



Universidad de Córdoba
Escuela Politécnica Superior

Proyecto Fin de Carrera

Ingeniería Informática

**Modelos de Redes Neuronales para
Regresión Ordinal basados en
Técnicas de Descenso por Gradiente**

Córdoba, Septiembre 2011

Directores:

Prof. Dr. Pedro Antonio Gutiérrez Peña

Prof. Dr. César Hervás Martínez

Autor:

Raúl Pérula Martínez

César Hervás Martínez, Catedrático de Universidad en Ciencias de la Computación e Inteligencia Artificial del Dpto. de Informática y Análisis Numérico en la Escuela Politécnica Superior de la Universidad de Córdoba.

Informa:

Que el presente proyecto fin de carrera titulado “*Modelos de Redes Neuronales para Regresión Ordinal basados en Técnicas de Descenso por Gradiente*”, constituye la memoria presentada por D. Raúl Pérula Martínez para aspirar al título de Ingeniero en Informática, ha sido realizado bajo mi dirección en la Escuela Politécnica Superior de la Universidad de Córdoba reuniendo, a mi juicio, las condiciones necesarios exigidas en este tipo de trabajos.

Y para que conste, se expide y firme el presente certificado en Córdoba, Septiembre de 2011.

El Director de proyecto:

Fdo: Prof. Dr. D. César Hervás Martínez

Pedro Antonio Gutiérrez Peña, Profesor Ayudante Doctor del Dpto. de Informática y Análisis Numérico en la Escuela Politécnica Superior de la Universidad de Córdoba.

Informa:

Que el presente proyecto fin de carrera titulado “*Modelos de Redes Neuronales para Regresión Ordinal basados en Técnicas de Descenso por Gradiente*”, constituye la memoria presentada por D. Raúl Pérula Martínez para aspirar al título de Ingeniero en Informática, ha sido realizado bajo mi dirección en la Escuela Politécnica Superior de la Universidad de Córdoba reuniendo, a mi juicio, las condiciones necesarios exigidas en este tipo de trabajos.

Y para que conste, se expide y firme el presente certificado en Córdoba, Septiembre de 2011.

El Director de proyecto:

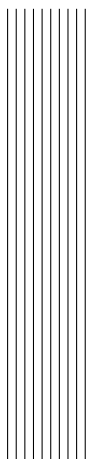
Fdo: Prof. Dr. D. Pedro Antonio Gutiérrez Peña

Agradecimientos

A mi madre por enseñarme todo lo que sé y soy.

A mis compañeros que siempre han estado ahí para dar un toque de alegría y pasar buenos momentos en esta carrera tan bonita y dura.

A Pedro por ser un director ejemplar, paciente y con mucho futuro y a César, por haber depositado su confianza en mí para la realización de este proyecto.



Índice general

| | |
|--|--------------|
| Índice general | I |
| Índice de ilustraciones | XIII |
| Índice de tablas | XVIII |
| 1. Introducción | 1 |
| 2. Definición del problema | 5 |
| 2.1. Definición del problema real | 5 |
| 2.2. Definición del problema técnico | 6 |
| 2.2.1. Funcionamiento | 6 |
| 2.2.2. Entorno | 7 |
| 2.2.3. Vida esperada | 8 |
| 2.2.4. Ciclo de mantenimiento | 8 |
| 2.2.5. Competencia | 9 |
| 2.2.6. Aspecto externo | 9 |
| 2.2.7. Estandarización | 10 |
| 2.2.8. Calidad y fiabilidad | 10 |

| | | |
|-----------|---|-----------|
| 2.2.9. | Programación de tareas | 11 |
| 2.2.10. | Pruebas | 11 |
| 2.2.11. | Seguridad | 12 |
| 2.2.12. | Licencia | 12 |
| 3. | Objetivos | 13 |
| 4. | Antecedentes | 15 |
| 4.1. | Grupo de Investigación AYRNA | 16 |
| 4.2. | Redes Neuronales | 17 |
| 4.2.1. | Regresión ordinal | 19 |
| 4.2.2. | Algoritmo Resilient Backpropagation (RPROP) . | 20 |
| 4.2.3. | Algoritmo iRPROP+ (RPROP mejorado) | 21 |
| 4.2.4. | Redes Neuronales basadas en el modelo Proportio- nal Odd Model (POM) | 22 |
| 4.2.5. | Medidas de rendimiento | 23 |
| 4.2.5.1. | Matriz de confusión | 23 |
| 4.2.5.2. | Ratio Correctamente Clasificado (Correctly Classified Rate (CCR)) | 26 |
| 4.2.5.3. | Error Absoluto Medio (Mean Absolute Error (MAE)) | 26 |
| 4.2.5.4. | Error Cuadrático Medio (Mean Square Error (MSE)) | 28 |
| 4.3. | Matlab | 29 |
| 4.3.1. | Toolbox <i>nnet</i> (Neural Network Toolbox) | 30 |
| 4.3.2. | Interfaces gráficas con Matlab | 31 |
| 5. | Restricciones | 33 |
| 5.1. | Factores dato | 33 |
| 5.2. | Factores estratégicos | 34 |

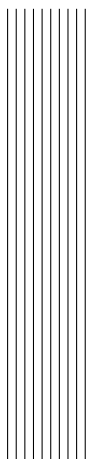
| | |
|---|-----------|
| 6. Recursos | 37 |
| 6.1. Recursos humanos | 37 |
| 6.2. Recursos materiales | 38 |
| 6.2.1. Recursos software | 38 |
| 6.2.2. Recursos hardware | 39 |
| 7. Especificación del sistema | 41 |
| 7.1. Algoritmo de Regresión Ordinal ORNNNet | 41 |
| 7.2. Descripción funcional del sistema | 48 |
| 7.2.1. Valores de los diferentes parámetros que utiliza el sistema | 49 |
| 7.2.2. Diagramas de Casos de Uso | 50 |
| 7.2.2.1. Perfiles de usuario | 51 |
| 7.2.2.2. Casos de Uso | 52 |
| 7.2.3. Diagrama de Secuencia | 52 |
| 8. Diseño del sistema | 57 |
| 8.1. Diseño de los módulos | 58 |
| 8.1.1. Módulo principal nnet | 58 |
| 8.1.1.1. Módulo @network | 60 |
| 8.1.1.2. Módulo nnformat | 60 |
| 8.1.1.3. Módulo nnnetwork | 61 |
| 8.1.1.4. Módulo nnother | 61 |
| 8.1.1.5. Módulo nnperformance | 61 |
| 8.1.1.6. Módulo nntrain | 63 |
| 8.2. Diseño de la interfaz | 64 |
| 8.2.1. Interfaz principal | 64 |
| 8.2.2. Interfaz de datos | 65 |
| 8.2.3. Interfaz para la creación de la red neuronal | 67 |
| 8.2.4. Interfaz de entrenamiento y simulación | 67 |
| 8.2.5. Interfaz de exportación de datos | 70 |

| | |
|--|-----------|
| 9. Implementación | 73 |
| 9.1. Entrenamiento y Simulación | 73 |
| 9.2. Entrenamiento ordinal | 74 |
| 9.3. Creación de una red neuronal ordinal | 76 |
| 9.4. Cálculo del número óptimo de neuronas en la capa oculta | 78 |
| 10.Pruebas | 81 |
| 10.1. Estrategia de pruebas | 82 |
| 10.1.1. Pruebas Unitarias | 83 |
| 10.1.2. Pruebas de Integración | 85 |
| 10.1.3. Pruebas del Sistema | 86 |
| 10.1.4. Pruebas de Aceptación | 87 |
| 11.Experimentación | 89 |
| 11.1. Configuración software y hardware | 89 |
| 11.2. Condiciones de los experimentos | 90 |
| 11.2.1. Diseño experimental | 90 |
| 11.3. Conjuntos de datos | 91 |
| 12.Resultados | 97 |
| 12.1. Resultados individuales | 97 |
| 12.1.1. Clasificación Nominal | 98 |
| 12.1.1.1. Conjunto de datos Automobile | 98 |
| 12.1.1.2. Conjunto de datos Balance Scale | 98 |
| 12.1.1.3. Conjunto de datos Bondrate | 98 |
| 12.1.1.4. Conjunto de datos Car | 105 |
| 12.1.1.5. Conjunto de datos Contact lenses | 107 |
| 12.1.1.6. Conjunto de datos Depression | 109 |
| 12.1.1.7. Conjunto de datos ERA | 109 |
| 12.1.1.8. Conjunto de datos ESL | 109 |
| 12.1.1.9. Conjunto de datos Eucalyptus | 115 |
| 12.1.1.10.Conjunto de datos LEV | 117 |

| | | |
|------------|--|-----|
| 12.1.1.11. | Conjunto de datos New Thyroid | 119 |
| 12.1.1.12. | Conjunto de datos Pasture | 121 |
| 12.1.1.13. | Conjunto de datos Squash Stored | 123 |
| 12.1.1.14. | Conjunto de datos Squash Unstored | 125 |
| 12.1.1.15. | Conjunto de datos SWD | 127 |
| 12.1.1.16. | Conjunto de datos TAE | 129 |
| 12.1.1.17. | Conjunto de datos Thyroid | 131 |
| 12.1.1.18. | Conjunto de datos Wine Quality Red | 133 |
| 12.1.1.19. | Conjunto de datos Wine Quality White | 135 |
| 12.1.2. | Clasificación Ordinal | 137 |
| 12.1.2.1. | Conjunto de datos Automobile | 137 |
| 12.1.2.2. | Conjunto de datos Balance Scale | 137 |
| 12.1.2.3. | Conjunto de datos Bondrate | 137 |
| 12.1.2.4. | Conjunto de datos Car | 144 |
| 12.1.2.5. | Conjunto de datos Contact lenses | 146 |
| 12.1.2.6. | Conjunto de datos Depression | 148 |
| 12.1.2.7. | Conjunto de datos ERA | 148 |
| 12.1.2.8. | Conjunto de datos ESL | 148 |
| 12.1.2.9. | Conjunto de datos Eucalyptus | 154 |
| 12.1.2.10. | Conjunto de datos LEV | 156 |
| 12.1.2.11. | Conjunto de datos New Thyroid | 158 |
| 12.1.2.12. | Conjunto de datos Pasture | 160 |
| 12.1.2.13. | Conjunto de datos Squash Stored | 162 |
| 12.1.2.14. | Conjunto de datos Squash Unstored | 164 |
| 12.1.2.15. | Conjunto de datos SWD | 166 |
| 12.1.2.16. | Conjunto de datos TAE | 168 |
| 12.1.2.17. | Conjunto de datos Thyroid | 170 |
| 12.1.2.18. | Conjunto de datos Wine Quality Red | 172 |
| 12.1.2.19. | Conjunto de datos Wine Quality White | 174 |
| 12.2. | Resultados generales | 176 |
| 12.2.1. | Clasificación Nominal | 176 |
| 12.2.2. | Clasificación Ordinal | 177 |

| | |
|--|------------|
| 12.3. Comparativa entre los resultados | 177 |
| 13. Conclusiones y Futuras Mejoras | 181 |
| 13.1. Conclusiones | 181 |
| 13.2. Futuras Mejoras | 183 |
| Bibliografía | 184 |
| A. Manual de usuario | 187 |
| A.1. Instalación y desinstalación | 187 |
| A.1.1. Instalación | 187 |
| A.1.2. Desinstalación | 191 |
| A.2. Uso de la aplicación | 191 |
| B. Manual de código | 211 |
| B.1. Carpeta nnet | 212 |
| B.1.1. Carpeta @network | 212 |
| B.1.1.1. Fichero osim.m | 212 |
| B.1.1.2. Fichero otrain.m | 213 |
| B.1.2. Carpeta nnformat | 214 |
| B.1.2.1. Fichero dividestra.m | 214 |
| B.1.3. Carpeta nnnetwork | 216 |
| B.1.3.1. Fichero newoff.m | 216 |
| B.1.4. Carpeta nnother | 219 |
| B.1.4.1. Fichero convdata.m | 219 |
| B.1.4.2. Fichero convoutputs.m | 220 |
| B.1.4.3. Fichero getstra.m | 222 |
| B.1.4.4. Fichero importfile.m | 223 |
| B.1.4.5. Fichero kfold.m | 223 |
| B.1.4.6. Fichero kfoldo.m | 225 |
| B.1.4.7. Fichero transdata.m | 227 |
| B.1.5. Carpeta nnperformance | 228 |

| | | |
|----------|-------------------------------|-----|
| B.1.5.1. | Fichero ccrcalc.m | 228 |
| B.1.5.2. | Fichero maecalc.m | 229 |
| B.1.6. | Carpeta nntrain | 230 |
| B.1.6.1. | Fichero trainirp.m | 230 |
| B.1.6.2. | Fichero trainirpo.m | 239 |
| B.2. | Carpeta nnguis | 248 |
| B.2.1. | Fichero onntool.m | 248 |



Índice de ilustraciones

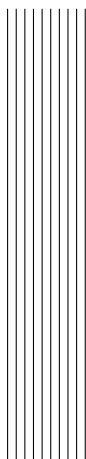
| | | |
|-------|---|-----|
| 4.1. | Red Neuronal Artificial | 18 |
| 4.2. | Funcionamiento de una Red Neuronal Artificial | 18 |
| 7.1. | Funcionamiento del algoritmo para redes neuronales artificiales ORNNet. | 43 |
| 7.2. | Diagrama de Contexto del Sistema. | 53 |
| 7.3. | Diagrama de caso de uso CU1: Tratamiento RNA. | 54 |
| 7.4. | Diagrama de Secuencia. | 56 |
| 8.1. | Diagrama de Paquetes principal | 59 |
| 8.2. | Diagrama de Paquetes del módulo nnet diseñado | 59 |
| 8.3. | Prototipo de la Interfaz principal | 64 |
| 8.4. | Prototipo de la Interfaz de datos | 66 |
| 8.5. | Prototipo de la Interfaz de red neuronal | 68 |
| 8.6. | Prototipo de la Interfaz de entrenamiento | 69 |
| 8.7. | Prototipo de la Interfaz de exportación | 71 |
| 12.1. | Matriz de confusión. Conjunto de datos Automobile. Clasificación nominal. | 100 |

| | |
|---|-----|
| 12.2. Matriz de confusión. Conjunto de datos Balance Scale. Clasificación nominal. | 102 |
| 12.3. Matriz de confusión. Conjunto de datos Bondrate. Clasi- ficación nominal. | 104 |
| 12.4. Matriz de confusión. Conjunto de datos Car. Clasificación nominal. | 105 |
| 12.5. Matriz de confusión. Conjunto de datos Contact Lenses. Clasificación nominal. | 107 |
| 12.6. Matriz de confusión. Conjunto de datos Depression. Cla- sificación nominal. | 110 |
| 12.7. Matriz de confusión. Conjunto de datos ERA. Clasifica- ción nominal. | 112 |
| 12.8. Matriz de confusión. Conjunto de datos ESL. Clasificación nominal. | 114 |
| 12.9. Matriz de confusión. Conjunto de datos Eucalyptus. Cla- sificación nominal. | 115 |
| 12.10 Matriz de confusión. Conjunto de datos LEV. Clasifica- ción nominal. | 117 |
| 12.11 Matriz de confusión. Conjunto de datos New Thyroid. Clasificación nominal. | 119 |
| 12.12 Matriz de confusión. Conjunto de datos Pasture. Clasifi- cación nominal. | 121 |
| 12.13 Matriz de confusión. Conjunto de datos Squash Stored. Clasificación nominal. | 123 |
| 12.14 Matriz de confusión. Conjunto de datos Squash Unstored. Clasificación nominal. | 125 |
| 12.15 Matriz de confusión. Conjunto de datos SWD. Clasifica- ción nominal. | 127 |
| 12.16 Matriz de confusión. Conjunto de datos TAE. Clasifica- ción nominal. | 129 |
| 12.17 Matriz de confusión. Conjunto de datos Thyroid. Clasifi- cación nominal. | 131 |

| | | |
|-------|---|-----|
| 12.18 | Matriz de confusión. Conjunto de datos Wine Quality Red. Clasificación nominal. | 133 |
| 12.19 | Matriz de confusión. Conjunto de datos Wine Quality White. Clasificación nominal. | 135 |
| 12.20 | Matriz de confusión. Conjunto de datos Automobile. Clasificación ordinal. | 139 |
| 12.21 | Matriz de confusión. Conjunto de datos Balance Scale. Clasificación ordinal. | 141 |
| 12.22 | Matriz de confusión. Conjunto de datos Bondrate. Clasificación ordinal. | 143 |
| 12.23 | Matriz de confusión. Conjunto de datos CAR. Clasificación ordinal. | 144 |
| 12.24 | Matriz de confusión. Conjunto de datos Contact Lenses. Clasificación ordinal. | 146 |
| 12.25 | Matriz de confusión. Conjunto de datos Depression. Clasificación ordinal. | 149 |
| 12.26 | Matriz de confusión. Conjunto de datos ERA. Clasificación ordinal. | 151 |
| 12.27 | Matriz de confusión. Conjunto de datos ESL. Clasificación ordinal. | 153 |
| 12.28 | Matriz de confusión. Conjunto de datos Eucalyptus. Clasificación ordinal. | 154 |
| 12.29 | Matriz de confusión. Conjunto de datos LEV. Clasificación ordinal. | 156 |
| 12.30 | Matriz de confusión. Conjunto de datos New Thyroid. Clasificación ordinal. | 158 |
| 12.31 | Matriz de confusión. Conjunto de datos Pasture. Clasificación ordinal. | 160 |
| 12.32 | Matriz de confusión. Conjunto de datos Squash Stored. Clasificación ordinal. | 162 |
| 12.33 | Matriz de confusión. Conjunto de datos Squash Unstored. Clasificación ordinal. | 164 |

| | | |
|-------|---|-----|
| 12.34 | Matriz de confusión. Conjunto de datos SWD. Clasificación ordinal. | 166 |
| 12.35 | Matriz de confusión. Conjunto de datos TAE. Clasificación ordinal. | 168 |
| 12.36 | Matriz de confusión. Conjunto de datos Thyroid. Clasificación ordinal. | 170 |
| 12.37 | Matriz de confusión. Conjunto de datos Wine Quality Red. Clasificación ordinal. | 172 |
| 12.38 | Matriz de confusión. Conjunto de datos Wine Quality White. Clasificación ordinal. | 174 |
| A.1. | Instalación. Archivo del código fuente comprimido. . . . | 188 |
| A.2. | Instalación. Carpeta del código fuente. | 188 |
| A.3. | Instalación. Menú File. | 189 |
| A.4. | Instalación. Añadir al Path. | 190 |
| A.5. | Instalación. Cuadro de diálogo. | 190 |
| A.6. | Desinstalación. Eliminación de contenido del Path. . . . | 191 |
| A.7. | Interfaz gráfica. Pantalla de bienvenida. | 193 |
| A.8. | Interfaz gráfica. Pantalla de carga de datos. | 194 |
| A.9. | Interfaz gráfica. Aviso de falta de carga de datos. . . . | 194 |
| A.10. | Interfaz gráfica. Carga de datos. | 195 |
| A.11. | Interfaz gráfica. Selección de la carpeta datasets. | 195 |
| A.12. | Interfaz gráfica. Selección de la carpeta 10-holdout. . . . | 196 |
| A.13. | Interfaz gráfica. Selección del fichero de entrenamiento. . | 196 |
| A.14. | Interfaz gráfica. Selección del fichero de test. | 197 |
| A.15. | Interfaz gráfica. Datos cargados. | 198 |
| A.16. | Interfaz gráfica. Pantalla de configuración de la red neuronal. | 199 |
| A.17. | Interfaz gráfica. Opción manual de configuración de la red neuronal. | 200 |
| A.18. | Interfaz gráfica. Opción de función de transferencia. . . . | 200 |
| A.19. | Interfaz gráfica. Opción para realizar un K-fold. | 201 |

| | |
|---|-----|
| A.20. Interfaz gráfica. Aviso de la opción K-fold. | 202 |
| A.21. Interfaz gráfica. Barra de espera de la opción K-fold. . . | 202 |
| A.22. Interfaz gráfica. Pantalla de entrenamiento. | 203 |
| A.23. Interfaz gráfica. Pantalla con red neuronal de forma gráfica. | 204 |
| A.24. Interfaz gráfica. Matriz de confusión. | 205 |
| A.25. Interfaz gráfica. Pantalla de entrenamiento de nnet. . . . | 206 |
| A.26. Interfaz gráfica. Muestra de los resultados obtenidos. . . | 207 |
| A.27. Interfaz gráfica. Pantalla de guardado de datos. | 208 |



Índice de tablas

| | |
|---|-----|
| 4.1. Matriz de confusión | 23 |
| 7.1. Diagrama de caso de uso CU0: Contexto del Sistema. . . | 53 |
| 7.2. Diagrama de caso de uso CU1. | 55 |
| 8.1. Especificación módulo nnet | 59 |
| 8.2. Especificación módulo @network | 60 |
| 8.3. Especificación módulo nnformat | 60 |
| 8.4. Especificación módulo nnnetwork | 61 |
| 8.5. Especificación módulo nnother | 62 |
| 8.6. Especificación módulo nnperformance | 63 |
| 8.7. Especificación módulo nntrain | 63 |
| 11.1. Conjuntos de datos usados. | 92 |
| 12.1. Resultados individuales. Conjunto de datos Automobile. Clasificación nominal. | 99 |
| 12.2. Resultados individuales. Conjunto de datos Balance Scale. Clasificación nominal. | 101 |

| | |
|---|-----|
| 12.3. Resultados individuales. Conjunto de datos Bondrate. Clasificación nominal. | 103 |
| 12.4. Resultados individuales. Conjunto de datos Car. Clasificación nominal. | 106 |
| 12.5. Resultados individuales. Conjunto de datos Contact Lenses. Clasificación nominal. | 108 |
| 12.6. Resultados individuales. Conjunto de datos Depression. Clasificación nominal. | 109 |
| 12.7. Resultados individuales. Conjunto de datos ERA. Clasificación nominal. | 111 |
| 12.8. Resultados individuales. Conjunto de datos ESL. Clasificación nominal. | 113 |
| 12.9. Resultados individuales. Conjunto de datos Eucalyptus. Clasificación nominal. | 116 |
| 12.10 Resultados individuales. Conjunto de datos LEV. Clasificación nominal. | 118 |
| 12.11 Resultados individuales. Conjunto de datos New Thyroid. Clasificación nominal. | 120 |
| 12.12 Resultados individuales. Conjunto de datos Pasture. Clasificación nominal. | 122 |
| 12.13 Resultados individuales. Conjunto de datos Squash Stored. Clasificación nominal. | 124 |
| 12.14 Resultados individuales. Conjunto de datos Squash Unstored. Clasificación nominal. | 126 |
| 12.15 Resultados individuales. Conjunto de datos SWD. Clasificación nominal. | 128 |
| 12.16 Resultados individuales. Conjunto de datos TAE. Clasificación nominal. | 130 |
| 12.17 Resultados individuales. Conjunto de datos Thyroid. Clasificación nominal. | 132 |
| 12.18 Resultados individuales. Conjunto de datos Wine Quality Red. Clasificación nominal. | 134 |

| | | |
|-------|---|-----|
| 12.19 | Resultados individuales. Conjunto de datos Wine Quality White. Clasificación nominal. | 136 |
| 12.20 | Resultados individuales. Conjunto de datos Automobile. Clasificación ordinal. | 138 |
| 12.21 | Resultados individuales. Conjunto de datos Balance Scale. Clasificación ordinal. | 140 |
| 12.22 | Resultados individuales. Conjunto de datos Bondrate. Clasificación ordinal. | 142 |
| 12.23 | Resultados individuales. Conjunto de datos Car. Clasificación ordinal. | 145 |
| 12.24 | Resultados individuales. Conjunto de datos Contact Lenses. Clasificación ordinal. | 147 |
| 12.25 | Resultados individuales. Conjunto de datos Depression. Clasificación ordinal. | 148 |
| 12.26 | Resultados individuales. Conjunto de datos ERA. Clasificación ordinal. | 150 |
| 12.27 | Resultados individuales. Conjunto de datos ESL. Clasificación ordinal. | 152 |
| 12.28 | Resultados individuales. Conjunto de datos Eucalyptus. Clasificación ordinal. | 155 |
| 12.29 | Resultados individuales. Conjunto de datos LEV. Clasificación ordinal. | 157 |
| 12.30 | Resultados individuales. Conjunto de datos New Thyroid. Clasificación ordinal. | 159 |
| 12.31 | Resultados individuales. Conjunto de datos Pasture. Clasificación ordinal. | 161 |
| 12.32 | Resultados individuales. Conjunto de datos Squash Stored. Clasificación ordinal. | 163 |
| 12.33 | Resultados individuales. Conjunto de datos Squash Unstored. Clasificación ordinal. | 165 |
| 12.34 | Resultados individuales. Conjunto de datos SWD. Clasificación ordinal. | 167 |

| | | |
|-------|---|-----|
| 12.35 | Resultados individuales. Conjunto de datos TAE. Clasificación ordinal. | 169 |
| 12.36 | Resultados individuales. Conjunto de datos Thyroid. Clasificación ordinal. | 171 |
| 12.37 | Resultados individuales. Conjunto de datos Wine Quality Red. Clasificación ordinal. | 173 |
| 12.38 | Resultados individuales. Conjunto de datos Wine Quality White. Clasificación ordinal. | 175 |
| 12.39 | Resultados generales. Clasificación nominal. | 176 |
| 12.40 | Resultados generales. Clasificación ordinal. | 177 |
| 12.41 | Comparativa entre resultados generales. | 179 |



1 Introducción

Este capítulo tiene como finalidad dar al lector una visión general del dominio del problema. Para ello se le intenta situar en el contexto del mismo, introduciéndole en el modelado de sistemas mediante el entrenamiento de redes neuronales artificiales haciendo uso de algoritmos de regresión ordinal.

Éste es uno de los principales campos de investigación del grupo AYRNA¹ (Aprendizaje y Redes Neuronales Artificiales) del Departamento de Informática y Análisis Numérico de la Universidad de Córdoba (UCO). Grupo en el cual queda situado y que marcará los objetivos del presente proyecto.

En el mundo real surge la necesidad de predecir datos que necesitamos o determinados sucesos que ocurren a partir del estudio de determinadas variables, es decir, es necesario anticiparnos a ellos para actuar en consecuencia. Para ello se utiliza el modelado de sistemas, el cual se presenta como uno de los problemas de mayor interés en numerosas ramas de la ciencia. Los modelos son abstracciones de la realidad que el ser humano

¹<http://www.uco.es/grupos/ayrna/>

realiza para llegar a un mayor grado de comprensión de ésta. Una solución al problema de predicción de un suceso es el modelado del mismo, es decir, realizar una abstracción del mismo que recoja sus características y comportamiento, para llegar a la comprensión de su realidad. El modelado de sistemas se puede aplicar en diversos campos de estudio científico. Para ello se realiza un modelado del sistema de manera que el modelo sea capaz de generalizar ante cualquier nueva situación, estudiada o no previamente, siempre dentro del dominio del problema. Para llegar a este modelo se realiza un entrenamiento previo del sistema a partir de los datos observados.

Algunos de los problemas más comunes con los que nos encontramos son la regresión y la clasificación. La regresión trata de establecer una relación funcional entre alguna de las variables dependientes que afectan al problema y que están en escala de ratios o intervalos, y las variables independientes del mismo, mientras que la clasificación intenta predecir la clase de pertenencia de los patrones a los que se aplica, que en este caso las variables dependientes podrán estar en escala nominal u ordinal.

La resolución de estos problemas se ha abordado clásicamente usando técnicas de optimización para minimizar una determinada función de error, previo establecimiento por parte del investigador del tipo de modelo a aplicar. Pero en muchas ocasiones, el modelo a aplicar no es lineal y además suele presentar una alta dimensionalidad en las variables independientes, lo que complica considerablemente el proceso de modelado.

Una de las alternativas más utilizadas en los últimos años para la resolución de este tipo de problemas ha sido la aplicación de las llamadas Redes Neuronales Artificiales.

Hoy día, el ser humano se enfrenta a problemas que debe resolver con éxito ya que, el simple hecho de distinguir entre un perro y un gato en

una imagen es una tarea que un niño de preescolar haría fácilmente. Pero, sin embargo, esta misma tarea podría confundir a un ordenador. Para llegar a una solución a estos problemas, el ser humano hace uso de sus capacidades físicas con el fin de resolverlo.

La Inteligencia Artificial trabaja desde hace años en el campo de las Redes Neuronales Artificiales con el fin de simular las capacidades físicas humanas para resolver problemas, ya sean cotidianos o no. Así, están consideradas un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. Dentro de esta vía de investigación se sitúa una parte del grupo de investigación AYRNA de la Universidad de Córdoba, concretamente, en la utilización de Redes Neuronales Evolutivas para la resolución de problemas de modelado.

Con el objetivo de satisfacer las necesidades de este área de investigación, el grupo de investigación AYRNA ha desarrollado de forma teórica varios algoritmos y en concreto el que nos atañe, para su posterior implementación y prueba de rendimiento y de este modo, aportar conocimientos a la comunidad científica.



2 Definición del problema

En este apartado se tratará de mostrar detalladamente el problema al que se pretende dar solución con la realización del proyecto. En su desarrollo se realizará una identificación de necesidades y establecimiento de objetivos del proceso general del proyecto. Para poder abordar este fin, el apartado se dividirá en la *definición del problema real* (problema definido desde el punto de vista del usuario final) y la *definición del problema técnico* (problema definido desde el punto de vista del ingeniero).

2.1. Definición del problema real

Las redes neuronales y en particular los algoritmos de regresión o clasificación ordinal se están utilizando cada vez más ya que ofrecen muy buenos resultados en problemas reales de esta tipología. Lo que se pretende realizar con este proyecto es implementar los algoritmos desarrollados teóricamente por los directores del proyecto y realizar una evaluación de los mismos en cuanto a eficiencia y eficacia.

La realización de este proyecto implica una alta complejidad y conlleva la realización del estudio completo de todo lo relacionado con redes

neuronales computacionales, algoritmos de inteligencia artificial tanto de clasificación como de regresión, y dentro de éstos los relacionados con el tema de la ordinalidad en la variable dependiente asociada al clasificador. Obteniendo así un estudio completo de la resolución de problemas mediante algoritmos de regresión ordinal o también llamado clasificación ordinal realizados con redes neuronales.

2.2. Definición del problema técnico

Para llevar a cabo la identificación del problema técnico se empleará una técnica de ingeniería de las más utilizadas en proyectos de éste ámbito denominada PDS (Product Design Specification). Ésta técnica permite realizar un análisis de los principales condicionantes técnicos del problema mediante la respuesta de una serie de cuestiones básicas e indispensables.

2.2.1. Funcionamiento

Los algoritmos de regresión ordinal son un método matemático que modela la relación entre una variable dependiente Y en escala ordinal, y las variables independientes X . Desde la perspectiva de una escala ordinal, a las clases se le asignan números para indicar el grado relativo en la que los objetos poseen ciertas características.

Una regresión ordinal nos permite determinar si un objeto tiene más o menos cantidad de cierta característica que algún otro objeto. De manera que, una escala ordinal nos indica la posición relativa, pero no la magnitud de las referencias entre los objetos con respecto a la variable dependiente del problema.

En investigación de mercados las escalas ordinales se utilizan para medir las actitudes, opiniones, percepciones y preferencias relativas.

2.2.2. Entorno

El entorno se define como el conjunto de aspectos o propiedades que rodean al problema y que aun presentándose de manera externa al mismo, pueden influir en el planteamiento de una solución, puesto que pueden afectar al sistema software desarrollado para tal fin.

En el análisis del entorno del proyecto que se pretende desarrollar se tendrán en cuenta los siguiente puntos de vista: entorno de programación, entorno software, entorno hardware y entorno de usuario.

