Universidad
Carlos III de Madrid

Monitorización del aprendizaje en redes de neuronas

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Autor: Roberto Pintos López 100304070
Tutor: José María Valls Ferrán

Índice general

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

1. Introducción

El proyecto consiste en la realización de una aplicación para visualizar gráficamente el proceso de aprendizaje de una red de neuronas de base radial con la finalidad de que aportar una perspectiva distinta que pueda enriquecer el proceso de aprendizaje de los alumnos del mencionado algoritmo.

1.1 Motivación

Las razones que me han movido a realizar este proyecto y no otro son las siguientes:

1. La aplicación sería de utilidad en caso de ser desarrollada porque actualmente no existe un software que muestre de forma gráfica este proceso de aprendizaje.

2. Comprender este tipo de algoritmos sin ningún tipo de soporte gráfico es complicado. Se puede adquirir una comprensión plena del algoritmo, pero dada la cantidad de factores a considerar es más fácil asimilar el algoritmo si una aplicación gráfica los plasma estos factores en una pantalla, ahorrando esfuerzo al aprendiz.

3. Sería una oportunidad para poder desarrollar bibliotecas de clases de redes de neuronas, bibliotecas de clases de extensiones a interfaces gráficas e integrarlas, lo cual mejorará mi formación como desarrollador de software.

1.2 Objetivos

Los objetivos que se persiguen con la realización de la aplicación son los siguientes:

1.2.1 Objetivos principales:

1. La aplicación debe ser capaz de visualizar el proceso de aprendizaje de las redes de neuronas de base radial. Debe ser capaz de mostrar los distintos elementos de la red durante el proceso de aprendizaje.

2. La aplicación debe ser capaz de servir como una herramienta complementaria para que los alumnos puedan comprender los distintos algoritmos de aprendizaje de las redes de base radial.

1.2.2 Objetivos secundarios

3. La aplicación debe poder ser implantada en cualquier maquina con sistema operativo Windows compatible con .NET Framework 4.0 o superior.

1.3 Estructura de la memoria

El documento está constituido por las siguientes secciones:

1. Introducción: en esta sección se describe de forma muy breve el proyecto a realizar así como las motivaciones que llevaron a realizarlo y los objetivos que se pretenden conseguir con la realización del proyecto.

2. Planteamiento del problema: en esta sección se realizará un análisis del estado del arte y se definirán los requisitos del proyecto. Finalmente se comentará brevemente el marco regulador que afecta al desarrollo del proyecto.

3. Diseño de la solución técnica: esta sección se realiza una descripción del diseño final adoptado, tomando justificadamente decisiones que permitan discernir la mejor alternativa de diseño de entre las planteadas.

4. Resultados y evaluación

5. Planificación y presupuesto del trabajo: en esta sección se detallará la planificación del proyecto haciendo uso de un diagrama de Gantt y se harán los cálculos relativos al coste del proyecto.

6. Conclusiones: tras la realización del proyecto se redactarán las impresiones que se han tenido y los objetivos que se ven cumplidos. También se comentarán posibles mejoras a realizar en el producto.

7. Anexo: contendrá un manual de usuario y un resumen del proyecto en inglés.

8. Referencias: Esta sección contendrá referencias a las fuentes que no sean de elaboración propia utilizadas para la realización de la memoria

2. Planteamiento del problema

2.1 Análisis del estado del arte

2.1.1 Introducción de las redes de neuronas de base radial.

Una red de neuronas de base radial es un tipo de red de neuronas artificial que es capaz de realizar un proceso de aprendizaje supervisado, más concretamente un proceso de regresión multidimensional Fueron introducidas en 1988 por D. S. Broomhead y David Lowe para ajustar funciones no lineales como alternativa al perceptrón multicapa.

Este tipo de redes dispone de una estructura rígida de tres capas: la capa de entrada, una única capa oculta y la capa de salida.

Las neuronas de la capa oculta disponen de dos elementos que permiten que cada neurona de la capa oculta se asocie con una función de ajuste del modelo definido por los patrones presentados a la red:

1. Centros: Un centro determina la posición de la neurona en el espacio de entrada. Cuanto más cerca esté una neurona de un patrón, más elevada será la activación de la mencionada neurona al ser introducido el patrón en la red

2. Desviaciones: la desviación determina la distribución de la curva en el espacio de entrada. Cuanto menor sea la desviación de la neurona más abrupta será la curva asociada a esta. El origen de este término radica en la aplicación tradicional de la función de densidad de la distribución gaussiana como función de activación de estas neuronas.

La función de activación de la capa oculta es una función de base radial. Las funciones de activación se aplicarán no sobre la entrada de la misma sino sobre una transformación que dará un valor directamente proporcional a la distancia euclidea entre el patrón proyectado en el espacio de entrada y el centro de la neurona e inversamente proporcional a la desviación de la neurona:

$$f(x) = \frac{\|X(n) - C_i\|}{d_i}$$

Ecuación 1: función de transformación de la entrada vectorial de las neuronas ocultas.

Entre las funciones de activación más utilizadas se encuentran las siguientes:

$$f(x) = e^{\frac{-x^2}{2}}$$

Ecuación 2: Función de activación gausiana

$$f(x) = \frac{1}{1 + x^2}$$

Ecuación 3: Función de activación inversa cuadrática

$$f(x) = \frac{1}{\sqrt{1 + x^2}}$$

Ecuación 4: Función de activación inversa multicuadrática

Aunque estas redes no son comúnmente utilizadas en aplicaciones que impliquen un alto volumen de patrones de entrenamiento, se le reconoce como una red con una alta eficiencia en la fase de entrenamiento.

2.1.2 Descripción de los algoritmos de aprendizaje:

El algoritmo de aprendizaje de las redes de base radial tiene dos variantes. En las dos variantes se tratará de ajustar los centros, desviaciones, pesos y umbrales. Sin embargo la forma de ajustar los centros y desviaciones difiere según los algoritmos a utilizar.

a) Algoritmo de aprendizaje totalmente supervisado:

Este algoritmo calculará mediante el método del descenso del gradiente cual es el ajuste de pesos que minimiza el error cuadrático medio cometido por la red con respecto al conjunto de patrones presentados a esta. Las expresiones que determinan los ajustes de los elementos de las neuronas en la introducción de un patrón en la red son las siguientes:

En su planteamiento original para la aplicación de la técnica del descenso del gradiente:

Sea $\omega_{i,k}$ el peso de la conexión de la neurona oculta i a la neurona de salida k.

Sea u_k el umbral hacia la neurona de salida k.

Sea $c_{i,j}$ la coordenada j del centro de la neurona oculta i.

Sea d_i la desviación de la neurona oculta i

Sea e el error cuadrático medio cometido por la red

$$\omega_{i,k}(n) = \omega_{i,k}(n-1) - \propto_1 \frac{\partial e(n)}{\omega_{i,k}}$$

Ecuación 5: ecuación de calibración de pesos (inicial)

$$u_k(n) = u_k(n-1) - \propto_1 \frac{\partial e(n)}{u_k}$$

Ecuación 6: ecuación de calibración de umbrales (inicial)

$$c_{i,j}(n) = c_{i,j}(n-1) - \propto_2 \frac{\partial e(n)}{c_{i,j}}$$

Ecuación 7: ecuación de calibración de centros (inicial)

$$d_i(n) = d_i(n-1) - \propto_3 \frac{\partial e(n)}{d_i}$$

Ecuación 8: ecuación de calibración de desviaciones (inicial)

Tras su manipulación para ser procesadas:

Sea $\omega_{i,k}$ el peso de la conexión de la neurona oculta i a la neurona de salida k.

Sea u_k el umbral hacia la neurona de salida k.

Sea $c_{i,j}$ la coordenada j del centro de la neurona oculta i.

Sea d_i la desviación de la neurona oculta i

Sea ϕ_i la salida de la neurona oculta i

Sea S_k la salida de la neurona de salida k

Sea X la entrada del patrón

Sea y_k el valor de salida k del patrón introducido en la red

Sea D_i la función de distancia aplicada en la neurona oculta i

$$\omega_{i,k}(n) = \omega_{i,k}(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)\phi_i(n)$$

Ecuación 9: ecuación de calibración de pesos (final)

$$u_k(n) = u_k(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)$$

Ecuación 10: ecuación de calibración de umbrales (final)

$$c_{i,j}(n) = c_{i,j}(n-1) + \propto_2 \left(\sum_{k=1}^{r}\left(\left(S_k(n) - y_k(n)\right)\omega_{i,k}\right)\right)\phi_i(n)\frac{\left(X_j - c_{i,j}\right)}{{d_i}^2}$$

Ecuación 11: ecuación de calibración de centros (final)

$$d_i(n) = d_i(n-1) + \propto_3 \left(\sum_{k=1}^{r}\left(\left(S_k(n) - y_k(n)\right)\omega_{i,k}\right)\right)\phi_i(n)\frac{D_i(X)}{{d_i}^2}$$

Ecuación 12: ecuación de calibración de desviaciones (final)

b) Algoritmo de aprendizaje híbrido.

Este algoritmo de aprendizaje consta de dos etapas:

En la primera los centros de las neuronas son situados siguiendo un algoritmo de agrupación sobre los patrones proyectados sobre el espacio de entrada. Los algoritmos más utilizados para esta labor son k-medias y mapas auto organizados de Kohonen.

Tras situar los centros se determinan las desviaciones de las neuronas. Las desviaciones deben tomar valores de tal forma que cada neurona oculta se encargue de la interpolación de la zona próxima a ella. Con este propósito se intentará que el solapamiento de las zonas de activación de dos neuronas sea lo más leve posible, para que la interpolación realizada sea suave.

Las desviaciones suelen ser calculadas mediante las siguientes funciones heurísticas (aunque estos los propósitos mencionados no se lograrán necesariamente):

1. Media uniforme de la distancia euclídea entre el centro de la neurona y sus n patrones vecinos más cercanos:

$$d_i = \frac{\sum_{j=1}^{n} \|C_i - C_j\|}{n}$$

Ecuación 13: media uniforme de distancias los n vecinos más cercanos

2. Media geométrica de la distancia euclídea entre el centro de la neurona y sus dos patrones vecinos más cercanos:

$$d_i = \sqrt{\|C_i - C_0\| \|C_i - C_1\|}$$

Ecuación 14: media geométrica de las distancias a los dos vecinos más cercanos

En la segunda etapa se aplica la técnica del descenso del gradiente para situar los pesos y umbrales procediendo de la misma forma que en la fase supervisada

$$\omega_{i,k}(n) = \omega_{i,k}(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)\phi_i(n)$$

Ecuación 15: ecuación de calibración de pesos (final) en el método híbrido

$$u_k(n) = u_k(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)$$

Ecuación 16: ecuación de calibración de umbrales (final) en el método híbrido

[1]**¡Error! No se encuentra el origen de la referencia.**

2.1.3 Software relacionado con redes de neuronas de base radial

El escaso número de aplicaciones y paquetes que se desarrollan tienen una finalidad práctica, centradas en la implementación de redes capaz de realizar procesos de aprendizaje de forma eficiente haciendo uso de las capacidades de multiprocesamiento de los computadores.

Apenas existen aplicaciones que ofrezcan siquiera una interfaz gráfica en comparación con la cantidad de paquetes desarrollados con la finalidad descrita anteriormente. Entre los paquetes desarrollados cabe destacar los siguientes:

- RSNNS de cran
- Encog Library
- DLib C++ Library

Entre el software encontrado que ofrezca interfaz gráfica, se ha encontrado muy poca información. La empresa Devitep ha desarrollado una aplicación con interfaz gráfica para el uso de redes de neuronas, sin embargo no llega al punto de mostrar el proceso de aprendizaje de forma detallada

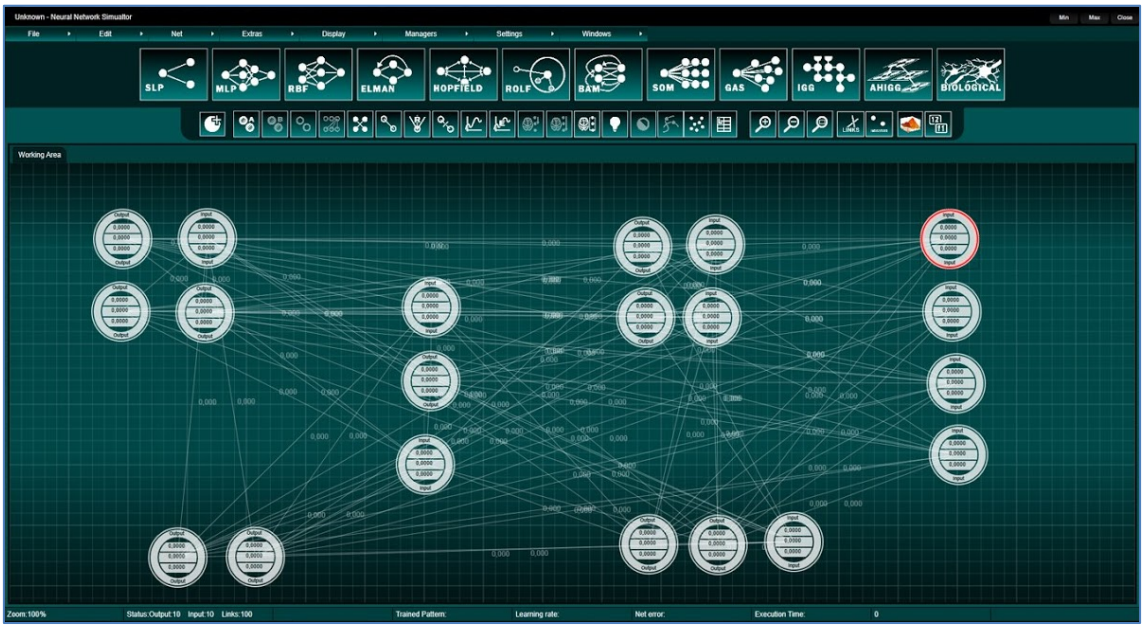


Ilustración 1: Simulador de redes de neuronas desarrollado por Devitep.

[3][4][5][6]

2.1.4 Conclusiones extraídas

Las conclusiones extraídas tras analizar el estado del arte son las siguientes:

1. Las redes de neuronas de base radial tienen múltiples aplicaciones actualmente, por lo que la se seguirá teniendo la necesidad de comprender su funcionamiento.
2. El algoritmo de aprendizaje no es trivial. Requiere de conocimientos de cálculo diferencial así como cierta capacidad de abstracción para comprender los algoritmos de aprendizaje, sus capacidades y sus limitaciones.
3. Existen suficientes paquetes que implementan redes de neuronas de base radial, pero no se dispone de ningún tipo de software que muestre de forma detallada el proceso de aprendizaje, por lo una herramienta que pueda dar esta funcionalidad podría ser bien recibida en círculos académicos donde más que la eficiencia no es tan importante como lograr la comprensión de funcionamiento de las redes.

