programa.
Utilizar estos programas internos parece lo más conveniente para lograr el objetivo del proyecto, pues también puede exportarse a otros ordenadores, es muy fácilmente modificable y no depende de la versión de Emergent, de forma que nuestra primera adaptación de la solución se enfocó por este camino.

La primera versión del proyecto consiste en un grupo de funciones dentro de un programa de Emergent que pueden realizar una búsqueda de parámetros para un grupo predefinido de parámetros. Este grupo de parámetros contiene los parámetros que más se utilizan en Emergent. Debido a que estos parámetros están fijados, no es posible realizar las mismas operaciones de búsqueda de parámetros si se desea realizarlas mediante otros parámetros; de forma que, la única forma de que el usuario cambie los parámetros es modificar gran parte del código de este programa y enlazarlo de nuevo con la red neuronal. Otro problema que apareció en esta solución es que, si el programa interno se mueve a otro proyecto diferente, el usuario tiene que comprobar y cambiar todos y cada uno de los enlaces que existen entre el programa y la red, puesto que los nombres de la red y de las capas son diferentes y únicos en cada proyecto de Emergent.

4.4.- El nuevo panel de SelectEdit

Después de contactar y hacer ciertas consultas al equipo de desarrollo de Emergent de la universidad de Colorado (Estados Unidos), se llegó a la conclusión de que la mejor forma de abordar el problema era modificar el código fuente de Emergent. Emergent está desarrollado en C++, uno de los lenguajes de programación más potentes, de forma que las limitaciones que antes nos impedían avanzar habían desaparecido. Trabajar con un lenguaje de alto nivel también significa que se necesita un mayor análisis y un mejor diseño para la solución. El equipo de desarrollo de Emergent facilitó una nueva opción para los paneles SelectEdit denominada Search. La nueva propiedad, Search, apareció en la versión 5.0.2, y puede verse cómo aparecía en el panel SelectEdit en la figura 4.4.
El propósito de la opción Search es activar o desactivar la búsqueda sistemática de parámetros en el panel SelectEdit. Los usuarios pueden, a partir de ahora, activar la opción Search y elegir un valor máximo, un valor mínimo y el tamaño de salto en el rango de valores. El valor “next step” es un valor de sólo lectura y los usuarios no pueden modificarlo; y muestra el valor que tendrá un parámetro la próxima vez que varíe. Este valor es útil para las ocasiones en las que se detenga el programa antes de que la búsqueda sistemática acabe, pues muestra en qué punto se ha detenido el programa.
En la figura 4.4, puede verse que hay dos parámetros con la opción Search activada y un tercero que tiene dicha función desactivada. La búsqueda de parámetros ignorará el parámetro cuya opción Search esté desactivada y sólo utilizará aquellas activadas.

4.5.- Diseño e implementación de la búsqueda sistemática de parámetros

En este proyecto, se está modificando un programa existente y, debido a ello, se puede ver una simplificación del estado del programa en la figura 4.5. Es una simplificación porque hay mucha información relacionada con Emergent que no influye al ámbito del proyecto y por eso se ha decidido omitir. A partir de aquí, se puede modificar el código fuente de Emergent. Para ello, tenemos dos opciones: Separar todo el código nuevo del resto de Emergent enlazando sólo lo que sea completamente necesario; o fusionar el código existente con el código nuevo. Las figuras 4.6 y 4.7 muestran ambas posibilidades. En ambas figuras, se han eliminado partes del diagrama por simplicidad, aunque no se eliminen ni modifiquen respecto al diseño original.

La primera solución separa la búsqueda de parámetros del resto de Emergent. Básicamente, se utiliza el patrón de diseño “Singleton”. En la figura 4.6, aparecen los cambios realizados en rojo,

destacando el objeto “Singleton” ParameterSearch. Este objeto está conectado a Program y no SelectEdit. De esta forma, el código de SelectEdit y el código de Program son independientes. El objeto añadido se encarga de calcular qué parámetros van a variar y cuántos saltos van a realizar para comunicar a los objetos Program qué valores deben utilizar cada vez que ejecuten la red neuronal. También proporciona una salida detallando la información necesaria por consola durante la ejecución de la red neuronal, y se encarga de crear una tabla de datos de salida para escribir los datos cada vez que se ejecute la red. Esta tabla es un nuevo tipo de objeto denominado ParameterSearchOutputData, que es un tipo de OutputData; y será donde se almacenen los resultados. Finalmente, el objeto Program tiene un nuevo método que se utiliza para preguntar al objeto “singleton” ParameterSearch qué parámetros deben ser cambiados para la siguiente ejecución y cuáles deben ser utilizados.

La segunda solución no utiliza ningún objeto nuevo, pues es añadir código dentro del diseño anterior. Las diferencias con el diseño original están marcadas en rojo en la figura 4.7. Los nuevos atributos en Parameter y el objeto ParameterSearch siguen existiendo, aunque la mayor diferencia se encuentra en los objetos Program y SelectEdit, pues están conectados internamente, sin un objeto Singleton, porque Program necesita obtener la información de SelectEdit sin ningún objeto adicional. Program tiene más funciones en su interior, pues estas funciones eran realizadas por el objeto “singleton” y debe responsabilizarse de ellas (Calcular el número de saltos de cada parámetro y utilizarlos, proporcionar la información por la consola durante la ejecución y crear la nueva tabla para almacenar los datos en ella.