- **Entorno de programación:** la implementación del algoritmo que se pretende desarrollar en el presente proyecto se realizará haciendo uso del lenguaje propio que tiene el programa de desarrollo matemático Matlab, así como de su interfaz gráfica y su editor de textos. El motivo de su uso se detallará más adelante.
- **Entorno software:** para el funcionamiento del algoritmo se hará uso de Matlab 2010a junto con el Toolbox propietario basado en el manejo y uso de redes neuronales, *nnet*.
- **Entorno hardware:** también conocido como entorno físico o de trabajo, hace referencia a las características del sistema informático en el que se ejecutará el algoritmo, así como el ambiente que lo rodea. El algoritmo y todas las pruebas de rendimiento se harán en un ordenador portátil personal, sin ningún requisito en especial.
- **Entorno de usuario:** los usuarios que puedan utilizar el código desarrollado o que podrán entender los resultados obtenidos serán, por lo general, investigadores que tengan conocimiento en Redes

Neuronales y del lenguaje de programación de Matlab. Por lo tanto, bajo este supuesto sólo serán necesarias ciertas nociones básicas sobre la temática para el entendimiento del proyecto en general.

2.2.3. Vida esperada

La vida esperada de un producto software puede definirse como el tiempo de vida estimado durante el cual puede realizarse una aplicación útil del mismo, en nuestro caso no es exactamente lo mismo ya que es una implementación que servirá para un conjunto específico de investigadores y no es un producto software como tal, de todos modos la definición es válida. Esta estimación es difícil de realizar al influir en la misma numerosos factores.

Al tratarse de un algoritmo basado en estudios de investigación, se estima que la vida esperada del mismo sea relativamente corta, pero teniendo en cuenta que, si el algoritmo se encuentra bien realizado, lo único que podrá tener es pequeñas modificaciones que lo mejoren en el sentido que sea. De todos modos, la implementación propuesta garantiza la escalabilidad del rendimiento del algoritmo. Por tanto, es lógico pensar que el presente proyecto pueda ser utilizado como punto de partida para trabajos posteriores.

2.2.4. Ciclo de mantenimiento

El ciclo de mantenimiento se identifica con el conjunto de modificaciones que será necesario realizar para que una determinada aplicación pueda hacer frente a diferentes circunstancias que puedan surgir o a nuevas exigencias procedentes del usuario final o del propio sistema.

A medio plazo se puede apreciar la necesidad de realizar un mantenimiento, fruto del surgimiento de nuevas necesidades o modificaciones

propuestas por el grupo de investigación AYRNA, el cual continuará realizando futuros proyectos en este ámbito.

2.2.5. Competencia

Puesto que este algoritmo tiene fines investigadores ya que se basa en un modelo teórico desarrollado por el grupo de investigación, no existe en la actualidad ninguno igual y de ahí la necesidad de su implementación para comprobar su correcto funcionamiento y eficacia.

2.2.6. Aspecto externo

La apariencia externa de una aplicación hace referencia no solamente al aspecto visual que tiene el usuario de la misma durante su ejecución, sino que también es necesario considerar la presentación física del sistema, los mecanismos de instalación proporcionados junto al mismo y los manuales que pueden acompañarlo.

Por todo esto, se ha decidido añadir el algoritmo al Toolbox del software de desarrollo matemático e investigador de Matlab, que ofrece una integración completa en un software de gran prestigio y usable en el ámbito científico y de investigación sobretodo cuando es necesario utilizar cálculos matriciales.

Para el almacenamiento de todo lo relacionado con el proyecto se hará uso del CD-ROM debido a la portabilidad, bajo coste, capacidad de almacenamiento, resistencia, seguridad y su uso generalizado entre los usuarios.

La implementación del algoritmo irá acompañada de un manual de usuario que se podrá consultar desde la propia ayuda de la aplicación de Matlab y donde se explicará, según el formato que proporciona Matlab en sus códigos y ayudas, el manejo del algoritmo de la forma más sencilla, para que su lectura sea más agradable y productiva.

Tanto el manual de usuario, el manual técnico y el manual de código se incluirán impresos y en formato PDF (Portable Document Format).

2.2.7. Estandarización

Con referencia a la estandarización en la programación, se procurará seguir los estándares más comunes aprendidos en la carrera tales como, correcta indentación de código, uso generalizado y especializado de comentarios de código, modularización del código y de las funciones, etc.

2.2.8. Calidad y fiabilidad

La calidad y la fiabilidad son dos factores muy importantes que hay que tener en cuenta en el desarrollo de cualquier aplicación, puesto que es necesario proporcionar al usuario final garantías que le permitan depositar su confianza en el producto, ya que, en caso contrario, podrían surgir reacciones adversas a su utilización.

La calidad de cualquier aplicación se asocia al hecho de que durante su ejecución no se produzcan errores que induzcan a su terminación irregular. La fiabilidad, por su parte, hace referencia a la capacidad del sistema para proporcionar datos reales, asegurando que las acciones realizadas durante el procesamiento resulten correctas y se lleven a cabo de manera óptima.

De este modo, la calidad del proyecto será directamente evaluada por el alumno proyectista junto con la colaboración de los directores del proyecto. A la finalización del mismo, será evaluada por el tribunal del proyecto.

La fiabilidad será un factor importante, ya que los datos o informes que obtiene el usuario deberán ser totalmente correctos, por lo que se realizarán pruebas que intenten minimizar el número de errores producidos.

Destacar que, a priori, los únicos errores que se deberían cometer en el sistema serán aquellos derivados de un uso incorrecto del mismo.

2.2.9. Programación de tareas

La programación de tareas se define como el conjunto de etapas y actividades que constituyen el proceso de desarrollo de una aplicación. A continuación se describirán las diferentes etapas en las que se puede organizar el proceso de desarrollo del algoritmo que se ha desarrollado:

1. Estudio técnico de conceptos y conocimientos generales sobre Redes Neuronales, Algoritmos de Regresión Ordinal y Nominal.
2. Estudio teórico del modelo de programación en Matlab.
3. Estudio de antecedentes sobre Algoritmos de Regresión Ordinal en Redes Neuronales.
4. Implementación del método desarrollado por los directores de proyecto en Matlab.
5. Realización de pruebas para la comprobación de la eficacia del algoritmo implementado.
6. Análisis de los resultados de las pruebas.
7. Obtención de conclusiones y presentación de las futuras mejoras.

En todo momento habrá una tarea que se dará durante todo el proyecto que es la realización de la documentación del proyecto.

2.2.10. Pruebas

Las pruebas se definen como el conjunto de acciones y datos que son utilizados para poder depurar la aplicación desarrollada, además del medio para demostrar la funcionalidad de la misma y su utilidad. Durante la

implementación, el algoritmo junto con el software en general del Toolbox *nnet* de Matlab será sometido a diversas pruebas para garantizar la corrección del software. En la fase de experimentación y resultados se someterá al algoritmo a pruebas de ejecución para estudiar su funcionamiento de acuerdo a los objetivos del proyecto.

2.2.11. Seguridad

El algoritmo a desarrollar será utilizado y distribuido por el grupo AYRNA de acuerdo a sus consideraciones, por lo que el desarrollo de mecanismos de seguridad contra copias o distribuciones no permitidas no tendrá sentido.

2.2.12. Licencia

Aunque el algoritmo está desarrollado para el software matemático Matlab no quiere decir que éste vaya a tener licencia privativa como el programa tiene, sino que se encontrará bajo una licencia libre GNU General Public License (GPL), por lo que podrá ser usado, modificado y distribuido libremente por cualquier usuario.



3 Objetivos

En el presente capítulo se expondrán todos los objetivos funcionales que se pretenden alcanzar con el desarrollo de este proyecto.

El propósito de este proyecto es desarrollar un algoritmo de regresión ordinal basado en redes neuronales, mediante la utilización de una función de *ranking* $f(x)$ formada por la suma ponderada de un conjunto de funciones de base (incluyendo funciones de tipo sigmoide, unidad producto, funciones de base radial, etc.).

En concreto, los objetivos perseguidos son los siguientes:

1. Realizar un estudio teórico sobre las temáticas relacionadas con el proyecto, incluyendo la regresión ordinal, la clasificación tradicional y los algoritmos de descenso por gradiente.
2. Implementar el algoritmo iRPROP+ (el cuál es una mejora del algoritmo RPROP), utilizando como base la implementación incluida en el paquete de redes neuronales (**nnet**) del software Matlab [8].
3. Desarrollar un algoritmo de regresión ordinal basado en Redes Neuronales. Este objetivo engloba varios sub-objetivos:

- a)* Modificación del modelo funcional estándar de redes neuronales para clasificación. Esto supone considerar una red neuronal de una sola neurona en capa de salida y aplicar una transformación de la salida que aproxime un valor de probabilidad de pertenencia a cada una de las clases. Esta transformación la basaremos en la Regresión Logística Ordinal estándar (*Proportional Odd Model*, [4]).
 - b)* Modificación de la función de error utilizada como base en el algoritmo iRProp+.
 - c)* Evaluar el algoritmo en un conjunto de bases de datos de prueba, comparar los resultados con los obtenidos por otros clasificadores ordinales y reformular la hipótesis y/o el algoritmo inicial, intentado conseguir (siempre que fuese posible) unos resultados competitivos.
4. Implementar una interfaz gráfica que facilite la utilización del algoritmo implementado.



4 Antecedentes

En este capítulo se pretende concretar el punto de partida del Proyecto. Este conocimiento base permitirá determinar en los siguientes capítulos las especificaciones del sistema a desarrollar.

En primer lugar, se hará una breve exposición acerca del grupo de investigación AYRNA, sus miembros, sus objetivos y las actividades e investigaciones que realiza.

Posteriormente, se analizará en profundidad los conceptos de redes neuronales y clasificación y regresión ordinal. Haciendo especial hincapié en los algoritmos previos de estudio en los que se basará el posterior algoritmo a implementar. También se hará un análisis de los parámetros de medida que se utilizarán para analizar el rendimiento de un clasificador realizando una explicación detallada de los mismos.

Por último, se analizará el entorno y lenguaje de programación del cual se hará uso, así como del toolbox o librería que facilitará la implementación del algoritmo y optimizará computacionalmente el sistema.

4.1. Grupo de Investigación AYRNA

El grupo de investigación de *Aprendizaje y Redes Neuronales Artificiales*, **AYRNA** (TIC-148 de la Junta de Andalucía) fue creado en 1994 por un pequeño grupo de investigadores interesados en el campo de las Redes Neuronales Artificiales (RNAs).

Durante los últimos años, el grupo ha diversificado sus áreas de interés, trabajando en la resolución de distintos problemas mediante técnicas de *soft computing* (redes neuronales artificiales, algoritmos evolutivos y otras meta-heurísticas).

En la actualidad, el grupo está formado por 9 investigadores doctores y 8 no doctores, siendo el investigador principal el Dr. César Hervás Martínez.

El grupo de investigación pertenece al Departamento de Informática y Análisis Numérico de la Universidad de Córdoba y tiene su sede en la 3ª planta del edificio *Albert Einstein* del Campus Universitario de Rabanales.

Actualmente, presenta las siguiente líneas de trabajo concentradas en:

- Redes Neuronales Evolutivas.
- Modelos Híbridos de Redes de Unidades Producto (UPs), Sigmoides (Perceptrón Multicapa, MLP), Funciones de Base radial (RBFs), etc.
- Algoritmos Híbridos en Computación Evolutiva.
- Funciones Multiobjetivo para Modelos de Redes y Modelos de Computación Evolutiva.

- Aplicaciones en Cinética Química, Predicción de Crecimiento Microbiano, Predicción de Polen, Teledetección, etc.
- Programación Evolutiva.
- Clasificación no balanceada.
- Clasificación Ordinal.
- Aplicaciones en biomedicina, transplantes hepáticos, etc.

4.2. Redes Neuronales

Una de las alternativas más utilizadas en los últimos años para la resolución de este tipo de problemas ha sido la aplicación de las llamadas redes neuronales artificiales. Ésta técnica de modelado es enormemente flexible y suele producir buenos resultados. Se fundamenta en la emulación de los sistemas nerviosos biológicos, combinando una gran cantidad de elementos simples de procesamiento altamente interconectado, cuya capacidad de cómputo se desarrolla mediante un proceso adaptativo de aprendizaje. Los elementos simples de procesamiento suelen denominarse neuronas y se agrupan en capas. Cada una de estas neuronas se encuentra interconectada con las neuronas de la capa anterior. Tradicionalmente, una red neuronal posee una capa de entrada (por la que se introducen los valores de las variables observadas), una o más capas ocultas (que realizan el procesamiento de dichos valores) y una capa de salida (en la que se pueden leer los valores predichos de salida). La Figura 4.1 muestra gráficamente el aspecto de una RNA.

Con el objetivo de lograr el aprendizaje de las redes neuronales, antes de enfrentarlas a un problema real, se requiere un proceso previo de entrenamiento que nos asegure que van a tener una buena capacidad de predicción para todos los casos, es decir, que el modelado sea eficaz.

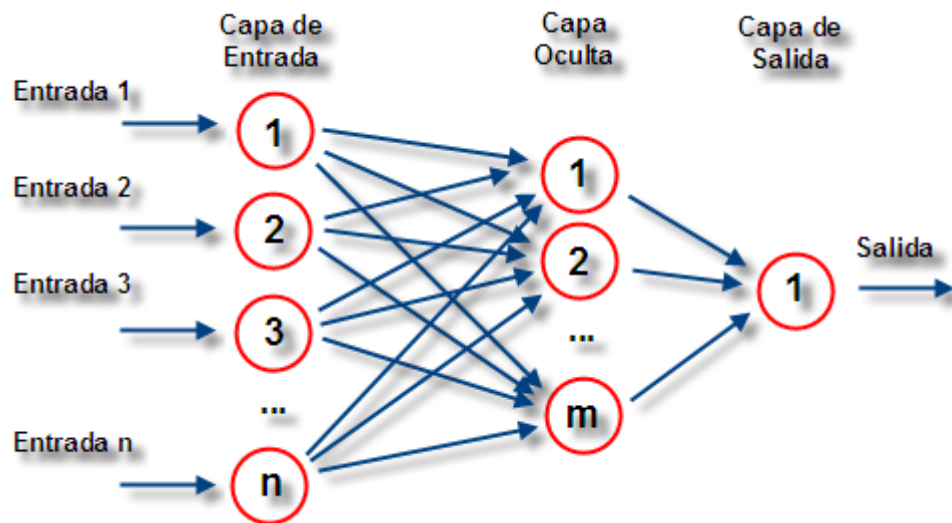


Figura 4.1: Red Neuronal Artificial

La Figura 4.2 muestra gráficamente el funcionamiento de una RNA.

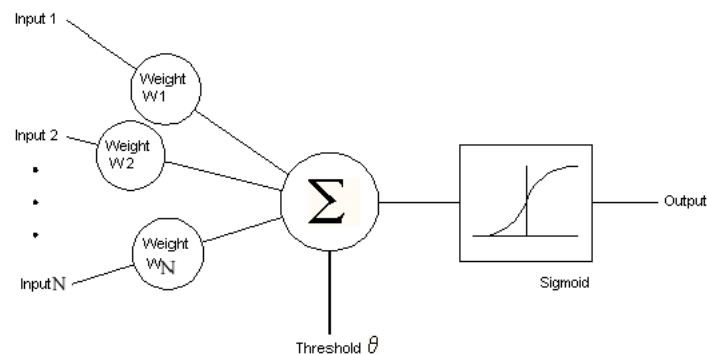


Figura 4.2: Funcionamiento de una Red Neuronal Artificial

En 1988, el Estudio sobre Redes Neuronales realizado por DARPA¹ listaba varias aplicaciones con redes neuronales. Dicho estudio sirvió para que saliesen otras aplicaciones comerciales, incluyendo un pequeño re-

¹Defense Advanced Research Projects Agency (Agencia de Investigación de Proyectos Avanzados de Defensa) es una agencia del Departamento de Defensa de los Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar.

conocedor de palabras, un monitor de procesos o un clasificador de sonar.

Las Redes Neuronales se han aplicado, además, en otros campos desde que DARPA escribió su informe. La siguiente lista contiene algunos de los campos en los que se utilizan actualmente las redes neuronales:

- Aeromodelismo espacial: simulación de vuelos, detección de fallos en componentes, etc.
- Automoción: sistemas de guía automáticos, análisis de garantías, etc.
- Banca: evaluación de tarjetas de crédito, etc.
- Defensa: búsqueda de objetivos, compresión de datos, extracción de características y supresión de ruidos, etc.
- Electrónica: predicción de códigos de secuencia, chip en circuitos integrados, etc.
- Entretenimiento: animaciones, efectos especiales, etc.
- Finanzas: análisis de uso de créditos, predicción de precios, etc.
- Manufacturías: control de procesos, diseño de productos, etc.
- Medicina: análisis de células cancerígenas, optimización de tiempos en trasplantes, etc.
- Robótica: control de trayectorias, controladores y manipuladores, sistemas de visión, etc.

4.2.1. Regresión ordinal

La regresión ordinal (RO) permite asignar una métrica óptima a los regresores discretos de un modelo de regresión múltiple. Se trata, en síntesis, de elegir la recodificación de los predictores de acuerdo con una

métrica ordinal tal, que se optimice el ajuste del modelo. De este modo, se extrae de cada regresor su mayor capacidad predictiva posible mediante una recodificación óptima de sus posibles valores en una nueva escala de naturaleza ordinal.

4.2.2. Algoritmo Resilient Backpropagation (RPROP)

Resilient backpropagation (RPROP) [6] es una técnica de optimización robusta basada en el gradiente, que ha sido comúnmente utilizada para el entrenamiento de redes neuronales artificiales. Se basa en el uso de un valor de velocidad de avance del algoritmo para la actualización de cada parámetro del modelo.

RPROP es una técnica ampliamente utilizada para el entrenamiento supervisado de redes neuronales artificiales tipo perceptrón multicapa, cuyo proceso de búsqueda es guiado por la primera derivada de la función $f(x)$; en este caso, $f'(x)$ es una medida de la diferencia entre la salida propuesta por la red neuronal y el valor esperado. RPROP difiere de la técnica clásica de propagación, hacia atrás, del error (o algoritmo *backpropagation*) en que las derivadas parciales de la función error sólo son usadas para determinar el sentido en que deben ser corregidos los pesos de la red pero no las magnitudes de los ajustes. Los algoritmos basados en *backpropagation* modifican los valores de los parámetros proporcionalmente al gradiente de la función de error, de tal forma que en regiones donde el gradiente tiende a ser plano el algoritmo avanza lentamente; esta modificación se hace con RPROP a través de un único parámetro que controla la velocidad de avance del algoritmo.

Para comprender mejor el funcionamiento que tiene, las ecuaciones 4.1 y 4.2 muestran el funcionamiento de esta técnica:

$$\Delta w_{ij}(t) = \begin{cases} -\Delta p_{ij}, & \text{si } \frac{\partial E}{\partial w_{ij}} > 0 \\ +\Delta p_{ij}, & \text{si } \frac{\partial E}{\partial w_{ij}} < 0 \\ 0, & \text{si } \frac{\partial E}{\partial w_{ij}} = 0 \end{cases} \quad (4.1)$$

$$\Delta p_{ij}(t) = \begin{cases} \alpha^+ \cdot \Delta w_{ij}(t-1), & \text{si } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) > 0 \\ \alpha^- \cdot \Delta w_{ij}(t-1), & \text{si } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) < 0 \\ \Delta w_{ij}(t-1), & \text{si } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) = 0 \end{cases} \quad (4.2)$$

Siendo,

w_{ij} los pesos o coeficientes del modelo,
 t el instante de tiempo,
 p_{ij} la actualización o cambio de los pesos,
 E la entropía,
 α el rango de aprendizaje.

Por otra parte, los parámetros obtenidos experimentalmente que se utilizan son,

$$\begin{aligned} \alpha^+ &= 1,2, \\ \alpha^- &= 0,5, \\ \Delta w(0) &= 0,5, \\ \Delta w(t)_{max} &= 50, \\ \Delta w(t)_{min} &= 0 \end{aligned}$$

4.2.3. Algoritmo iRPROP+ (RPROP mejorado)

La variante de RPROP, iRProp+ [7], añade un paso de vuelta atrás al algoritmo, que permite evitar los óptimos locales. La idea es que, cuando el cambio en un parámetro de la red haya producido un aumento del valor de la función de error, vuelva al estado anterior antes de producirse dicho cambio. De esta forma, se tienen las ecuaciones:

$$\Delta w_{ij}(t) = \begin{cases} \alpha^+ \cdot \Delta w_{ij}(t-1), & \text{si } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) > 0 \\ \Delta w_{ij}(t-1) - \Delta w_{ij}(t-2), & \text{si } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) < 0 \\ & \text{y } E(t) > E(t-1) \\ \Delta w_{ij}(t-1), & \text{si } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) = 0 \end{cases}$$

4.2.4. Redes Neuronales basadas en el modelo Proportional Odd Model (POM)

Para poder utilizar el algoritmo iRProp+ para regresión ordinal, nos vamos a basar en el modelo *Proportional Odd Model* (POM) [4].

Este modelo de regresión ordinal entra dentro del grupo de modelos de umbral, que se basan en suponer que la respuesta ordinal está asociada a una variable artificial medida en escala continua y modelada mediante intervalos de clase sobre la recta real. La mayoría de estos modelos se pueden representar mediante una función de rango $f(x)$ la cual transforma el vector de variables de entrada en valores de la recta real y un conjunto de umbrales $\{\beta_0, \dots, \beta_J\}$ que delimitan las categorías en esta recta. Diferentes formas de esta función de rango proporcionan modelos de clasificación ordinal como los que se han mencionado en el capítulo de Introducción.

Bajo un punto de vista probabilístico, el modelo POM [4] es el método estadístico básico de regresión ordinal y puede considerarse también como un modelo de umbral, donde $f(x)$ es una combinación lineal ponderada de las variables de entrada.

La idea fundamental de este proyecto es basarnos en dicho modelo probabilístico para poder definir una red neuronal para regresión ordinal y utilizar una modificación del algoritmo iRProp+ (por su mayor eficiencia

y eficacia) para optimizar los parámetros del modelo.

4.2.5. Medidas de rendimiento

Para comprobar que los resultados obtenidos después del entrenamiento y la simulación de una red neuronal artificial son buenos, es decir, que los resultados obtenidos al aplicar un algoritmo específico, se corresponden con un buen clasificador; existen unas medidas que se utilizan para poder ver la calidad o eficiencia de los resultados.

Algunos de estas medidas se mencionarán y explicarán a continuación. Aunque existen más, estas son las más relevantes para el tema concreto que se está tratando que es la regresión ordinal mediante redes neuronales artificiales.

4.2.5.1. Matriz de confusión

Una matriz de confusión contiene información acerca de las clasificaciones actuales o predichas realizadas por un sistema de clasificación. El rendimiento de tales sistemas se evalúa, normalmente, usando los datos de esta matriz. La Tabla 4.1 muestra la matriz de confusión para un clasificador de dos clases.

| | | Predicho | |
|--------|----------|----------|----------|
| | | Negativo | Positivo |
| Actual | Negativo | a | b |
| | Positivo | c | d |

Tabla 4.1: Matriz de confusión

Las entradas de la matriz de confusión tienen el siguiente significado en el contexto de nuestro estudio:

- a es el número de predicciones **correctas** en la que una instancia es negativa.
- b es el número de predicciones **incorrectas** en la que una instancia es positiva.
- c es el número de predicciones **incorrectas** en la que una instancia es negativa.
- d es el número de predicciones **correctas** en la que una instancia es positiva.

En el campo de la Inteligencia Artificial, una matriz de confusión es una herramienta de visualización usada normalmente en aprendizaje supervisado (en aprendizaje no supervisado se le suele llamar matriz de coincidencias). Uno de los beneficios de usar una matriz de confusión es, que es muy fácil ver si el sistema distingue entre las dos clases.

Los siguientes términos son importantes para la comprensión de los valores de la matriz de confusión del ejemplo:

La precisión (AC - accuracy) es la proporción del número total de predicciones que fueron correctas. Se calcula usando la siguiente ecuación:

$$AC = \frac{a + d}{a + b + c + d} \quad (4.3)$$

Un verdadero positivo (TP - true positive) es la proporción de casos positivos que se identificaron correctamente, se calcula mediante la siguiente ecuación:

$$TP = \frac{d}{c + d} \quad (4.4)$$

Un falso positivo (FP - false positive) es la proporción de casos negativos que se clasificaron incorrectamente como positivos, se calcula mediante la siguiente ecuación:

$$FP = \frac{b}{a + b} \quad (4.5)$$

Un verdadero negativo (TN - true negative) se define como la proporción de casos negativos que se clasificaron correctamente, se calcula mediante la siguiente ecuación:

$$TN = \frac{a}{a + b} \quad (4.6)$$

Un falso negativo (FN- false negative) es la proporción de casos positivos que fueron incorrectamente clasificados como negativos, se calcula mediante la siguiente ecuación:

$$FN = \frac{c}{c + d} \quad (4.7)$$

Por último, la proporción (P) es la proporción de los casos positivos predichos que fueron correctos, se calcula mediante la siguiente ecuación:

$$P = \frac{d}{b + d} \quad (4.8)$$

4.2.5.2. Ratio Correctamente Clasificado (Correctly Classified Rate (CCR))

Cuando se tiene más de dos clases en el modelo se calcula la precisión o CCR, que es el número de elementos en la diagonal principal de la matriz de confusión, es decir, la suma del número de predicciones correctas en la que una instancia es negativa más el número de predicciones correctas en la que una instancia es positiva, dividido por el número total de elementos de la matriz.

Anteriormente se ha visto como la precisión o AC, pero generalizando para este caso en concreto, se puede decir que corresponde a la siguiente ecuación:

$$CCR = \frac{\text{nº de bien clasificados}}{\text{nº total de patrones}} \quad (4.9)$$

4.2.5.3. Error Absoluto Medio (Mean Absolute Error (MAE))

En estadística, el error absoluto medio es una cantidad usada para medir como de buenos son los pronósticos o predicciones de los resultados. El error absoluto medio (MAE) viene dado por la siguiente ecuación:

$$MAE = \frac{1}{n} \sum_{i=1}^n |f_i - y_i| = \sum_{i=1}^n |e_i| \quad (4.10)$$

Como su propio nombre indica, el error absoluto medio es una media de los errores absolutos, $e_i = f_i - y_i$, donde f_i es la predicción e y_i es el valor verdadero. Existe una notación alternativa en la que se puede incluir frecuencias relativas como factores de peso.

El error absoluto medio es una medida común de pronóstico de errores en análisis temporales de series, donde el término “error absoluto medio” se utiliza en algunas ocasiones confundiéndolo con la definición más estandarizada de desviación media absoluta.

El MAE mide la magnitud media de los errores en un conjunto de pronóstico, sin considerar su dirección. Así, se puede decir que mide la precisión para variables continuas.

También es una medida específica para clasificación ordinal. Donde se realiza el sumatorio de los errores cometidos por un clasificador en valor absoluto asignando una etiqueta numérica a cada clase. Para ahorrar coste computacional, el error absoluto medio también se puede calcular en base a la matriz de confusión explicada anteriormente, multiplicando ésta por una matriz de costes del estilo a la mostrada:

$$MP = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \quad (4.11)$$

Multiplicando elemento a elemento, no de forma matricial, la matriz de confusión por esta otra matriz de pesos, resultará una matriz de errores absolutos. Dándole un peso proporcional a la distancia entre aquellos

elementos mal clasificados. En este momento, haciendo el sumatorio de todos los elementos de la matriz y dividiéndolo por el número total de elementos, se obtendrá el MAE. La siguiente ecuación muestra lo explicado anteriormente:

$$MAE = \frac{\sum (MC \cdot MP)}{TOTAL} \quad (4.12)$$

4.2.5.4. Error Cuadrático Medio (Mean Square Error (MSE))

En estadística, el error cuadrático medio o MSE de un estimador es uno de los posibles caminos para cuantificar la diferencia entre un estimador y el valor verdadero de la cantidad que se va a estimar, es decir, es la media aritmética de los cuadrados de las desviaciones del estimador $\hat{\theta}$ respecto al valor verdadero del estadístico que se trata de estimar.

Se diferencia de la varianza en que, en ésta, las desviaciones son con respecto a la media aritmética. Lógicamente, cuando el estimador sea insesgado, el error cuadrático medio será igual a la varianza.

El MSE de un estimador con respecto al parámetro estimado $\hat{\theta}$ se define como:

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] \quad (4.13)$$

El MSE también se puede calcular sumando la varianza al cuadrado del sesgo o *bias* del estimador:

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + \left(Bias(\hat{\theta}, \theta) \right)^2 \quad (4.14)$$

4.3. Matlab

MATLAB es un lenguaje de computación de alto nivel y un entorno interactivo para desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico. Con MATLAB, se puede resolver problemas de cálculo más rápidamente que con lenguajes de programación tradicionales, tales como C, C++ o Fortran.

El lenguaje de MATLAB incluye operaciones vectoriales y matriciales que son fundamentales para resolver los problemas científicos y de ingeniería. Agiliza tanto el desarrollo como la ejecución.

Con el lenguaje de MATLAB, se puede programar y desarrollar algoritmos más rápidamente que con los lenguajes tradicionales porque ya no hay que realizar tareas administrativas de bajo nivel, tales como declarar variables, especificar tipos de datos o asignar memoria. En muchos casos, MATLAB elimina la necesidad de bucles “for”, o condiciones explícitas “if”. Por ello, una línea de código de MATLAB generalmente reemplaza a varias líneas de código en C/C++, Fortran o JAVA.

Al mismo tiempo, MATLAB ofrece todas las características de los lenguajes de programación tradicionales, que incluyen operadores aritméticos, control de flujo, estructuras de datos, tipos de datos, programación orientada a objetos (OOP) y depuración.

MATLAB permite ejecutar comandos o grupos de comandos uno a uno, sin compilar ni enlazar, y repetir su ejecución hasta lograr la solución óptima.

También ofrece todas las funciones gráficas necesarias para visualizar datos de ingeniería y científicos. Incluye funciones de representación de diagramas bidimensionales y tridimensionales, visualización de volumen tridimensional, herramientas para crear diagramas en forma interactiva y la posibilidad de exportar los resultados a los formatos gráficos más conocidos. Se puede personalizar los diagramas añadiendo varios ejes, cambiando los colores de las líneas y marcadores, añadiendo anotaciones, ecuaciones LaTeX, leyendas y trazando formas.

MATLAB es un software multiplataforma y eso facilita que cualquiera pueda hacer desarrollos en Matlab en su propio sistema operativo.

4.3.1. Toolbox *nnet* (Neural Network Toolbox)

Neural Network Toolbox proporciona herramientas para el diseño, implementación, visualización y simulación de redes neuronales artificiales. Las redes neuronales se utilizan para aplicaciones donde el análisis formal es difícil o imposible, tales como reconocimiento de patrones o los sistemas no lineales de identificación y control. El toolbox soporta redes con conexiones hacia adelante o “feedforward”, redes de base radial, redes dinámicas, mapas auto-organizados u otros paradigmas de red probadas.

Las redes neuronales se componen de elementos simples funcionando en paralelo. Estos elementos están inspirados en los sistemas nerviosos biológicos. Al igual que en la naturaleza, las conexiones entre los elementos determinan en gran medida la función de red. Se puede entrenar una red neuronal para realizar una función en particular, mediante el ajuste de los valores de las conexiones (pesos) entre los elementos.

Por lo general, las redes neuronales se ajustan o se entrenan de modo que una determinada entrada conducirá a una salida específica.

4.3.2. Interfaces gráficas con Matlab

Matlab proporciona la posibilidad de crear Interfaces Gráficas de Usuario (GUI²) de varias formas, por ejemplo, a través de librerías de interfaz externas que permiten implementarlas escritas en C/C++, JAVA y Fortran o utilizando su aplicación gráfica para la creación de interfaces o realizando la interfaz programándola directamente a través de ficheros MAT.