Todas estas conclusiones refuerzan mi creencia inicial de la existencia de la necesidad de un software destinado a mostrar de forma gráfica el proceso de aprendizaje de las redes de neuronas de base radial.

2.2 Requisitos.

A continuación se especifica cuáles deben ser las funcionalidades de la aplicación a desarrollar, por medio de requisitos.

Cada requisito se define en una tabla con los siguientes campos:

- Identificador (ID). Cadena de caracteres que identifica al requisito de forma unívoca. Su estructura es R<tipo>-<número>, donde:
  - Tipo. Es el tipo del requisito. Pueden ser:
    - Funcional (FU). Indica qué debe hacer la aplicación.
    - No funcional (NF). Indica cómo debe hacer algo la aplicación.
    - Negativo (NE). Indica qué no se puede hacer bajo ciertas condiciones.
  - Número. Conjunto de cifras para numerar los requisitos que comienzan en 001.
- Necesidad. Sirve para indicar cómo de importante es el requisito para garantizar la calidad de la aplicación. Puede ser:
  - Esencial
  - Deseable
  - Opcional
- Prioridad. Útil para establecer un orden de implementación de los requisitos. Puede tomar tres valores:
  - Alta
  - Media
  - Baja
- Fuente. Indica quién es la persona que dio origen al requisito, si el tutor o el alumno.
- Verificabilidad. Señala la facilidad con que es posible comprobar que la aplicación cumple el requisito. Puede ser:
  - Alta
  - Media
  - Baja
- Descripción. Se trata de una explicación breve, en la medida de lo posible, del requisito.

2.2.1 Requisitos funcionales

2.2.1.1 Requisitos relacionados la entrada y salida de datos

ID	RFU-001		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como entrada un fichero seleccionado por el usuario.		

Tabla 1 Requisito funcional 1.

ID	RFU-002		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como parámetro de los algoritmos de aprendizaje las redes de neuronas el número de neuronas de entrada.		

Tabla 2: Requisito funcional 2.

ID	RFU-003		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como parámetro de los algoritmos de aprendizaje las redes de neuronas el número de neuronas de la capa oculta.		

Tabla 3: Requisito funcional 3.

ID	RFU-004		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como parámetro de los algoritmos de aprendizaje las distintas tasas de aprendizaje.		

Tabla 4: Requisito funcional 4.

ID	RFU-005		
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como parámetro de la interaz 3d el número de puntos a calcular para renderizar el modelo generado por la red en la fase de aprendizaje.		

Tabla 5: Requisito funcional 5.

ID	RFU-006		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como parámetro de entrada las dimensiones a utilizar como dimensiones de entrada y la dimensión a utilizar como dimensión de salida.		

Tabla 6: Requisito funcional 6.

ID	RFU-007		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☐Alta	☒ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación recibirá como parámetro de entrada un punto de proyección del espacio.		

Tabla 7: Requisito funcional 7.

ID	RFU-008		
Necesidad	☐Esencial	☐ Deseable	☒ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación podrá cargar redes de neuronas creadas con ella misma		

Tabla 8: Requisito funcional 8

ID	RFU-009		
Necesidad	☐Esencial	☐ Deseable	☒ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación podrá guardar redes de neuronas creadas.		

Tabla 9: Requisito funcional 9

Requisitos relacionados con los algoritmos a implementar

ID	RFU-010		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dispondrá de la implementación del algoritmo de aprendizaje supervisado de una red de neuronas de base radial.		

Tabla 10: Requisito funcional 10.

ID	RFU-011		
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dispondrá de la implementación del algoritmo de aprendizaje híbrido de una red de neuronas de base radial.		

Tabla 11: Requisito funcional 11.

ID	RFU-012		
Necesidad	☐Esencial	☐ Deseable	☒ Opcional
Prioridad	☐Alta	☐ Media	☒ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dispondrá de la implementación del algoritmo de aprendizaje del perceptrón multicapa.		

Tabla 12: Requisito funcional 12.

Requisitos relacionados con la interfaz

ID	RFU-013		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dispondrá de una interfaz gráfica		

Tabla 13: Requisito funcional 13.

ID	RFU-014		
Necesidad	☐ Esencial	☐ Deseable	☒Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación podrá trabajar con múltiples redes simultáneamente.		

Tabla 14: Requisito funcional 14.

ID	RFU-015		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dispondrá de un motor gráfico que permita visualizar una proyección cónica de una proyección sobre un espacio tridimensional del espacio de entrada-salida de las redes		

Tabla 15: Requisito funcional 15.

ID	RFU-016		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dispondrá de un medio de seleccionar por parte del usuario el tipo de algoritmo de aprendizaje y el tipo de red de neuronas a aplicar en el aprendizaje de un modelo.		

Tabla 16: Requisito funcional 16.

ID	RFU-017		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación mostrará durante el proceso de aprendizaje de una red una malla representando el modelo generado por esta.		

Tabla 17: Requisito funcional 17.

ID	RFU-018		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒Tutor	☐Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Desde la interfaz se podrá comenzar, pausar y continuar el proceso de aprendizaje de las redes.		

Tabla 18: Requisito funcional 18.

ID	RFU-019		
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Desde la interfaz será posible desplazarse por el mundo renderizado asociado a una red.		

Tabla 19: Requisito funcional 19.

ID	RFU-020		
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☐ Media	☒ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☐Alta	☒ Media	☐Baja
Descripción	Desde la interfaz será posible realizar una transformación de escalado del mundo renderizado		

Tabla 20: Requisito funcional 20.

ID	RFU-021		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dibujará las curvas gausianas generadas por cada neurona oculta en el proceso de aprendizaje de las redes de base radial de función de activación gausiana.		

Tabla 21: Requisito funcional 21.

ID	RFU-022		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Durante la fase de entrenamiento, el usuario podrá limitar el periodo de duración mínimo de un ciclo de entrenamiento.		

Tabla 22: Requisito funcional 22.

2.2.2 Requisitos no funcionales

2.2.2.1 Requisitos relacionados la entrada y salida de datos

ID	RNF-001		
Necesidad	☐ Esencial	☐ Deseable	☒Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación solo garantizará su correcto funcionamiento si los ficheros de entrada hacen dan valores normalizados a sus patrones.		

Tabla 23: Requisito no funcional 1.

ID	RNF-002		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación solo garantizará su correcto funcionamiento si los ficheros de entrada respetan el formato arff		

Tabla 24: Requisito no funcional 2.

ID	RNF-003		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El valor de las tasas de aprendizaje introducidas estará comprendió entre cero (excluido) y uno (incluido)		

Tabla 25: Requisito no funcional 3.

ID	RNF-004		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El valor de las tasas del número de neuronas de entrada debe ser un número natural.		

Tabla 26: Requisito no funcional 4.

ID	RNF-005		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El valor de las tasas del número de neuronas de la capa oculta debe ser un número natural.		

Tabla 27: Requisito no funcional 5.

Requisitos relacionados con los algoritmos a implementar

ID	RNF-006		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema utilizará el formato de coma flotante de doble precisión para almacenar el valor de los pesos de las redes de neuronas		

Tabla 28: Requisito no funcional 6.

ID	RNF-007		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema utilizará el formato de coma flotante de doble precisión para almacenar el valor de los umbrales de las redes de neuronas		

Tabla 29: Requisito no funcional 7.

ID	RNF-008		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	En la creación de una red de neuronas, la aplicación inicializará los pesos a valores aleatorios comprendidos entre 0 (excluido) y uno (incluido)		

Tabla 30: Requisito no funcional 8.

ID	RNF-009		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	En la creación de una red de neuronas, la aplicación inicializará los umbrales a valores aleatorios comprendidos entre 0 (incluido) y uno (incluido)		

Tabla 31: Requisito no funcional 9.

ID	RNF-010		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	En la creación de una red de neuronas de base radial, la aplicación inicializará los centros a valores aleatorios comprendidos entre 0 (incluido) y uno (incluido)		

Tabla 32: Requisito no funcional 10.

ID	RNF-011		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	En la creación de una red de neuronas de base radial, la aplicación inicializará las desviaciones a valores aleatorios comprendidos entre 0 (excluido) y uno (incluido)		

Tabla 33: Requisito no funcional 11.

Requisitos relacionados con la interfaz

ID	RNF-012		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La interfaz gráfica estará basada en formularios de Windows		

Tabla 34: Requisito no funcional 12.

ID	RNF-013		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La interfaz de la aplicación gestionará las distintas redes creadas mediante un conjunto de formularios encargándose cada formulario red determinada por la aplicación.		

Tabla 35: Requisito no funcional 13.

ID	RNF-014		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Las dimensiones de entrada se asociarán en el modelo tridimensional a las dimensiones X y Z respectivamente.		

Tabla 36: Requisito no funcional 14.

ID	RNF-015		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Las dimensión de salida se asociará en el modelo tridimensional a la dimensión Y de este.		

Tabla 37: Requisito no funcional 15.

ID	RFU-016		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El modelo generado hará un barrido en las dimensiones de entrada especificadas y en las dimensiones no especificadas como de entrada o salida asignará como valor el valor de el punto de proyección introducido.		

Tabla 38: Requisito no funcional 16.

ID	RNF-017		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación dibujará con un único color el modelo generado por la red y con otro color las distintas gausianas generadas por las distintas neuronas ocultas de la red		

Tabla 1: Requisito no funcional 17.

Requisitos relacionados con las herramientas de desarrollo a utilizar

ID	RNF-018		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación estará implementada haciendo uso del lenguaje de programación C#.		

Tabla 39: Requisito no funcional 18.

ID	RNF-019		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación estará implementada haciendo uso de .NET Framework 4.0		

Tabla 40: Requisito no funcional 19.

ID	RNF-020		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación será desarrollada haciendo uso del entorno Visual Studio 2010.		

Tabla 41: Requisito funcional 20.

ID	RNF-021		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☐Tutor	☒ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	La aplicación estará preparada para ser implantada en entornos con un sistema operativo Windows que soporte .NET Framework 4.0		

Tabla 42: Requisito funcional 21.

2.2.3 Requisitos negativos

ID	RNE-001		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El usuario no podrá modificar la topología de la red una vez creada la misma.		

Tabla 43: Requisito negativo 1.

ID	RNE-002		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El usuario no podrá modificar los pesos de una red		

Tabla 44: Requisito negativo 2.

ID	RNE-003		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El usuario no podrá modificar los umbrales de una red		

Tabla 45: Requisito negativo 3.

ID	RNE-004		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El usuario no podrá modificar los centros de una red de base radial		

Tabla 46: Requisito negativo 4.

ID	RNE-005		
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Fuente	☒ Tutor	☐ Alumno	
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El usuario no podrá modificar las desviacioens de una red de base radial		

Tabla 47: Requisito negativo 5.

2.3 Marco Regulador

El marco regulador legal que afecta a la aplicación, tanto a los usuarios que la utilicen como en el desarrollo de la misma es el que se describe a continuación

La aplicación desarrollada puede leer ficheros de datos, cuya obtención/generación y uso deben estar siempre regidos por la *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal*. Deben cumplir esta ley tanto los ficheros que se han usado para desarrollo y pruebas sobre la aplicación como los utilizados por usuarios.

Respecto al software utilizado para el desarrollo de la aplicación, hay que tener en cuenta que al adquirirlo y utilizarlo se acepta un acuerdo de licencia que se debe cumplir. Se han respetado las licencias de:

- Microsoft Windows 8.
- Microsoft Office 2007
- Microsoft Visual Studio 2010

Finalmente se debe respetar el *Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia*. En este proyecto, se ha adquirido conocimiento y se han expuesto datos de diferentes fuentes, las cuales serán referenciadas en el apartado de referencias de este documento.

3. Diseño de la solución técnica

3.1 Alternativas de diseño.

A la hora de realizar el diseño se plantearon las siguientes alternativas:

3.1.1 Componentes del sistema

Hay un gran abanico de posibilidades a la hora de diseñar cuales serán los componentes que compondrán el sistema a desarrollar. De primeras queda descartado que el sistema este formado por un único componente porque debido a la complejidad del proyecto iría contra los principios de reutilización de código y escalabilidad del sistema, aparte que dificultaría un futuro mantenimiento del mismo.

Una vez determinado que van a diseñarse múltiples componentes que van a integrar el sistema, es necesario diseñarlos procurando maximizar la cohesión de cada componente y minimizar el acoplamiento entre los componentes. Para ello se realizó una lista con los diversos grupos de funcionalidades que tendría la aplicación. Los grupos de funcionalidades son los siguientes:

1. Funcionalidades de interfaz
   1.1. Formularios
   1.2. Manejadores
   1.3. Elementos de sincronización de hilos
2. Extensión para cálculos matemáticos
3. Algoritmia
   3.1. Redes de neuronas
   3.2. Renderizado 3D
4. Entrada y salida de datos

Todos estos grupos pueden combinarse formando un conjunto bastante amplio de posibilidades. Procurando conseguir un producto mantenible y escalable por encima de otras características determiné que la mejor disposición de componentes sería la siguiente:

1. Un componente que implemente los algoritmos de redes de neuronas.
2. Un componente que implemente los algoritmos de renderizado 3D.
3. Un componente que implemente las funcionalidades de entrada y salida de datos
4. Un componente que implemente los formularios, sus manejadores y enriquezca estos con mecanismos de sincronización

Con respecto a la posibilidad de añadir un componente que diese una extensión a las funcionalidades matemáticas que aporta el lenguaje, se descartó puesto que se pensó que sería una mejor alternativa que cada componente implementase

internamente las funcionalidades específicas que necesitase de esta índole puesto que eran claramente asociables a cada componente de tal forma que no había dos componentes que necesitasen de un mismo paquete de extensión del lenguaje.