Ambas soluciones se compararon para elegir cuál era la mejor. La primera, permite que el código fuente esté bien estructurado y sea leído fácilmente por aquellos que quieran modificar el código en el futuro. El problema reside en que Emergent tiene muchos ficheros en su código fuente, y añadir más ficheros a los ya existentes no permite que el código sea más fácil de interpretar; de modo que no podemos aprovecharnos de la mayor ventaja de esta solución. También produce una pérdida de eficiencia, puesto que dividir el código para añadir partes ralentiza el propio Emergent; y uno de los objetivos del proyecto era conseguir minimizar el impacto en Emergent de la búsqueda de parámetros sistemática. La segunda solución no mantiene el código tan bien estructurado, pero el impacto en el tiempo de ejecución de Emergent es casi inexistente. Debido a esto, se decidió que la segunda solución es la mejor para esta situación debido al uso de este programa.
El diagrama de la figura 4.7 muestra cómo es la solución elegida. Aun así, algunas restricciones no han podido ser representadas con el UML estándar y deben tenerse en cuenta:
La primera afecta a los pocos parámetros que existen en una red neuronal de Emergent que no tienen un valor numérico, cuyo valor es alfanumérico. Estos parámetros no pueden aprovecharse del desarrollo de este proyecto, pues no pueden incluirse en ningún rango numérico.
Otra restricción que se debe tener en cuenta es que los usuarios pueden crear más de un panel SelectEdit. Se dan dos opciones diferentes: Realizar una ejecución de un programa para todos los parámetros de todos los paneles SelectEdit o realizar dicha ejecución utilizando sólo un panel SelectEdit. Esta decisión no es tan importante, puesto que la mayoría de los programas tendrán sólo un panel SelectEdit, pero se debe considerar esta posibilidad. Esta segunda opción permite a los usuarios crear grupos de parámetros que quieren probar en diferentes ejecuciones, aunque los usuarios deberían ser capaces de elegir un panel antes de ejecutar el programa. Esta solución requiere modificar el interfaz gráfico de Emergent, pero esto choca con los deseos del equipo de desarrollo de Emergent. Afortunadamente, Emergent elige automáticamente el último panel que el usuario haya modificado: es lógico que el usuario modifique el panel que quiere utilizar. De forma que la segunda opción tiene poco impacto en la solución y es lo más intuitivo para el usuario, con lo cual se eligió esa opción.
La última restricción tiene que ver con el número máximo de parámetros que puede utilizar Emergent. Emergent puede trabajar con matrices de, como mucho, siete dimensiones. Éste es el límite de parámetros sobre los que se puede realizar una búsqueda de parámetros utilizando este proyecto. Para que se pueda utilizar un mayor número de parámetros, se debe modificar el núcleo de Emergent, y esta decisión sólo concierne al equipo de desarrollo de Emergent. Si

dicho equipo ha decidido que el número máximo de parámetros sea siete, es porque un mayor número de parámetros tendría un impacto negativo en el tiempo de ejecución de Emergent.

4.6.- Diseño e implementación de representación de datos

Los datos que se representan en la figura 4.12 son difíciles de interpretar, así que es mejor aislar cada pieza de información para obtener unas conclusiones. Los usuarios deberían poder elegir una columna de esa tabla de salida para crear una matriz que muestre los valores que aparecen en dicha columna relacionándolos con la combinación de los valores de los parámetros que se utilizaron para obtener esa salida. Para ello, Emergent creará una tabla de datos a partir de la columna mediante el objeto DataTable, creando otra tabla del tipo OutputData. En la figura 4.13, aparece el diagrama UML que representa la funcionalidad que se va a añadir a Emergent en esta ocasión.
DataTable obtendrá un nuevo método llamado VisualizaParameters para crear la nueva tabla que será llamada VisualParameterSearchData.
Esta tabla es diferente a las otras, pues sólo contiene una celda. Dicha celda, contiene una matriz con los datos de salida ordenados para poder representarlos gráficamente. El nombre de las columnas y de las filas de dicha matriz viene dado por el nombre de los parámetros que se han usado en la búsqueda y el paso del intervalo en el que se ha obtenido ese valor de salida tal y como puede verse en la figura 4.15. Dado que cada dato es un escalar, puede representarse de forma gráfica a color como se muestra en la figura 4.16.

4.7.- Programa interno para Emergent 5.0.2

Una vez apareció la versión 5.0.2 de Emergent, se discutió con los desarrolladores de Emergent la posibilidad de mover todas estas modificaciones hechas en el código fuente de Emergent a un programa interno de Emergent tal y como se comentó anteriormente. De esta forma, cualquier usuario podría modificar el programa interno para adaptarlo a la funcionalidad que desee sin necesidad de cambiar el código de Emergent. La mejora de Emergent en el código fuente funciona correctamente en modo recursivo, pero Emergent tiene problemas a la hora de gestionar la recursividad en programas internos. Debido a esto, se facilitó también la posibilidad de que la ejecución fuera iterativa añadiendo una opción en el menú principal del programa interno. La versión iterativa produce el mismo resultado que la recursiva, aunque no sea tan elegante como ésta última. De todas formas, el código recursivo del programa interno se ha mantenido puesto que los desarrolladores de Emergent están tratando de buscar la solución al problema. Si Emergent se modifica para poder trabajar con más de siete dimensiones y se arregla el problema de la recursividad, sólo es necesario cambiar un valor al número de dimensiones para actualizar el programa interno; mientras que la versión iterativa debería modificarse con una nueva serie de líneas de código.