Una Interfaz Gráfica de Usuario es una exposición gráfica en una o más ventanas que contiene controles, componentes de llamada, que permite al usuario realizar tareas interactivas con un sistema. El usuario de la GUI no tiene porque crear un script o escribir comandos en la línea de comandos para realizar las tareas. Ésto ayuda a los usuarios a comprender los detalles de cómo se realizan las tareas.

Los componentes que puede incluir una GUI pueden ser tales como menús, barras de herramientas, botones de varios tipos, listas de selección, etc. Las GUIs creadas usando las herramientas de MATLAB pueden realizar cualquier tipo de computación, como por ejemplo, leer y escribir en ficheros externos, comunicarse con otras GUIs o mostrar datos como gráficos.


Como se ha comentado antes, se podrán realizar GUIs con MATLAB de dos maneras posibles:

- Usando la herramienta para creación de interfaces gráficas que contiene llamada GUIDE (GUI Development Environment)³.
- O creando los ficheros de código fuente que generarán las GUIs como funciones o scripts (a esta técnica se la denomina *construcción*

²Graphical User Interface.

³Entorno de Desarrollo para GUI. La aplicación gráfica es muy simple y se asemeja mucho a la aplicación QtCreator para creación de interfaces gráficas con C++.

programada de GUIs).



5 Restricciones

En este capítulo se expondrán todas las restricciones, o factores limitativos, existentes en el ámbito del diseño y que condicionan la elección de una u otra alternativa. Los factores limitativos pueden estructurarse en dos grupos:

- **Factores dato:** son aquellos que no pueden ser modificados durante el transcurso del proyecto, como puede ser el presupuesto económico asignado al proyecto o la duración estimada del mismo.
- **Factores estratégicos:** representan variables de diseño que permiten la elección entre diferentes alternativas por parte del ingeniero. En función de la opción escogida, podrá alterarse el proceso de desarrollo y el propio producto final obtenido, con lo que resultará necesario analizar las posibilidades existentes en las primeras etapas del proceso.

5.1. Factores dato

En el desarrollo de este proyecto se van a considerar los siguientes factores dato impuestos por el tipo de proyecto:

- **Restricciones humanas:** al ser éste un proyecto final de carrera, este factor lo condiciona el director de proyecto (perteneciente al grupo de investigación AYRNA), restringiendo el proyecto a un número determinado de personas. En nuestro caso, un solo alumno para el desarrollo del mismo.
- **Restricciones temporales:** estará condicionado al cumplimiento de los objetivos previamente establecidos, aunque por lo general, se suele establecer limitaciones con el fin de no demorar en exceso la presentación del Proyecto. En este caso, el tiempo de elaboración de este Proyecto se espera que no sobrepase la convocatoria de septiembre del curso 2010/2011, a expensas de no sufrir contratiempos.
- **Restricciones hardware:** dado a la cantidad de cálculos que realizarán los algoritmos desarrollados, desde el cliente nos imponen ciertas restricciones hardware (como la potencia de los equipos en los que se utilizará el sistema) a tener en cuenta a la hora de elegir ciertos parámetros en el futuro de diseño de la aplicación.
- **Restricciones software:** según las características del proyecto y el fin que tendrá, meramente investigador. No se impone un software específico ni un lenguaje de programación en especial, pero sí que tenga la suficiente potencia para desarrollar el proyecto.

5.2. Factores estratégicos

Los principales factores estratégicos que afectan al presente proyecto, así como los diferentes motivos que justifican la elección realizada en cada restricción, son los que se comentan a continuación:

- Los algoritmos que se desea desarrollar deberán estar bien modularizados, de manera que permita de manera fácil y con la mayor agilidad posible la realización de futuras modificaciones y ampliaciones que puedan considerarse necesarias.

- Los algoritmos se desarrollarán en el entorno de desarrollo Matlab, porque la potencia que ofrece este software de desarrollo matemático y la utilidad de tener a disposición un Toolbox sobre redes neuronales que facilitan la elaboración del proyecto.
- Para la elaboración de la documentación en \LaTeX se utilizará el editor Texmaker, ya que es de libre distribución.
- Para la elaboración de cualquier tipo de diagramas se hará uso de la herramienta Día, disponible en libre distribución para los distintos sistemas operativos.



6 Recursos

En este capítulo se expondrán de forma clara y concisa los recursos humanos y materiales necesarios para este proyecto. Los recursos se definen como aquellos medios de los que se dispone para abordar el proceso de desarrollo del proyecto. El análisis de los recursos existentes se realizará atendiendo a una doble perspectiva:

- **Recursos humanos:** son aquellos que están constituidos por toda persona que intervenga en el proceso de desarrollo del sistema.
- **Recursos materiales:** son aquellos que pueden definirse como el conjunto de todas las entidades no animadas que permiten realizar el proceso de desarrollo de la aplicación, así como la generación de la documentación relativa a la misma.

6.1. Recursos humanos

El conjunto de personas que intervendrán durante el proceso de desarrollo del presente proyecto se muestran a continuación:

- **Directores:**

- Prof. Dr. Pedro Antonio Gutiérrez Peña.

Profesor Ayudante Doctor del Dpto. de Informática y Análisis Numérico y miembro investigador del grupo AYRNA.

- Prof. Dr. César Hervás Martínez.

Catedrático del Dpto. de Informática y Análisis Numérico y director del grupo AYRNA.

Los directores se encargarán de supervisar las tareas de desarrollo para comprobar que los resultados obtenidos se corresponden con los requisitos planteados. Además, facilitarán aquellos recursos materiales que resulten necesarios para abordar con éxito el proceso de desarrollo.

■ **Autor:**

- Raúl Pérula Martínez.

Ingeniero Técnico Informático de Sistemas.

6.2. Recursos materiales


6.2.1. Recursos software

- Sistema operativo Ubuntu 10.04.
- Lenguaje de programación y herramienta MATLAB 2010a.
- Lenguaje \LaTeX para la realización de la documentación.
- Texmaker 2.1 como editor de documentos \LaTeX .
- Programa de edición de diagramas, dia 0.97.1.

6.2.2. Recursos hardware

Equipo portátil con las siguiente características técnicas:

- Procesador Intel Core 2 Duo T7300 (Santa Rosa) a 2 GHz (4 MB memoria caché L2).
- 2048 MB de memoria RAM.
- Disco duro de 120 GB.



7 Especificación del sistema

En los capítulos anteriores se ha proporcionado una visión general del problema a resolver. En este capítulo, dedicado a la especificación, se va a detallar el funcionamiento del algoritmo de regresión ordinal basado en redes neuronales artificiales y se realizará un análisis del coste computacional de sus fases atendiendo a posibles optimizaciones.

7.1. Algoritmo de Regresión Ordinal ORNNNet

En esta sección se explicará el funcionamiento detallado del algoritmo a implementar.

El modelo de la red neuronal se basa en una capa de entrada, dos capas ocultas y una capa de salida. La capa de entrada tomará los valores de entrenamiento de entrada (x_1, x_2, \dots, x_i) que para pasar a la siguiente capa obtendrá nuevos valores (B_1, B_2, \dots, B_i) , cambiando los valores de los pesos (w_1, w_2, \dots, w_i) . Para actualizar estos valores se utilizará una función de transferencia (sigmoide), por ejemplo, una *logsig* o *tansig*. La primera capa oculta obtendrá los nuevos valores a partir de los datos elegidos, en cambio, la segunda capa oculta, que tendrá solo una neuro-

na, obtendrá los valores de la capa anterior y no tendrá *bias*. Esta capa tendrá como función de transferencia fija del tipo *purelin*. Por último, para obtener los valores de la capa de salida, que tendrá el mismo número de neuronas que la capa de entrada, se pondrá el valor de los pesos fijo a 1 y se utilizará la función de transferencia *logsig*. Una de las condiciones que hay para la capa de salida es que el valor de las *bias* tiene que mantener el orden, es decir, $\beta_0^1 < \beta_0^2 < \dots < \beta_0^{J-1}$. Una vez obtenida la salida se hará una transformación de los valores para poder tratarlos.

La Figura 7.1 muestra gráficamente dicho funcionamiento.

La mayoría de los modelos de regresión ordinal se representan como en la ecuación 7.1.

$$C(\mathbf{x}) = \begin{cases} c_1, & \text{si } f(\mathbf{x}, \boldsymbol{\theta}) \leq \beta_0^1 \\ c_2, & \text{si } \beta_0^1 < f(\mathbf{x}, \boldsymbol{\theta}) \leq \beta_0^2 \\ \dots & \\ c_J, & \text{si } f(\mathbf{x}, \boldsymbol{\theta}) > \beta_0^{J-1} \end{cases} \quad (7.1)$$

siendo c_1, c_2, \dots, c_J las clases ordenadas, $\beta_0^1 < \beta_0^2 < \dots < \beta_0^{J-1}$ los sesgos o *bias* de las funciones discriminantes y $f(\mathbf{x}, \boldsymbol{\theta})$ la función de ranking discriminante (en este caso, no lineal).

Nuestro modelo, que llamaremos ORNNet, es similar al modelo POM¹, pero usando transformaciones básicas no lineales de las entradas en lugar de las variables de entrada. De este modo, las funciones discriminantes a utilizar son ahora:

¹Proportional Odds Model, modelo explicado en el capítulo de Antecedentes.

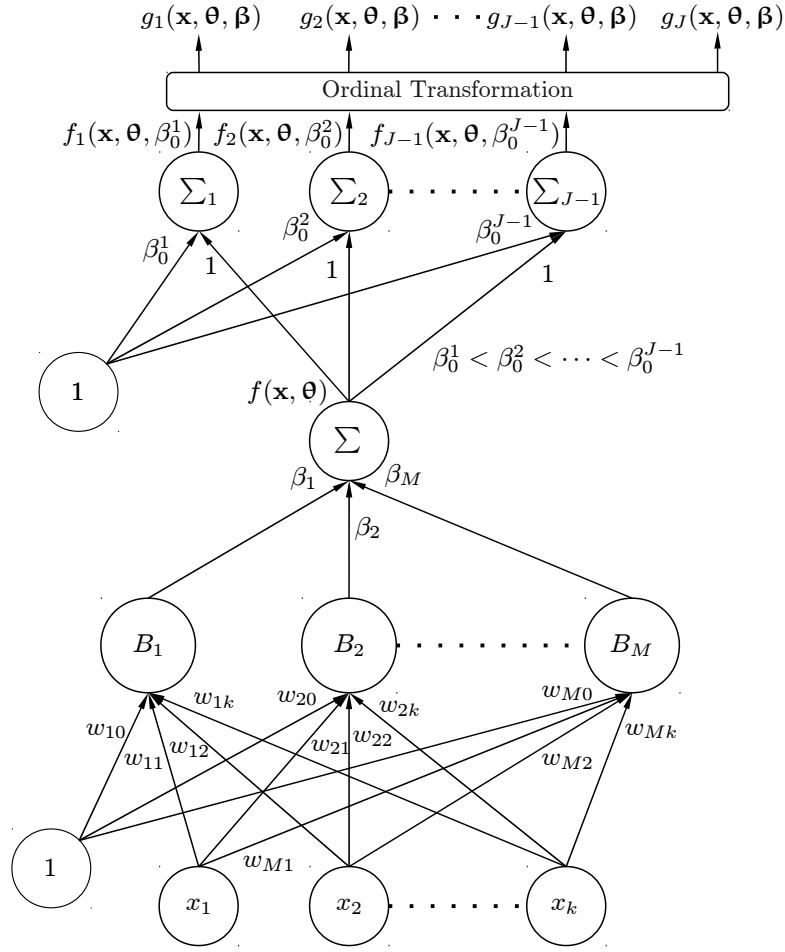


Figura 7.1: Funcionamiento del algoritmo para redes neuronales artificiales ORNNet.

$$f_l(\mathbf{x}, \boldsymbol{\theta}, \beta_0^l) = f(\mathbf{x}, \boldsymbol{\theta}) - \beta_0^l; \quad 1 \leq l \leq J-1 \quad (7.2)$$

siendo $f(\mathbf{x}, \boldsymbol{\theta})$ la función de base del modelo, como por ejemplo, un sumatorio de transformaciones no lineales de las variables de entrada:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^M \beta_i B_i(\mathbf{x}, \mathbf{w}_i) \quad (7.3)$$

donde $B_i(x, w_i)$ son las funciones de base, que este proyectos se han utilizado del tipo *logsig*, *tansig* y *purelin*.

Haciendo uso del modelo POM, las probabilidades acumuladas, las posibilidades acumuladas² y los logits acumulados serían:

$$P(Y \leq l) = p_1 + \cdots + p_l \quad (7.4)$$

$$odds(Y \leq l) = \frac{P(Y \leq l)}{1 - P(Y \leq l)} \quad (7.5)$$

$$logit(Y \leq l) = \ln \left(\frac{P(Y \leq l)}{1 - P(Y \leq l)} \right) \quad (7.6)$$

para $1 \leq l \leq J - 1$.

Y en el modelo de ORNNet:

$$logit(Y \leq l) = f_l(\mathbf{x}, \boldsymbol{\theta}, \beta_0^l) = f(\mathbf{x}, \boldsymbol{\theta}) - \beta_0^l; \quad 1 \leq l \leq J - 1 \quad (7.7)$$

$$P(Y \leq l) = \frac{1}{1 + \exp(f(\mathbf{x}, \boldsymbol{\theta}) - \beta_0^l)}; \quad 1 \leq l \leq J - 1 \quad (7.8)$$

$$P(Y \leq J) = 1 \quad (7.9)$$

²Cumulative odds.

Esto permite expresar las probabilidades como:

$$P(Y = 1) = g_1(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) = P(Y \leq 1) \quad (7.10)$$

$$P(Y = l) = g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) = P(Y \leq l) - P(Y \leq l - 1) \quad (7.11)$$

donde $2 \leq l \leq J$.

En el problema de clasificación ordinal, las medidas x_i , $i = 1, 2, \dots, k$, se toman de manera individual (o como un objeto) y éstos se clasifican en una de las J clases en base a dichas mediciones. Se asume que J es finito y que las medidas de x_i son observaciones aleatorias de esas clases.

Sea $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, \dots, N\}$ un conjunto de datos de entrenamiento, donde $\mathbf{x}_n = (x_{1n}, \dots, x_{kn})$ es el vector de mediciones tomando valores en $\boldsymbol{\Omega} \subset \mathbb{R}^k$, e \mathbf{y}_n el nivel de clase o etiqueta del n -ésimo individuo. Se adoptará la técnica más común para representar niveles de clases usando un vector de codificación “1-de- J ”, $\mathbf{y} = (y^{(1)}, y^{(2)}, \dots, y^{(J)})$, tal que $y^{(l)} = 1$ si \mathbf{x} corresponde a un elemento perteneciente a la clase l y, en otro caso, a $y^{(l)} = 0$.

La función que se usará para evaluar una red ORNNet es la del error cuadrático medio (MSE) que viene expresada a continuación:

$$l(\boldsymbol{\beta}, \boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^N \sum_{l=1}^J \left(g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) - y_n^{(l)} \right)^2 \quad (7.12)$$

donde $\boldsymbol{\beta} = (\beta_0^1, \dots, \beta_0^{J-1})$ es el vector de sesgos o *bias* y el modelo es:

$$g_1(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{1 + \exp(f_1(\mathbf{x}, \boldsymbol{\theta}, \beta_0^1))} \quad (7.13)$$

$$g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{1 + \exp(f_l(\mathbf{x}, \boldsymbol{\theta}, \beta_0^l))} - \frac{1}{1 + \exp(f_{l-1}(\mathbf{x}, \boldsymbol{\theta}, \beta_0^{l-1}))} \quad (7.14)$$

$$g_J(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) = 1 - \frac{1}{1 + \exp(f_{J-1}(\mathbf{x}, \boldsymbol{\theta}, \beta_0^{J-1}))} \quad (7.15)$$

donde $l = 2, \dots, J - 1$.

Para el propósito que de este algoritmo, se ha realizado una adaptación del procedimiento para óptimos locales en regresión ordinal, *iRprop+*, y la función del MSE (Ecuación 7.12). En este caso, el vector gradiente sigue la siguiente formulación.

$$\nabla l(\boldsymbol{\beta}, \beta_1, \dots, \beta_M, \mathbf{w}_1, \dots, \mathbf{w}_M) = \left(\frac{\partial l}{\partial \boldsymbol{\beta}}, \frac{\partial l}{\partial \beta_1}, \dots, \frac{\partial l}{\partial \beta_M}, \frac{\partial l}{\partial \mathbf{w}_1}, \dots, \frac{\partial l}{\partial \mathbf{w}_M} \right)$$

donde β_1, \dots, β_M y w_1, \dots, w_M son los coeficientes de la función de base mostrada en la Figura 7.1.

Siendo η alguno de los parámetros $\boldsymbol{\beta}$ o $\boldsymbol{\theta}$. Con lo que:

$$\frac{\partial l}{\partial \eta} = \frac{1}{N} \sum_{n=1}^N \sum_{l=1}^J 2 \cdot \left(g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) - y_n^{(l)} \right) \cdot \frac{\partial g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \eta}.$$

Derivando ahora la función g :

$$\frac{\partial g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta})}{\partial \eta} = \begin{cases} \left(\frac{-e^{f_1}}{(1 + e^{f_1})^2} \right) \frac{\partial f_1}{\partial \eta}, & l = 1 \\ \left(\frac{-e^{f_l}}{(1 + e^{f_l})^2} \right) \frac{\partial f_l}{\partial \eta} - \left(\frac{-e^{f_{l-1}}}{(1 + e^{f_{l-1}})^2} \right) \frac{\partial f_{l-1}}{\partial \eta}, & l = 2, \dots, J - 1 \\ - \left(\frac{-e^{f_{J-1}}}{(1 + e^{f_{J-1}})^2} \right) \frac{\partial f_{J-1}}{\partial \eta}, & l = J \end{cases}$$

donde $g_l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\beta}) = g_l$ y $f_l(\mathbf{x}, \boldsymbol{\theta}, \beta_0^l) = f_l$.

La siguiente expresión incluye las derivadas de los coeficientes para la capa de salida:

$$\frac{\partial f_l}{\partial \beta_0^k} = \begin{cases} 0 & k \neq l \\ -1 & k = l \end{cases},$$

$$\frac{\partial f_l}{\partial \beta_s} = B_s(\mathbf{x}, \mathbf{w}_s); \quad 1 \leq s \leq M$$

El gradiente para las capas ocultas dependerá del tipo de función base. Si $B_s(\mathbf{x}, \mathbf{w}_s)$ son nodos sigmoïdales; cuya ecuación es:

$$B_s(\mathbf{x}, \mathbf{w}_s) = \sigma \left(\sum_{i=1}^k w_{is} x_i \right), \sigma(x) = \frac{1}{1 + e^{-x}} \quad (7.16)$$

Entonces,

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{w}_s} &= \left(\frac{\partial f_l}{\partial w_{s1}}, \dots, \frac{\partial f_l}{\partial w_{sk}} \right) \\ \frac{\partial f_l}{\partial w_{st}} &= \beta_s^l \sigma' \left(\sum_{i=1}^k w_{is} x_i \right) x_t \end{aligned}$$

donde $l = 1, 2, \dots, J - 1$, $t = 1, 2, \dots, k$ y $s = 1, 2, \dots, M$.

7.2. Descripción funcional del sistema

En esta sección, se describirá el sistema con detalle, haciendo uso de técnicas de modelado que mostrarán de un modo inequívoco cómo interviene el usuario en el sistema, cuáles son los ítems que componen el sistema y qué operaciones soportan estos ítems. Para ello, se empleará la metodología UML (Lenguaje Unificado de Modelado) [12].

UML es un lenguaje de modelado que mediante un determinado vocabulario y un conjunto de reglas es capaz de representar conceptual y físicamente un sistema. El motivo de escoger esta metodología es que incorpora técnicas sencillas que dan una visión más cercana de lo que el usuario espera obtener.

7.2.1. Valores de los diferentes parámetros que utiliza el sistema

Con respecto a los principales valores del conjunto de datos que se vaya a usar, el número de capas ocultas, las funciones de las capas ocultas, etc. serán valores que se decidirán en secciones posteriores, durante la experimentación o en cuanto se pueda evaluar el funcionamiento del módulo, de manera que se consiga una mejor solución.

A continuación se describen los parámetros más comunes a definir por el usuario para el algoritmo o la creación de la red neuronal artificial:

- *Conjunto de datos*: son los datos que se especificarán para que el sistema pueda funcionar y para que la red neuronal pueda entrenar y simular los resultados de salida.
 - **Datos de entrada**: son los datos de entrada del sistema, éstos vendrán definidos para cada conjunto de datos con los que se pruebe el sistema y tendrán un formato ordinal.
 - **Datos objetivos**: son los datos objetivo del sistema, éstos se utilizarán para comprobar cuanto de buena es la red neuronal y el algoritmo implementado conteniendo los resultados reales de la clasificación.
- *Número de neuronas en las capas ocultas*: este parámetro podrá ser variable en función de cuantas capas ocultas y cuantas neuronas se desee poner en la red neuronal, por defecto se establecerán dos capas en la que la primera capa oculta contendrá 20 neuronas por defecto y la segunda capa oculta, que será fija para el sistema, contendrá 1 neurona. Adicionalmente, se proporcionará la opción de calcular el número óptimo de neuronas que debería haber en las capas ocultas.
- *Número de neuronas en la capa de salida*: este parámetro es fijo, ya que será el mismo número que haya en la entrada del sistema.

- *Funciones de transferencia de las capas*: este parámetro podrá contener alguno de los siguientes valores: *logsig*, *tansig*, *purelin*, *softmax*; para las capas ocultas que se definan. Adicionalmente, para las capas fijas, la última capa de la capa oculta y la capa de salida, tendrán el valor de *purelin* y *logsig* respectivamente.
- *Función de entrenamiento*: la función de entrenamiento que se usará en el sistema será la implementada para el algoritmo ORNNet que será una modificación del algoritmo visto con anterioridad iRProp+.
- *Función de aprendizaje*: este parámetro vendrá por defecto en el sistema al crear la red neuronal y se utilizará el valor: *learnngdm*, la cual se basa en descenso de gradiente.
- *Función de precisión*: para este parámetro también se usará el valor por defecto: *mse*, aunque se podrá modificar en cuanto convenga. Los distintos valores que podrá tomar son: *mse* y *mae*.

7.2.2. Diagramas de Casos de Uso

Para obtener el modelo de objetos del software a desarrollar, se partirá de un conjunto de casos de uso desde un alto nivel de abstracción hasta un nivel de detalle más elevado, pudiendo obtener de esta forma todos los requisitos necesarios y los objetos que conformarán el sistema. Un caso de uso especifica el comportamiento de un sistema o de una parte del mismo y es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor.

Los casos de uso se emplean para capturar el comportamiento deseado del sistema en desarrollo sin tener que especificar cómo se implementa ese comportamiento, además de proporcionar un medio para que los desarrolladores, los usuarios finales del sistema y los expertos del dominio lleguen a una comprensión común del sistema.

7.2.2.1. Perfiles de usuario

En el sistema de resolución de problemas de modelado se pueden distinguir distintos tipos de actores, un actor representa un conjunto coherente de roles que juegan los usuarios de los casos de uso al interactuar con él. Los actores pueden ser personas o sistemas externos, establecidos por la siguiente clasificación:

- *Principales*: personas que usan el sistema, serán los usuarios del sistema. Este proyecto no se basa en la interacción con el usuario del sistema ya que retoma más importancia la implementación y la obtención de resultados del algoritmo propuesto, no obstante, existirá una interfaz gráfica con la que el usuario podrá interactuar. El usuario podrá ejecutar el algoritmo y obtener los resultados. Para alcanzar dicho fin, el usuario tendrá que introducir unos parámetros de entrada, es decir, un conjunto de datos de entrada y de objetivos para el entrenamiento y el testeo de la red neuronal. Una vez especificados, habrá que crear la red neuronal y seguidamente realizar el entrenamiento y la simulación de la misma. Por último, tendrá que recoger las salidas. Las personas que usarán el sistema serán aquellas que tengan conocimientos acerca del uso y funcionamiento de redes neuronales artificiales.
- *Secundarios*: personas que mantienen o administran el sistema. El mantenimiento de esta aplicación va a ser llevada por otras personas que realicen mejoras o actualizaciones debido a evoluciones que sufra el toolbox nnet de Matlab.
- *Material externo*: dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- *Otros sistemas*: sistemas con los que la aplicación interactúa. Sistema Operativo: indica el sistema sobre el cual trabajará la aplicación, que dado a que el lenguaje es proporcionado por Matlab y éste es

multiplataforma, podrá ser cualquiera de las arquitecturas soportadas.

7.2.2.2. Casos de Uso

El modelado de Casos de Uso es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los Casos de Uso se utilizan para modelar cómo un sistema funciona actualmente o cómo los usuarios desean que funcione. No es realmente una aproximación a la orientación a objetos, realmente es una forma de modelar procesos. Además es, sin embargo, una manera muy eficaz de dirigirse hacia el análisis de sistemas orientados a objetos, aunque en este caso no será ese su fin. Los casos de uso son, generalmente, el punto de partida del análisis con UML.

La Figura 7.2 y la Tabla 7.1 representan el caso de uso correspondiente al contexto del sistema.

La Figura 7.3 y la Tabla 7.2 representan el caso de uso correspondiente al tratamiento de la RNA.

7.2.3. Diagrama de Secuencia

Los diagramas de Secuencia son un tipo de diagrama usados para modelar interacciones entre objetos en un sistema según UML. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Mientras que el diagrama de casos de uso permite el modelado de una vista *business* del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

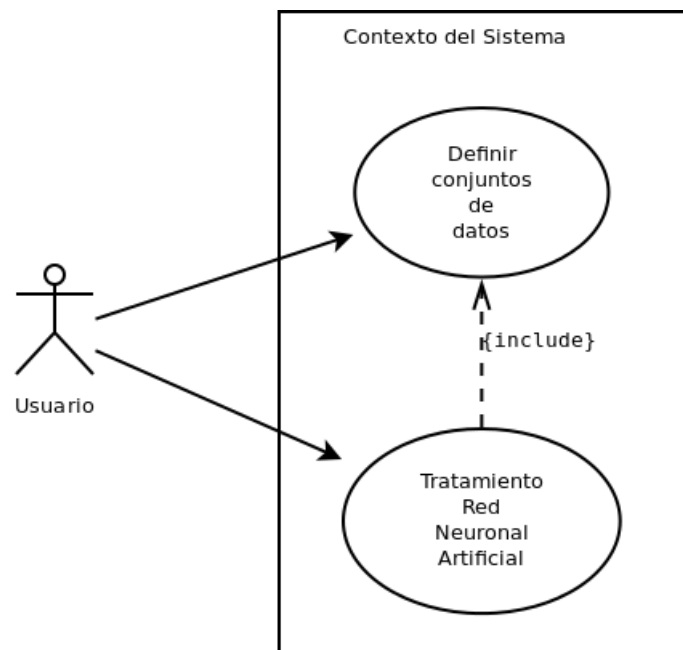


Figura 7.2: Diagrama de Contexto del Sistema.

| | |
|----------------------------|--|
| Nombre | Contexto del Sistema. |
| Actores | Usuario. |
| Descripción | El sistema proporciona al usuario la opción de definir el conjunto de datos o de tratar la red neuronal artificial, ya sea creándola, entrenándola u obteniendo los resultados. |
| Casos de Uso | <p>CU1. Definir el conjunto de datos. Si el usuario quiere definir el conjunto de datos tendrá que elegir esta opción.</p> <p>CU2. Tratamiento Red Neuronal Artificial. Si el usuario desea realizar el tratamiento de la red neuronal, ya sea crearla, entrenarla u obtener los resultados.</p> |
| Flujo Principal de Eventos | El usuario comenzará eligiendo la opción que desea ejecutar, que podrá ser o definir el conjunto de datos a usar o tratar la red neuronal artificial. |

Tabla 7.1: Diagrama de caso de uso CU0: Contexto del Sistema.

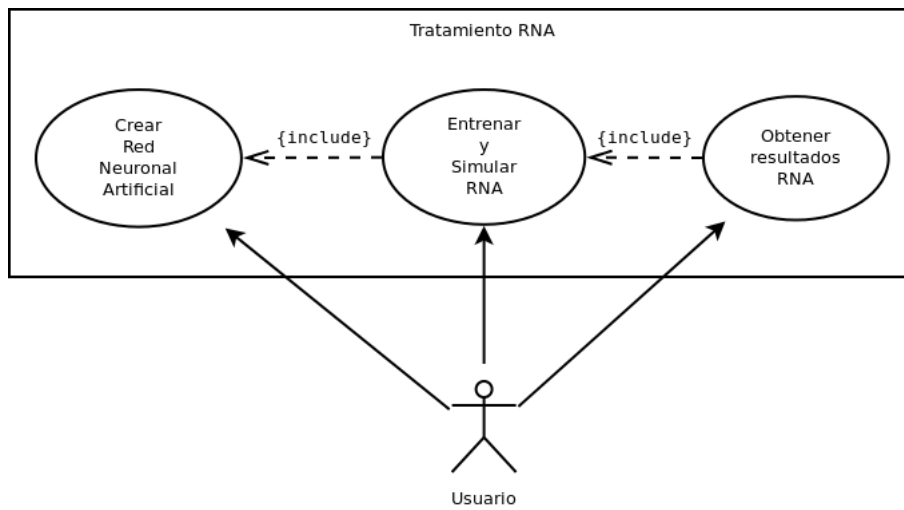


Figura 7.3: Diagrama de caso de uso CU1: Tratamiento RNA.

La Figura 7.4 representa el diagrama de secuencia correspondiente al funcionamiento del sistema.

| | |
|----------------------------|---|
| Nombre | Tratamiento RNA. |
| Actores | Usuario. |
| Descripción | El sistema proporciona las opciones para el tratamiento de una red neuronal artificial para regresión ordinal. |
| Casos de Uso | <p>Crear RNA: consiste en la creación de una red neuronal artificial para regresión ordinal.</p> <p>Entrenar y simular RNA: consiste en realizar el entrenamiento de la red neuronal aplicando el algoritmo ORNNet explicado en la sección 7.1.</p> <p>Obtener resultados RNA: consiste en obtener los resultados generados por el entrenamiento y simulación del conjunto de datos en la red neuronal aplicando el algoritmo desarrollado de regresión ordinal.</p> |
| Flujo Principal de Eventos | El usuario crea la red neuronal de la que, a partir de ella y del conjunto de datos cargado previamente, se realiza el entrenamiento y la simulación de la red haciendo uso del algoritmo implementado, ORNNet, del cual se obtienen los resultados para el posterior análisis de los mismos. |

Tabla 7.2: Diagrama de caso de uso CU1.