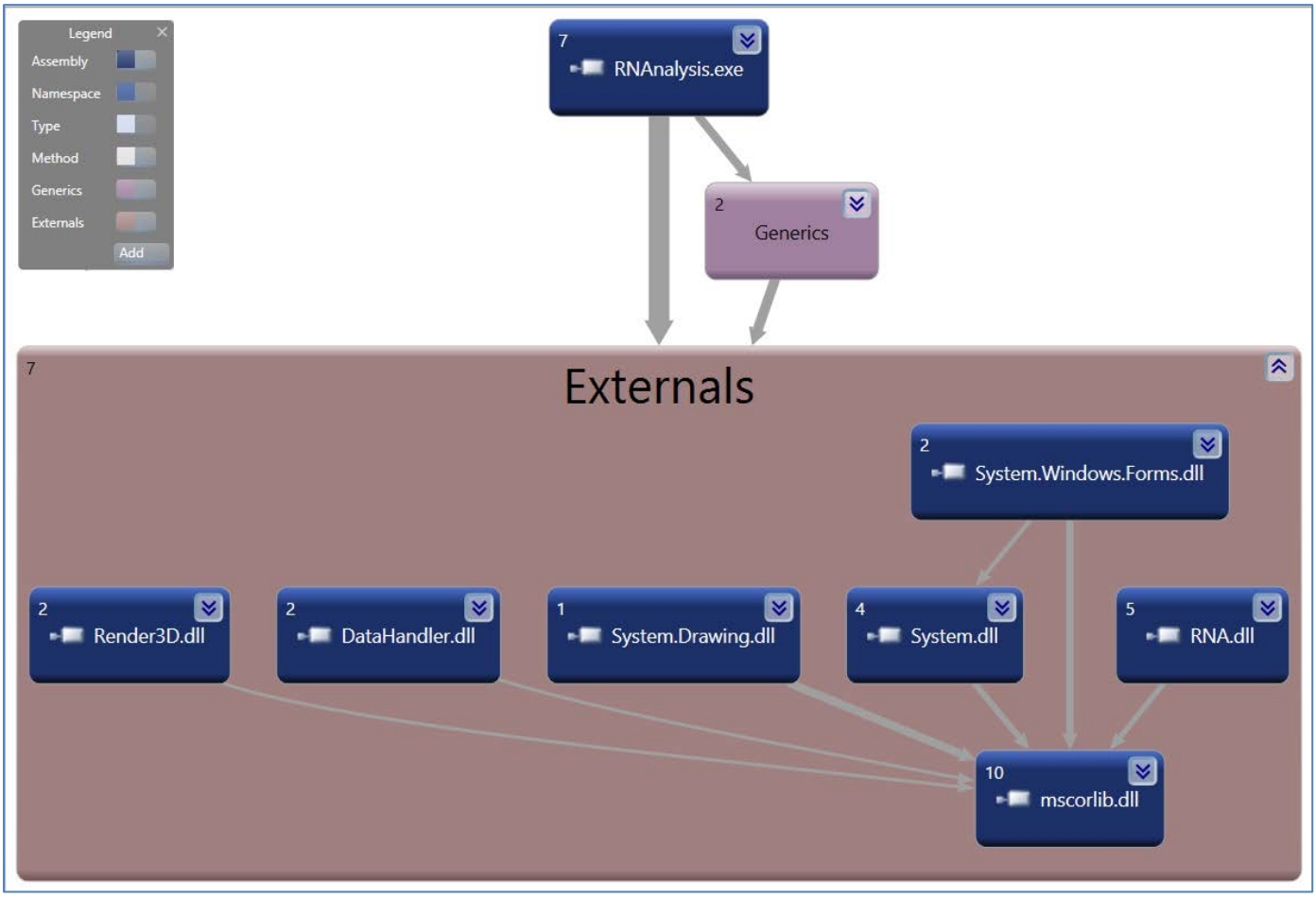


Ilustración 2: Modelo de componentes del sistema desde el punto de vista del componente gestor de la interfaz.

En la ilustración anterior se puede ver el modelo de componentes final de la aplicación.

RNAAnalisis: Componente principal del sistema. Se encarga de gestionar la interfaz del componente.

System.Windows.Forms: Componente de Microsoft que implementa la sección de .NET Framework relacionada con la gestión de formularios

Render3D: Componente que se encarga del proceso de renderizado de los modelos.

DataHandler: Componente encargado de aportar la funcionalidad relacionada con la entrada y salida de datos a excepción de la entrada directa sobre los formularios.

System.Drawing: Componente de Microsoft encargado de dar la funcionalidad relacionada con el procesamiento de imágenes.

System: Componente de Microsoft encargado de dar una implementación parcial de .NET Framework

RNA: Componente que almacena la implementación de las redes de neuronas de base radial.

mscorlib: biblioteca de clases de Microsoft que aporta una implementación parcial de .NET Framework en su capa más baja.

3.1.2 Formularios del componente principal

Con respecto a este componente se plantearon dos alternativas:

1. Un conjunto de formularios descentralizado.

2. Un conjunto de formularios contenido en un formulario principal.

Se optó por la segunda opción puesto que el hecho de anidar los formularios de las redes en un formulario padre da la posibilidad de introducir un layout que los muestre de forma organizada y más fácilmente manipulable por parte del usuario.

Además, debido a la necesidad de trabajar con varios hilos accediendo a zonas de memoria compartida existe la necesidad de trabajar con mecanismos de sincronización. Para ello han sido utilizados mútex, los cuales pueden dormir hilos para prevenir que varios procesos accedan simultáneamente a una sección crítica.

3.1.3 Grado de abstracción de la implementación de los algoritmos de las redes de neuronas:

Puesto que se intenta que el software sea escalable y mantenible se optó por tratar de maximizar el grado de abstracción en el diseño de las estructuras de datos asociados a las distintas redes para que utilizando los mecanismos de herencia del lenguaje orientado a objetos utilizado se pueda añadir de forma inmediata las funcionalidades comunes que puedan compartir las redes de neuronas, como puede ser, por ejemplo el proceso de transmisión de las señales o el proceso del cálculo del error cometido por la red con respecto a un conjunto de patrones.

3.1.4 Motor gráfico:

La disyuntiva entre utilizar un motor gráfico externo o implementar uno es menos evidente de lo aparente.

Por un lado los motores gráficos no son ni de lejos triviales de implementar por la complejidad algorítmica existente y existe la necesidad de comunicarse con la tarjeta gráfica de la máquina en caso de que se quiera minimizar el tiempo de respuesta del componente.

Sin embargo, una de las motivaciones de la realización del proyecto era el crecimiento como desarrollador software. Si a esto unimos el hecho de que el motor gráfico que se necesitaría no tendría que tener todas las coberturas que ofrecen los motores gráficos, esta opción parece bastante atractiva. Además el hecho de desarrollar el motor gráfico aporta a la aplicación control total del proceso de renderizado, por lo que se puede evitar que se realicen diversas optimizaciones que se realizan de forma casi imperceptible por los motores gráficos para mejorar el tiempo de respuesta de los mismos pero que pueden causar una alteración no deseada en el resultado final de la renderización.

Por los motivos anteriormente expuestos se llevará a cabo la implementación del motor gráfico.

3.2 Diseño de la aplicación

A continuación se muestra diagramas con los que se ilustrará el diseño de la aplicación

Debido a la complejidad del mismo no se recurrirá a un único diagrama. En lugar de recurrir a los tradicionales diagramas UML se recurrirá a diagramas de dependencias generados con Visual Studio.

Dada la complejidad del proyecto no se mostrará el diseño en profundidad del proyecto. Implicaría aumentar considerablemente la extensión del documento innecesariamente puesto que se puede alcanzar una comprensión del diseño de la aplicación desde un enfoque descendente profundizando exclusivamente en las secciones del proyecto más importantes. De todas formas si se desea consultar el conjunto completo de dependencias del proyecto se adjuntará un al diagrama de dependencias completo de este en formato dgml, el cual podrá ser abierto por cualquier versión de Visual Studio 2010 o posterior.

3.2.1 Componente RNAAnalisis

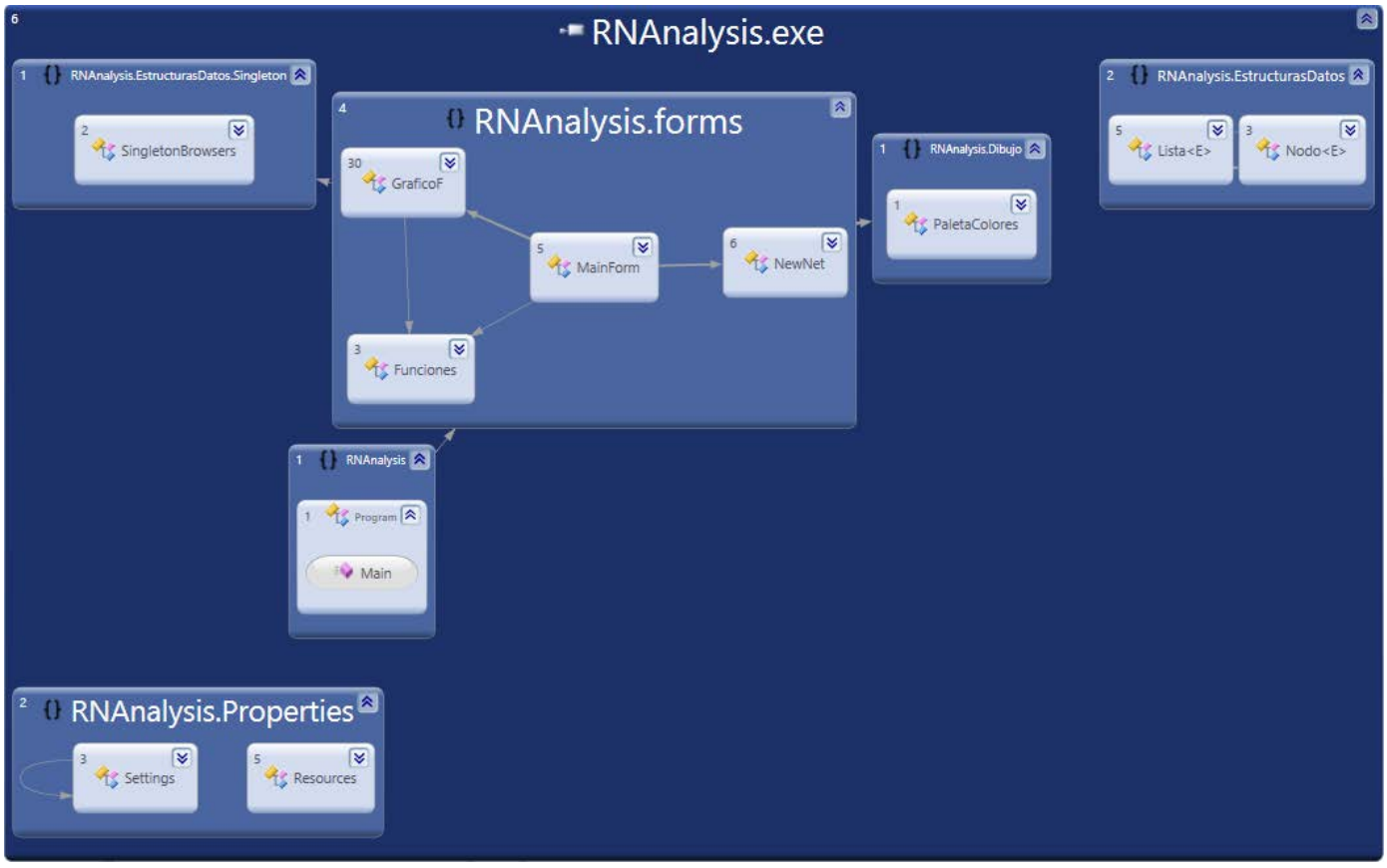


Ilustración 3: Gráfico de dependencias del componente RNAAnalisis

El componente principal consta de los siguientes espacios de nombres:

1. RNAAnalisis.Dibujo: contiene una clase para almacenar en una instancia un conjunto de colores.

2. RNAAnalisis.EstructurasDatos: contiene la implementación de una lista doblemente enlazada.

3. RNAAnalisis.EstructurasDatos.Singleton: contiene gestores de objetos Singleton útiles para los formularios.

4. RNAAnálisis.forms: contiene las implementaciones de los formularios y una pequeña clase funciones que es utilizado como un módulo con funciones auxiliares

5. RNAAnalisis.Properties: espacio de nombres añadido por defecto para la gestión de la configuración y los recursos de los formularios de Windows.

3.1.2.2 Componente RNA

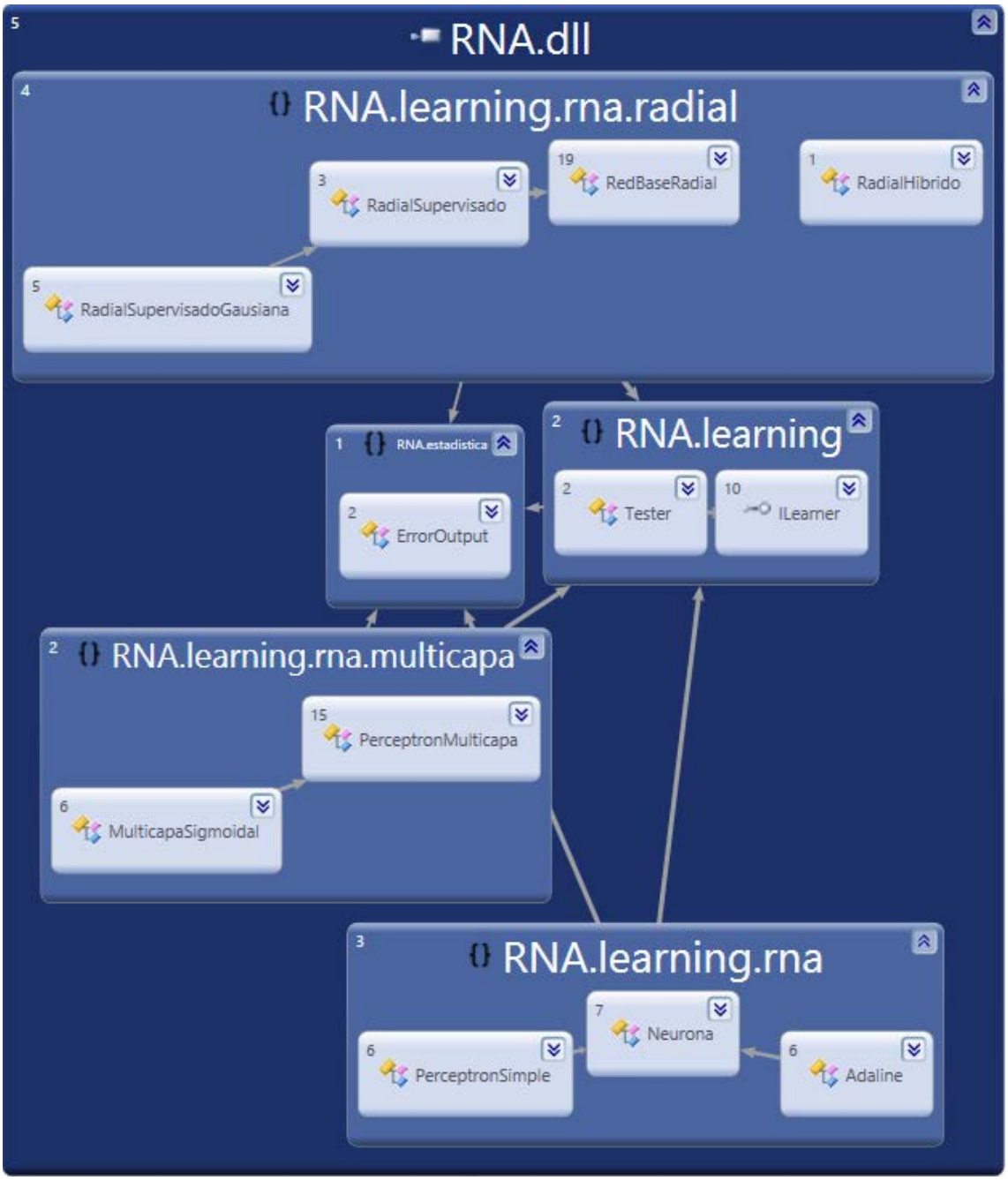


Ilustración 4: Gráfico de dependencias del componente RNA

El componente contiene los siguientes espacios de nombres:

1. RNA.estadistica: contiene una clase cuyas instancias servirán para almacenar y comunicar los errores cometidos por las redes.