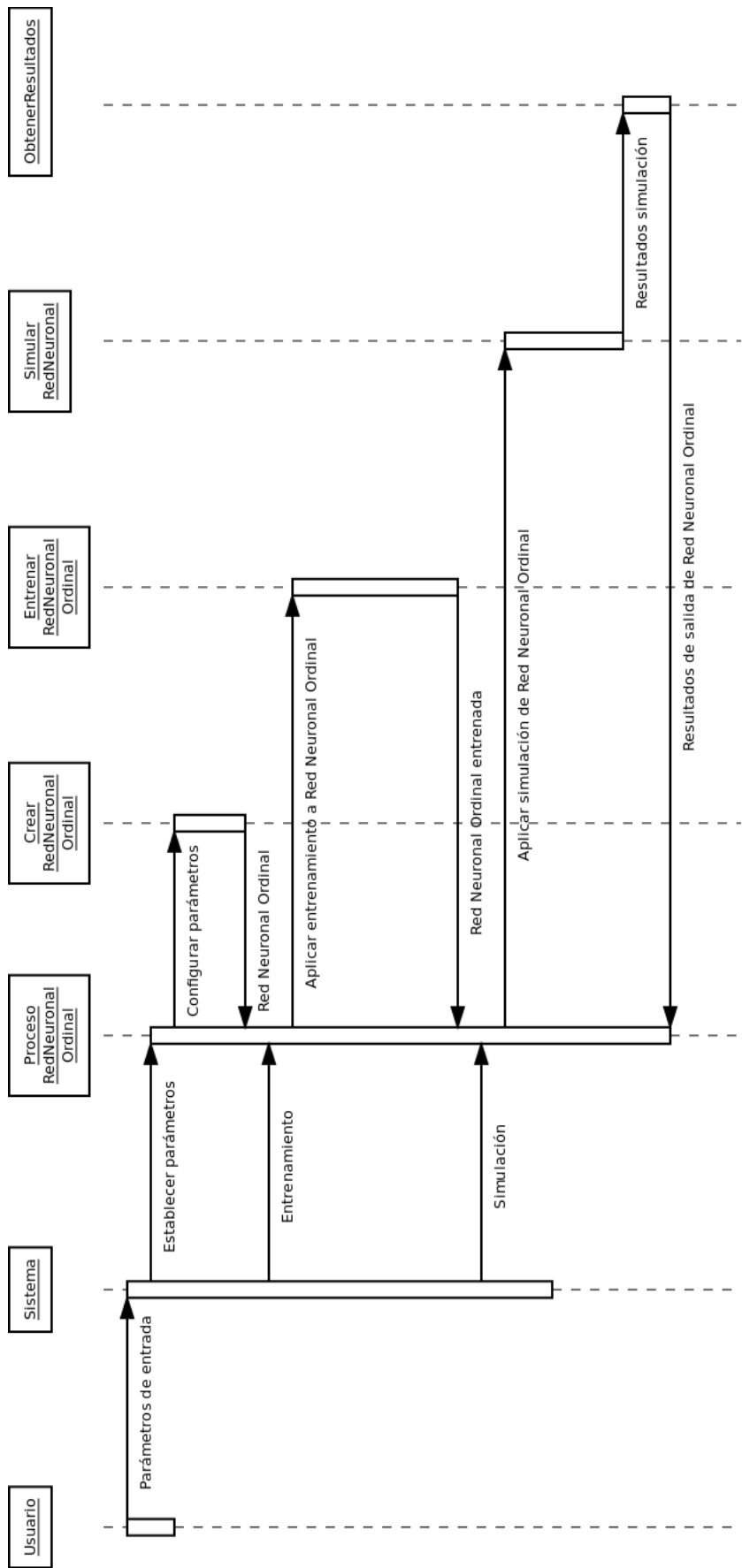
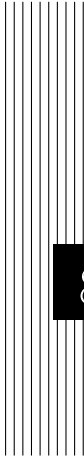


Figura 7.4: Diagrama de Secuencia.



8 Diseño del sistema

En este capítulo se detallarán los módulos que serán necesarios utilizar e implementar. Para cada módulo se especificará el nombre del módulo, una descripción general del módulo, las variables que usa y las funciones que utiliza. Estos módulos se extraen tanto de los diagramas de casos de uso y de los diagramas de secuencia como de la propia definición del problema. En cada uno de los módulos que se describirán a continuación se distinguirán principalmente los siguiente apartados:

- *Nombre del módulo:* se especificará el nombre del módulo que se vaya a explicar. Este nombre será identificativo y se corresponderá con el estilo que tiene el toolbox nnet de Matlab para que la integración sea lo más correcta posible.
- *Descripción general del módulo:* se hará una descripción general de la funcionalidad y el manejo que tendrá el módulo.
- *Funciones del módulo:* en el caso de que el módulo en sí no sea una función, se detallarán las funciones que se usan para el correcto funcionamiento de dicho módulo.

8.1. Diseño de los módulos

En esta sección se va a realizar la especificación de los módulos que contendrá el sistema. Para ello, se hará uso de diagramas de paquetes y así poder ver la estructuración del sistema.

En el Lenguaje Unificado de Modelado, los diagramas de Paquetes se usan para reflejar la organización de paquetes y sus elementos. Los usos más comunes de para los diagrama de paquete son para organizar diagramas de casos de uso y diagramas de clases, estos paquetes son como grandes contenedores de clases. Los elementos contenidos en un paquete comparten el mismo espacio de nombres, esto significa que los elementos contenidos en un mismo espacio de nombres específico deben tener nombres únicos. Como otra característica de estos diagramas, cada paquete se debe identificar con un nombre único y opcionalmente mostrar todos los elementos dentro del mismo.

Antes de realizar la especificación de cada módulo por separado, en la Figura 8.1 se muestra la estructuración general del sistema diseñado para que se ajuste lo más posible y con una estructura similar al toolbox nnet de Matlab.

8.1.1. Módulo principal nnet

Este será el módulo principal del sistema el cual contendrá los submódulos necesarios para el funcionamiento del sistema. Cada submódulo contendrá las funciones necesarias para la creación y tratamiento de la red neuronal artificial junto con el algoritmo propuesto en este proyecto. La organización de este módulo contiene una estructura similar al del toolbox nnet de Matlab.

La Figura 8.2 muestra el diagrama de Paquetes para este módulo.

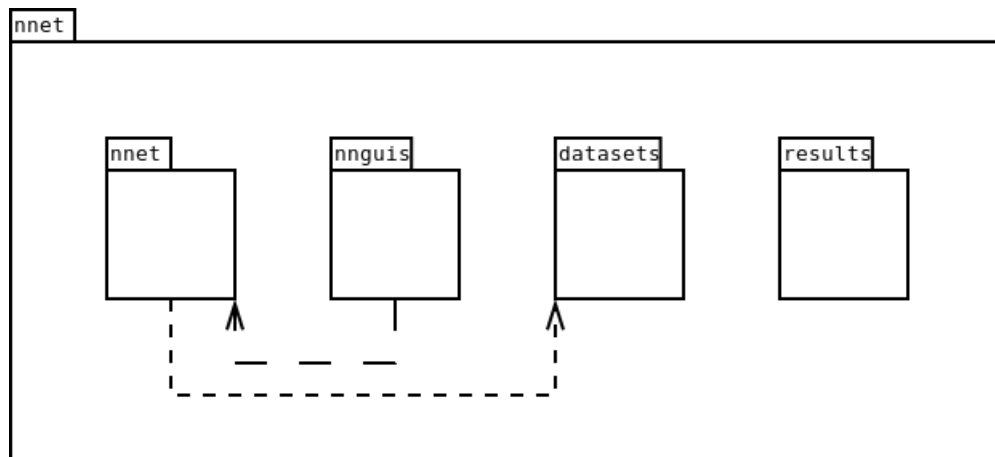
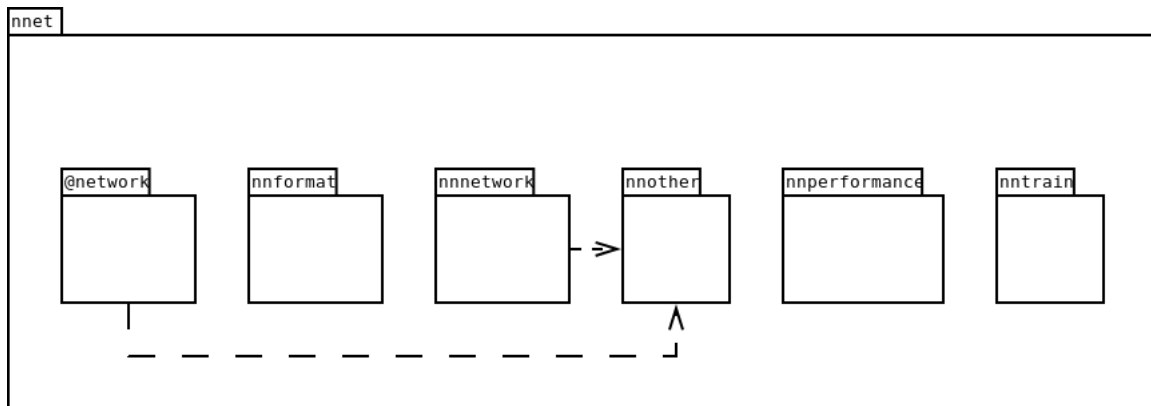


Figura 8.1: Diagrama de Paquetes principal

Figura 8.2: Diagrama de Paquetes del módulo `nnet` diseñado

La Tabla 8.1 muestra la especificación breve del módulo principal.

| | |
|-------------|--|
| Nombre | <code>nnet</code> |
| Descripción | Módulo principal del sistema. Contendrá los principales submódulos para que el sistema funcione. |

Tabla 8.1: Especificación módulo `nnet`

8.1.1.1. Módulo @network

La Tabla 8.2 muestra la especificación detallada del módulo así como la descripción de las funciones que contiene.

| | |
|-------------|---|
| Nombre | @network |
| Descripción | Módulo que contiene las funciones para el entrenamiento y simulación de la red neuronal pero ajustada para que los métodos se basen en la ordinalidad. |
| Funciones | <i>OSIM</i> : función para simular una red neuronal ordinal a partir de los datos de entrada del conjunto de datos seleccionado y que devolverá las salidas obtenidas. <i>OTRAIN</i> : función para entrenar una red neuronal ordinal a partir de los datos de entrada y los datos objetivo del conjunto de datos seleccionado. El entrenamiento se producirá haciendo uso del algoritmo propuesto en este proyecto, es decir, ORNnet. |

Tabla 8.2: Especificación módulo @network

8.1.1.2. Módulo nnformat

La Tabla 8.3 muestra la especificación detallada del módulo así como la descripción de las funciones que contiene.

| | |
|-------------|--|
| Nombre | nnformat |
| Descripción | Módulo que contiene la función para realizar una división estratificada de los datos. |
| Funciones | <i>DIVIDESTRA</i> : función que realiza una división estratificada de los datos a partir de los datos objetivos los cuales son tratados para obtener un vector de elementos con identificación a la clase a la que pertenecen. |

Tabla 8.3: Especificación módulo nnformat

8.1.1.3. Módulo `nnnetwork`

La Tabla 8.4 muestra la especificación detallada del módulo así como la descripción de las funciones que contiene.

| | |
|-------------|--|
| Nombre | <code>nnnetwork</code> |
| Descripción | Módulo que contiene la función necesaria para crear una red neuronal ordinal. |
| Funciones | <i>NEWOFF</i> : función que crea una red neuronal ordinal a partir de los datos de entrada y objetivo del conjunto de datos seleccionado, por defecto habrá al menos dos capas ocultas donde la última capa oculta tendrá solamente una neurona sin bias y se usará como función de transferencia lineal. La capa de salida tendrá por defecto como función de transferencia una sigmoide logarítmica. La función de entrenamiento especificada por defecto será la de entrenamiento basado en el algoritmo modificado <i>iRProp+</i> ajustado para la implementación ordinal (ORNnet). Por último, la función por defecto para análisis será la del Error Cuadrado Medio (MSE) y se utilizará una división estratificada de los datos para la simulación de la red. |

Tabla 8.4: Especificación módulo `nnnetwork`

8.1.1.4. Módulo `nnother`

La Tabla 8.5 muestra la especificación detallada del módulo así como la descripción de las funciones que contiene.

8.1.1.5. Módulo `nnperformance`

La Tabla 8.6 muestra la especificación detallada del módulo así como la descripción de las funciones que contiene.

| | |
|-------------|---|
| Nombre | nnother |
| Descripción | Módulo que contiene las funciones auxiliares para carga de ficheros, realización de pre o post procesamiento de los conjuntos de datos de entrada u objetivos para convertirlos o transformarlos en conjuntos de datos válidos para su posterior tratado o realización de una división estratificada. |
| Funciones | <p><i>CONVDATA</i>: función que realiza, a partir del conjunto de datos de entrada y de objetivos con el formato proporcionado por NNEP de la librería del grupo AYRNA, JCLEC, la creación de los nuevos conjuntos de datos compatibles para el tratado por Matlab.</p> <p><i>CONVOUTPUTS</i>: función que realiza el tratado del conjunto de datos de salida proporcionado por la simulación de la red neuronal ordinal convirtiéndolo en un conjunto de datos de salida válido para su posterior procesado.</p> <p><i>GETSTRA</i>: función que realiza, a partir de la transformación de un conjunto de datos objetivos, la creación del vector de elementos estratificados.</p> <p><i>IMPORTFILE</i>: función que realiza la carga de datos de un fichero a partir del nombre del fichero.</p> <p><i>KFOLD</i>: función que realiza una validación de cruce haciendo un k-Fold, que por defecto será de diez folds, a partir de los datos de entrada y objetivo de la red devolviendo el número óptimo de neuronas en la capa oculta.</p> <p><i>TRANSDATA</i>: función que realiza la transformación de los datos objetivos para obtener como resultado un nuevo conjunto de datos objetivos compatible para el procesamiento del MSE.</p> |

Tabla 8.5: Especificación módulo nnother

| | |
|-------------|---|
| Nombre | nnperformance |
| Descripción | Módulo que contiene las funciones para medición del rendimiento de las salidas proporcionadas al entrenar y simular la red neuronal ordinal. |
| Funciones | <i>CCRCALC</i> : función que realiza el cálculo del rango correctamente clasificado a partir de la matriz de confusión resultante de los datos de salida proporcionados al simular la red neuronal ordinal.. <i>MAECALC</i> : función que realiza el cálculo de error medio absoluto a partir de la matriz de confusión resultante de los datos de salida proporcionados al simular la red neuronal ordinal. |

Tabla 8.6: Especificación módulo nnperformance

8.1.1.6. Módulo nntrain

La Tabla 8.7 muestra la especificación detallada del módulo así como la descripción de las funciones que contiene.

| | |
|-------------|--|
| Nombre | nntrain |
| Descripción | Módulo que contiene las funciones de entrenamiento para una red neuronal. Para una red neuronal ordinal se ha implementado la función de entrenamiento iRProp+ modificado con las especificaciones del algoritmo ORNnet. |
| Funciones | <i>TRAINIRP</i> : función que realiza el entrenamiento de una red neuronal artificial aplicando el algoritmo iRProp+. <i>TRAINIRPO</i> : función que realiza el entrenamiento de una red neuronal ordinal aplicando el algoritmo modificado iRProp+ con las modificaciones necesarias y especificadas por ORNnet. |

Tabla 8.7: Especificación módulo nntrain

8.2. Diseño de la interfaz

En esta sección se va a especificar cómo será el entorno que el usuario perciba al poner en funcionamiento la aplicación. Para ello se va a hacer uso de herramientas de prototipado de interfaces, concretamente de la herramienta web Balsamiq. En cada uno de los prototipos de la interfaz se especificarán cada uno de los elementos que contiene detalladamente.

8.2.1. Interfaz principal

La *Interfaz principal* poseerá una estructura como la que se puede apreciar en la Figura 8.3.

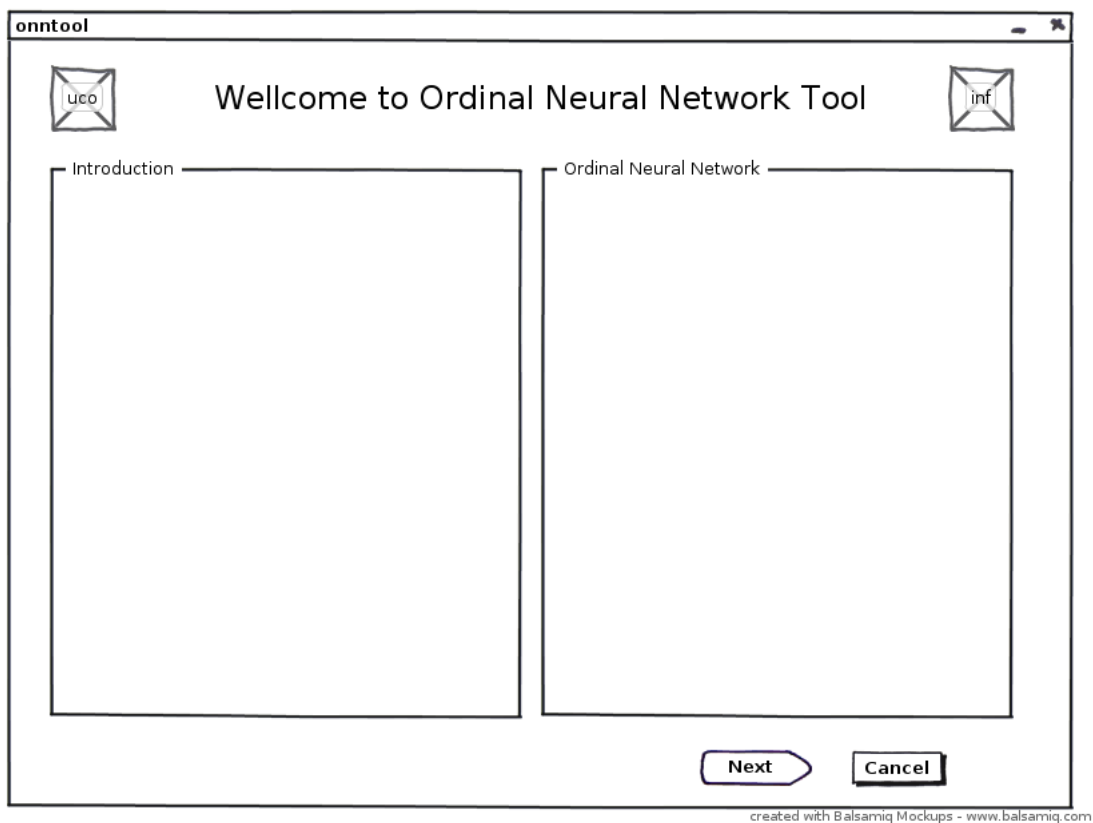


Figura 8.3: Prototipo de la Interfaz principal

La interfaz contendrá unas dimensiones de 800x600 píxeles en todo momento y, concretamente en ésta, los siguientes elementos ya sean interactivos o no:

- **Título:** contiene el título principal de bienvenida a la aplicación.
- **Imagen UCO:** imagen corporativa de la Universidad de Córdoba.
- **Imagen Informática:** imagen corporativa de los Ingenieros Informáticos.
- **Información Introducción:** cuadro que proporciona una breve información general sobre el uso de la aplicación.
- **Información Redes Neuronales Ordinales:** cuadro que proporciona una breve información sobre las redes neuronales artificiales ordinales.
- **Botón Siguiente:** botón que sirve para pasar a la siguiente interfaz.
- **Botón Cancelar:** botón que sirve para cerrar la interfaz en cualquier momento.

Los elementos comunes que aparezcan en las siguientes interfaces no se volverán a explicar para que no exista redundancia de información.

8.2.2. Interfaz de datos

La *Interfaz de datos* poseerá una estructura como la que se puede apreciar en la Figura 8.4.

La interfaz contendrá los siguientes elementos, ya sean interactivos o no:

- **Título:** contiene el título representativo de la interfaz.
- **Botón de selección de datos de entrada (train y test):** botón que muestra la información del conjunto de datos de entrada.

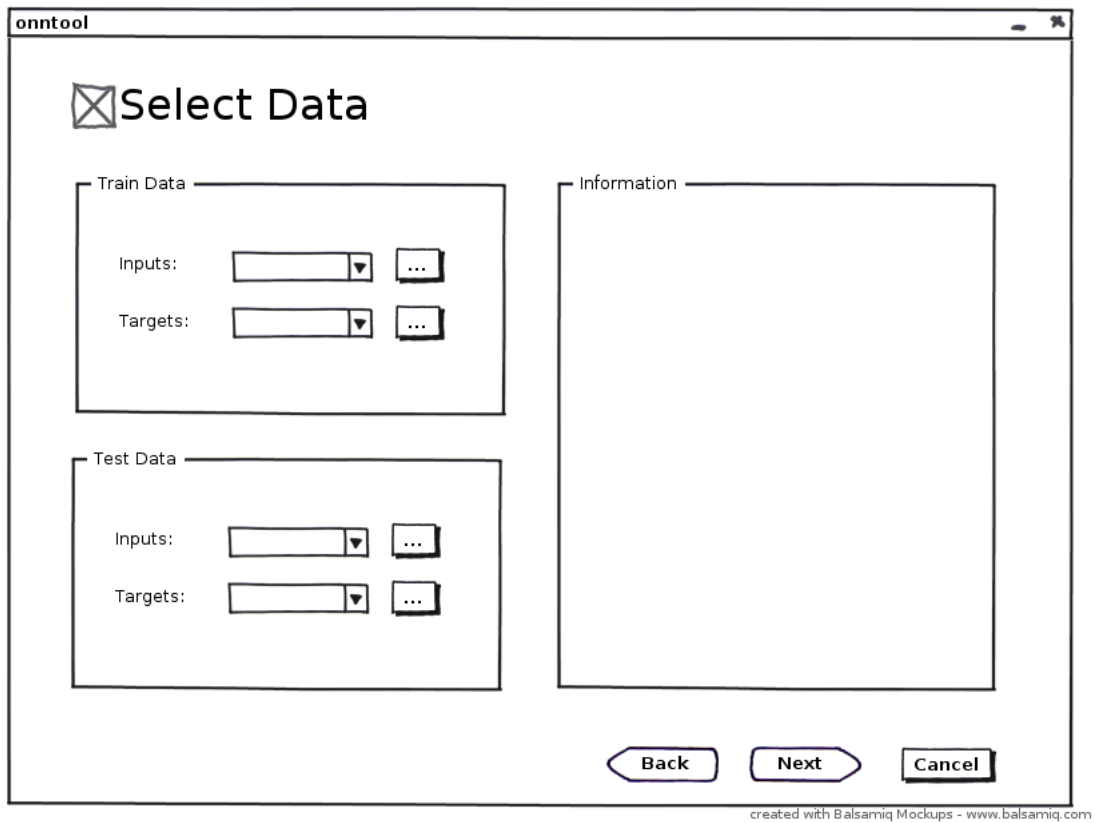


Figura 8.4: Prototipo de la Interfaz de datos

- **Botón de selección de datos objetivos (train y test):** botón que muestra la información del conjunto de datos objetivos.
- **Botones de selección de datos (train y test):** botón que, a partir de una interfaz de carga de datos, selecciona un fichero externo y carga los datos para tratarlos.
- **Información de ayuda:** cuadro que proporciona una breve información de la interfaz.
- **Botón Atrás:** botón que sirve para pasar a la interfaz anterior.

8.2.3. Interfaz para la creación de la red neuronal

La *Interfaz de red neuronal* poseerá una estructura como la que se puede apreciar en la Figura 8.5.

La interfaz contendrá los siguientes elementos, ya sean interactivos o no:

- **Opción de realización de k-fold:** botón que da la opción de realizar un k-fold para los conjuntos de datos.
- **Número de neuronas:** en caso de no activar la opción de k-fold se podrá especificar el número de neuronas que tendrá la capa oculta.
- **Opción para la función de transferencia:** al igual que el anterior, cuando no esté activa la opción del k-fold, se tendrá que especificar la función de transferencia.
- **Botón de Diagrama:** botón que da la posibilidad de mostrar de forma gráfica como se organiza la red neuronal artificial ordinal.

8.2.4. Interfaz de entrenamiento y simulación

La *Interfaz de entrenamiento y simulación* poseerá una estructura como la que se puede apreciar en la Figura 8.6.

onntool

Network Configuration

Parameters

k-fold? Yes ▼
No

If no selected k-fold, put number of neurons in hidde layer:

Neurons:

Transfer Function: tansig ▼
logsig

Information

Diagram Back Next Cancel

created with Balsamiq Mockups - www.balsamiq.com

Figura 8.5: Prototipo de la Interfaz de red neuronal

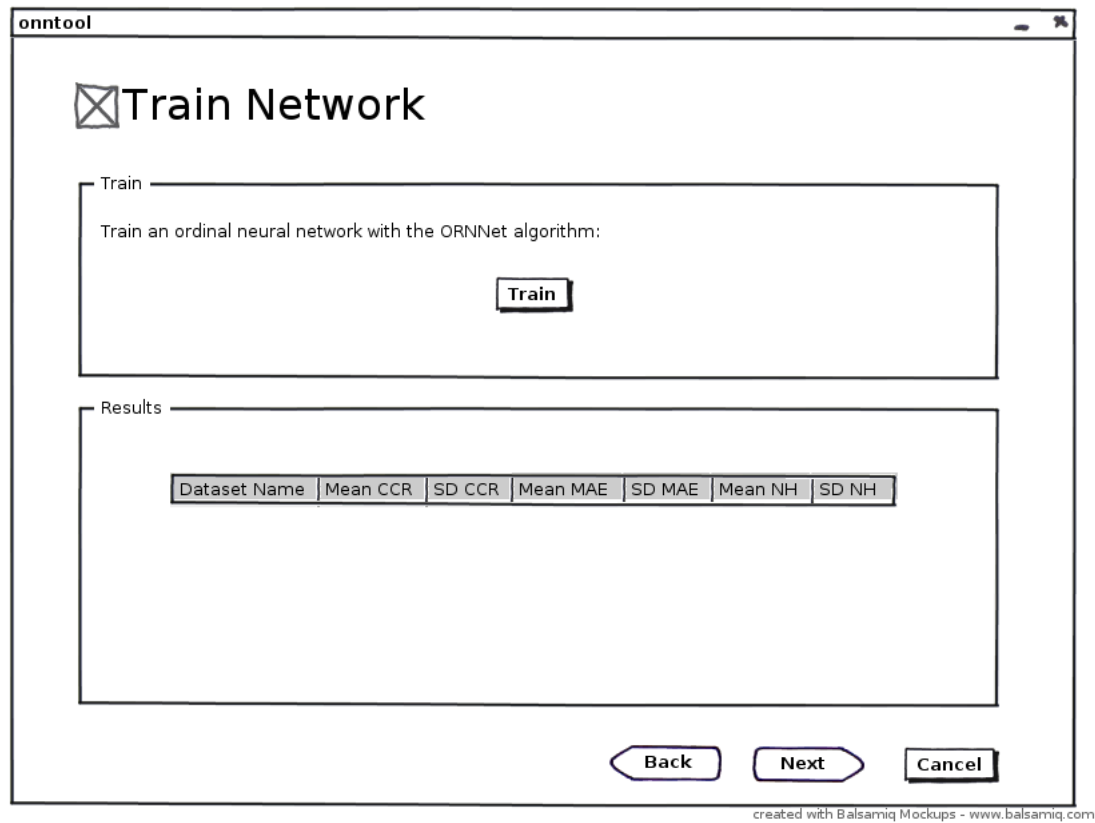


Figura 8.6: Prototipo de la Interfaz de entrenamiento

La interfaz contendrá los siguientes elementos, ya sean interactivos o no:

- **Botón de entrenamiento:** botón que entrena y simula la red neuronal a partir de los datos de entrenamiento y test del conjunto de datos.
- **Resultados:** muestra los resultados obtenidos a partir de la simulación.

8.2.5. Interfaz de exportación de datos

La *Interfaz de exportación* poseerá una estructura como la que se puede apreciar en la Figura 8.7.

La interfaz contendrá los siguientes elementos, ya sean interactivos o no:

- **Nombre para guardar la red neuronal:** campo para especificar como se llamará la variable que almacenará la estructura de la red neuronal.
- **Nombre para guardar las salidas obtenidas:** campo para especificar como se llamará la variable que almacenará la matriz de salida.
- **Nombre para guardar los resultados de salida (CCR, MAE, NH):** campo para especificar como se llamará la variable que almacenará la matriz de resultados.
- **Botón Guardar Resultados:** botón que guardará los resultados en el espacio de trabajo de Matlab.
- **Botón Finalizar:** botón que cerrará la interfaz al dar el proceso por finalizado.

onntool

Save Results

Save Data

Save network to MATLAB network object named:

Save outputs to MATLAB matrix named:

Save results to MATLAB object named:

created with Balsamiq Mockups - www.balsamiq.com

Figura 8.7: Prototipo de la Interfaz de exportación



9 Implementación

En este capítulo se detallarán aquellas particularidades para la implementación real del algoritmo de regresión ordinal y las redes neuronales. Para ello, se han seguido los manuales y tutoriales, además de las guías de referencia que el propio Matlab proporciona [9, 10, 11].

9.1. Entrenamiento y Simulación

Función `osim`

Esta función tiene como particularidades que hay que hacer un tratamiento de las salidas después de simular la red neuronal ordinal, es decir, hay que poner la última línea de resultados a 1 para que la probabilidad acumulada sea correcta en todos los casos alcanzando este valor. Después de este tratamiento, se hace uso de la función *convoutputs* la cual realiza una conversión de las salidas para que se convierta de una salida con probabilidades acumuladas a una salida con resultados binarios, donde se pondrá un 1 cuando sea la clase con mayor probabilidad y un 0 en el resto.

Se muestra el código parcial para que quede clara la implementación.

```

1  %%%
2  %%SIMULATION
3  %simulating the net to obtain the outputs (test set)
4  testOutputs = sim(net,testInputs);
5
6  %adding ones to the last output
7  testOutputs(size(testOutputs,1)+1,:) = ones(1,size(testOutputs,2));
8
9  %converting outputs values
10 testOutputs = convoutputs(testOutputs);

```

Listing 9.1: Archivo osim.m

Función otrain

Esta función tiene como particularidad la llamada ajustada a la función de entrenamiento que se ha implementado en este proyecto y que posteriormente se mostrará con sus particularidades. Esta llamada se encuentra ajustada ya que los valores objetivos de la última entrada no hace falta tratarlos cuando se trate de una red neuronal ordinal.

```

1  %%%
2  %%TRAIN FUNCTION CALLED
3  [net,tr] = train(net,X,T(1:(size(T,1)-1),:));

```

Listing 9.2: Archivo otrain.m

9.2. Entrenamiento ordinal

Función trainirpo

Esta función tiene las mayores peculiaridades en comparación con el resto, ya que es la que aplica la actualización y ajuste del algoritmo iR-Prop+ para redes neuronales ordinales. En la línea 15 se muestra el tratado especial que hay que tener para que las bias se analicen en un paso

posterior y que los pesos no se modifiquen. A partir de la línea 21 se hace el nuevo procesamiento de los sesgos o *bias*, ya que en este caso, las *bias* hay que tratarlas para que se mantenga la condición, $b_1 < b_2 < \dots < b_J - 1$.

```

1 üüüüéí
2  %%APPLY iRPROP+ UPDATE  %%%
3  ggX = gX.*gX_old;
4
5  %- valor delt_inc = 1.2, delt_dec = 0.5
6  %- deltaX = matriz del tamaño de los pesos con valor delta0
7  deltaX = ((ggX>0)*delt_inc + (ggX<0)*delt_dec + (ggX==0)).*deltaX;
8  deltaX = (ggX==0).*deltaX + (ggX>0).*min(deltaX,deltaMAX)+ (ggX<0).*max
    (deltaX,deltaMIN);
9  dX = (ggX>0 | ggX==0).*deltaX.*sign(gX) + (ggX<0 & perf>perf_old).*dX;
10 ddX = (((ggX>0 | ggX==0).*dX)-((ggX<0 & perf>perf_old).*dX));
11
12 %- actualizacion de pesos, modificacion para que los pesos de la
    salida
13 %- siempre valgan uno
14 len = net.outputs{net.numLayers}.size;
15 X = X+ddX.*[ones(length(X)-2*len,1); zeros(2*len,1)];
16
17 %- actualizacion de bias
18 %- controlar la condicion de beta1 < beta2 < ... < betaN,
19 %- siendo las betas el valor de las bias de la capa de salida, que en
    este
20 %- caso
21 for i=(length(X)-len+1):length(X)
22     %- comprobar que el siguiente elemento es mayor
23     if i == length(X)
24         if X(i-1) < (X(i)+ddX(i))
25             X(i) = X(i)+ddX(i);
26         else
27             gX(i) = 0;
28         end
29     else
30         if (X(i)+ddX(i)) < X(i+1)
31             X(i) = X(i)+ddX(i);
32         else
33             gX(i) = 0;
34         end
35     end
36 end
37
38 gX = (~(ggX<0)).*gX;

```

```

39
40 net = setx(net,X);

```

Listing 9.3: Archivo trainirpo.m

9.3. Creación de una red neuronal ordinal

Función newoff

Esta función tiene bastantes peculiaridades ya que es de las más importantes al crear e inicializar la red neuronal artificial de forma ordinal. Desde la línea 3 hasta la 23 se muestra como, a partir de los parámetros de entrada, se crea la red neuronal con los parámetros ajustados y fijos necesarios para que se realice el proceso para una red neuronal ordinal. Entre los ajustes que se realizan, se pueden observar que las dos últimas capas son fijas, las funciones de transferencia para esas capas también lo son. Además, la función de entrenamiento es la comentada anteriormente y la función para la división inicial de los datos es una división estratificada y no la que vendría por defecto.

Las líneas a partir de la 25 serían los ajustes posteriores a la creación de la red neuronal y que no se pueden establecer en el momento de la creación de la misma. Entre otros, se determina los pesos fijos que debe tener la última capa, la eliminación del sesgo intermedio, la inicialización de los *bias* para que cumplan la condición que antes se ha comentado y los porcentajes de cada uno de los patrones para los conjuntos de entrenamiento, test y validación.

```

1  %%%
2  %%CREATE NEURAL NETWORK
3  if (nargin == 2)
4      net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[20
        1],{'tansig','purelin','logsig'},'trainirpo',...
5      'learngdm','mse',{'fixunknowns','removeconstantrows','mapminmax'},{},
        'dividestra');

```

```

6 elseif (nargin == 3)
7     net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[
            varargin{3} 1],{'tansig','purelin','logsig'},'trainirpo',...
8         'learngdm','mse',{'fixunknowns','removeconstantrows','mapminmax'},{},{},
            'dividestra');
9 elseif (nargin == 4)
10    if isa(varargin{4},'cell') %transformation if it is a cell array
11        aux = varargin{4};
12        aux(:,length(aux)+1:length(aux)+2) = {'purelin','logsig'};
13        varargin{4} = aux;
14
15        net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[
            varargin{3} 1],varargin{4},'trainirpo',...
16        'learngdm','mse',{'fixunknowns','removeconstantrows','mapminmax'}
            ,{},{},'dividestra');
17    else
18        net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[
            varargin{3} 1],{varargin{4},'purelin','logsig'},'trainirpo',...
19        'learngdm','mse',{'fixunknowns','removeconstantrows','mapminmax'}
            ,{},{},'dividestra');
20    end
21 else
22     error('NNET:Arguments','Input arguments incorrect');
23 end
24
25 %% ADJUST PARAMETERS
26
27 % set the name of de ANN
28 net.name = 'Ordinal Neural Network';
29 % delete bias of the last hidden layer
30 net.biasConnect(net.numLayers-1) = 0;
31 % put weight of output layer to one fixed
32 net.LW{net.numLayers,net.numLayers-1} = ones(net.outputs{net.numLayers}.
            size,1);
33 % initialize with correct bias values
34 net.b{net.numLayers} = (-5*((size(net.b{net.numLayers},1)-1)/2):5:5*((
            size(net.b{net.numLayers},1)-1)/2))';
35
36 % adjust dataset divide
37 net.divideParam.trainRatio = 0.8;
38 net.divideParam.valRatio = 0.2;
39 net.divideParam.testRatio = 0;
40 net.divideParam.targets = getstra(varargin{2});

```

Listing 9.4: Archivo newoff.m

9.4. Cálculo del número óptimo de neuronas en la capa oculta

Función kfoldo

Esta función tiene como peculiaridad la potencia que Matlab tiene para realizar cálculos y procesamiento de grandes operaciones computacionales. Para comenzar, se realiza una llamada a una de las funciones implementadas la cual obtiene las clases para un conjunto de datos, en la línea 3. Seguidamente, la función *crossvalind* obtiene los índices de una forma semialeatoria realizando un K-Fold¹. A partir de esos índices, se definen los nuevos conjuntos de datos de entrenamiento y, de este modo, realizar, un número determinado de veces, la creación, entrenamiento y simulación de la red neuronal para posteriormente obtener los datos de las salidas y poder calcular la matriz de confusión. A partir de la matriz de confusión y realizando la media del número determinado de ejecuciones del mismo, se obtendrá el *MAE*. Para determinar el número de neuronas en capa oculta se seleccionará aquel número que conforme modelos que proporcionan un menor *MAE*.

```

1  %%%
2  %% establecimiento de las clases
3  classes = getstra(trainTargets);
4
5  %% K-FOLD CALCULATING
6
7  % getting indices from stratified set of targets
8  indices = crossvalind('Kfold', classes, kFold);
9
10 v = zeros(length(NH), 1);
11
12 for i=1:NH
13     aux = 0;
14     for k=1:kFold
15         % calculating and dividing folds
16         testing = (indices == k)';
17         training = ~testing;

```

¹Método explicado en el capítulo de Experimentación.


```

18
19 % Training set
20 trainX = trainInputs(:,training);
21 trainY = trainTargets(:,training);
22 trainYNew = transdata(trainY);
23
24 % Test set
25 testX = trainInputs(:,testing);
26 testY = trainTargets(:,testing);
27
28 for j=1:numIter
29     % creating the neural network
30     net = newoff(trainX,trainY,i,'tansig');
31     net.trainParam.showWindow = false; %don't show training interface
32
33     % training the net
34     net = otrain(net,trainX,trainYNew);
35
36     % simulating the net to obtain the outputs (test set)
37     testOutputs = osim(net,testX);
38
39     % calculating confusion matrix and mae
40     [c,cm,ind,per] = confusion(testY,testOutputs);
41     aux = aux+maecalc(cm, size(testX,2));
42 end
43 end
44 % obtaining mean error
45 v(NH == i) = aux/(kFold*numIter);
46 end
47
48 %% return values
49 NHO = NH(v == min(v));
50 E = min(v);

```

Listing 9.5: Archivo kfoldo.m



10 Pruebas

La fase de pruebas es una parte fundamental en el desarrollo de cualquier sistema ya que determinará la calidad del mismo. A lo largo del desarrollo del sistema se realizan un conjunto de controles que determina de esta forma la calidad del mismo y para poder detectar errores cuanto antes, ya que a medida que el desarrollo avanza, cualquier corrección del sistema puede ser mucho más costosa de solucionar tanto en tiempo como en esfuerzo.

Si se considera el sistema como un conjunto de componentes que interactúan entre sí, esto da lugar a la necesidad de probar estos componentes por separado para demostrar que cada uno de ellos funciona correctamente y que produce unos resultados satisfactorios, de esta forma es necesario incluir pruebas de los componentes antes de realizar las pruebas de todo el conjunto.

Con muchas de estas pruebas no sólo se busca un correcto funcionamiento, sino también óptimo, ya que es necesario el entrenamiento de diferentes parámetros que con determinados valores pueden dar resultados buenos pero inferiores al máximo esperado, de esta forma es necesario

realizar pruebas que permitan optimizar los valores de entrada del sistema.

Se va a mostrar una introducción teórica sobre el fundamento y finalidad de las pruebas y a continuación el catálogo de pruebas.

Las características fundamentales de las pruebas se pueden resumir en los siguientes puntos:

1. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.
2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
3. La filosofía más adecuada para las pruebas consiste en planificarlas y diseñarlas de la forma más sistemática posible para poder detectar el máximo número y variedad de defectos con el mínimo consumo de tiempo y esfuerzo.

10.1. Estrategia de pruebas

Hay que intentar diseñar un conjunto de casos de prueba que permita conseguir una confianza aceptable que encuentre los defectos existentes y a la vez que los resultados obtenidos seas lo suficientemente buenos con respecto a un margen establecido teniendo que optimizar en caso contrario.

Como entorno de pruebas para este sistema se han desarrollado una serie de pruebas denominadas *pruebas de caja negra* y *pruebas de caja blanca*. Los resultados de estas pruebas han llevado a la detección de errores que se han ido solucionando volviendo a fases anteriores y volviendo a realizar seguidamente las pruebas hasta obtener un resultado satisfactorio.

El procedimiento utilizado para eliminar errores es, por tanto, la *vuelta atrás*, localizando el síntoma que indujo a pensar que había un error e ir hacia atrás hasta llegar a la causa del mismo. A su vez, el conjunto de casos de prueba realizados se pueden clasificar en 4 tipos que se irán explicando en las siguientes subsecciones.

10.1.1. Pruebas Unitarias

El conjunto de pruebas que forman esta parte se trata de un conjunto de pruebas unitarias de los componentes del sistema y, por lo tanto, la verificación de la menor unidad del diseño del software, el módulo. Se considera como módulo a un bloque básico de construcción de programas, una parte del código que implementa una función simple o un fragmento de código que se puede compilar y/o probar independientemente. Normalmente tienen una longitud menor de 500 líneas de código. Éste se compone de dos tipos de pruebas, como se ha comentado anteriormente, pruebas estructurales o *pruebas de caja blanca* y pruebas funcionales o *pruebas de caja negra*.

Las pruebas de caja blanca están relacionadas con la fase de diseño, así, se intenta mostrar la validación de los componentes que conforman el sistema por lo que es necesario probar la funcionalidad de cada uno de los componentes por separado demostrando que cumplen con su tarea de forma óptima y no causan ningún error.

Básicamente, se intenta probar cada una de las líneas de código que conforman el sistema, normalmente las combinaciones que se pueden llevar a cabo de la ejecución del sistema son muy numerosas debido a la existencia de bucles, sentencias condicionales, etc. Sin embargo, es necesario probar que cada una de esas sentencias realiza una acción de forma correcta, es decir, será necesario realizar una cobertura de sentencias que

demuestre que ese trozo de código se ejecuta correctamente.

Los resultados de estas pruebas han llevado a la detección de errores solucionados en la fase de codificación, buscando la causa a partir de los síntomas que delataban el error y de este modo, realizando las oportunas acciones para corregirlos.

De esta forma y a medida que se iban implementando las distintas funciones y módulos de ejecución del sistema, se iba comprobando su eficiencia y estructura para comprobar que no se producía ningún tipo de error.

Estas pruebas se han realizado de la siguiente forma:

1. Se ha comprobado, para cada módulo, si se almacenaba y trataba correctamente toda la información contenida en las estructuras internas de datos.
2. Se ha comprobado que el flujo de información se realiza correctamente para todas las ramas posibles del módulo.
3. Se han realizado pruebas en los límites de los bucles.
4. Se ha comprobado la ejecución de cada una de las sentencias que forman parte del módulo, analizando la ejecución completa de cada módulo.

Las pruebas de caja negra se han desarrollado de forma paralela y posterior a la composición de todos los componentes que forman parte del sistema. Dichas pruebas se centran en las acciones visibles al usuario y salidas reconocibles desde el sistema en lo que se espera de un módulo. Por ello, se denominan pruebas funcionales, limitándose a suministrar datos como entrada y estudiar la salida sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra se apoyan en la especificación del sistema. Así, para cada módulo:

1. Se ha realizado un estudio de los posibles valores correctos en los parámetros de tipo dato, centrándose en las condiciones límite.
2. Se ha analizado qué valores se esperaban como resultado del módulo y en qué condiciones habrían de darse, se proporcionaron datos de entrada a los algoritmos y se comprobó que la salida era la esperada.
3. Para realizar todas estas pruebas se han utilizado pequeños programas de prueba en los que se cargaban las funciones o los módulos con los valores necesarios para cubrir el caso de prueba a realizar, se realizaba el método y se mostraban los resultados.

10.1.2. Pruebas de Integración

De igual modo, para comprobar que la estructura del sistema y el control del mismo se ejecutan de forma correcta, es necesario llevar a cabo un conjunto de pruebas de integración que permitan comprobar que el flujo de datos entre módulos es correcto.

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo del software una vez que se han aprobado las pruebas unitarias de dichos módulos. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso hecha en conjunto.

Las pruebas de integración se han realizado de forma ascendente, es decir, se trata mediante un proceso incremental en el cual se comprueba el siguiente módulo con el conjunto de módulos que ya han sido probados, este proceso continua hasta llegar a probar el sistema completo. De esta forma se comienza por los módulos de más bajo nivel hasta alcanzar el

sistema al completo.

De este modo, estas pruebas se han realizado de la siguiente forma:

1. Se ha comprobado en cada paso incremental que la interacción entre el módulo añadido con el conjunto de módulos ya probados funciona correctamente.
2. Se ha comprobado que el flujo de información se realiza correctamente entre los diferentes módulos.
3. Se ha comprobado que la ejecución de todas las ramas posibles del conjunto de módulos del sistema hasta que el punto alcanzado sea correcto.

Otro apartado importante a puntualizar es la integración del sistema en Matlab como parte del toolbox nnet, así, fue necesario realizar una serie de pruebas que demostraran que el sistema podría funcionar perfectamente en concordancia con el toolbox original, administrando los valores de entrada dados por éste y proporcionándole posteriormente los resultados como valores de salida.

10.1.3. Pruebas del Sistema

Tras la finalización de las pruebas anteriores se obtendrá un sistema completo funcionalmente correcto y del que han sido solventados todos los posibles errores que hayan ido surgiendo gracias a dichas pruebas, por lo que se puede comenzar una serie final de pruebas del software para verificar que se han integrado adecuadamente todos los elementos del sistema y que se realizan las funciones apropiadas.

La fase de pruebas del sistema tiene como objetivo verificar el sistema software para comprobar si éste cumple los requisitos. Dentro de esta fase se pueden desarrollar varios tipos distintos de pruebas en función de

los objetivos. Algunos tipos son, pruebas funcionales, pruebas de usabilidad, pruebas de rendimiento, pruebas de seguridad, etc.

También se ha tenido muy en cuenta la velocidad de ejecución de los algoritmos desarrollados, ya que en este tipo de sistemas característicos por una aplicación iterativa de cálculos, pequeños detalles de implementación pueden influir fuertemente en el rendimiento final. Se ha preferido una mayor velocidad de cómputo frente a la restricción del consumo de memoria. A pesar de esto, se ha realizado un control del consumo de recursos del sistema aunque no muy exhaustivo.

En este proyecto, estas pruebas se realizarán durante las pruebas de experimentación que se pueden encontrar en el siguiente capítulo. De esta forma, se cumple con el doble objetivo de ejercitar a fondo el sistema cumpliendo con las pruebas del sistema y de comprobar si la utilización de los modelos teóricos propuestos ha merecido la pena en comparación con los resultados de los que se dispone en otros medios.

10.1.4. Pruebas de Aceptación

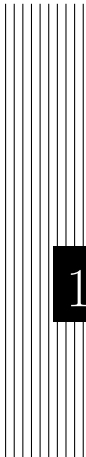
Estas pruebas las realiza el usuario final. Son básicamente pruebas funcionales sobre el sistema completado y buscan una cobertura de la especificación del sistema. Estas pruebas no se realizan durante el desarrollo pues no tendría mucho sentido de cara al usuario, sino una vez pasadas todas las pruebas de integración.

La experiencia muestra que aun después del más cuidadoso proceso de pruebas por parte del desarrollador quedan una serie de errores que sólo aparecen cuando el usuario se pone a usarlo.

Este tipo de pruebas se han desarrollado de dos formas, las primeras pruebas del proceso consisten en invitar al usuario a que venga al entorno

de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el usuario tiene siempre a un experto a mano para ayudarlo a usar el sistema y para analizar los resultados.

Las siguientes pruebas se desarrollan en el entorno del usuario, buscando un entorno que esté fuera del control. Ahí, el usuario se queda a solas con el sistema y trata de encontrarle fallos (reales o en su opinión) de los que informa al desarrollador. En este proceso, el usuario es el único con posibilidad de encontrar cualquier error que se le haya pasado al desarrollador o de buscar una mayor comodidad o eficiencia del sistema, éste es un proceso que puede llevar semanas o incluso meses dependiendo de la complejidad del sistema y del usuario final.



11 Experimentación

El capítulo de experimentación tiene como finalidad dar una idea de las estrategias seguidas en el desarrollo del software y de los resultados obtenidos en las mismas. El proceso de experimentación consiste en someter al software a una evaluación para comprobar que se comporta de acuerdo a las especificaciones y poder obtener conclusiones de su ejecución y sus resultados. La información obtenida en la experimentación ayudará a definir los parámetros idóneos para el algoritmo y de este modo mejorar su ejecución.

11.1. Configuración software y hardware

En esta sección se detalla la configuración software y hardware de la máquina donde se ejecutará la experimentación:

Software

- Sistema Operativo Ubuntu Server 10.04.2 LTS.
- Matlab Version 7.8.0.347 (R2009a).

Hardware

- Microprocesadores Intel Xeon E5405 2.00GHz.
- Memoria RAM 8 GB.

11.2. Condiciones de los experimentos

La aplicación de los experimentos se ha realizado de manera sistemática, procurando encontrar siempre los valores óptimos de los parámetros de configuración para resolver de la mejor forma posible los diversos problemas que puedan surgir. Para poder llevar a cabo dichos experimentos, es necesario establecer el diseño experimental de los mismo y utilizar un conjunto de bases de datos reales que permitan llevar a cabo las pruebas.

11.2.1. Diseño experimental

Para la cálculo del número de neuronas óptimo en la capa oculta se utilizará:

k-fold: Proceso de validación cruzada en el que el conjunto de datos se divide al azar en k conjuntos de datos. De esos k conjuntos de datos, uno es utilizado como datos de validación para testear el modelo, mientras que las restantes se usan como datos de entrenamiento. El proceso de validación cruzada es, entonces, repetido k veces, usando cada vez un conjunto de datos diferente para cada validación. Los k resultados obtenidos se combinan luego (normalmente mediante el cálculo de la media de la medida de rendimiento utilizada) para obtener una única estimación.

Una vez obtenidos los resultados, se realiza un análisis indicando si concuerdan con los esperados e incluso si han superado las expectativas o el margen de error que se producido es menor. Para valorar los modelos obtenidos se utilizará, el ratio de patrones correctamente clasificados

(CCR) y el error absoluto medio (MAE), explicados con anterioridad en el capítulo de Antecedentes.

11.3. Conjuntos de datos

Se ha llevado a cabo experimentos sobre 20 conjuntos de datos de dominio público. Estos conjuntos de datos presentan una buena diversidad respecto a diferentes características.

Los conjuntos de datos se obtendrán de los proporcionados por el grupo de investigación, éstos se encontrarán divididos mediante un Hold-out en dos, un subconjunto de entrenamiento y otro de test, siendo el de entrenamiento mayor que el de test, aproximadamente un 75 % por ciento para entrenamiento y un 25 %.

En la Tabla 11.1 se detallan las características de los conjuntos de datos empleados. Los atributos que se mostrarán en la tabla son, el nombre del conjunto de datos, número total de patrones de entrenamiento, el número total de patrones de test, el número de atributos y número de clases.

A continuación se hará una descripción de cada uno de los conjuntos de datos:

- *Automobile*: este conjunto de datos consiste en tres tipos de entidades: 1) la especificación de una automóvil en términos de varias características. 2) su clasificación de riesgo asignado inseguro, 3) sus pérdidas normalizadas en uso en comparación con otros coches. La segunda clasificación corresponde con el grado en el cual un automóvil tiene más riesgo del que su precio indica. Entonces, si es más arriesgado (o menos), ésto se ajusta por el movimiento hacia arriba (o hacia abajo) de la escala. A este proceso lo suelen llamar

| Nombre | N.T.P. Entrenamiento | N.T.P. Test | Núm. Atributos | Núm. Clases |
|-------------------|----------------------|-------------|----------------|-------------|
| Automobile | 1530 | 520 | 71 | 6 |
| Balance Scale | 4680 | 1570 | 4 | 3 |
| Bondrate | 420 | 150 | 37 | 5 |
| Car | 12960 | 4320 | 21 | 4 |
| Contact Lenses | 180 | 60 | 6 | 3 |
| Depression | 1010 | 1320 | 7 | 3 |
| ERA | 7500 | 2500 | 4 | 9 |
| ESL | 3660 | 1220 | 4 | 9 |
| Eucalyptus | 5520 | 1840 | 91 | 5 |
| LEV | 7500 | 2500 | 4 | 5 |
| New Thyroid | 1610 | 540 | 5 | 3 |
| Pasture | 270 | 90 | 25 | 3 |
| Squash Stored | 390 | 130 | 51 | 3 |
| Squash Unstored | 390 | 130 | 52 | 3 |
| SWD | 7500 | 2500 | 10 | 4 |
| TAE | 1130 | 380 | 54 | 3 |
| Thyroid | 54000 | 18000 | 21 | 3 |
| Winequality Red | 11990 | 4000 | 11 | 6 |
| Winequality White | 36730 | 12250 | 11 | 7 |

Tabla 11.1: Conjuntos de datos usados.

”simboling”.

El tercer factor es el pago relativo medio de pérdida por vehículos inseguros por año. Este valor está normalizado para todos los automóviles sin un tamaño particular en la clasificación (pequeño de dos puertas, furgonetas, deportivos, etc.) y representa la media de pérdidas por coche por año.

- *Balance Scale*: este conjunto de datos se generó a partir de los resultados de un modelo experimental psicológico. Cada ejemplo se clasifica como la punta de la balanza a la derecha, a la izquierda o balanceado. Los atributos son el peso de la punta izquierda, el de la derecha y la distancia correcta. El camino correcto para encontrar la clase es el mayor entre (distancia izquierda * peso izquierda)

y (distancia derecha * peso derecha). Si son iguales, se encuentra balanceada.

- *Bondrate*: conjunto de datos basado en el rango de bondad que existe en datos de varias ciudades a partir de datos como el número de personas residentes, el rango de ingresos entrantes, etc.
- *Car*: el conjunto de datos de evaluación de coches se derivó desde un modelo simple de decisión jerárquico. Debido a la estructura conocida, este conjunto de datos puede ser particularmente útil para testeo de inducción constructiva y métodos de descubrimiento de estructuras.
- *Contact Lenses*: este conjunto de datos contiene 3 tipos de clases, la primera clase se refiere al paciente que debería llevar unas lentes de contacto duras, la segunda clase se refiere al paciente que debería de llevar unas lentes de contacto blandas y por último, la tercera clase sería que el paciente no debe de llevar lentes de contacto.
- *Depression*: conjunto de datos que contiene 3 tipos de clases, unidad espacial con depresión, unidad espacial sin depresión y unidad espacial donde no hay depresión. Este conjunto de datos proviene de casos reales obtenidos en Andalucía, al sur de España.
- *ERA*: el ECMWF ERA-40 Re-Analysis Project consiste en un número de conjuntos de datos climáticos que abarcan el periodo entre mediados de 1957 a agosto de 2002 usando un modelo consistente.
- *ESL*: el conjunto de datos ESL (Employee Selection) contiene perfiles de solicitantes de ciertos puestos de trabajo industriales. Expertos psicólogos de una compañía de reclutamiento, basándose en resultados de test psicométricos y entrevistas con los candidatos, determinaron los valores de los atributos de entrada. La salida es la clasificación general correspondiente a los grados de aptitud del candidato para ese tipo de trabajo.

- *Eucalyptus*: este conjunto de datos tiene como objetivo determinar qué lotes de semillas son mejores para la conservación del suelo en una región montañosa y en una estación seca. La determinación de esto se haya mediante la medición de parámetros como la medición de la altura, diámetro por altura, supervivencia y otros factores contribuyentes.
- *LEV*: conjunto de datos que se usa para entrenamiento y simulación de redes neuronales.
- *New Thyroid*: este conjunto de datos esta reducido a partir de otro y provee resultados de clasificación del funcionamiento normal, hipoadactivo o hiperactivo de la glándula tiroides en base a 5 atributos.
- *Pasture*: el conjunto de datos de producción de pastos tiene como objetivo predecir la producción de pasto mediante una variedad de factores biofísicos. Las variables de vegetación y suelo de las zonas de pastoreo de North Island, país montañoso el cual tiene diversas aplicaciones tales como aplicación de fertilizantes o paso con animales de carga.
- *Squash Stored*: este conjunto de datos tiene como objetivos determinar qué variables antes de la cosecha dan un buen sabor a la calabaza después de diversos periodos de tiempo almacenada. Esto se determina mediante una medida de aceptabilidad categorizada como aceptable, no aceptable o excelente.
- *Squash Unstored*: este conjunto de datos tiene como objetivos los mismo que el conjunto de datos anterior, la única diferencia es que en éste, el fruto no se almacena antes de medirse, por lo que carece de uno de los atributos, el peso de la fruta después de su almacenaje.
- *SWD*: el conjunto de datos Social Workers Decisions (Ordinal SWD) contiene las evaluaciones del mundo real de trabajadores sociales cualificados en relación con el riesgo al que se enfrentan los niños

si ellos se quedasen con sus familias en sus casas. Esta valoración de la evaluación de riesgos es, a menudo, presentada a las cortes judiciales para ayudar a decidir que es lo mejor para un niño que presuntamente padece abusos o está descuidado.

- *TAE*: el conjunto de datos Teaching Assistant Evaluation consiste en evaluaciones del desempeño docente sobre tres semestres regulares y dos semestres de verano de asistencia educacional asignada al Departamento de Estadística de la Universidad de Wisconsin (Madison). Las puntuaciones se dividieron en tres categorías de iguales tamaños (bajo (1), medio (2), alto (3)) para formar la variable de clases.
- *Thyroid*: este conjunto de datos es uno de las variadas bases de datos sobre Tiroides disponible en el repositorio de la UCI. La tarea que tiene es detectar si un paciente dado es normal (1) o sufre de hipertiroidismo (2) o de hipotiroidismo (3).
- *Winequality Red*: este conjunto de datos se encuentra relacionado a la variante roja del vino de Portuguese Vinho Verde. Debido a los problemas de privacidad y logística, solo las variables físico-químicas (entradas) y sensoriales (salidas) están disponibles. Este conjunto de datos se puede usar para tareas tanto de clasificación como de regresión. Las clases están ordenadas y no balanceadas.
- *Winequality White*: este conjunto de datos se encuentra relacionado a la variante blanca del vino de Portuguese Vinho Verde. Debido a los problemas de privacidad y logística, solo las variables físico-químicas (entradas) y sensoriales (salidas) están disponibles. Este conjunto de datos se puede usar para tareas tanto de clasificación como de regresión. Las clases están ordenadas y no balanceadas.

El aprendizaje a partir de datos no balanceados, como son algunos de los conjuntos de datos utilizados, puede resultar dificultoso. Se dice que

un conjunto de datos se encuentra no balanceado cuando presenta una desigualdad marcada en la distribución de las clases, es decir, que hay diferencias acusadas entre el número de ejemplos de las distintas clases. El problema surge porque, normalmente, los algoritmos de aprendizaje y clasificación en computación genética o evolutiva o mediante redes neuronales, que se usan para construir los clasificadores, tienden a centrarse en las clases mayoritarias y a pasar por alto las minoritarias. La consecuencia directa de este comportamiento es que el clasificador no será capaz de clasificar correctamente los ejemplos correspondientes a las clases menos frecuentes.

La estrategia de aprendizaje no es la única cuestión que hay que afrontar cuando se trabaja con datos no balanceados. Otro punto fundamental es el de cómo medir la calidad de los clasificadores obtenidos. En clasificación, el rendimiento se suele medir en términos de la tasa de aciertos, es decir, la proporción de ejemplos correctamente clasificados o en términos de errores, ya sean medios o absolutos.

El tamaño del conjunto de datos es hasta el momento el factor más relevante que limita la resolución de problemas de clasificación mediante este tipo de métodos debido a que el tiempo computacional requerido para clasificar grandes conjuntos de datos es excesivamente alto.



12 Resultados

En este capítulo se detallarán los resultados de las ejecuciones del sistema con el objetivo de analizar los resultados y de este modo, obtener conclusiones que apoyen los objetivos del proyecto.

12.1. Resultados individuales

En esta sección se presentarán los resultados individuales generados por cada conjunto de datos al aplicar el algoritmo realizado para el entrenamiento y simulación de las redes neuronales artificiales ya sea de forma ordinal o nominal.

Para la obtención de los datos se ha realizado un 10-fold de los conjuntos de datos de entrenamiento y test de los que se partía. Además se ha repetido 3 veces para que el resultado sea más fiable. Los valores que podrá tomar k serán potencias de 2 desde 0 hasta 5, es decir, $2^0 - 2^5$.

A excepción del conjunto de datos *Depression*, que sólo tiene un par completo, el resto de conjuntos de datos se encuentra dividido en 10 ficheros de datos en forma de pares entrenamiento-test.

12.1.1. Clasificación Nominal

Para cada uno de los conjuntos se mostrarán los datos obtenidos correspondientes a aplicar un modelo de clasificación donde no se tiene en cuenta el orden en las etiquetas y la matriz de confusión del mejor resultado.

12.1.1.1. Conjunto de datos Automobile

En la Tabla 12.1 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Automobile, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.1.

12.1.1.2. Conjunto de datos Balance Scale

En la Tabla 12.2 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Balance Scale, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.2.

12.1.1.3. Conjunto de datos Bondrate

En la Tabla 12.3 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.480769 | 0.653846 | 32 | 93.094352 |
| 2 | 0.538462 | 0.788462 | 32 | 94.889793 |
| 3 | 0.730769 | 0.365385 | 32 | 96.891213 |
| 4 | 0.653846 | 0.442308 | 32 | 82.703460 |
| 5 | 0.423077 | 0.730769 | 32 | 83.732745 |
| 6 | 0.711538 | 0.365385 | 32 | 85.084591 |
| 7 | 0.596154 | 0.653846 | 32 | 74.788456 |
| 8 | 0.480769 | 0.711538 | 32 | 76.002375 |
| 9 | 0.730769 | 0.326923 | 32 | 77.727133 |
| 10 | 0.576923 | 0.750000 | 16 | 63.014245 |
| 11 | 0.519231 | 0.673077 | 16 | 64.158049 |
| 12 | 0.538462 | 0.692308 | 16 | 65.511677 |
| 13 | 0.538462 | 0.634615 | 32 | 67.701905 |
| 14 | 0.576923 | 0.615385 | 32 | 68.513606 |
| 15 | 0.692308 | 0.480769 | 32 | 69.283688 |
| 16 | 0.596154 | 0.596154 | 32 | 56.909019 |
| 17 | 0.673077 | 0.480769 | 32 | 57.548274 |
| 18 | 0.653846 | 0.480769 | 32 | 58.238033 |
| 19 | 0.653846 | 0.500000 | 32 | 55.620439 |
| 20 | 0.692308 | 0.442308 | 32 | 56.458364 |
| 21 | 0.538462 | 0.692308 | 32 | 57.064387 |
| 22 | 0.576923 | 0.596154 | 32 | 54.333290 |
| 23 | 0.615385 | 0.653846 | 32 | 54.953139 |
| 24 | 0.596154 | 0.557692 | 32 | 55.760442 |
| 25 | 0.519231 | 0.769231 | 32 | 60.943545 |
| 26 | 0.500000 | 0.788462 | 32 | 61.680323 |
| 27 | 0.557692 | 0.653846 | 32 | 62.469532 |
| 28 | 0.634615 | 0.461538 | 32 | 55.954954 |
| 29 | 0.826923 | 0.250000 | 32 | 56.800467 |
| 30 | 0.576923 | 0.596154 | 32 | 57.397560 |

Tabla 12.1: Resultados individuales. Conjunto de datos Automobile. Clasificación nominal.