2. RNA.learning: espacio de nombres de los algoritmos de aprendizaje. Contiene la interfaz que deberán implementar todas las redes de neuronas para tener funcionalidades comunes a todas las técnicas de aprendizaje.

3. RNA.learning.rna: espacio de nombres de las redes de neuronas. Contiene la implementación de una neurona genérica y de redes neuronales de una sola neurona.

4. RNA.learning.rna.multicapa: contiene la implementación del perceptrón multicapa genérico así como la de un perceptrón multicapa con una función de activación sigmoidal

5. RNA.learning.rna.radial: contiene la implementación de las redes de base radial.

3.2.3 Componente Render3D

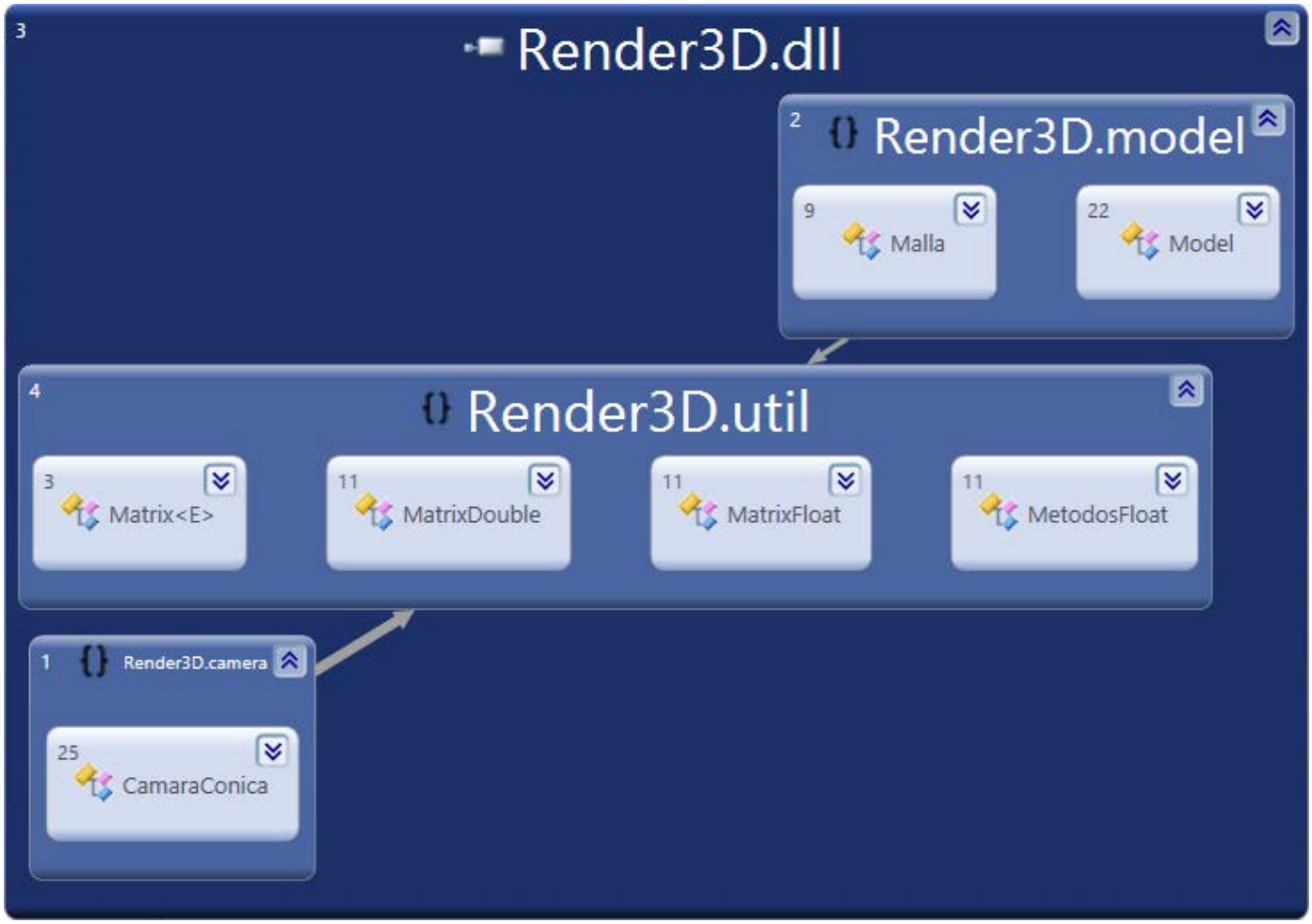


Ilustración 5: Grafico de dependencias del componente Render3D

El componente contiene los siguientes espacios de nombres:

1. Render3D.camera: contiene la implementación de la cámara cónica, la cual es capaz de proyectar un conjunto de segmentos determinador por puntos y conexiones entre puntos sobre un plano dado un punto de visión. Este punto de visión estará determinado por un vector de posición y un conjunto de vectores de orientación X, Y, Z que formarán necesariamente una base ortonormal del espacio tridimensional.

2. Render3D.model: contiene una clase que puede ser manejada por la cámara para renderizar objetos. Incluye a su vez métodos para generar objetos geométricos comunes como cubos y esferas.

3. Render3D.util: contiene clases con métodos de cálculo auxiliares.

3.2.4. Componente DataHandler

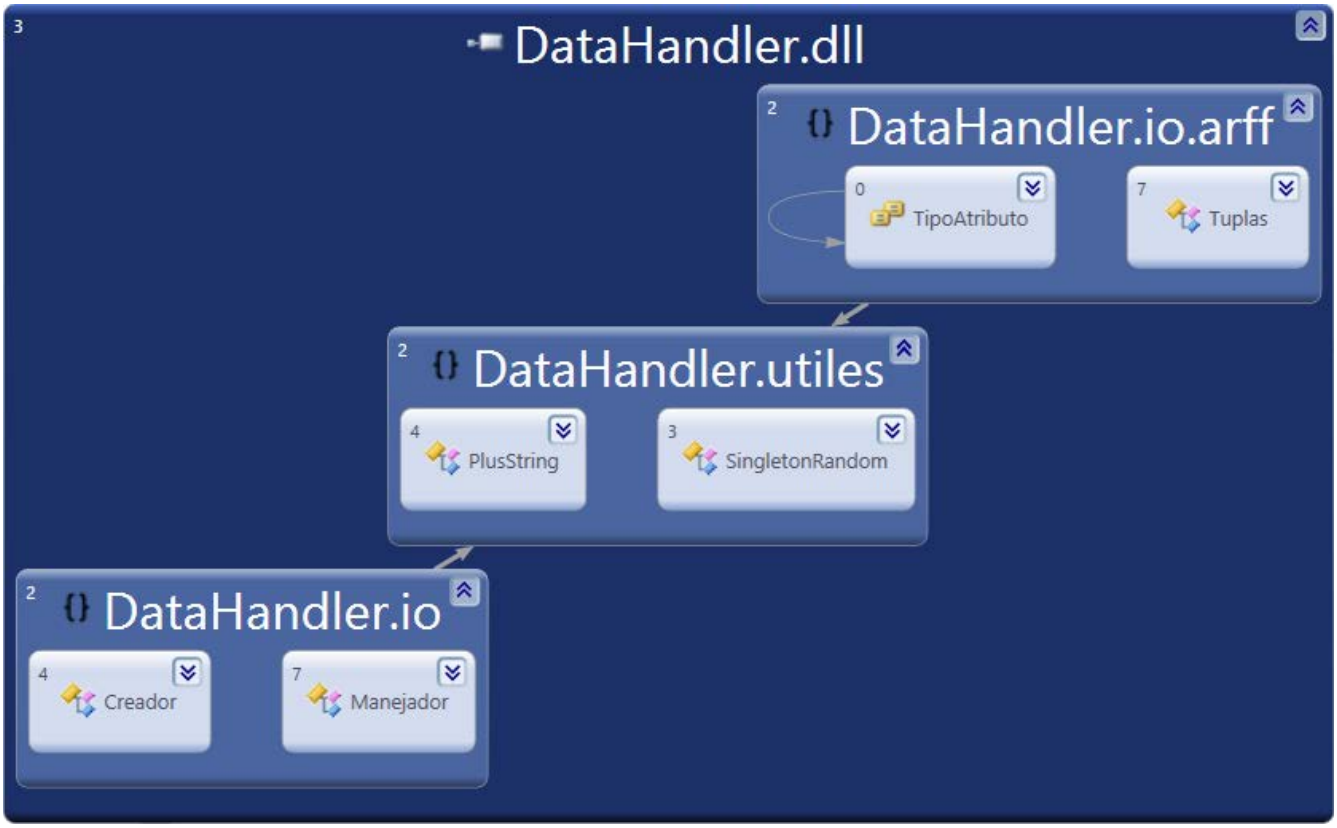


Ilustración 6: Gráfico de dependencias del componente DataHandler

El componente contiene los siguientes espacios de nombres:

1. DataHandler.io: contiene clases con métodos para generar distribuciones aleatorias de patrones así como manipular conjuntos de patrones.

2. DataHandler.io.arff: contiene una clase para leer patrones escritos en formato arff.

3. DataHandler.utiles: proporcionan funcionalidad auxiliar que puedan requerirse en los espacios de nombres anteriores.

4. Evaluación

A continuación se presentan las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación. Las pruebas se clasificarán en los siguientes grupos:

1. Pruebas de funcionalidad de formularios
2. Pruebas de los métodos de entrada
3. Pruebas del motor gráfico

A continuación se muestran las pruebas realizadas. Tan solo se mostrará una parte de las pruebas realizadas para no extender innecesariamente el documento, pero en la realidad por cada prueba aportada se han tenido en cuenta posibles valores límite y se han realizado tantas pruebas como se ha considerado para abarcar todas las posibilidades.

4.1 Pruebas de funcionalidad de formularios

Las pruebas deben garantizar que los formularios se muestran correctamente y que realizan las comprobaciones pertinentes para que los datos introducidos sean válidos. Las pruebas a realizar serán las siguientes:

4.1.1 Crear múltiples redes y eliminarlas. Con esto se comprobará que el formulario principal se muestra correctamente, que los formularios hijos son debidamente contenidos en el formulario principal y que el layout del formulario principal gestiona correctamente la visualización de los formularios hijos.

Para ello ejecutaremos la aplicación y crearemos cuatro redes. Para crear una red haremos click en Archivo ➔Nuevo ➔Red y rellenaremos el formulario que aparecerá a continuación:

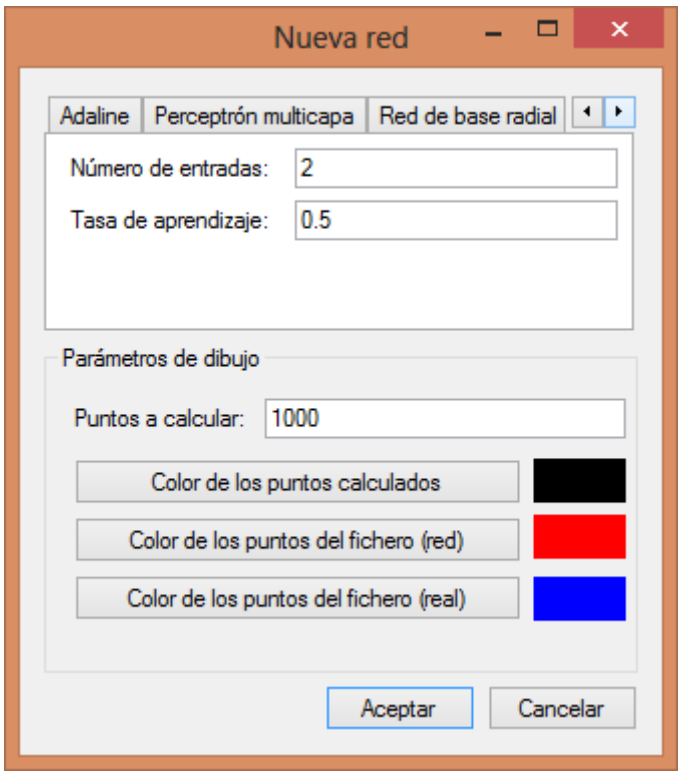


Ilustración 7: Formulario de creación de redes

Tras repetir el proceso cuatro veces se obtiene lo siguiente:

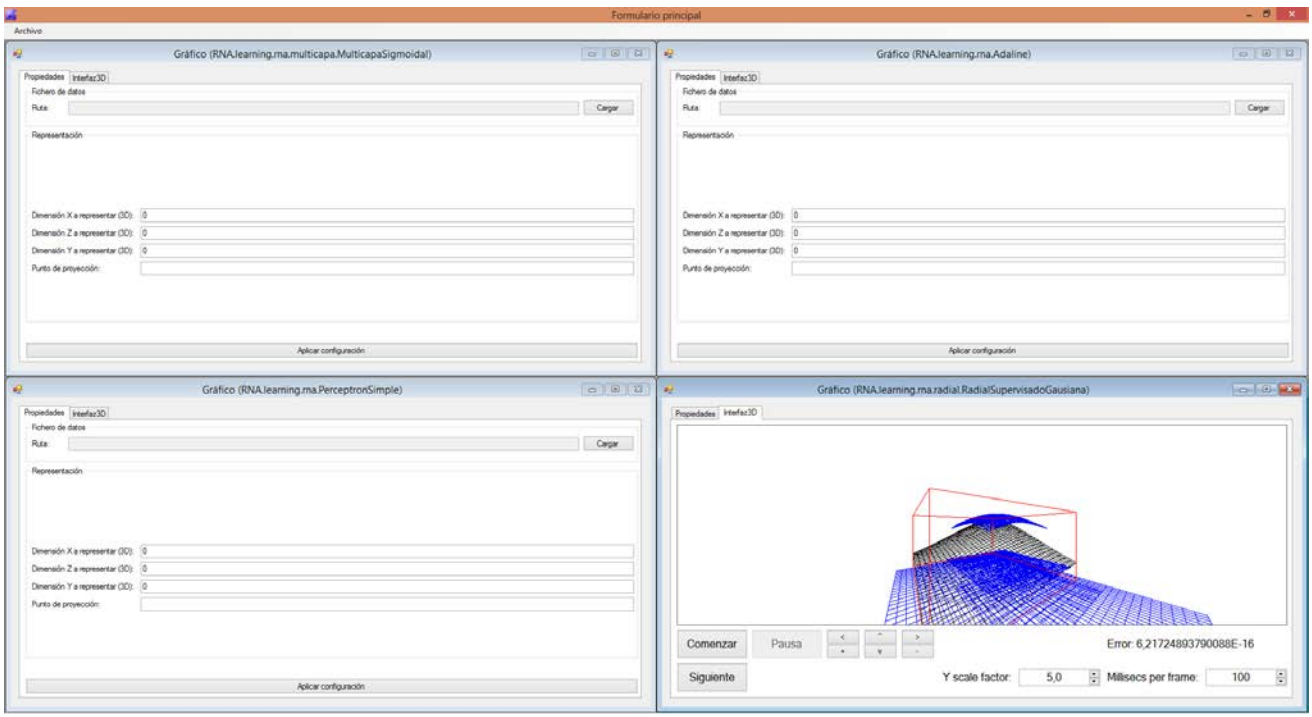


Ilustración 8: Creación de múltiples redes en la interfaz.

Nótese que la última red ha sido modificada, pero eso es intrascendente en nuestra prueba. Ahora procederemos a eliminar las redes y comprobaremos que el layout ajusta el tamaño de los formularios hijos.

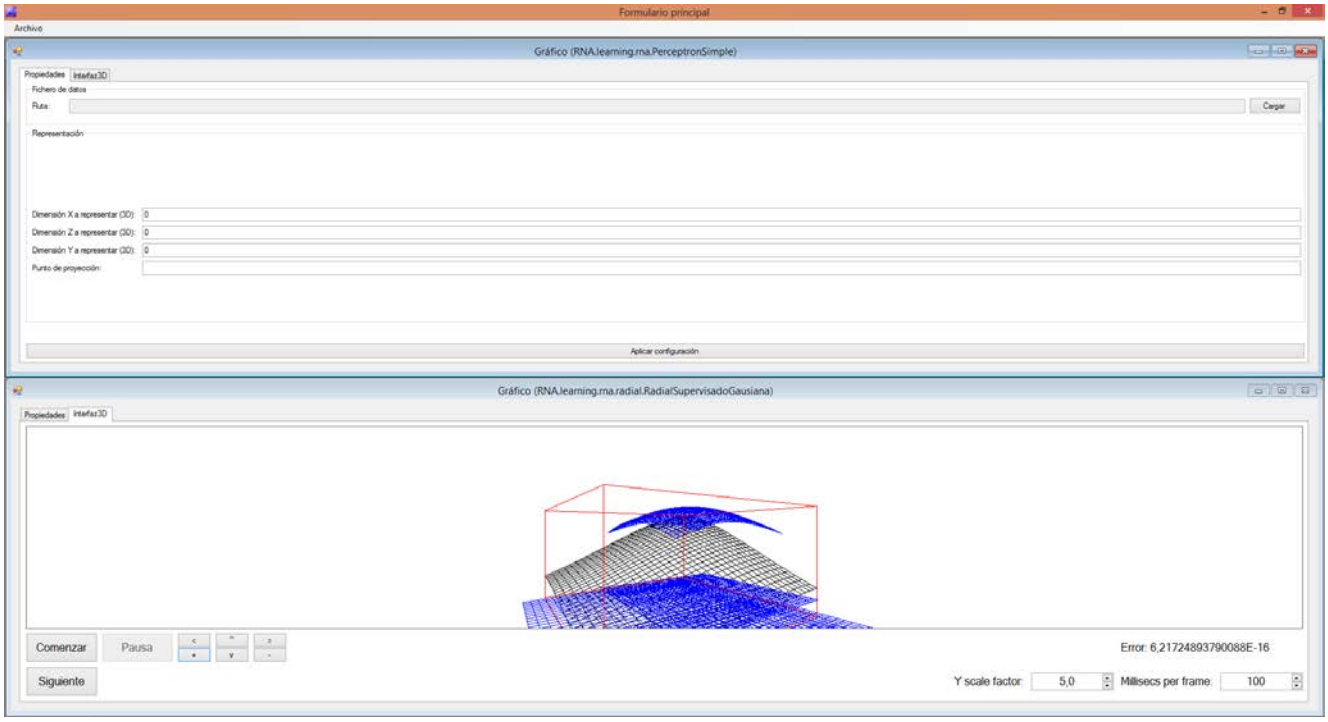


Ilustración 9: Formulario principal tras la eliminación de dos de sus hijos.

Dado que se ha gestionado correctamente la visualización de los formularios hijos la aplicación ha pasado satisfactoriamente la prueba.

4.1.2 Maximizar y minimizar formularios asociados a redes: con esta prueba comprobaremos que el formulario padre ajusta correctamente los tamaños de los formularios cuando estos se maximizan o se minimizan.

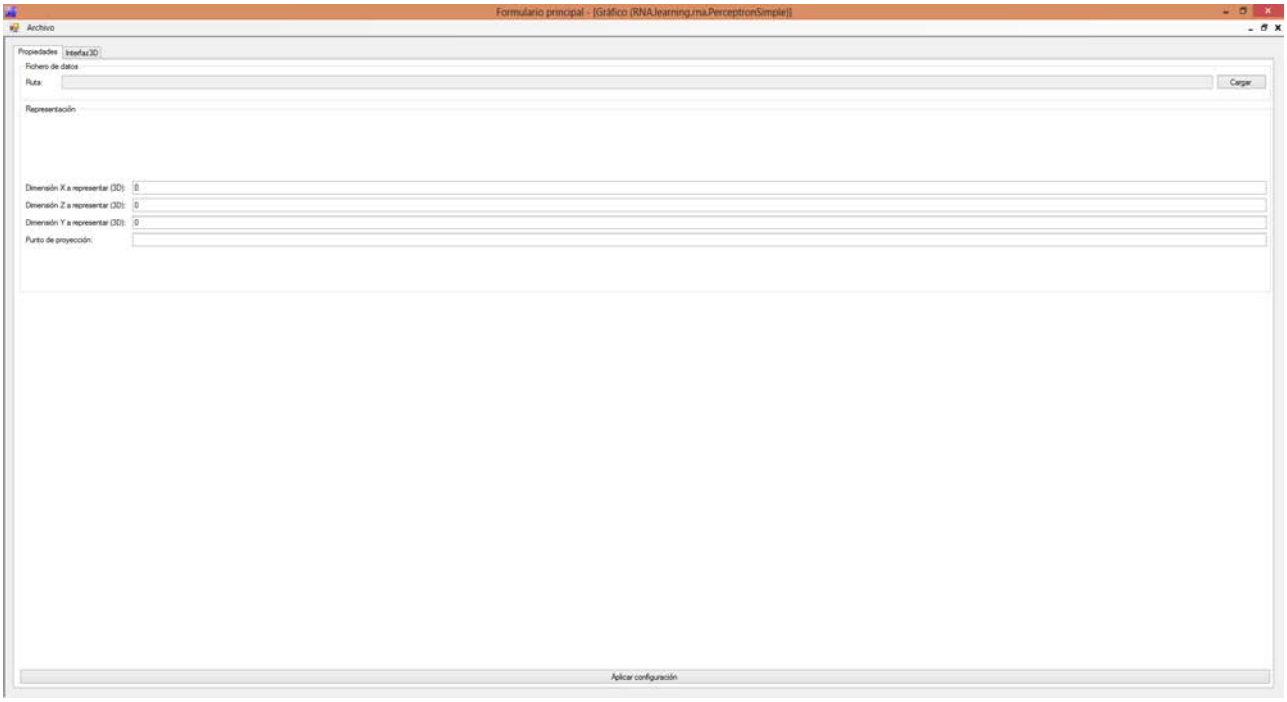


Ilustración 10: Formulario maximizado

Ilustración 11: Formulario minimizado

Los formularios se muestran correctamente por lo que la prueba ha sido superada con éxito

4.2 Pruebas de los métodos de entrada

4.2.1. Prueba de la carga de los ficheros en formato arff.

Para ello cargaremos un fichero en formato arff y comprobaremos mediante el depurador de Visual Studio que el fichero ha sido cargado exitosamente. Para ello en el formulario de la red haremos click en el botón “Cargar”. Se muestra el siguiente formulario:

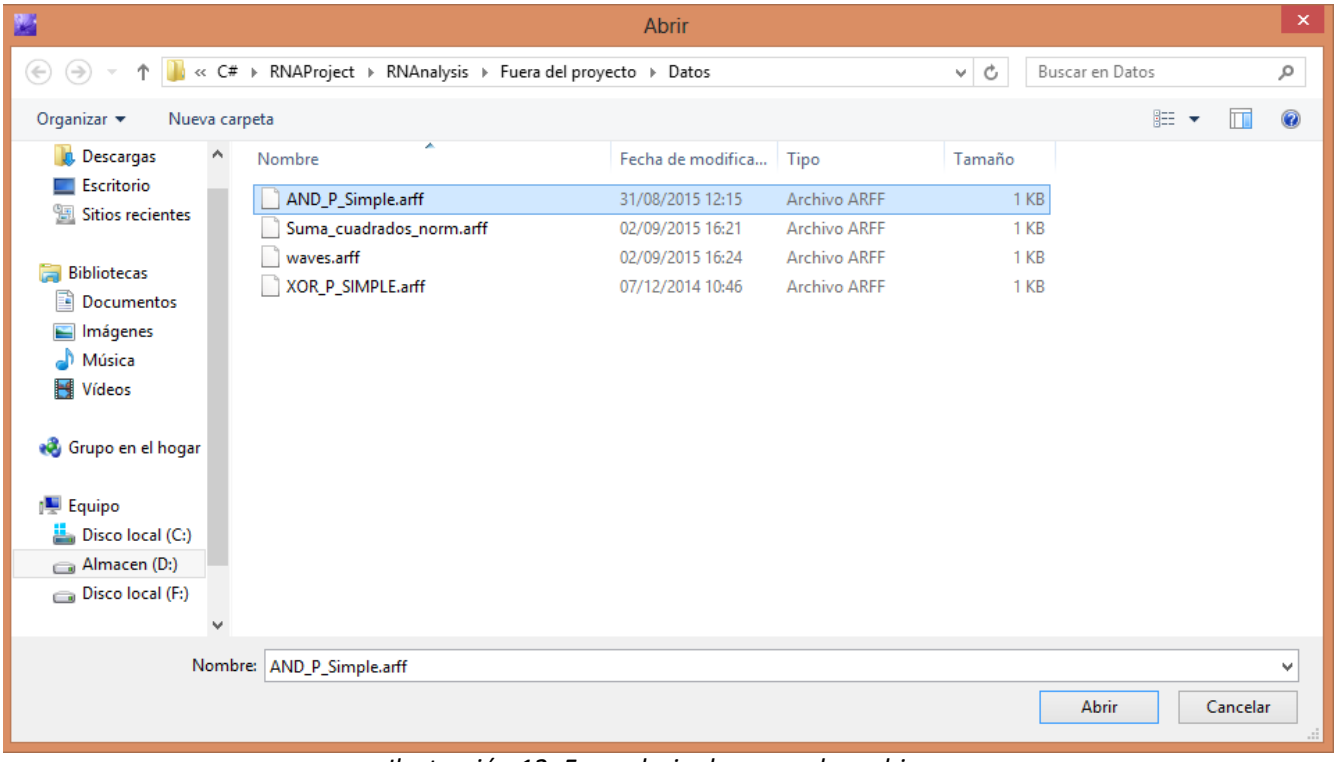


Ilustración 12: Formulario de carga de archivos.