Confusion Matrix

| | | | | | | | |
|---|--------------|----------------|----------------|---------------|----------------|--------------|----------------|
| 1 | 0 0.0% | 1 1.9% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0.0% 100% |
| 2 | 1 1.9% | 5 9.6% | 1 1.9% | 0 0.0% | 0 0.0% | 0 0.0% | 71.4% 28.6% |
| 3 | 0 0.0% | 0 0.0% | 13 25.0% | 1 1.9% | 0 0.0% | 0 0.0% | 92.9% 7.1% |
| 4 | 0 0.0% | 0 0.0% | 0 0.0% | 12 23.1% | 2 3.8% | 0 0.0% | 85.7% 14.3% |
| 5 | 0 0.0% | 0 0.0% | 2 3.8% | 0 0.0% | 6 11.5% | 0 0.0% | 75.0% 25.0% |
| 6 | 0 0.0% | 0 0.0% | 1 1.9% | 0 0.0% | 0 0.0% | 7 13.5% | 87.5% 12.5% |
| | 0.0% 100% | 83.3% 16.7% | 76.5% 23.5% | 92.3% 7.7% | 75.0% 25.0% | 100% 0.0% | 82.7% 17.3% |
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| | Target Class | | | | | | |

Output Class

Figura 12.1: Matriz de confusión. Conjunto de datos Automobile. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.910828 | 0.101911 | 32 | 57.696494 |
| 2 | 0.942675 | 0.082803 | 32 | 58.481274 |
| 3 | 0.904459 | 0.114650 | 32 | 59.114637 |
| 4 | 0.942675 | 0.063694 | 32 | 57.062635 |
| 5 | 0.949045 | 0.057325 | 32 | 57.857067 |
| 6 | 0.968153 | 0.038217 | 32 | 58.746454 |
| 7 | 0.904459 | 0.114650 | 16 | 56.484056 |
| 8 | 0.917197 | 0.089172 | 16 | 57.043869 |
| 9 | 0.929936 | 0.082803 | 16 | 57.741806 |
| 10 | 0.923567 | 0.076433 | 16 | 53.879592 |
| 11 | 0.961783 | 0.057325 | 16 | 54.778862 |
| 12 | 0.949045 | 0.063694 | 16 | 55.408901 |
| 13 | 0.949045 | 0.063694 | 16 | 54.753924 |
| 14 | 0.853503 | 0.210191 | 16 | 55.274971 |
| 15 | 0.898089 | 0.121019 | 16 | 55.811972 |
| 16 | 0.929936 | 0.076433 | 32 | 57.704460 |
| 17 | 0.923567 | 0.082803 | 32 | 58.302384 |
| 18 | 0.936306 | 0.063694 | 32 | 59.076371 |
| 19 | 0.955414 | 0.050955 | 16 | 59.304432 |
| 20 | 0.904459 | 0.108280 | 16 | 59.867301 |
| 21 | 0.968153 | 0.038217 | 16 | 60.641971 |
| 22 | 0.968153 | 0.038217 | 32 | 54.123307 |
| 23 | 0.949045 | 0.063694 | 32 | 55.157696 |
| 24 | 0.904459 | 0.101911 | 32 | 55.683436 |
| 25 | 0.898089 | 0.114650 | 32 | 58.392956 |
| 26 | 0.910828 | 0.101911 | 32 | 58.976779 |
| 27 | 0.949045 | 0.050955 | 32 | 59.621647 |
| 28 | 0.923567 | 0.089172 | 16 | 56.255009 |
| 29 | 0.904459 | 0.108280 | 16 | 56.830125 |
| 30 | 0.929936 | 0.089172 | 16 | 57.412558 |

Tabla 12.2: Resultados individuales. Conjunto de datos Balance Scale. Clasificación nominal.

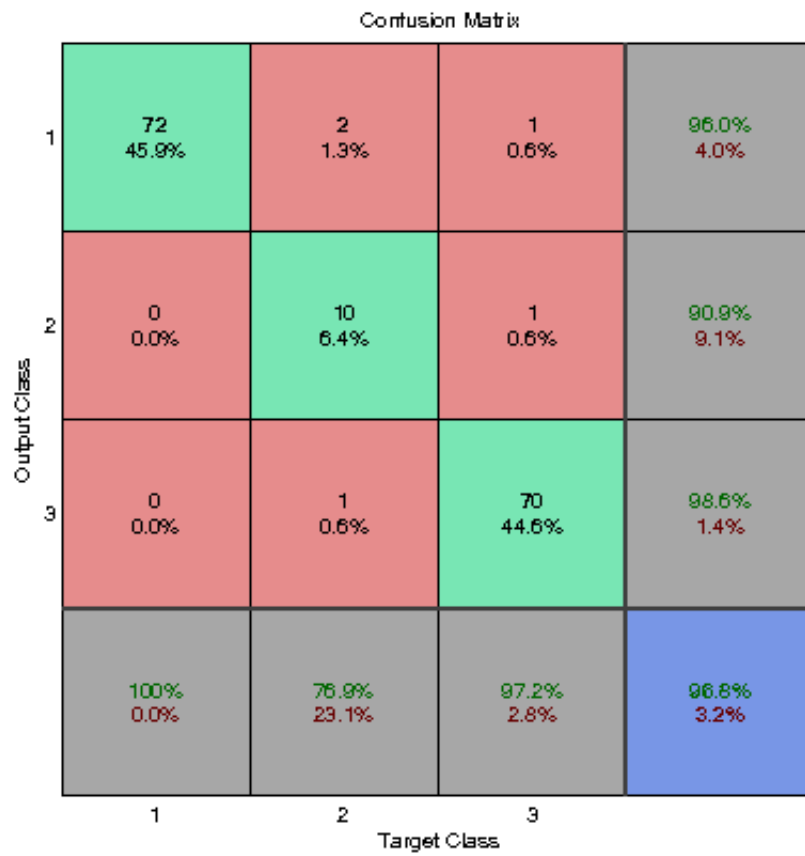


Figura 12.2: Matriz de confusión. Conjunto de datos Balance Scale. Clasificación nominal.

artificial no ordinal para el conjunto de datos Bondrate, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.666667 | 0.400000 | 16 | 32.132868 |
| 2 | 0.533333 | 0.666667 | 16 | 32.659737 |
| 3 | 0.600000 | 0.466667 | 16 | 33.241910 |
| 4 | 0.666667 | 0.400000 | 4 | 31.763658 |
| 5 | 0.533333 | 0.733333 | 4 | 32.362206 |
| 6 | 0.533333 | 0.666667 | 4 | 32.926967 |
| 7 | 0.533333 | 0.600000 | 16 | 31.035668 |
| 8 | 0.600000 | 0.533333 | 16 | 31.556096 |
| 9 | 0.600000 | 0.533333 | 16 | 32.081018 |
| 10 | 0.600000 | 0.600000 | 16 | 31.820959 |
| 11 | 0.600000 | 0.600000 | 16 | 32.360409 |
| 12 | 0.666667 | 0.466667 | 16 | 32.891236 |
| 13 | 0.533333 | 0.666667 | 32 | 33.563667 |
| 14 | 0.400000 | 0.866667 | 32 | 34.098812 |
| 15 | 0.533333 | 0.600000 | 32 | 34.634752 |
| 16 | 0.533333 | 0.666667 | 32 | 31.884221 |
| 17 | 0.533333 | 0.666667 | 32 | 32.405902 |
| 18 | 0.666667 | 0.400000 | 32 | 32.961736 |
| 19 | 0.600000 | 0.600000 | 32 | 31.754275 |
| 20 | 0.666667 | 0.533333 | 32 | 32.286801 |
| 21 | 0.533333 | 0.533333 | 32 | 32.830340 |
| 22 | 0.600000 | 0.600000 | 16 | 32.910873 |
| 23 | 0.533333 | 0.666667 | 16 | 33.731668 |
| 24 | 0.333333 | 0.800000 | 16 | 34.463909 |
| 25 | 0.533333 | 0.733333 | 32 | 36.406399 |
| 26 | 0.533333 | 0.800000 | 32 | 37.316972 |
| 27 | 0.466667 | 0.800000 | 32 | 38.429347 |
| 28 | 0.600000 | 0.533333 | 8 | 35.388399 |
| 29 | 0.466667 | 0.600000 | 8 | 36.044441 |
| 30 | 0.600000 | 0.466667 | 8 | 36.831163 |

Tabla 12.3: Resultados individuales. Conjunto de datos Bondrate. Clasificación nominal.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.3.

Confusion Matrix

| | | | | | | |
|---|--------------|----------------|----------------|--------------|--------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 1 6.7% | 8 53.3% | 2 13.3% | 0 0.0% | 0 0.0% | 72.7% 27.3% |
| 3 | 0 0.0% | 0 0.0% | 1 6.7% | 0 0.0% | 0 0.0% | 100% 0.0% |
| 4 | 0 0.0% | 1 6.7% | 0 0.0% | 1 6.7% | 1 6.7% | 33.3% 66.7% |
| 5 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 0.0% 100% | 88.9% 11.1% | 33.3% 66.7% | 100% 0.0% | 0.0% 100% | 66.7% 33.3% |
| | 1 | 2 | 3 | 4 | 5 | |
| | Target Class | | | | | |

Output Class

Figura 12.3: Matriz de confusión. Conjunto de datos Bondrate. Clasificación nominal.

12.1.1.4. Conjunto de datos Car

En la Tabla 12.4 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Car, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.4.

Confusion Matrix

| | | | | | | |
|--------------|---|---------------|---------------|---------------|--------------|---------------|
| Output Class | 1 | 302 69.9% | 0 0.0% | 0 0.0% | 0 0.0% | 100% 0.0% |
| | 2 | 1 0.2% | 95 22.0% | 1 0.2% | 0 0.0% | 97.9% 2.1% |
| | 3 | 0 0.0% | 1 0.2% | 16 3.7% | 0 0.0% | 94.1% 5.9% |
| | 4 | 0 0.0% | 0 0.0% | 0 0.0% | 16 3.7% | 100% 0.0% |
| | | 99.7% 0.3% | 99.0% 1.0% | 94.1% 5.9% | 100% 0.0% | 99.3% 0.7% |
| | | 1 | 2 | 3 | 4 | |
| | | Target Class | | | | |

Figura 12.4: Matriz de confusión. Conjunto de datos Car. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.958333 | 0.043981 | 32 | 226.393012 |
| 2 | 0.944444 | 0.062500 | 32 | 228.835668 |
| 3 | 0.965278 | 0.037037 | 32 | 230.871909 |
| 4 | 0.979167 | 0.025463 | 32 | 202.167168 |
| 5 | 0.983796 | 0.016204 | 32 | 205.027874 |
| 6 | 0.981481 | 0.023148 | 32 | 207.829281 |
| 7 | 0.972222 | 0.032407 | 32 | 174.261568 |
| 8 | 0.974537 | 0.032407 | 32 | 178.433216 |
| 9 | 0.967593 | 0.037037 | 32 | 180.947405 |
| 10 | 0.979167 | 0.027778 | 16 | 194.213574 |
| 11 | 0.983796 | 0.018519 | 16 | 196.910010 |
| 12 | 0.967593 | 0.037037 | 16 | 199.067734 |
| 13 | 0.981481 | 0.023148 | 32 | 182.252793 |
| 14 | 0.962963 | 0.041667 | 32 | 184.494332 |
| 15 | 0.983796 | 0.018519 | 32 | 186.581949 |
| 16 | 0.981481 | 0.018519 | 32 | 223.682423 |
| 17 | 0.988426 | 0.011574 | 32 | 225.426108 |
| 18 | 0.983796 | 0.016204 | 32 | 226.728119 |
| 19 | 0.974537 | 0.027778 | 32 | 217.977243 |
| 20 | 0.967593 | 0.043981 | 32 | 219.510728 |
| 21 | 0.972222 | 0.032407 | 32 | 220.887031 |
| 22 | 0.993056 | 0.006944 | 32 | 258.591102 |
| 23 | 0.993056 | 0.009259 | 32 | 260.277011 |
| 24 | 0.986111 | 0.016204 | 32 | 261.883019 |
| 25 | 0.983796 | 0.018519 | 32 | 244.065082 |
| 26 | 0.979167 | 0.027778 | 32 | 248.886129 |
| 27 | 0.986111 | 0.013889 | 32 | 253.490846 |
| 28 | 0.979167 | 0.023148 | 32 | 226.566350 |
| 29 | 0.986111 | 0.016204 | 32 | 229.881659 |
| 30 | 0.990741 | 0.009259 | 32 | 233.665970 |

Tabla 12.4: Resultados individuales. Conjunto de datos Car. Clasificación nominal.

12.1.1.5. Conjunto de datos Contact lenses

En la Tabla 12.5 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Contact Lenses, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.5.



Figura 12.5: Matriz de confusión. Conjunto de datos Contact Lenses. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.666667 | 0.333333 | 32 | 37.287027 |
| 2 | 1.000000 | 0.000000 | 32 | 37.774725 |
| 3 | 0.833333 | 0.333333 | 32 | 38.320482 |
| 4 | 0.333333 | 1.166667 | 1 | 35.215124 |
| 5 | 0.833333 | 0.333333 | 1 | 35.796073 |
| 6 | 0.166667 | 1.500000 | 1 | 36.343266 |
| 7 | 0.833333 | 0.166667 | 32 | 33.590399 |
| 8 | 0.500000 | 0.500000 | 32 | 34.065188 |
| 9 | 0.333333 | 1.000000 | 32 | 34.540757 |
| 10 | 0.666667 | 0.500000 | 16 | 34.664080 |
| 11 | 0.666667 | 0.333333 | 16 | 35.187971 |
| 12 | 0.500000 | 0.666667 | 16 | 35.671418 |
| 13 | 0.833333 | 0.333333 | 32 | 33.283002 |
| 14 | 0.666667 | 0.666667 | 32 | 33.828097 |
| 15 | 0.833333 | 0.333333 | 32 | 34.321844 |
| 16 | 0.666667 | 0.333333 | 2 | 35.278676 |
| 17 | 0.666667 | 0.500000 | 2 | 35.753695 |
| 18 | 1.000000 | 0.000000 | 2 | 36.317772 |
| 19 | 0.666667 | 0.500000 | 32 | 36.388497 |
| 20 | 0.500000 | 0.833333 | 32 | 36.888895 |
| 21 | 0.500000 | 0.833333 | 32 | 37.425510 |
| 22 | 0.666667 | 0.333333 | 32 | 34.556766 |
| 23 | 0.500000 | 0.500000 | 32 | 35.035464 |
| 24 | 0.833333 | 0.166667 | 32 | 35.573715 |
| 25 | 0.333333 | 0.833333 | 32 | 35.650899 |
| 26 | 0.666667 | 0.500000 | 32 | 36.136839 |
| 27 | 0.666667 | 0.500000 | 32 | 36.640007 |
| 28 | 0.833333 | 0.333333 | 32 | 34.155435 |
| 29 | 0.833333 | 0.333333 | 32 | 34.648851 |
| 30 | 0.833333 | 0.333333 | 32 | 35.169734 |

Tabla 12.5: Resultados individuales. Conjunto de datos Contact Lenses. Clasificación nominal.

12.1.1.6. Conjunto de datos Depression

En la Tabla 12.6 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Depression, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.772727 | 0.280303 | 32 | 31.300865 |
| 2 | 0.734848 | 0.340909 | 32 | 31.798644 |
| 3 | 0.757576 | 0.318182 | 32 | 32.305562 |

Tabla 12.6: Resultados individuales. Conjunto de datos Depression. Clasificación nominal.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.6.

12.1.1.7. Conjunto de datos ERA

En la Tabla 12.7 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos ERA, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.7.

12.1.1.8. Conjunto de datos ESL

En la Tabla 12.8 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos ESL, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa

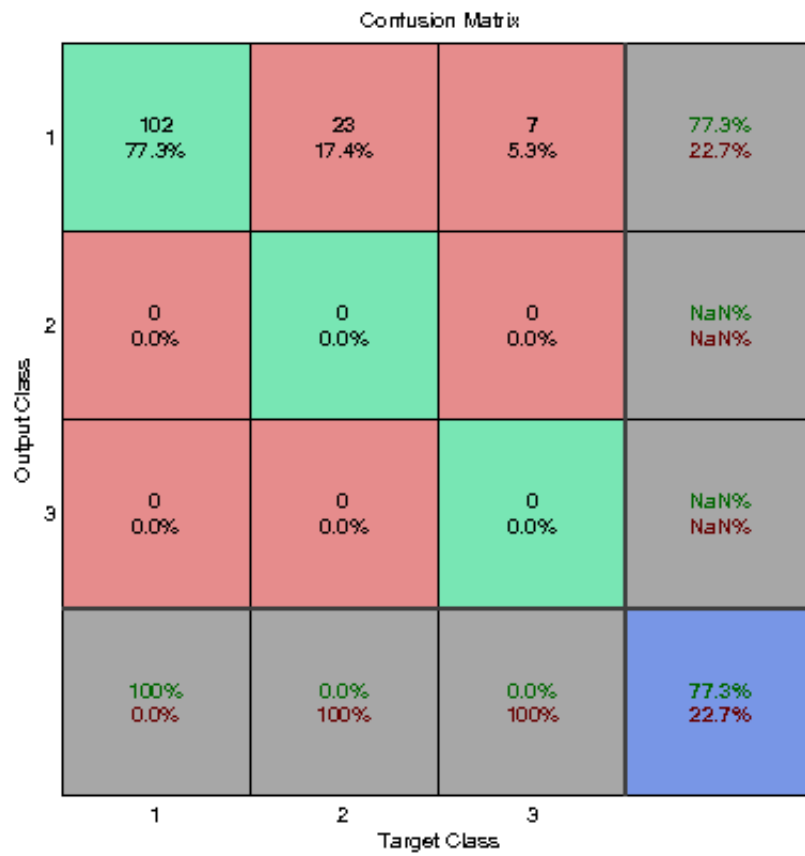


Figura 12.6: Matriz de confusión. Conjunto de datos Depression. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.260000 | 1.408000 | 8 | 94.649243 |
| 2 | 0.300000 | 1.196000 | 8 | 95.796501 |
| 3 | 0.260000 | 1.360000 | 8 | 96.789635 |
| 4 | 0.260000 | 1.312000 | 8 | 99.080923 |
| 5 | 0.264000 | 1.248000 | 8 | 100.664679 |
| 6 | 0.244000 | 1.260000 | 8 | 101.947881 |
| 7 | 0.268000 | 1.264000 | 32 | 108.534575 |
| 8 | 0.220000 | 1.336000 | 32 | 109.572379 |
| 9 | 0.284000 | 1.288000 | 32 | 110.630043 |
| 10 | 0.252000 | 1.504000 | 8 | 98.960875 |
| 11 | 0.220000 | 1.404000 | 8 | 99.985403 |
| 12 | 0.252000 | 1.300000 | 8 | 100.972921 |
| 13 | 0.216000 | 1.376000 | 8 | 93.895894 |
| 14 | 0.224000 | 1.356000 | 8 | 94.921070 |
| 15 | 0.244000 | 1.392000 | 8 | 95.808354 |
| 16 | 0.240000 | 1.340000 | 8 | 94.701149 |
| 17 | 0.280000 | 1.216000 | 8 | 95.659855 |
| 18 | 0.240000 | 1.392000 | 8 | 96.469038 |
| 19 | 0.248000 | 1.364000 | 32 | 96.448068 |
| 20 | 0.228000 | 1.388000 | 32 | 97.569403 |
| 21 | 0.164000 | 1.996000 | 32 | 98.379240 |
| 22 | 0.224000 | 1.552000 | 8 | 101.227930 |
| 23 | 0.272000 | 1.388000 | 8 | 102.495461 |
| 24 | 0.272000 | 1.280000 | 8 | 103.388947 |
| 25 | 0.248000 | 1.408000 | 32 | 95.006399 |
| 26 | 0.264000 | 1.396000 | 32 | 96.430592 |
| 27 | 0.252000 | 1.464000 | 32 | 97.639657 |
| 28 | 0.280000 | 1.196000 | 8 | 99.315351 |
| 29 | 0.268000 | 1.216000 | 8 | 100.248671 |
| 30 | 0.232000 | 1.208000 | 8 | 101.279933 |

Tabla 12.7: Resultados individuales. Conjunto de datos ERA. Clasificación nominal.

Confusion Matrix

| | | | | | | | | | | | |
|--------------|---|----------------|----------------|----------------|----------------|----------------|---------------|----------------|--------------|--------------|----------------|
| Output Class | 1 | 12 4.8% | 4 1.6% | 4 1.6% | 0 0.0% | 1 0.4% | 1 0.4% | 0 0.0% | 0 0.0% | 0 0.0% | 54.5% 45.5% |
| | 2 | 7 2.8% | 13 5.2% | 10 4.0% | 4 1.6% | 0 0.0% | 2 0.8% | 1 0.4% | 0 0.0% | 0 0.0% | 35.1% 64.9% |
| | 3 | 0 0.0% | 3 1.2% | 14 5.6% | 12 4.8% | 8 3.2% | 4 1.6% | 2 0.8% | 0 0.0% | 0 0.0% | 32.6% 67.4% |
| | 4 | 3 1.2% | 10 4.0% | 11 4.4% | 13 5.2% | 16 6.4% | 6 2.4% | 6 2.4% | 1 0.4% | 0 0.0% | 19.7% 80.3% |
| | 5 | 1 0.4% | 5 2.0% | 5 2.0% | 8 3.2% | 8 3.2% | 8 3.2% | 4 1.6% | 0 0.0% | 0 0.0% | 20.5% 79.5% |
| | 6 | 0 0.0% | 1 0.4% | 0 0.0% | 3 1.2% | 4 1.6% | 2 0.8% | 0 0.0% | 0 0.0% | 0 0.0% | 20.0% 80.0% |
| | 7 | 0 0.0% | 0 0.0% | 1 0.4% | 3 1.2% | 3 1.2% | 6 2.4% | 9 3.6% | 6 2.4% | 0 0.0% | 32.1% 67.9% |
| | 8 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 9 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 1 0.4% | 4 1.6% | 80.0% 20.0% |
| | | 52.2% 47.8% | 36.1% 63.9% | 31.1% 68.9% | 30.2% 69.8% | 20.0% 80.0% | 6.9% 93.1% | 40.9% 59.1% | 0.0% 100% | 100% 0.0% | 30.0% 70.0% |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| | | Target Class | | | | | | | | | |

Figura 12.7: Matriz de confusión. Conjunto de datos ERA. Clasificación nominal.

oculta calculado y el tiempo total de la ejecución.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.704918 | 0.311475 | 32 | 94.380922 |
| 2 | 0.713115 | 0.311475 | 32 | 95.571342 |
| 3 | 0.704918 | 0.311475 | 32 | 96.393981 |
| 4 | 0.696721 | 0.319672 | 16 | 100.530087 |
| 5 | 0.655738 | 0.377049 | 16 | 101.459313 |
| 6 | 0.713115 | 0.303279 | 16 | 103.064496 |
| 7 | 0.696721 | 0.319672 | 32 | 93.188031 |
| 8 | 0.655738 | 0.360656 | 32 | 94.382841 |
| 9 | 0.647541 | 0.360656 | 32 | 95.480957 |
| 10 | 0.713115 | 0.303279 | 16 | 104.481647 |
| 11 | 0.696721 | 0.336066 | 16 | 105.388066 |
| 12 | 0.745902 | 0.286885 | 16 | 106.853371 |
| 13 | 0.737705 | 0.286885 | 16 | 94.363731 |
| 14 | 0.704918 | 0.311475 | 16 | 95.263599 |
| 15 | 0.704918 | 0.319672 | 16 | 96.607124 |
| 16 | 0.688525 | 0.319672 | 32 | 98.371956 |
| 17 | 0.721311 | 0.286885 | 32 | 99.422718 |
| 18 | 0.721311 | 0.303279 | 32 | 100.451141 |
| 19 | 0.672131 | 0.336066 | 32 | 100.510673 |
| 20 | 0.672131 | 0.352459 | 32 | 101.379594 |
| 21 | 0.614754 | 0.418033 | 32 | 102.258125 |
| 22 | 0.704918 | 0.344262 | 32 | 99.686464 |
| 23 | 0.647541 | 0.393443 | 32 | 100.819624 |
| 24 | 0.696721 | 0.336066 | 32 | 101.956361 |
| 25 | 0.696721 | 0.327869 | 16 | 100.868638 |
| 26 | 0.721311 | 0.303279 | 16 | 102.200773 |
| 27 | 0.713115 | 0.311475 | 16 | 103.114766 |
| 28 | 0.672131 | 0.360656 | 32 | 92.145573 |
| 29 | 0.704918 | 0.327869 | 32 | 93.067119 |
| 30 | 0.696721 | 0.336066 | 32 | 94.263664 |

Tabla 12.8: Resultados individuales. Conjunto de datos ESL. Clasificación nominal.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.8.

| | | | | | | | | | | |
|--------------|---|--------------|----------------|----------------|----------------|----------------|----------------|----------------|--------------|----------------|
| Output Class | 1 | 0 0.0% | 1 0.8% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0.0% 100% |
| | 2 | 0 0.0% | 2 1.6% | 3 2.5% | 1 0.8% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 33.3% 66.7% |
| | 3 | 0 0.0% | 0 0.0% | 5 4.1% | 1 0.8% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 83.3% 16.7% |
| | 4 | 0 0.0% | 0 0.0% | 1 0.8% | 18 14.8% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 94.7% 5.3% |
| | 5 | 0 0.0% | 0 0.0% | 0 0.0% | 3 2.5% | 26 21.3% | 4 3.3% | 0 0.0% | 0 0.0% | 78.8% 21.2% |
| | 6 | 0 0.0% | 0 0.0% | 0 0.0% | 2 1.6% | 3 2.5% | 27 22.1% | 3 2.5% | 0 0.0% | 77.1% 22.9% |
| | 7 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 3 2.5% | 13 10.7% | 5 4.1% | 59.1% 40.9% |
| | 8 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 9 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | | NaN% NaN% | 66.7% 33.3% | 55.6% 44.4% | 72.0% 28.0% | 89.7% 10.3% | 79.4% 20.6% | 81.2% 18.8% | 0.0% 100% | 0.0% 100% |
| Target Class | | | | | | | | | | |

Figura 12.8: Matriz de confusión. Conjunto de datos ESL. Clasificación nominal.

12.1.1.9. Conjunto de datos Eucalyptus

En la Tabla 12.9 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Eucalyptus, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.9.

Confusion Matrix

| | | | | | | |
|---|---------------|----------------|----------------|----------------|----------------|----------------|
| 1 | 42 22.8% | 8 4.3% | 1 0.5% | 2 1.1% | 1 0.5% | 77.8% 22.2% |
| 2 | 3 1.6% | 10 5.4% | 4 2.2% | 0 0.0% | 0 0.0% | 58.8% 41.2% |
| 3 | 0 0.0% | 7 3.8% | 24 13.0% | 9 4.9% | 1 0.5% | 58.5% 41.5% |
| 4 | 0 0.0% | 1 0.5% | 1 0.5% | 34 18.5% | 13 7.1% | 68.4% 30.6% |
| 5 | 0 0.0% | 1 0.5% | 2 1.1% | 9 4.9% | 11 6.0% | 47.8% 52.2% |
| | 93.3% 6.7% | 37.0% 63.0% | 75.0% 25.0% | 63.0% 37.0% | 42.3% 57.7% | 65.8% 34.2% |
| | 1 | 2 | 3 | 4 | 5 | |
| | Target Class | | | | | |

Output Class

Figura 12.9: Matriz de confusión. Conjunto de datos Eucalyptus. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.657609 | 0.418478 | 32 | 86.983433 |
| 2 | 0.657609 | 0.423913 | 32 | 88.444341 |
| 3 | 0.635870 | 0.429348 | 32 | 90.359443 |
| 4 | 0.559783 | 0.510870 | 16 | 103.153928 |
| 5 | 0.635870 | 0.407609 | 16 | 104.392842 |
| 6 | 0.608696 | 0.483696 | 16 | 105.320662 |
| 7 | 0.570652 | 0.510870 | 32 | 137.773911 |
| 8 | 0.603261 | 0.516304 | 32 | 139.822835 |
| 9 | 0.608696 | 0.500000 | 32 | 141.585741 |
| 10 | 0.543478 | 0.538043 | 16 | 127.898603 |
| 11 | 0.554348 | 0.543478 | 16 | 129.476039 |
| 12 | 0.603261 | 0.445652 | 16 | 130.853702 |
| 13 | 0.521739 | 0.592391 | 16 | 129.088374 |
| 14 | 0.581522 | 0.494565 | 16 | 131.118791 |
| 15 | 0.619565 | 0.472826 | 16 | 133.236031 |
| 16 | 0.630435 | 0.434783 | 32 | 132.230281 |
| 17 | 0.619565 | 0.407609 | 32 | 134.780232 |
| 18 | 0.592391 | 0.429348 | 32 | 137.684820 |
| 19 | 0.635870 | 0.451087 | 16 | 138.123127 |
| 20 | 0.603261 | 0.516304 | 16 | 140.024503 |
| 21 | 0.657609 | 0.391304 | 16 | 142.181114 |
| 22 | 0.625000 | 0.418478 | 32 | 134.150526 |
| 23 | 0.597826 | 0.445652 | 32 | 136.870417 |
| 24 | 0.586957 | 0.494565 | 32 | 139.516383 |
| 25 | 0.592391 | 0.472826 | 16 | 124.936632 |
| 26 | 0.586957 | 0.500000 | 16 | 126.976282 |
| 27 | 0.510870 | 0.625000 | 16 | 128.238887 |
| 28 | 0.592391 | 0.516304 | 32 | 124.574774 |
| 29 | 0.603261 | 0.445652 | 32 | 126.291711 |
| 30 | 0.592391 | 0.472826 | 32 | 127.615487 |

Tabla 12.9: Resultados individuales. Conjunto de datos Eucalyptus. Clasificación nominal.

12.1.1.10. Conjunto de datos LEV

En la Tabla 12.10 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos LEV, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.10.

Confusion Matrix

| | | | | | | |
|---|----------------|----------------|----------------|----------------|----------------|----------------|
| 1 | 16 6.4% | 2 0.8% | 0 0.0% | 0 0.0% | 0 0.0% | 88.9% 11.1% |
| 2 | 6 2.4% | 53 21.2% | 22 8.8% | 2 0.8% | 0 0.0% | 63.9% 36.1% |
| 3 | 2 0.8% | 13 5.2% | 68 27.2% | 21 8.4% | 2 0.8% | 64.2% 35.8% |
| 4 | 0 0.0% | 2 0.8% | 10 4.0% | 26 10.4% | 3 1.2% | 63.4% 36.6% |
| 5 | 0 0.0% | 0 0.0% | 0 0.0% | 1 0.4% | 1 0.4% | 50.0% 50.0% |
| | 66.7% 33.3% | 75.7% 24.3% | 68.0% 32.0% | 52.0% 48.0% | 16.7% 83.3% | 65.6% 34.4% |
| | 1 | 2 | 3 | 4 | 5 | |
| | Target Class | | | | | |

Output Class

Figura 12.10: Matriz de confusión. Conjunto de datos LEV. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.616000 | 0.408000 | 32 | 80.266864 |
| 2 | 0.604000 | 0.456000 | 32 | 81.113165 |
| 3 | 0.604000 | 0.432000 | 32 | 81.791046 |
| 4 | 0.556000 | 0.488000 | 8 | 83.787376 |
| 5 | 0.584000 | 0.456000 | 8 | 84.630677 |
| 6 | 0.540000 | 0.492000 | 8 | 85.191049 |
| 7 | 0.572000 | 0.448000 | 8 | 77.563661 |
| 8 | 0.600000 | 0.420000 | 8 | 78.216927 |
| 9 | 0.544000 | 0.496000 | 8 | 78.778827 |
| 10 | 0.612000 | 0.412000 | 32 | 78.572674 |
| 11 | 0.644000 | 0.380000 | 32 | 79.450127 |
| 12 | 0.644000 | 0.384000 | 32 | 80.136519 |
| 13 | 0.520000 | 0.516000 | 8 | 78.218553 |
| 14 | 0.612000 | 0.412000 | 8 | 78.919299 |
| 15 | 0.608000 | 0.412000 | 8 | 79.893889 |
| 16 | 0.572000 | 0.476000 | 16 | 80.619956 |
| 17 | 0.604000 | 0.448000 | 16 | 81.362095 |
| 18 | 0.600000 | 0.444000 | 16 | 82.447892 |
| 19 | 0.564000 | 0.448000 | 16 | 98.045986 |
| 20 | 0.584000 | 0.424000 | 16 | 99.722214 |
| 21 | 0.580000 | 0.428000 | 16 | 101.483137 |
| 22 | 0.652000 | 0.368000 | 32 | 101.355211 |
| 23 | 0.640000 | 0.388000 | 32 | 102.908453 |
| 24 | 0.640000 | 0.384000 | 32 | 103.668111 |
| 25 | 0.592000 | 0.428000 | 32 | 85.416408 |
| 26 | 0.604000 | 0.432000 | 32 | 86.095126 |
| 27 | 0.608000 | 0.408000 | 32 | 87.076322 |
| 28 | 0.656000 | 0.376000 | 16 | 81.639785 |
| 29 | 0.600000 | 0.444000 | 16 | 82.864609 |
| 30 | 0.612000 | 0.424000 | 16 | 84.223846 |

Tabla 12.10: Resultados individuales. Conjunto de datos LEV. Clasificación nominal.