El fichero en cuestión es el siguiente:

@relation AND

@attribute X numeric

@attribute Y numeric

@attribute CLASE {-1,1}

@data

0,0, -1

0,1, -1

1,0, -1

1,1, 1

La información obtenida del depurador es la siguiente:

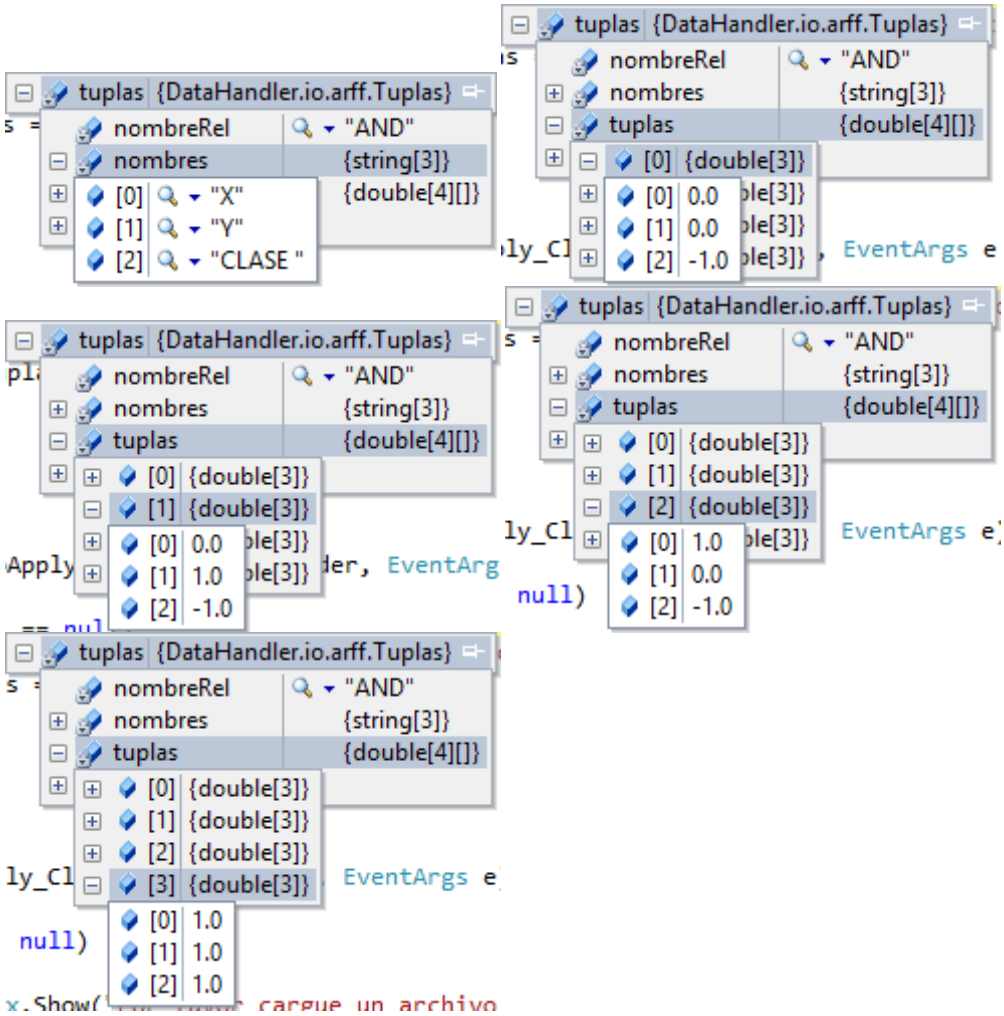


Ilustración 13: Información obtenida del depurador.

Como se puede apreciar el fichero ha sido cargado correctamente.

4.2.2. Introducción de datos en los formularios.

A continuación se comprobará que los formularios solo permiten la introducción de datos en los rangos adecuados.

Tras la introducción de los datos en los formularios se comprobó que los mensajes de error se mostraban al introducir cualquier valor incorrecto y que el formulario interrumpía el manejador asociado a los datos introducidos.

4.3 Pruebas del motor gráfico.

4.3.1. Desplazamiento por el mundo renderizado con el hilo encargado del proceso de aprendizaje detenido.

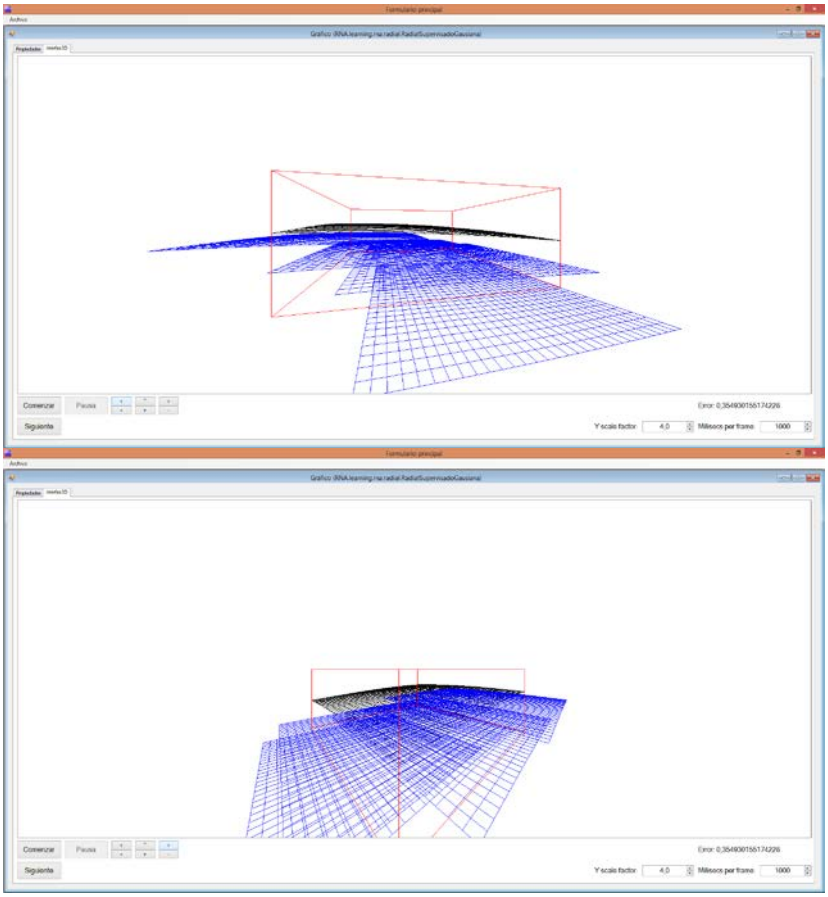


Ilustración 14: Desplazamiento por el mundo renderizado con el hilo encargado del proceso de aprendizaje detenido.

Tras pausar el proceso y desplazarnos por el mundo podemos comprobar el buen funcionamiento del motor.

4.3.1. Desplazamiento por el mundo renderizado con el hilo encargadodel proceso de aprendizaje en ejecución.

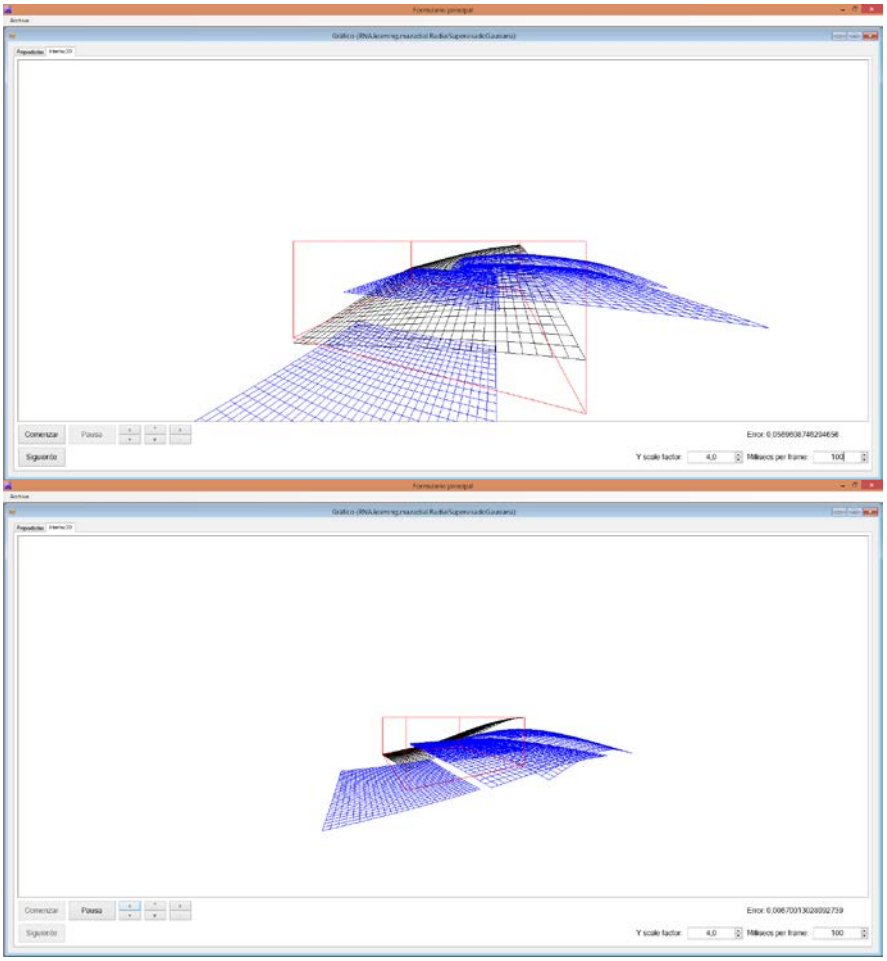


Ilustración 15: Desplazamiento por el mundo renderizado con el hilo encargado del proceso de aprendizaje en ejecución.

El motor funciona correctamente cuando además el hilo encargado del proceso de aprendizaje se encuentra en ejecución.

5. Planificación y entorno socioeconómico:

5.1 Planificación.

La planificación para la realización del proyecto es la siguiente:

Actividad	Fecha de inicio	Duración (días)	Fecha de fin	Dependencias
1. Obtención de requisitos	01/08/2015	2	02/08/2015	-
2. Diseño de la aplicación	03/08/2015	3	05/08/2015	1
3. Implementación	06/08/2015	20	25/08/2015	1,2
4. Pruebas unitarias	26/08/2015	8	02/09/2015	3
5. Redacción de la memoria del trabajo	03/08/2015	10	12/09/2015	1,2,3,4

Tabla 48: planificación del proyecto

Se asume que cada día se trabajará 4 horas.

El camino crítico es el único camino realizable, el que abarca las tareas de la 1 a la cinco en orden ascendente.

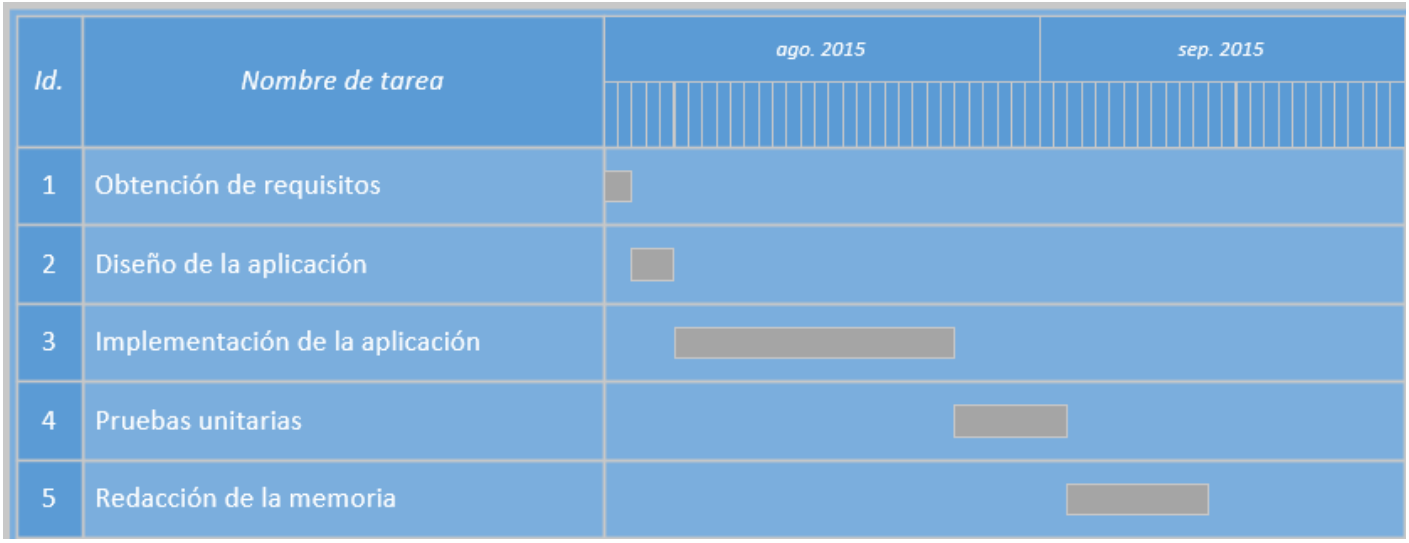


Ilustración 16: Diagrama de Gantt del proyecto.

La planificación del proyecto consta de cinco tareas que se realizarán secuencialmente en orden descendente. En la tarea de pruebas se incluye toda labor de implementación destinada a corregir errores detectados en esta fase.

5.2 Entorno socioeconómico.

A continuación se procede a realizar el cálculo de costes del proyecto dentro del entorno socioeconómico actual

5.2.1 Costes derivados del capital humano

Salario percibido por hora de trabajo: 20 €.

Horas trabajadas: 172.

$$Coste_{trabajador} = Horas \cdot Salario_{hora} = 3340€$$

Ecuación 17: Costes del trabajador

5.2.2 Costes derivados de los equipos informáticos.

Para el desarrollo se ha utilizado un computador personalizado por componentes por un valor de 758€. Debido a que el periodo de amortización es de tres años y el ordenador será utilizado durante cuarenta y tres días, el coste asociado al uso de la máquina es el siguiente:

$$Coste_{equipos} = \frac{43}{365 \cdot 4} \cdot 758\ € = 22{,}32\ €$$

Ecuación 18: Costes de los equipos

5.2.3 Costes derivados del uso de software

Licencia	Coste	Coste imputable
Licencia de Microsoft Windows 8.1 64bits OEM	104€	3,06€
Microsoft Office	-	-
Microsoft Visual Studio 2010	-	-
Microsoft Visio 2013	-	-

Tabla 49: Costes derivados del uso de software

Las licencias de Microsoft Office, Microsoft Visual Studio y Microsoft Visio fueron adquiridas de forma gratuita por la condición de estudiante de la Universidad Carlos III de Madrid.

$$Coste_{software} = 3{,}06€$$

Ecuación 19: Costes del software

5.2.4 Costes del entorno de trabajo.

Descripción	Coste
Electricidad	15,93€
Internet	71,51€

Tabla 50: costes del entorno de trabajo.

$$Coste_{entorno} = Coste_{electricidad} + Coste_{internet} = 87{,}44\ €$$

Ecuación 20: Costes del entorno de trabajo

El coste total del proyecto sin IVA será el siguiente:

$$Coste_{total} = Coste_{trabajador} + Coste_{equipos} + Coste_{software} + Coste_{entorno} = 3452{,}86\ €$$

Ecuación 21: Coste total del proyecto sin IVA.

6. Conclusiones

Tras la realización del proyecto se ha satisfecho el objetivo de formarme como desarrollador así como el objetivo de satisfacer la necesidad de una aplicación que muestre de forma gráfica el proceso de aprendizaje de una red de neuronas de base radial.

Espero que gracias a esta herramienta los alumnos tengan la oportunidad de comprender el funcionamiento de las redes de neuronas de base radial desde una perspectiva más visual, aunque sin dejar de lado los fundamentos matemáticos de los algoritmos. Tras utilizar la aplicación queda patente el hecho de que, si bien esta herramienta es útil, es un mero complemento en el proceso de la comprensión de las redes. Esta herramienta no debe ser utilizada para sustituir al profesor ni al estudio de los fundamentos matemáticos de los algoritmos.

Gracias al desarrollo de esta aplicación hay finalmente un software centrado en el análisis de las redes por encima de la mera aplicación eficiente de los algoritmos.

Personalmente la realización de este proyecto ha incrementado mis conocimientos en muchos ámbitos. Desde el punto de vista de la ingeniería del software me ha permitido adquirir experiencia en el desarrollo de productos software y me ha servido para repasar algunos de los procedimientos adquiridos en el grado. También he adquirido cierta soltura con el lenguaje de programación C# y he aprendido a utilizar herramientas de Visual Studio que desconocía que ofrecía, como los proyectos de modelado.

6.1 Tendencias futuras

Si bien esta aplicación es novedosa es cierto que es mejorable en muchos aspectos. Sería conveniente incrementar el número de redes de neuronas implementadas para enriquecer la aplicación y que no solo sea útil para el estudio de redes de neuronas de base radial sino que también sea útil para comprender otros tipos de redes de neuronas. También podría añadirse al proyecto implementaciones de otros algoritmos para añadir la posibilidad de contrastar los resultados obtenidos con cada técnica.

7. Anexos

7.1 Abstract

7.1.1 Introduction.

The goal of this project is to develop an application able to display a 3D view of a radial basis function neural network´s learning algorithm to help students in the process of learning and understanding these algorithms.

7.1.1.1 Motivation

The reasons that made I started the project are the following ones:

1. The application would be useful because nowadays there is no software which displays this learning process graphically.

2. The comprehension of these algorithms with this tool would be easier.

3. It would be a great chance to develop class libraries and acquire experience as software developer

7.1.1.2 Objectives

Main objectives:

1. The application must be able to display a 3D view of a radial basis function neural network´s learning algorithm. It must be able to display the neural network´s elements during the learning process.

2. The application must be a complementary tool which helps the student to understand the algorithms.

Secondary objectives:

3. The application must be able to work in Windows environments with .NET Framework 4.0 or later.

7.1.2 Problem approach

7.1.2.1 State of the art

A radial basis function neural network is a kind of artificial neural network able to do supervised learning process. This process is a regression process based on base radial functions.

A radial basis function neural network has an input layer, a hidden layer and an output lawyer. The input layer has as many neurons as desired and it´s used as the input of the net. The hidden layer has as many neurons as desired. Each neuron is connected with every input layer neuron by a not weighted link.

Every hidden layer neuron has the following elements:

1. Center: the center of the neuron will determinate its position in the input domain.

2. Derivation: a value that determinates the form of the curve

3. Radial basis activating function.

The input of the function is the result of the application of the following function to the input of the neuron:

$$f(x) = \frac{\|X(n) - C_i\|}{d_i}$$

Equation 22: hidden layer neuron´s input transformation function.

The most frequently radial basis function used are the following ones:

$$f(x) = e^{\frac{-x^2}{2}}$$

Equation 23: gaussian activation function

$$f(x) = \frac{1}{1 + x^2}$$

Equation 24: inverse quadratic function

$$f(x) = \frac{1}{\sqrt{1 + x^2}}$$

Equation 25: inverse multiquadratic function

Every hidden layer neuron is connected with the output layer neuron by a weighted link.

Learning algorithms

There are two variants in the learning algorithm of the radial basis neural networks:

The fully supervised method applies a gradient descent algorithm to calculate all the parameters of the net. The hybrid method applies an unsupervised learning algorithm to calculate the centers of the net, then calculates the derivations of the net with heuristic functions and finally determinates the weights and thresholds with the gradient descendent technique.

Now it will be described the expressions to apply in the learning processes.

If $\omega_{i,k}$ is the weight of the link between the ith hidden neuron and the kth output neuron.

If u_k is the threshold of the kth output neuron.

If $c_{i,j}$ is the jth coordinate of the ith hidden layer neuron´s center.

If d_i is the derivation of the ith hidden layer neuron.

If ϕ_i is the output of the ith hidden layer neuron.

If S_k is the output of the kth output layer neuron.

If X is the pattern input

If y_k is the kth value of the pattern introduced into the net.

If D_i is the distance function of the ith hidden layer neuron.

$$\omega_{i,k}(n) = \omega_{i,k}(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)\phi_i(n)$$

Equation 26: weight calibration formula

$$u_k(n) = u_k(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)$$

Equation 27: threshold calibration formula

$$c_{i,j}(n) = c_{i,j}(n-1) + \propto_2 \left(\sum_{k=1}^{r} \left(\left(S_k(n) - y_k(n)\right)\omega_{i,k}\right)\right)\phi_i(n)\frac{\left(X_j - c_{i,j}\right)}{{d_i}^2}$$

Equation 28: center calibration formula

$$d_i(n) = d_i(n-1) + \propto_3 \left(\sum_{k=1}^{r} \left(\left(S_k(n) - y_k(n)\right)\omega_{i,k}\right)\right)\phi_i(n)\frac{D_i(X)}{{d_i}^2}$$

Equation 29: derivation calibration formula

Centers: as it´s said the centers are calculated using an unsupervised learning algorithm such as K-means or Kohonen SOM.

Derivations: they are calculated using the following heuristic functions:

1. Uniform mean of the Euclidean distance between the center of the neuron and the k nearest patterns to the neuron.

$$d_i = \frac{\sum_{j=1}^{n} \|C_i - C_j\|}{n}$$

Equation 30: uniform mean of Euclidean distances.

2. Geometric mean of the Euclidean distance between the center of the neuron and the two nearest patterns to the neuron

$$d_i = \sqrt{\|C_i - C_0\| \|C_i - C_1\|}$$

Equation 31: geometric mean of Euclidean distances.

Weights and thresholds are calculated with the same expressions of the fully supervised algorithm:

$$\omega_{i,k}(n) = \omega_{i,k}(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)\phi_i(n)$$

Equation 32: weight calibration formula (hybrid method)

$$u_k(n) = u_k(n-1) + \propto_1 \left(S_k(n) - y_k(n)\right)$$

Equation 33: threshold calibration formula (hybrid method)

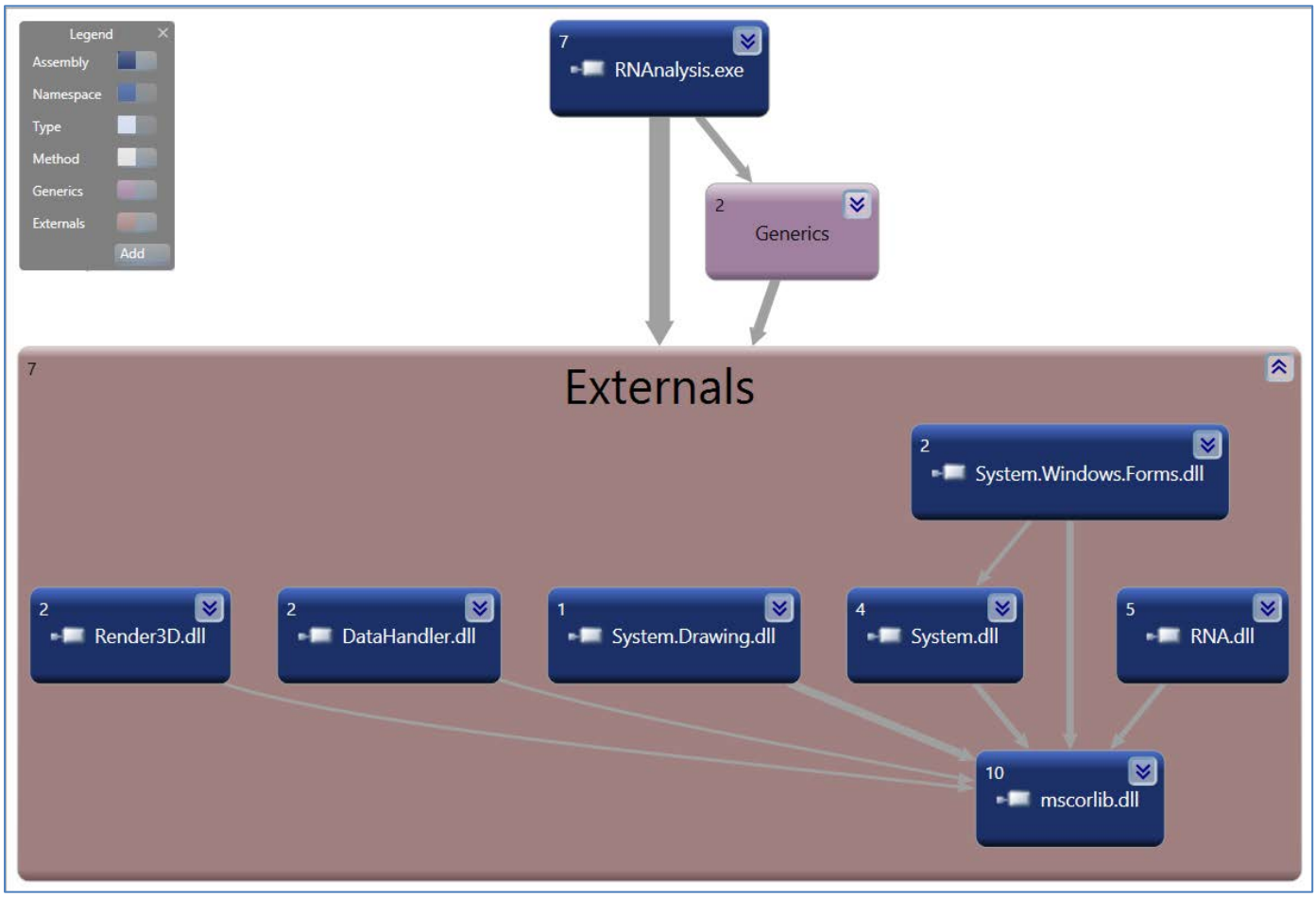
7.1.3 Technical solution design

7.1.3.1 Design alternatives

System components

There are a lot of possible distributions of components to develop the system. The system won´t be composed by one component because the project is too complex to develop one reusable and scalable system with this design.

The components of the system are the following ones:



RNAAnalisis: Main component of the system. It manages the system graphic interface.

System.Windows.Forms: Microsoft component that implements form Management .NET Framework´s section.

Render3D: Component that will render 3D models of the nets.

DataHandler: Component that implements the functionality of the input-output interface of the System.

System.Drawing: Microsoft component that implements the graphical process of .NET Framework.

RNA: Component with the artificial neural networks implementation.

Mscorlib: Microsoft core library of .NET Framework.

Main component forms

There are two alternatives:

1. **Decentralized system of forms associated to nets.**
2. **Forms associated to nets contained in a parent form**

The component will be designed with forms associated to nets contained in a parent form because the parent container will be able to manage the correct display of the forms, making the navigation of the forms easier to the user.

Abstraction level of the neural network algorithms implementation

To allow the software to be maintainable and scalable we will try to maximize the level of abstraction in the neural network algorithms implementation.

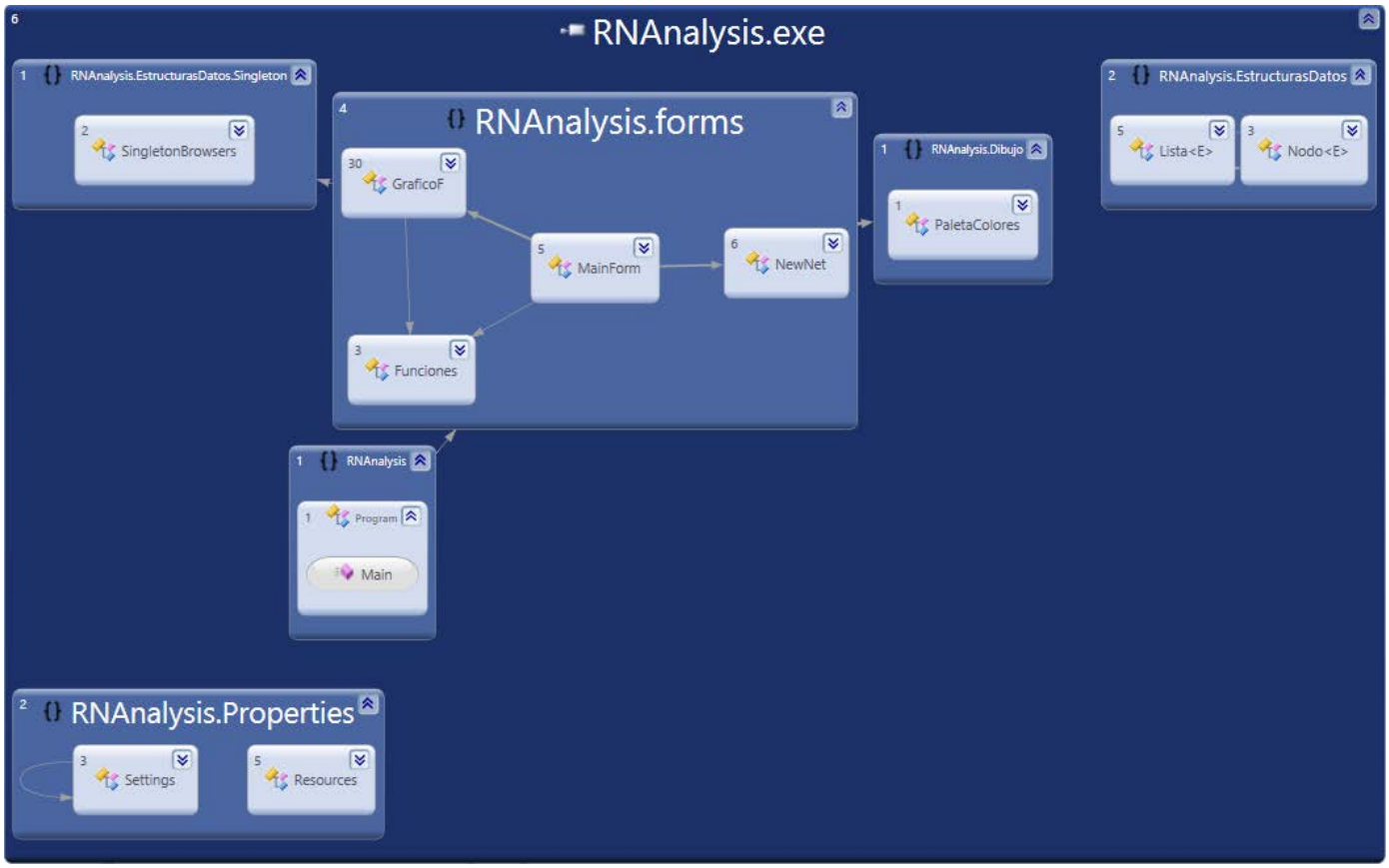
Graphic engine.