12.1.1.11. Conjunto de datos New Thyroid

En la Tabla 12.11 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos New Thyroid, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.11.

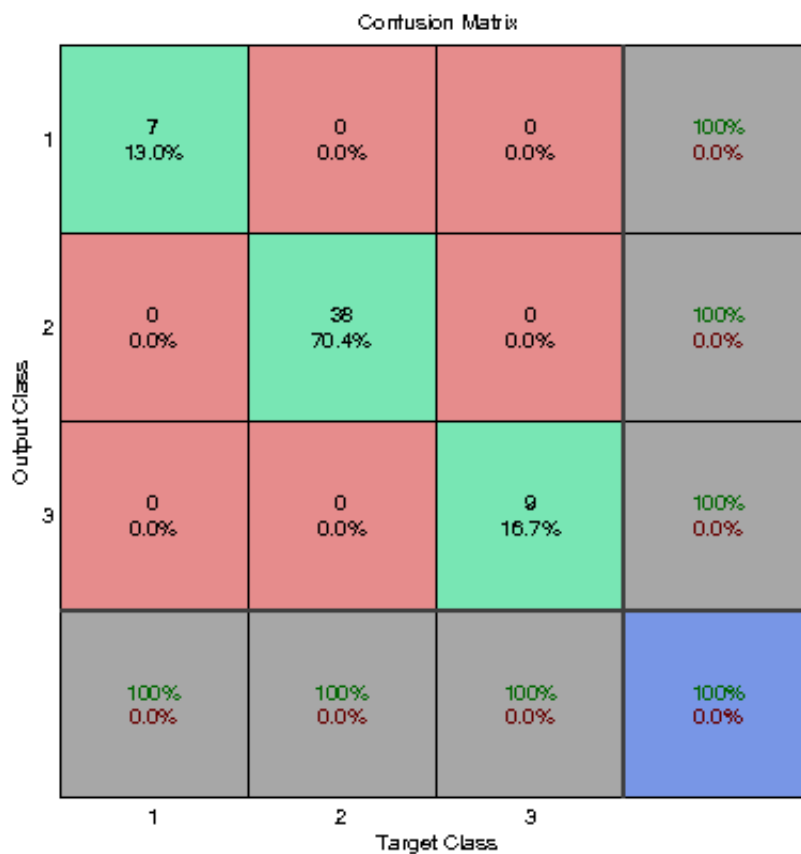


Figura 12.11: Matriz de confusión. Conjunto de datos New Thyroid. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.944444 | 0.055556 | 8 | 56.391590 |
| 2 | 0.962963 | 0.037037 | 8 | 57.152590 |
| 3 | 0.962963 | 0.037037 | 8 | 57.881753 |
| 4 | 0.981481 | 0.018519 | 8 | 56.624350 |
| 5 | 0.907407 | 0.092593 | 8 | 57.198756 |
| 6 | 0.981481 | 0.018519 | 8 | 57.909303 |
| 7 | 0.944444 | 0.055556 | 8 | 58.100112 |
| 8 | 0.944444 | 0.055556 | 8 | 58.813688 |
| 9 | 0.944444 | 0.055556 | 8 | 59.369146 |
| 10 | 0.981481 | 0.018519 | 32 | 56.105439 |
| 11 | 0.981481 | 0.018519 | 32 | 56.912449 |
| 12 | 0.981481 | 0.018519 | 32 | 57.755937 |
| 13 | 0.944444 | 0.055556 | 32 | 58.384526 |
| 14 | 1.000000 | 0.000000 | 32 | 58.962763 |
| 15 | 1.000000 | 0.000000 | 32 | 59.537113 |
| 16 | 0.981481 | 0.018519 | 16 | 51.235736 |
| 17 | 0.925926 | 0.074074 | 16 | 51.771279 |
| 18 | 0.962963 | 0.037037 | 16 | 52.420730 |
| 19 | 0.944444 | 0.055556 | 16 | 56.154932 |
| 20 | 0.907407 | 0.092593 | 16 | 56.690510 |
| 21 | 0.944444 | 0.055556 | 16 | 57.343875 |
| 22 | 0.851852 | 0.148148 | 32 | 61.574390 |
| 23 | 0.944444 | 0.055556 | 32 | 62.214086 |
| 24 | 0.962963 | 0.037037 | 32 | 62.928800 |
| 25 | 1.000000 | 0.000000 | 32 | 54.516787 |
| 26 | 0.962963 | 0.037037 | 32 | 55.096721 |
| 27 | 0.981481 | 0.018519 | 32 | 55.696183 |
| 28 | 0.981481 | 0.018519 | 32 | 54.400743 |
| 29 | 0.962963 | 0.037037 | 32 | 55.023636 |
| 30 | 0.925926 | 0.074074 | 32 | 55.661342 |

Tabla 12.11: Resultados individuales. Conjunto de datos New Thyroid. Clasificación nominal.

12.1.1.12. Conjunto de datos Pasture

En la Tabla 12.12 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Pasture, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.12.

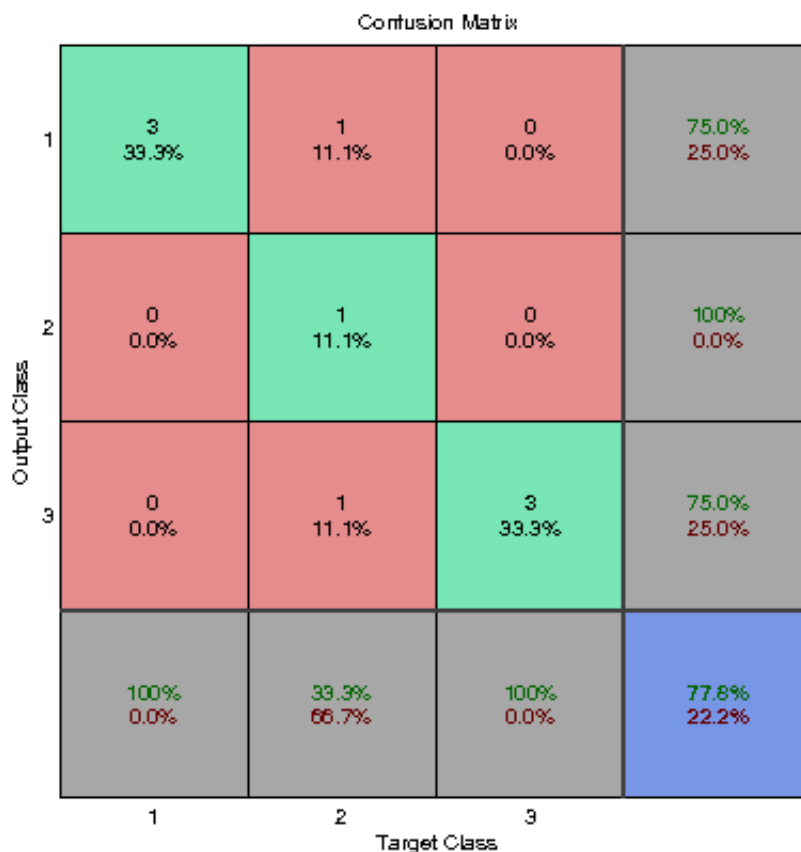


Figura 12.12: Matriz de confusión. Conjunto de datos Pasture. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.666667 | 0.333333 | 16 | 32.848349 |
| 2 | 0.777778 | 0.333333 | 16 | 33.347032 |
| 3 | 0.444444 | 0.666667 | 16 | 33.859302 |
| 4 | 0.666667 | 0.444444 | 16 | 32.436724 |
| 5 | 0.666667 | 0.333333 | 16 | 32.995108 |
| 6 | 0.777778 | 0.222222 | 16 | 33.620799 |
| 7 | 0.777778 | 0.222222 | 32 | 33.168610 |
| 8 | 0.444444 | 0.555556 | 32 | 33.689824 |
| 9 | 0.555556 | 0.444444 | 32 | 34.201492 |
| 10 | 0.555556 | 0.555556 | 16 | 31.480980 |
| 11 | 0.777778 | 0.222222 | 16 | 32.000769 |
| 12 | 0.777778 | 0.222222 | 16 | 32.519514 |
| 13 | 0.777778 | 0.333333 | 16 | 31.884995 |
| 14 | 0.777778 | 0.333333 | 16 | 32.395708 |
| 15 | 0.666667 | 0.333333 | 16 | 32.901453 |
| 16 | 0.666667 | 0.444444 | 32 | 32.755855 |
| 17 | 0.777778 | 0.222222 | 32 | 33.269784 |
| 18 | 0.777778 | 0.222222 | 32 | 33.843322 |
| 19 | 0.666667 | 0.333333 | 8 | 35.256478 |
| 20 | 0.777778 | 0.222222 | 8 | 35.846588 |
| 21 | 0.666667 | 0.333333 | 8 | 36.382160 |
| 22 | 0.777778 | 0.222222 | 16 | 32.476834 |
| 23 | 0.777778 | 0.222222 | 16 | 32.985779 |
| 24 | 0.888889 | 0.222222 | 16 | 33.507996 |
| 25 | 1.000000 | 0.000000 | 16 | 30.053371 |
| 26 | 0.666667 | 0.444444 | 16 | 30.558909 |
| 27 | 0.777778 | 0.222222 | 16 | 31.086716 |
| 28 | 0.888889 | 0.111111 | 8 | 31.983072 |
| 29 | 0.666667 | 0.333333 | 8 | 32.487174 |
| 30 | 0.777778 | 0.222222 | 8 | 33.014020 |

Tabla 12.12: Resultados individuales. Conjunto de datos Pasture. Clasificación nominal.

12.1.1.13. Conjunto de datos Squash Stored

En la Tabla 12.13 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Squash Stored, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.13.

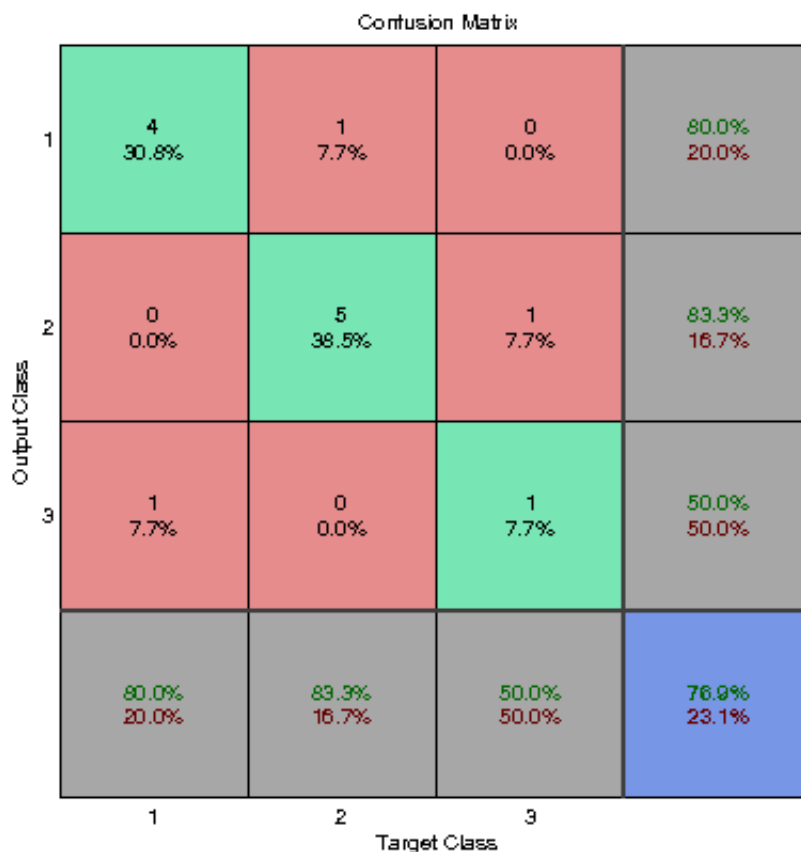


Figura 12.13: Matriz de confusión. Conjunto de datos Squash Stored. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.692308 | 0.307692 | 32 | 33.018301 |
| 2 | 0.769231 | 0.230769 | 32 | 33.628657 |
| 3 | 0.615385 | 0.384615 | 32 | 34.220098 |
| 4 | 0.615385 | 0.461538 | 32 | 34.090601 |
| 5 | 0.461538 | 0.615385 | 32 | 34.621919 |
| 6 | 0.615385 | 0.461538 | 32 | 35.174346 |
| 7 | 0.846154 | 0.153846 | 16 | 33.108731 |
| 8 | 0.846154 | 0.153846 | 16 | 33.677626 |
| 9 | 0.846154 | 0.230769 | 16 | 34.273966 |
| 10 | 0.461538 | 0.615385 | 16 | 34.157002 |
| 11 | 0.384615 | 0.692308 | 16 | 34.737629 |
| 12 | 0.461538 | 0.615385 | 16 | 35.286944 |
| 13 | 0.615385 | 0.384615 | 32 | 32.994004 |
| 14 | 0.692308 | 0.307692 | 32 | 33.542206 |
| 15 | 0.538462 | 0.461538 | 32 | 34.125155 |
| 16 | 0.846154 | 0.153846 | 32 | 36.852136 |
| 17 | 0.769231 | 0.230769 | 32 | 37.460467 |
| 18 | 0.846154 | 0.153846 | 32 | 38.035101 |
| 19 | 0.538462 | 0.461538 | 32 | 33.857159 |
| 20 | 0.692308 | 0.384615 | 32 | 34.403277 |
| 21 | 0.769231 | 0.230769 | 32 | 35.252874 |
| 22 | 0.615385 | 0.384615 | 32 | 35.817345 |
| 23 | 0.846154 | 0.153846 | 32 | 36.377911 |
| 24 | 0.692308 | 0.307692 | 32 | 36.960697 |
| 25 | 0.769231 | 0.307692 | 32 | 35.224621 |
| 26 | 0.615385 | 0.384615 | 32 | 35.823314 |
| 27 | 0.692308 | 0.307692 | 32 | 36.384940 |
| 28 | 0.538462 | 0.461538 | 32 | 34.645769 |
| 29 | 0.538462 | 0.461538 | 32 | 35.204983 |
| 30 | 0.461538 | 0.538462 | 32 | 35.872486 |

Tabla 12.13: Resultados individuales. Conjunto de datos Squash Stored. Clasificación nominal.

12.1.1.14. Conjunto de datos Squash Unstored

En la Tabla 12.14 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Squash Unstored, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.14.

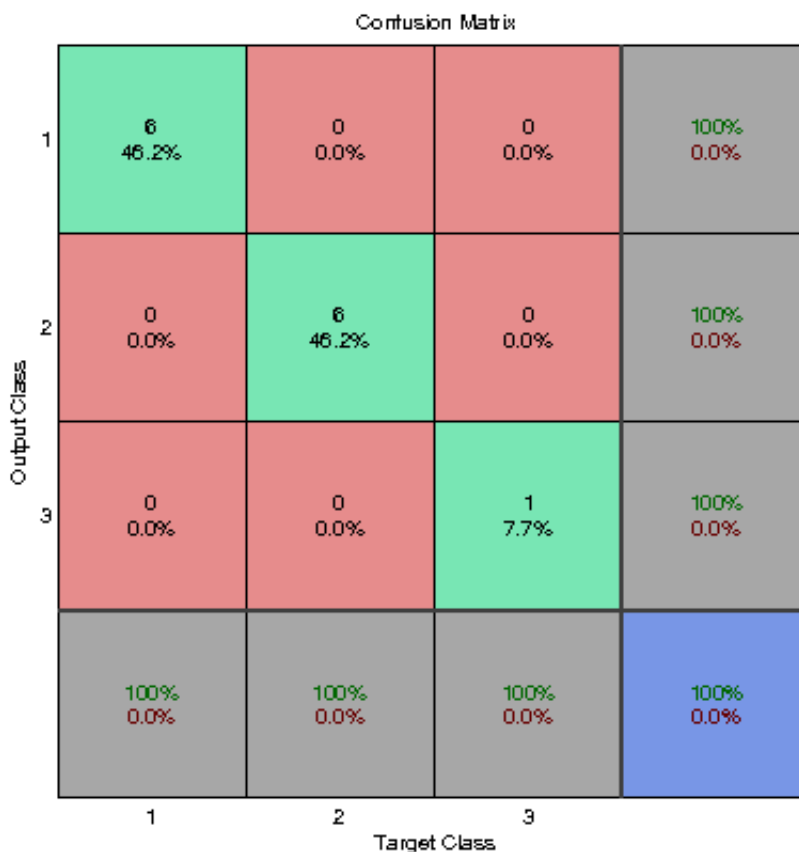


Figura 12.14: Matriz de confusión. Conjunto de datos Squash Unstored. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.692308 | 0.307692 | 16 | 38.555008 |
| 2 | 0.692308 | 0.307692 | 16 | 39.083568 |
| 3 | 0.769231 | 0.230769 | 16 | 39.616729 |
| 4 | 0.692308 | 0.307692 | 32 | 37.501771 |
| 5 | 0.846154 | 0.153846 | 32 | 38.234359 |
| 6 | 0.692308 | 0.307692 | 32 | 38.964258 |
| 7 | 0.538462 | 0.461538 | 8 | 38.250728 |
| 8 | 0.538462 | 0.461538 | 8 | 38.816182 |
| 9 | 0.615385 | 0.384615 | 8 | 39.375852 |
| 10 | 0.769231 | 0.230769 | 16 | 37.538394 |
| 11 | 0.615385 | 0.384615 | 16 | 38.065717 |
| 12 | 0.769231 | 0.230769 | 16 | 38.611913 |
| 13 | 0.692308 | 0.307692 | 32 | 38.534547 |
| 14 | 0.692308 | 0.307692 | 32 | 39.131538 |
| 15 | 0.769231 | 0.230769 | 32 | 39.745798 |
| 16 | 0.769231 | 0.230769 | 32 | 37.765312 |
| 17 | 0.923077 | 0.076923 | 32 | 38.336859 |
| 18 | 0.615385 | 0.384615 | 32 | 38.886442 |
| 19 | 0.615385 | 0.384615 | 16 | 38.103891 |
| 20 | 1.000000 | 0.000000 | 16 | 38.652105 |
| 21 | 0.615385 | 0.384615 | 16 | 39.178142 |
| 22 | 0.538462 | 0.461538 | 8 | 40.237490 |
| 23 | 0.538462 | 0.461538 | 8 | 40.856357 |
| 24 | 0.538462 | 0.461538 | 8 | 41.398202 |
| 25 | 0.692308 | 0.307692 | 16 | 36.013871 |
| 26 | 0.692308 | 0.307692 | 16 | 36.580837 |
| 27 | 0.769231 | 0.230769 | 16 | 37.161816 |
| 28 | 0.461538 | 0.538462 | 32 | 35.405337 |
| 29 | 0.461538 | 0.538462 | 32 | 36.020152 |
| 30 | 0.461538 | 0.538462 | 32 | 36.722897 |

Tabla 12.14: Resultados individuales. Conjunto de datos Squash Unstored. Clasificación nominal.

12.1.1.15. Conjunto de datos SWD

En la Tabla 12.15 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos SWD, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.15.

Confusion Matrix

| | | | | | | |
|--------------|---|--------------|----------------|----------------|----------------|----------------|
| Output Class | 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 2 | 8 3.2% | 62 24.8% | 22 8.8% | 2 0.8% | 66.0% 34.0% |
| | 3 | 0 0.0% | 24 9.6% | 68 27.2% | 28 11.2% | 56.7% 43.3% |
| | 4 | 0 0.0% | 2 0.8% | 10 4.0% | 24 9.6% | 66.7% 33.3% |
| | | 0.0% 100% | 70.5% 29.5% | 68.0% 32.0% | 44.4% 55.6% | 61.6% 38.4% |
| | | 1 | 2 | 3 | 4 | |
| | | Target Class | | | | |

Figura 12.15: Matriz de confusión. Conjunto de datos SWD. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.552000 | 0.472000 | 16 | 69.075582 |
| 2 | 0.580000 | 0.444000 | 16 | 70.213074 |
| 3 | 0.552000 | 0.484000 | 16 | 71.404527 |
| 4 | 0.576000 | 0.464000 | 16 | 66.856671 |
| 5 | 0.584000 | 0.444000 | 16 | 67.917688 |
| 6 | 0.584000 | 0.460000 | 16 | 68.998294 |
| 7 | 0.616000 | 0.400000 | 4 | 68.606040 |
| 8 | 0.572000 | 0.468000 | 4 | 69.373810 |
| 9 | 0.600000 | 0.428000 | 4 | 70.056346 |
| 10 | 0.544000 | 0.484000 | 8 | 74.103700 |
| 11 | 0.592000 | 0.440000 | 8 | 75.320514 |
| 12 | 0.564000 | 0.460000 | 8 | 76.344302 |
| 13 | 0.604000 | 0.420000 | 8 | 70.214325 |
| 14 | 0.576000 | 0.444000 | 8 | 70.914067 |
| 15 | 0.592000 | 0.436000 | 8 | 71.507590 |
| 16 | 0.552000 | 0.480000 | 16 | 78.526392 |
| 17 | 0.548000 | 0.472000 | 16 | 79.437080 |
| 18 | 0.580000 | 0.440000 | 16 | 80.001902 |
| 19 | 0.596000 | 0.416000 | 8 | 84.808671 |
| 20 | 0.584000 | 0.432000 | 8 | 85.435177 |
| 21 | 0.540000 | 0.488000 | 8 | 86.182248 |
| 22 | 0.532000 | 0.516000 | 8 | 91.165473 |
| 23 | 0.524000 | 0.520000 | 8 | 91.918158 |
| 24 | 0.560000 | 0.468000 | 8 | 92.581271 |
| 25 | 0.540000 | 0.512000 | 32 | 94.583567 |
| 26 | 0.528000 | 0.524000 | 32 | 95.351649 |
| 27 | 0.464000 | 0.572000 | 32 | 96.007213 |
| 28 | 0.512000 | 0.504000 | 16 | 90.477380 |
| 29 | 0.560000 | 0.468000 | 16 | 91.031556 |
| 30 | 0.544000 | 0.476000 | 16 | 91.777624 |

Tabla 12.15: Resultados individuales. Conjunto de datos SWD. Clasificación nominal.

12.1.1.16. Conjunto de datos TAE

En la Tabla 12.16 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos TAE, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.16.

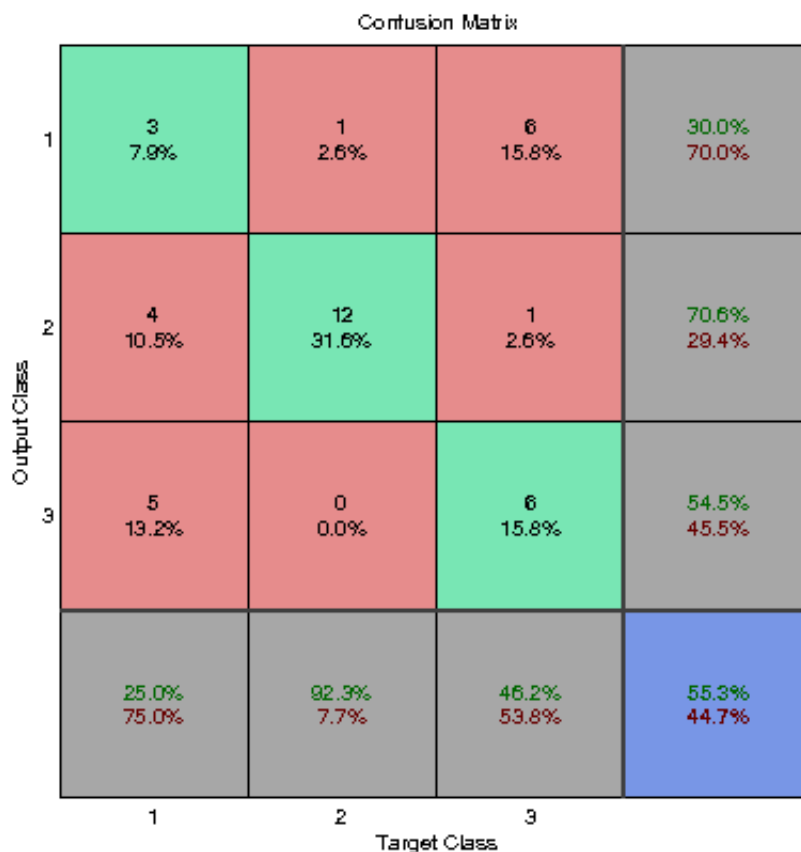


Figura 12.16: Matriz de confusión. Conjunto de datos TAE. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.447368 | 0.710526 | 32 | 44.093534 |
| 2 | 0.473684 | 0.710526 | 32 | 44.692259 |
| 3 | 0.500000 | 0.605263 | 32 | 45.362586 |
| 4 | 0.473684 | 0.605263 | 16 | 44.227137 |
| 5 | 0.500000 | 0.578947 | 16 | 44.868795 |
| 6 | 0.394737 | 0.710526 | 16 | 45.475661 |
| 7 | 0.394737 | 0.894737 | 4 | 44.590749 |
| 8 | 0.342105 | 0.763158 | 4 | 45.174024 |
| 9 | 0.421053 | 0.578947 | 4 | 45.728800 |
| 10 | 0.447368 | 0.710526 | 16 | 43.698689 |
| 11 | 0.526316 | 0.736842 | 16 | 44.715966 |
| 12 | 0.447368 | 0.710526 | 16 | 45.414942 |
| 13 | 0.447368 | 0.710526 | 32 | 43.306576 |
| 14 | 0.315789 | 0.894737 | 32 | 43.843100 |
| 15 | 0.421053 | 0.815789 | 32 | 44.405563 |
| 16 | 0.552632 | 0.736842 | 8 | 45.855199 |
| 17 | 0.500000 | 0.710526 | 8 | 46.438884 |
| 18 | 0.236842 | 1.184211 | 8 | 46.966350 |
| 19 | 0.447368 | 0.657895 | 32 | 45.703597 |
| 20 | 0.473684 | 0.631579 | 32 | 46.555656 |
| 21 | 0.473684 | 0.684211 | 32 | 47.233145 |
| 22 | 0.447368 | 0.789474 | 4 | 43.738797 |
| 23 | 0.263158 | 1.026316 | 4 | 44.310920 |
| 24 | 0.315789 | 1.026316 | 4 | 44.864981 |
| 25 | 0.236842 | 1.026316 | 16 | 43.165465 |
| 26 | 0.394737 | 0.815789 | 16 | 43.934546 |
| 27 | 0.421053 | 0.763158 | 16 | 44.567184 |
| 28 | 0.500000 | 0.631579 | 16 | 45.620270 |
| 29 | 0.289474 | 0.789474 | 16 | 46.241698 |
| 30 | 0.342105 | 0.736842 | 16 | 46.918825 |

Tabla 12.16: Resultados individuales. Conjunto de datos TAE. Clasificación nominal.

12.1.1.17. Conjunto de datos Thyroid

En la Tabla 12.17 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Thyroid, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.17.



Figura 12.17: Matriz de confusión. Conjunto de datos Thyroid. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.939444 | 0.110000 | 32 | 478.542093 |
| 2 | 0.942222 | 0.105556 | 32 | 487.274355 |
| 3 | 0.942222 | 0.106667 | 32 | 495.896737 |
| 4 | 0.943333 | 0.103333 | 32 | 626.213597 |
| 5 | 0.942778 | 0.105556 | 32 | 634.430876 |
| 6 | 0.946667 | 0.098889 | 32 | 646.226503 |
| 7 | 0.940000 | 0.110000 | 16 | 784.618538 |
| 8 | 0.943889 | 0.101667 | 16 | 798.528671 |
| 9 | 0.926111 | 0.125000 | 16 | 800.407434 |
| 10 | 0.941667 | 0.107778 | 32 | 708.396940 |
| 11 | 0.943889 | 0.101111 | 32 | 716.113162 |
| 12 | 0.942778 | 0.103889 | 32 | 729.212300 |
| 13 | 0.961111 | 0.070556 | 32 | 832.914136 |
| 14 | 0.943333 | 0.106111 | 32 | 838.570121 |
| 15 | 0.942778 | 0.107778 | 32 | 842.982983 |
| 16 | 0.941667 | 0.109444 | 32 | 798.487837 |
| 17 | 0.940556 | 0.110556 | 32 | 803.753550 |
| 18 | 0.941111 | 0.110000 | 32 | 814.128440 |
| 19 | 0.956111 | 0.079444 | 32 | 749.626394 |
| 20 | 0.941111 | 0.108889 | 32 | 754.637222 |
| 21 | 0.942222 | 0.107222 | 32 | 761.907159 |
| 22 | 0.939444 | 0.110000 | 32 | 754.395814 |
| 23 | 0.943889 | 0.106111 | 32 | 760.422473 |
| 24 | 0.945000 | 0.103889 | 32 | 768.763667 |
| 25 | 0.956111 | 0.080556 | 32 | 794.443878 |
| 26 | 0.943333 | 0.106111 | 32 | 800.816268 |
| 27 | 0.951667 | 0.092222 | 32 | 835.714167 |
| 28 | 0.940556 | 0.106111 | 16 | 646.169532 |
| 29 | 0.973333 | 0.043889 | 16 | 682.267846 |
| 30 | 0.940556 | 0.108889 | 16 | 688.232570 |

Tabla 12.17: Resultados individuales. Conjunto de datos Thyroid. Clasificación nominal.

12.1.1.18. Conjunto de datos Wine Quality Red

En la Tabla 12.18 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Wine Quality Red, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.18.

Confusion Matrix

| | | | | | | | |
|---|--------------|--------------|----------------|----------------|----------------|--------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 0 0.0% | 0 0.0% | 1 0.2% | 1 0.2% | 0 0.0% | 0 0.0% | 0.0% 100% |
| 3 | 2 0.5% | 12 3.0% | 127 31.8% | 48 12.0% | 1 0.2% | 0 0.0% | 66.8% 33.2% |
| 4 | 0 0.0% | 1 0.2% | 40 10.0% | 97 24.2% | 31 7.8% | 2 0.5% | 56.7% 43.3% |
| 5 | 0 0.0% | 0 0.0% | 3 0.8% | 13 3.2% | 18 4.5% | 3 0.8% | 48.6% 51.4% |
| 6 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 0.0% 100% | 0.0% 100% | 74.3% 25.7% | 61.0% 39.0% | 36.0% 64.0% | 0.0% 100% | 60.5% 39.5% |
| | 1 | 2 | 3 | 4 | 5 | 6 | |

Target Class

Figura 12.18: Matriz de confusión. Conjunto de datos Wine Quality Red. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.605000 | 0.420000 | 8 | 180.303867 |
| 2 | 0.602500 | 0.422500 | 8 | 181.382909 |
| 3 | 0.600000 | 0.422500 | 8 | 183.102819 |
| 4 | 0.570000 | 0.485000 | 32 | 171.313057 |
| 5 | 0.575000 | 0.470000 | 32 | 172.927601 |
| 6 | 0.557500 | 0.492500 | 32 | 174.178725 |
| 7 | 0.577500 | 0.462500 | 32 | 157.054650 |
| 8 | 0.580000 | 0.457500 | 32 | 158.521862 |
| 9 | 0.572500 | 0.470000 | 32 | 159.526745 |
| 10 | 0.597500 | 0.440000 | 32 | 147.390033 |
| 11 | 0.597500 | 0.437500 | 32 | 150.120504 |
| 12 | 0.590000 | 0.457500 | 32 | 155.625960 |
| 13 | 0.550000 | 0.477500 | 16 | 153.326494 |
| 14 | 0.565000 | 0.465000 | 16 | 157.211427 |
| 15 | 0.547500 | 0.480000 | 16 | 159.101383 |
| 16 | 0.590000 | 0.445000 | 32 | 153.285101 |
| 17 | 0.577500 | 0.457500 | 32 | 155.354207 |
| 18 | 0.597500 | 0.442500 | 32 | 157.800509 |
| 19 | 0.547500 | 0.495000 | 32 | 112.647250 |
| 20 | 0.557500 | 0.482500 | 32 | 113.851294 |
| 21 | 0.562500 | 0.477500 | 32 | 115.152849 |
| 22 | 0.582500 | 0.460000 | 32 | 212.181674 |
| 23 | 0.577500 | 0.465000 | 32 | 218.060627 |
| 24 | 0.592500 | 0.455000 | 32 | 222.497155 |
| 25 | 0.575000 | 0.470000 | 32 | 123.334716 |
| 26 | 0.602500 | 0.432500 | 32 | 124.460066 |
| 27 | 0.570000 | 0.472500 | 32 | 125.807377 |
| 28 | 0.570000 | 0.470000 | 32 | 220.395902 |
| 29 | 0.582500 | 0.452500 | 32 | 223.221931 |
| 30 | 0.575000 | 0.457500 | 32 | 225.463287 |

Tabla 12.18: Resultados individuales. Conjunto de datos Wine Quality Red. Clasificación nominal.