We will develop our graphic engine of the system to improve our experience as software programmers and to be able to control the rendering process. It will be possible due to the simple 3D world we must render.

7.1.3.2 Application design

The design of the application is the following one:

RNAAnalisis Component



Picture 17: Dependency graph of RNAAnalisis component.

The namespaces of the component are the following ones:

1. RNAAnalisis.Dibujo: contains a class to store in a single instance a color palette.

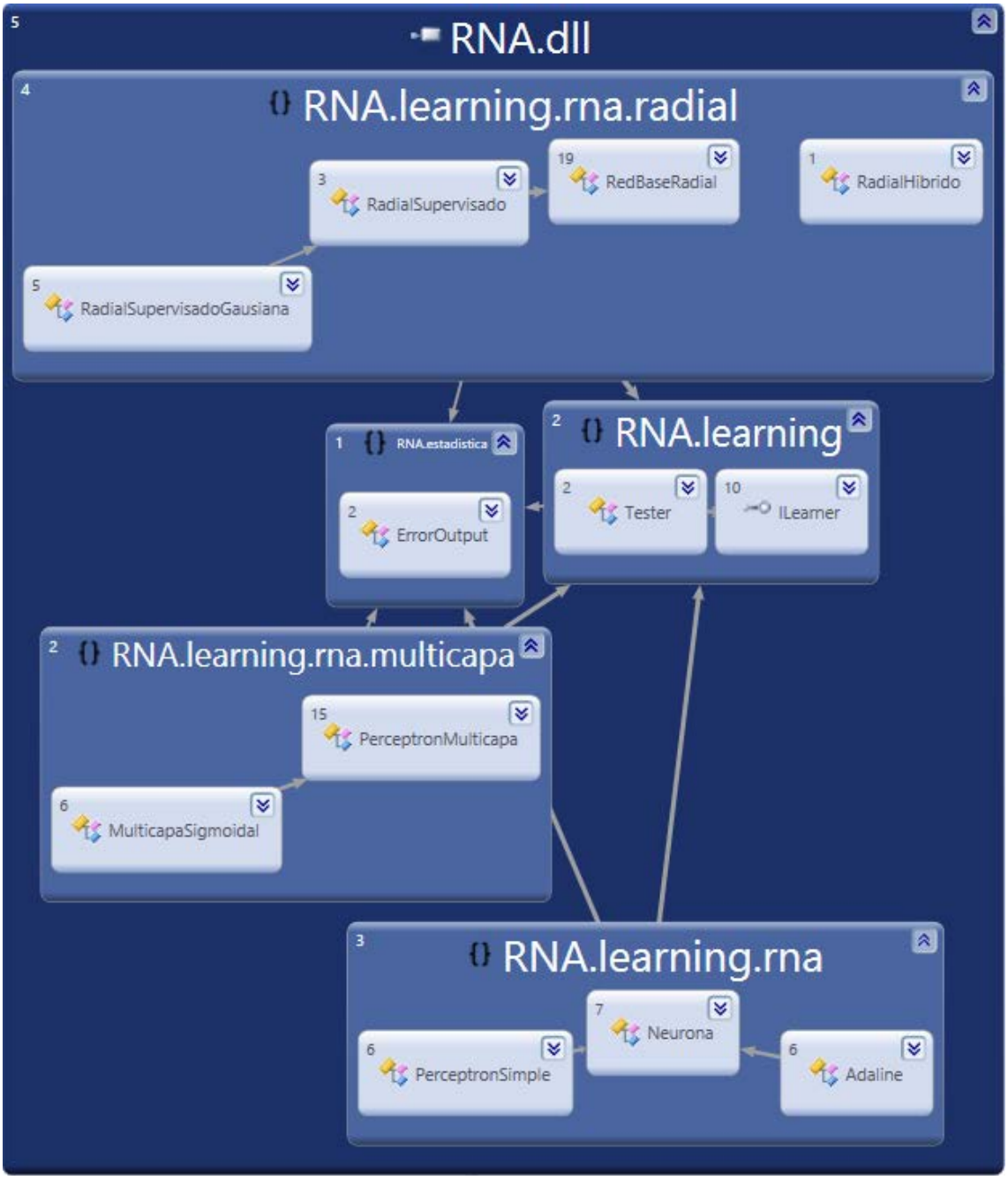
2. RNAAnalisis.EstructurasDatos: contains the implementation of a double linked list

3. RNAAnalisis.EstructurasDatos.Singleton: contains singleton object managers useful to the forms.

4. RNAAnálisis.forms: contains forms implementations and a module with auxiliary functions.

5. RNAAnalisis.Properties: default namespace to manage resources and settings of the forms.

RNA Component



Picture 18: Dependency graph of RNA component.

The namespaces of the component are the following ones:

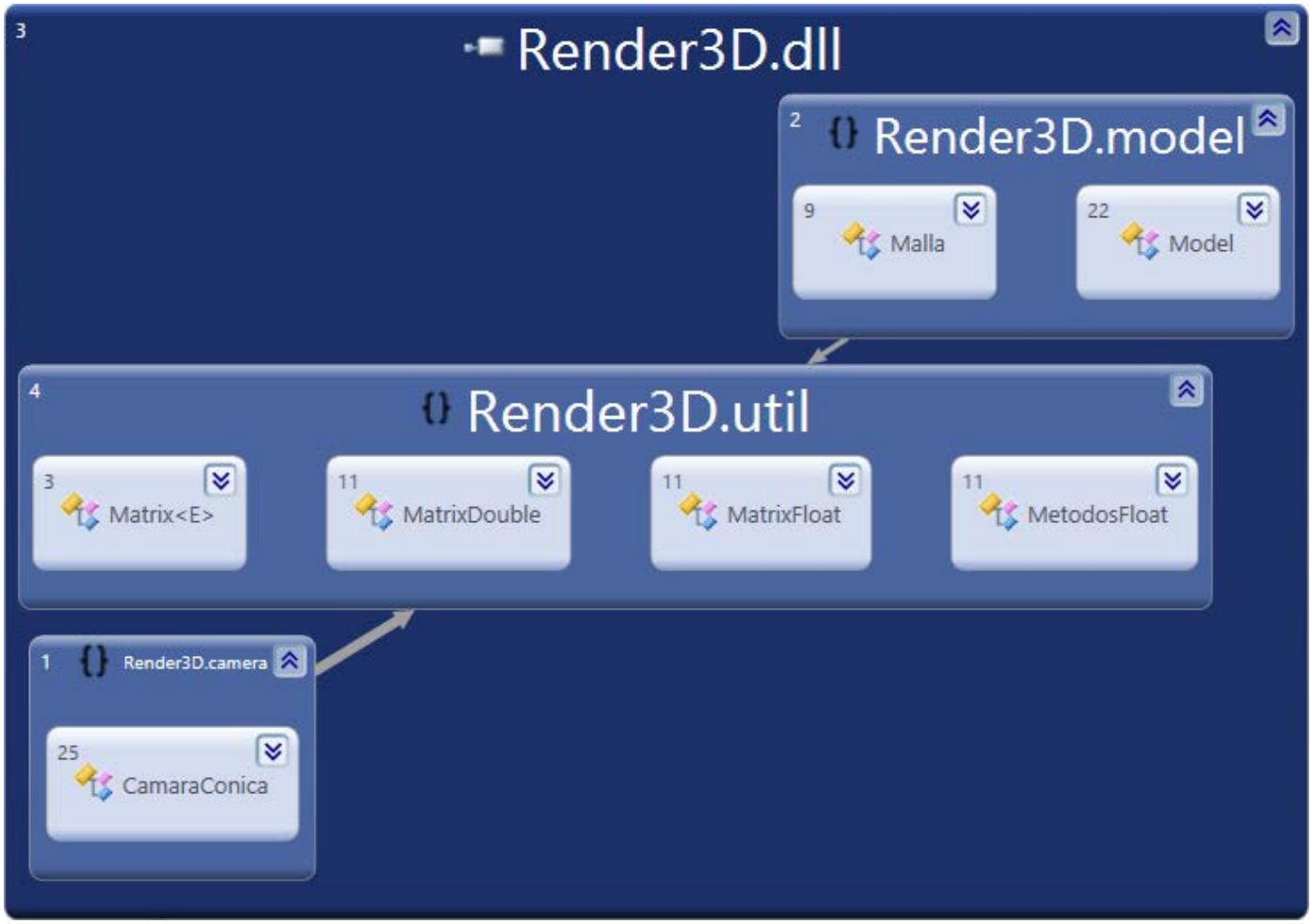
1. **RNA.estadistica**: contains a class to manage the errors made by the nets

2. **RNA.learning**: learning algorithms namespace. Contains the common interface of all the learning techniques of the component.

3. **RNA.learning.rna**: artificial neural net namespace. Contains the implementation of the default neuron and single neuron networks implementation..

4. **RNA.learning.rna.multicapa**: contains the implementation of multilayer perceptron.

5. **RNA.learning.rna.radial**: contains the implementation of the radial basis function neural network.



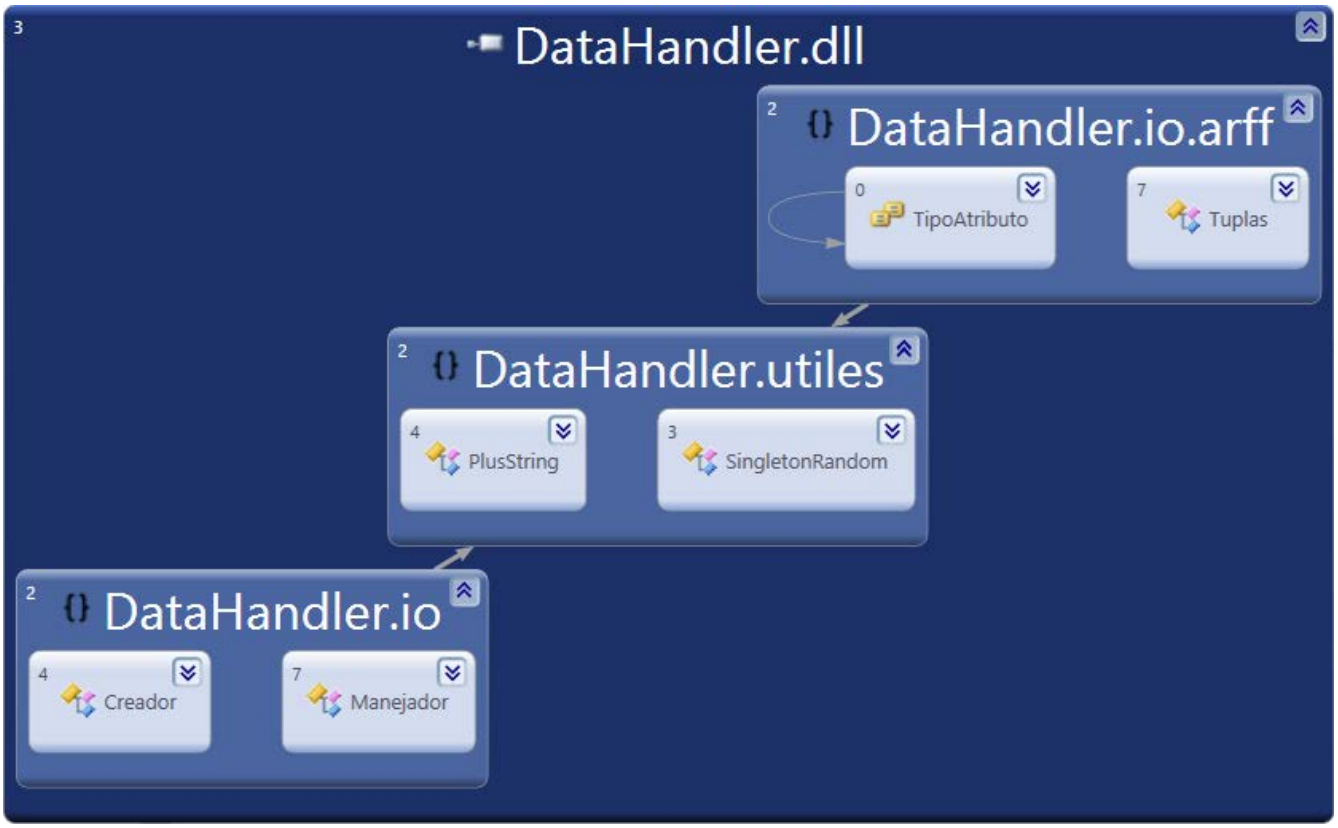
Picture 19: Dependency graph of Render3D component

The namespaces of the component are the following ones:

1. **Render3D.camera**: contains the implementation of the camera.

2. **Render3D.model**: contains the implementation of models that can be loaded in world of the camera

3. **Render3D.util**: contains classes with calculus methods.



Picture 20: Dependency graph of DataHandler component

The namespaces of the component are the following ones:

1. DataHandler.io: contains classes with methods to generate random patterns and manage patterns.

2. DataHandler.io.arff: contains a class to read arff files.

3. DataHandler.utiles: implements methods that can be useful in the component.

7.1.4 Conclusion

After having developed the application, we have achieve the goal of improving as software developer and the goal of developing an application able to display a 3D view of the radial basis function neural network´s learning algorithms.

I Hope this tool could help students in the process of understanding these algorithms with a new perspective. This tool is only a complement in the process; no student should hope this tool will make him understand the algorithm without a minimum mathematical base.

8. Referencias

[1] Isasi, P., Galván, I.M., *Redes de Neuronas Artificiales. Un enfoque práctico.* Pearson 2004.

[2] Valls, J. M., Galván, I.M. Material de Clase: Tema 6, desde el sitio Web de OCW – UC3M. Disponible en: http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas-artificiales/transparencias/material-de-clase.-tema-6

[3] Neural Network Simulator. Disponible en: http://devitep.de/Software/NeuralNetworkSimulator

[4] Bergmeir Christoph, Benítez, J. M, (12/06/2015) Neural Networks in R using the Stuttgart Neural Network Simulator (SNNS). Disponible en: https://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf

[5] encog-java-core (GitHub project). Disponible en: https://github.com/encog/encog-java-core

[6] DLib C++ Library. Disponible en: http://dlib.net/ml.html