12.1.1.19. Conjunto de datos Wine Quality White

En la Tabla 12.19 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de clasificación nominal mediante una red neuronal artificial no ordinal para el conjunto de datos Wine Quality White, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.19.

Confusion Matrix

| | | | | | | | | |
|---|--------------|---------------|----------------|----------------|----------------|--------------|--------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 0 0.0% | 1 0.1% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 100% 0.0% |
| 3 | 2 0.2% | 26 2.1% | 207 16.9% | 104 8.5% | 13 1.1% | 4 0.3% | 0 0.0% | 58.1% 41.9% |
| 4 | 2 0.2% | 12 1.0% | 152 12.4% | 415 33.9% | 143 11.7% | 23 1.9% | 1 0.1% | 55.5% 44.5% |
| 5 | 1 0.1% | 2 0.2% | 5 0.4% | 31 2.5% | 64 5.2% | 17 1.4% | 0 0.0% | 53.3% 46.7% |
| 6 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 7 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 0.0% 100% | 2.4% 97.6% | 56.9% 43.1% | 75.5% 24.5% | 29.1% 70.9% | 0.0% 100% | 0.0% 100% | 56.1% 43.9% |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| | Target Class | | | | | | | |

Output Class

Figura 12.19: Matriz de confusión. Conjunto de datos Wine Quality White. Clasificación nominal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.525714 | 0.528980 | 32 | 472.718929 |
| 2 | 0.537143 | 0.515918 | 32 | 478.898004 |
| 3 | 0.533878 | 0.526531 | 32 | 486.452733 |
| 4 | 0.553469 | 0.502041 | 32 | 492.413339 |
| 5 | 0.548571 | 0.517551 | 32 | 498.748994 |
| 6 | 0.546939 | 0.511837 | 32 | 504.331909 |
| 7 | 0.535510 | 0.516735 | 32 | 502.586573 |
| 8 | 0.539592 | 0.515918 | 32 | 505.444831 |
| 9 | 0.544490 | 0.510204 | 32 | 509.019749 |
| 10 | 0.559184 | 0.496327 | 32 | 439.600623 |
| 11 | 0.551020 | 0.505306 | 32 | 442.201230 |
| 12 | 0.555918 | 0.511837 | 32 | 444.709318 |
| 13 | 0.560000 | 0.508571 | 32 | 426.057543 |
| 14 | 0.560816 | 0.501224 | 32 | 433.531953 |
| 15 | 0.547755 | 0.509388 | 32 | 440.001835 |
| 16 | 0.528980 | 0.528163 | 16 | 421.275589 |
| 17 | 0.526531 | 0.535510 | 16 | 427.596872 |
| 18 | 0.542857 | 0.524082 | 16 | 432.304687 |
| 19 | 0.546939 | 0.510204 | 32 | 443.623211 |
| 20 | 0.549388 | 0.511837 | 32 | 450.717200 |
| 21 | 0.549388 | 0.511837 | 32 | 460.445679 |
| 22 | 0.536327 | 0.517551 | 32 | 298.064680 |
| 23 | 0.536327 | 0.524898 | 32 | 302.440650 |
| 24 | 0.545306 | 0.515102 | 32 | 306.669210 |
| 25 | 0.551020 | 0.512653 | 32 | 267.801558 |
| 26 | 0.549388 | 0.515102 | 32 | 270.181043 |
| 27 | 0.555102 | 0.508571 | 32 | 272.601412 |
| 28 | 0.519184 | 0.538776 | 32 | 260.526432 |
| 29 | 0.541224 | 0.508571 | 32 | 263.842295 |
| 30 | 0.533878 | 0.520000 | 32 | 266.384360 |

Tabla 12.19: Resultados individuales. Conjunto de datos Wine Quality White. Clasificación nominal.

12.1.2. Clasificación Ordinal

Para cada uno de los conjuntos se mostrarán los datos obtenidos correspondientes a aplicar un modelo de clasificación donde se tiene en cuenta el orden de las etiquetas y la matriz de confusión del mejor resultado.

12.1.2.1. Conjunto de datos Automobile

En la Tabla 12.20 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Automobile, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.20.

12.1.2.2. Conjunto de datos Balance Scale

En la Tabla 12.21 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Balance Scale, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.21.

12.1.2.3. Conjunto de datos Bondrate

En la Tabla 12.22 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Bondrate, donde se

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.596154 | 0.519231 | 16 | 85.747170 |
| 2 | 0.634615 | 0.461538 | 16 | 87.920509 |
| 3 | 0.673077 | 0.423077 | 16 | 89.453910 |
| 4 | 0.750000 | 0.288462 | 32 | 82.084880 |
| 5 | 0.730769 | 0.346154 | 32 | 83.918852 |
| 6 | 0.673077 | 0.365385 | 32 | 85.304232 |
| 7 | 0.615385 | 0.423077 | 16 | 86.627201 |
| 8 | 0.692308 | 0.365385 | 16 | 88.482431 |
| 9 | 0.326923 | 0.884615 | 16 | 89.425752 |
| 10 | 0.557692 | 0.538462 | 32 | 96.619424 |
| 11 | 0.596154 | 0.442308 | 32 | 98.211527 |
| 12 | 0.576923 | 0.480769 | 32 | 99.974041 |
| 13 | 0.673077 | 0.423077 | 32 | 95.484119 |
| 14 | 0.634615 | 0.576923 | 32 | 97.477025 |
| 15 | 0.653846 | 0.480769 | 32 | 99.114709 |
| 16 | 0.653846 | 0.461538 | 8 | 87.888840 |
| 17 | 0.692308 | 0.403846 | 8 | 89.611253 |
| 18 | 0.596154 | 0.480769 | 8 | 91.341020 |
| 19 | 0.692308 | 0.384615 | 32 | 106.156652 |
| 20 | 0.788462 | 0.230769 | 32 | 107.093248 |
| 21 | 0.615385 | 0.461538 | 32 | 107.825167 |
| 22 | 0.730769 | 0.346154 | 16 | 90.299342 |
| 23 | 0.730769 | 0.326923 | 16 | 91.175830 |
| 24 | 0.615385 | 0.442308 | 16 | 91.869013 |
| 25 | 0.576923 | 0.500000 | 32 | 90.537970 |
| 26 | 0.576923 | 0.538462 | 32 | 91.272478 |
| 27 | 0.576923 | 0.557692 | 32 | 92.074490 |
| 28 | 0.692308 | 0.346154 | 32 | 79.185589 |
| 29 | 0.730769 | 0.346154 | 32 | 81.014182 |
| 30 | 0.596154 | 0.461538 | 32 | 82.121662 |

Tabla 12.20: Resultados individuales. Conjunto de datos Automobile. Clasificación ordinal.

Confusion Matrix

| | | | | | | | |
|---|--------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 1 1.9% | 1 1.9% | 1 1.9% | 0 0.0% | 0 0.0% | 0 0.0% | 33.3% 66.7% |
| 3 | 0 0.0% | 3 5.8% | 12 23.1% | 1 1.9% | 2 3.8% | 0 0.0% | 66.7% 33.3% |
| 4 | 0 0.0% | 1 1.9% | 4 7.7% | 8 15.4% | 2 3.8% | 1 1.9% | 50.0% 50.0% |
| 5 | 0 0.0% | 0 0.0% | 0 0.0% | 3 5.8% | 4 7.7% | 0 0.0% | 57.1% 42.9% |
| 6 | 0 0.0% | 0 0.0% | 0 0.0% | 2 3.8% | 0 0.0% | 6 11.5% | 75.0% 25.0% |
| | 0.0% 100% | 20.0% 80.0% | 70.6% 29.4% | 57.1% 42.9% | 50.0% 50.0% | 85.7% 14.3% | 59.6% 40.4% |
| | 1 | 2 | 3 | 4 | 5 | 6 | |

Target Class

Figura 12.20: Matriz de confusión. Conjunto de datos Automobile. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.955414 | 0.050955 | 16 | 65.743054 |
| 2 | 0.961783 | 0.057325 | 16 | 66.649109 |
| 3 | 0.968153 | 0.031847 | 16 | 67.307758 |
| 4 | 0.974522 | 0.025478 | 16 | 69.137743 |
| 5 | 0.974522 | 0.025478 | 16 | 70.339063 |
| 6 | 0.936306 | 0.063694 | 16 | 70.976060 |
| 7 | 0.974522 | 0.025478 | 32 | 70.769429 |
| 8 | 0.974522 | 0.025478 | 32 | 71.519097 |
| 9 | 0.980892 | 0.019108 | 32 | 72.149978 |
| 10 | 0.974522 | 0.025478 | 16 | 65.603547 |
| 11 | 0.980892 | 0.025478 | 16 | 66.198870 |
| 12 | 0.955414 | 0.044586 | 16 | 66.870230 |
| 13 | 0.968153 | 0.031847 | 16 | 65.753556 |
| 14 | 0.955414 | 0.057325 | 16 | 66.665470 |
| 15 | 0.936306 | 0.063694 | 16 | 67.410740 |
| 16 | 0.980892 | 0.025478 | 16 | 63.400353 |
| 17 | 0.968153 | 0.031847 | 16 | 64.024448 |
| 18 | 0.993631 | 0.006369 | 16 | 64.806097 |
| 19 | 0.961783 | 0.044586 | 32 | 65.129104 |
| 20 | 0.968153 | 0.031847 | 32 | 65.748934 |
| 21 | 0.961783 | 0.038217 | 32 | 66.529389 |
| 22 | 0.980892 | 0.019108 | 8 | 67.015900 |
| 23 | 0.910828 | 0.101911 | 8 | 67.608999 |
| 24 | 0.923567 | 0.076433 | 8 | 68.264202 |
| 25 | 0.955414 | 0.050955 | 32 | 62.827839 |
| 26 | 0.955414 | 0.044586 | 32 | 63.467910 |
| 27 | 0.968153 | 0.031847 | 32 | 64.267416 |
| 28 | 0.974522 | 0.038217 | 32 | 62.237717 |
| 29 | 0.980892 | 0.019108 | 32 | 63.262287 |
| 30 | 0.961783 | 0.038217 | 32 | 63.862226 |

Tabla 12.21: Resultados individuales. Conjunto de datos Balance Scale. Clasificación ordinal.

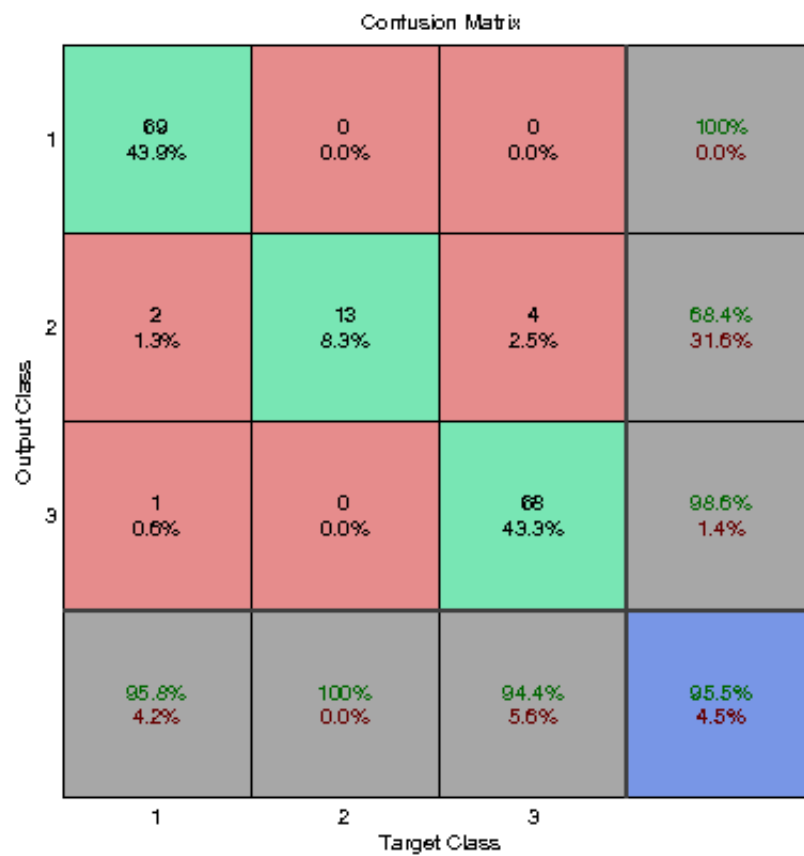


Figura 12.21: Matriz de confusión. Conjunto de datos Balance Scale. Clasificación ordinal.

muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.666667 | 0.533333 | 8 | 40.546571 |
| 2 | 0.600000 | 0.600000 | 8 | 41.116781 |
| 3 | 0.466667 | 0.666667 | 8 | 41.701918 |
| 4 | 0.533333 | 0.600000 | 4 | 40.624989 |
| 5 | 0.533333 | 0.600000 | 4 | 41.296239 |
| 6 | 0.600000 | 0.600000 | 4 | 41.878668 |
| 7 | 0.600000 | 0.533333 | 8 | 39.773189 |
| 8 | 0.600000 | 0.533333 | 8 | 40.344563 |
| 9 | 0.666667 | 0.466667 | 8 | 40.907160 |
| 10 | 0.533333 | 0.666667 | 8 | 40.084405 |
| 11 | 0.600000 | 0.600000 | 8 | 40.669137 |
| 12 | 0.600000 | 0.600000 | 8 | 41.248813 |
| 13 | 0.466667 | 0.733333 | 16 | 40.631065 |
| 14 | 0.533333 | 0.466667 | 16 | 41.257619 |
| 15 | 0.466667 | 0.666667 | 16 | 41.862001 |
| 16 | 0.666667 | 0.400000 | 32 | 40.102882 |
| 17 | 0.600000 | 0.666667 | 32 | 40.703840 |
| 18 | 0.600000 | 0.466667 | 32 | 41.318584 |
| 19 | 0.600000 | 0.533333 | 4 | 40.196201 |
| 20 | 0.600000 | 0.600000 | 4 | 40.778431 |
| 21 | 0.733333 | 0.333333 | 4 | 41.406988 |
| 22 | 0.600000 | 0.600000 | 8 | 40.481213 |
| 23 | 0.600000 | 0.600000 | 8 | 41.061030 |
| 24 | 0.600000 | 0.533333 | 8 | 41.649272 |
| 25 | 0.466667 | 0.733333 | 32 | 41.032928 |
| 26 | 0.333333 | 0.733333 | 32 | 41.687307 |
| 27 | 0.400000 | 0.666667 | 32 | 42.326742 |
| 28 | 0.600000 | 0.600000 | 4 | 40.187696 |
| 29 | 0.600000 | 0.600000 | 4 | 40.789510 |
| 30 | 0.600000 | 0.600000 | 4 | 41.374077 |

Tabla 12.22: Resultados individuales. Conjunto de datos Bondrate. Clasificación ordinal.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.22.

Confusion Matrix

| | | | | | | |
|---|--------------|--------------|----------------|--------------|--------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 1 6.7% | 9 60.0% | 2 13.3% | 1 6.7% | 1 6.7% | 64.3% 35.7% |
| 3 | 0 0.0% | 0 0.0% | 1 6.7% | 0 0.0% | 0 0.0% | 100% 0.0% |
| 4 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 5 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 0.0% 100% | 100% 0.0% | 33.3% 66.7% | 0.0% 100% | 0.0% 100% | 66.7% 33.3% |
| | 1 | 2 | 3 | 4 | 5 | |

Target Class

Figura 12.22: Matriz de confusión. Conjunto de datos Bondrate. Clasificación ordinal.

12.1.2.4. Conjunto de datos Car

En la Tabla 12.23 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos Car, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.23.

Confusion Matrix

| | | | | | | |
|--------------|---|---------------|---------------|----------------|---------------|----------------|
| Output Class | 1 | 289 66.9% | 1 0.2% | 0 0.0% | 0 0.0% | 99.7% 0.3% |
| | 2 | 14 3.2% | 92 21.3% | 4 0.9% | 0 0.0% | 83.6% 16.4% |
| | 3 | 0 0.0% | 3 0.7% | 13 3.0% | 1 0.2% | 76.5% 23.5% |
| | 4 | 0 0.0% | 0 0.0% | 0 0.0% | 15 3.5% | 100% 0.0% |
| | | 95.4% 4.6% | 95.8% 4.2% | 76.5% 23.5% | 93.8% 6.2% | 94.7% 5.3% |
| | | 1 | 2 | 3 | 4 | |
| | | Target Class | | | | |

Figura 12.23: Matriz de confusión. Conjunto de datos CAR. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.946759 | 0.053241 | 32 | 151.887842 |
| 2 | 0.965278 | 0.034722 | 32 | 153.286324 |
| 3 | 0.946759 | 0.053241 | 32 | 154.580300 |
| 4 | 0.990741 | 0.009259 | 32 | 255.452242 |
| 5 | 0.960648 | 0.039352 | 32 | 257.318668 |
| 6 | 0.969907 | 0.030093 | 32 | 259.064653 |
| 7 | 0.969907 | 0.030093 | 32 | 223.374036 |
| 8 | 0.962963 | 0.037037 | 32 | 225.835342 |
| 9 | 0.967593 | 0.032407 | 32 | 230.982927 |
| 10 | 0.976852 | 0.023148 | 32 | 219.152456 |
| 11 | 0.956019 | 0.046296 | 32 | 222.734081 |
| 12 | 0.972222 | 0.030093 | 32 | 226.209490 |
| 13 | 0.969907 | 0.030093 | 32 | 224.324821 |
| 14 | 0.956019 | 0.043981 | 32 | 227.877676 |
| 15 | 0.972222 | 0.027778 | 32 | 231.555674 |
| 16 | 0.986111 | 0.013889 | 32 | 226.990909 |
| 17 | 0.981481 | 0.018519 | 32 | 230.807854 |
| 18 | 0.974537 | 0.025463 | 32 | 234.681901 |
| 19 | 0.979167 | 0.020833 | 32 | 241.254538 |
| 20 | 0.962963 | 0.037037 | 32 | 244.268623 |
| 21 | 0.962963 | 0.037037 | 32 | 248.276521 |
| 22 | 0.976852 | 0.023148 | 32 | 263.952307 |
| 23 | 0.986111 | 0.013889 | 32 | 269.189546 |
| 24 | 0.981481 | 0.018519 | 32 | 273.857618 |
| 25 | 0.969907 | 0.030093 | 32 | 261.397656 |
| 26 | 0.972222 | 0.027778 | 32 | 262.945080 |
| 27 | 0.958333 | 0.041667 | 32 | 264.379748 |
| 28 | 0.965278 | 0.034722 | 16 | 248.642184 |
| 29 | 0.956019 | 0.043981 | 16 | 249.802035 |
| 30 | 0.967593 | 0.032407 | 16 | 251.448573 |

Tabla 12.23: Resultados individuales. Conjunto de datos Car. Clasificación ordinal.

12.1.2.5. Conjunto de datos Contact lenses

En la Tabla 12.24 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos Contact Lenses, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.24.

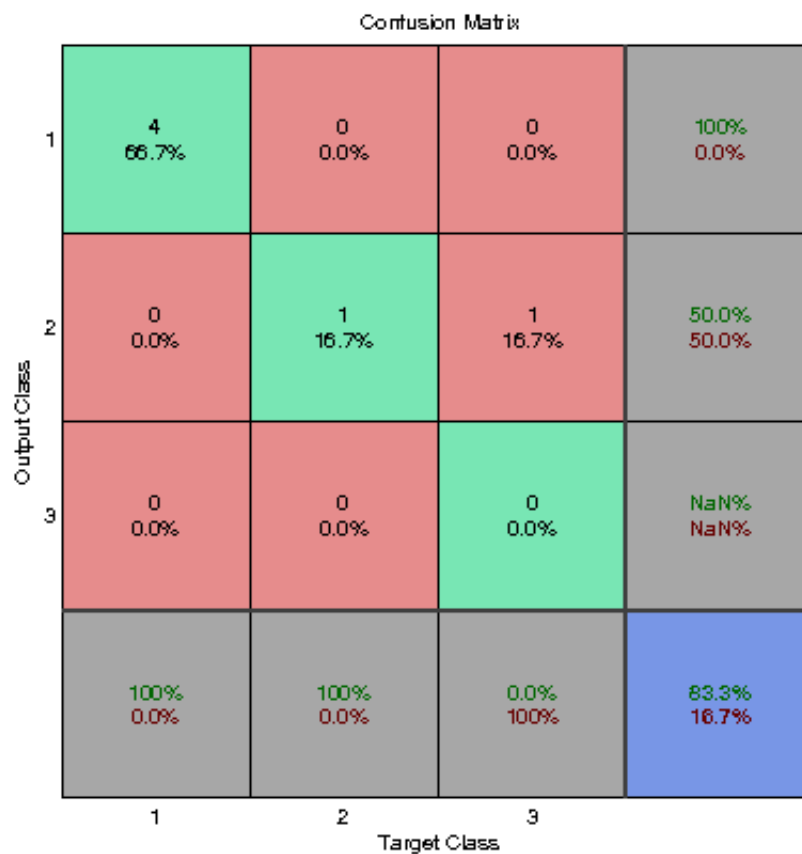


Figura 12.24: Matriz de confusión. Conjunto de datos Contact Lenses. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.833333 | 0.166667 | 8 | 51.198859 |
| 2 | 0.666667 | 0.333333 | 8 | 51.744441 |
| 3 | 0.833333 | 0.333333 | 8 | 52.360742 |
| 4 | 0.500000 | 0.500000 | 16 | 56.885605 |
| 5 | 0.500000 | 0.500000 | 16 | 57.525216 |
| 6 | 0.500000 | 0.500000 | 16 | 58.067246 |
| 7 | 0.333333 | 0.833333 | 4 | 47.291733 |
| 8 | 0.500000 | 0.666667 | 4 | 47.890259 |
| 9 | 0.333333 | 0.666667 | 4 | 48.471094 |
| 10 | 0.666667 | 0.333333 | 8 | 54.795185 |
| 11 | 0.666667 | 0.333333 | 8 | 55.640687 |
| 12 | 0.666667 | 0.333333 | 8 | 56.247796 |
| 13 | 0.666667 | 0.500000 | 8 | 52.674727 |
| 14 | 0.666667 | 0.500000 | 8 | 53.311953 |
| 15 | 0.500000 | 0.666667 | 8 | 53.874049 |
| 16 | 0.833333 | 0.166667 | 32 | 51.466867 |
| 17 | 0.833333 | 0.166667 | 32 | 52.038862 |
| 18 | 0.833333 | 0.166667 | 32 | 52.605775 |
| 19 | 0.500000 | 0.833333 | 4 | 50.051482 |
| 20 | 0.666667 | 0.500000 | 4 | 50.652619 |
| 21 | 0.666667 | 0.500000 | 4 | 51.263870 |
| 22 | 0.833333 | 0.333333 | 16 | 57.963919 |
| 23 | 0.833333 | 0.333333 | 16 | 58.541606 |
| 24 | 0.833333 | 0.333333 | 16 | 59.156156 |
| 25 | 0.833333 | 0.166667 | 16 | 52.982173 |
| 26 | 0.666667 | 0.333333 | 16 | 53.646113 |
| 27 | 0.666667 | 0.333333 | 16 | 54.223429 |
| 28 | 0.500000 | 0.500000 | 32 | 56.966584 |
| 29 | 0.666667 | 0.500000 | 32 | 57.501004 |
| 30 | 0.500000 | 0.500000 | 32 | 58.527886 |

Tabla 12.24: Resultados individuales. Conjunto de datos Contact Lenses. Clasificación ordinal.

12.1.2.6. Conjunto de datos Depression

En la Tabla 12.25 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos Depression, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.750000 | 0.303030 | 2 | 43.105924 |
| 2 | 0.757576 | 0.295455 | 2 | 43.707960 |
| 3 | 0.750000 | 0.325758 | 2 | 44.347971 |

Tabla 12.25: Resultados individuales. Conjunto de datos Depression. Clasificación ordinal.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.25.

12.1.2.7. Conjunto de datos ERA

En la Tabla 12.26 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos ERA, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.26.

12.1.2.8. Conjunto de datos ESL

En la Tabla 12.27 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos ESL, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas

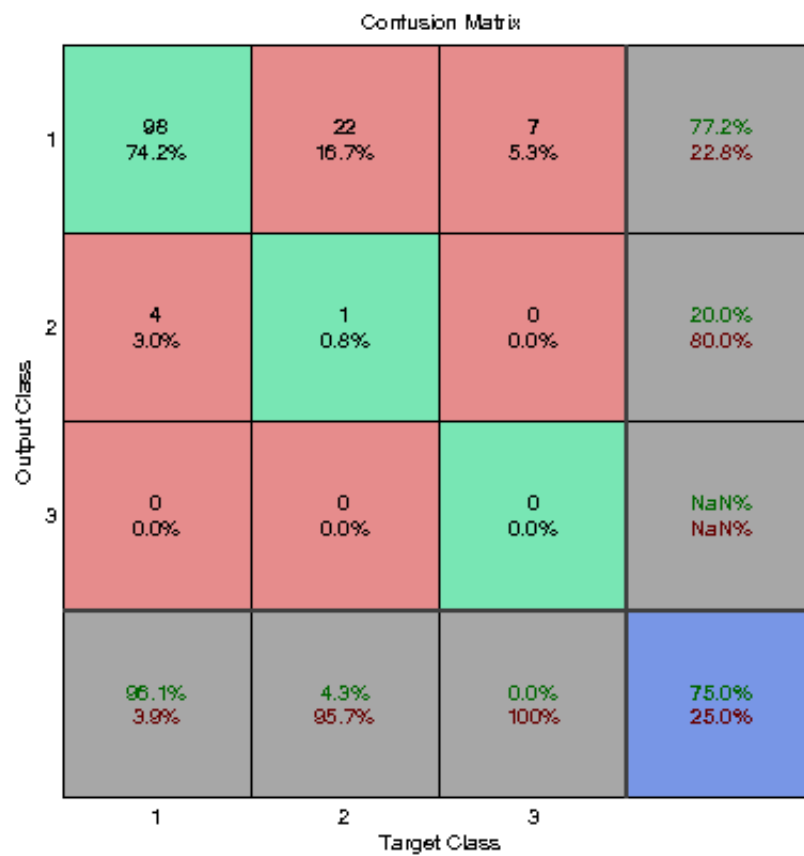


Figura 12.25: Matriz de confusión. Conjunto de datos Depression. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.320000 | 1.156000 | 8 | 87.299913 |
| 2 | 0.332000 | 1.164000 | 8 | 88.511411 |
| 3 | 0.256000 | 1.232000 | 8 | 89.481787 |
| 4 | 0.276000 | 1.172000 | 16 | 92.553754 |
| 5 | 0.240000 | 1.260000 | 16 | 93.346466 |
| 6 | 0.232000 | 1.256000 | 16 | 94.618155 |
| 7 | 0.288000 | 1.188000 | 16 | 88.554431 |
| 8 | 0.248000 | 1.268000 | 16 | 89.796162 |
| 9 | 0.296000 | 1.216000 | 16 | 90.997336 |
| 10 | 0.280000 | 1.180000 | 16 | 90.757959 |
| 11 | 0.284000 | 1.300000 | 16 | 92.294864 |
| 12 | 0.264000 | 1.264000 | 16 | 93.218881 |
| 13 | 0.244000 | 1.252000 | 1 | 98.233644 |
| 14 | 0.276000 | 1.196000 | 1 | 99.107357 |
| 15 | 0.236000 | 1.368000 | 1 | 99.968383 |
| 16 | 0.280000 | 1.208000 | 32 | 89.114847 |
| 17 | 0.296000 | 1.140000 | 32 | 90.517757 |
| 18 | 0.284000 | 1.160000 | 32 | 91.947673 |
| 19 | 0.264000 | 1.184000 | 4 | 90.715769 |
| 20 | 0.256000 | 1.340000 | 4 | 91.736844 |
| 21 | 0.252000 | 1.280000 | 4 | 92.716020 |
| 22 | 0.276000 | 1.212000 | 8 | 90.651273 |
| 23 | 0.256000 | 1.248000 | 8 | 91.893868 |
| 24 | 0.272000 | 1.200000 | 8 | 92.971112 |
| 25 | 0.280000 | 1.288000 | 32 | 92.285472 |
| 26 | 0.252000 | 1.472000 | 32 | 93.534491 |
| 27 | 0.272000 | 1.436000 | 32 | 94.667319 |
| 28 | 0.216000 | 1.260000 | 32 | 88.347897 |
| 29 | 0.264000 | 1.140000 | 32 | 89.220068 |
| 30 | 0.236000 | 1.164000 | 32 | 90.009554 |

Tabla 12.26: Resultados individuales. Conjunto de datos ERA. Clasificación ordinal.

Confusion Matrix

| | | | | | | | | | | | |
|--------------|---|------------|----------------|----------------|----------------|----------------|----------------|---------------|----------------|--------------|----------------|
| Output Class | 1 | 12 4.8% | 4 1.6% | 4 1.6% | 0 0.0% | 1 0.4% | 1 0.4% | 0 0.0% | 0 0.0% | 0 0.0% | 54.5% 45.5% |
| | 2 | 4 1.6% | 10 4.0% | 9 3.6% | 3 1.2% | 0 0.0% | 2 0.8% | 1 0.4% | 0 0.0% | 0 0.0% | 34.5% 65.5% |
| | 3 | 4 1.6% | 7 2.8% | 16 6.4% | 10 4.0% | 12 4.8% | 4 1.6% | 3 1.2% | 0 0.0% | 0 0.0% | 28.6% 71.4% |
| | 4 | 1 0.4% | 9 3.6% | 9 3.6% | 12 4.8% | 7 2.8% | 4 1.6% | 1 0.4% | 0 0.0% | 0 0.0% | 27.9% 72.1% |
| | 5 | 2 0.8% | 6 2.4% | 5 2.0% | 15 6.0% | 15 6.0% | 10 4.0% | 6 2.4% | 1 0.4% | 0 0.0% | 25.0% 75.0% |
| | 6 | 0 0.0% | 0 0.0% | 1 0.4% | 0 0.0% | 2 0.8% | 2 0.8% | 2 0.8% | 0 0.0% | 0 0.0% | 28.6% 71.4% |
| | 7 | 0 0.0% | 0 0.0% | 1 0.4% | 3 1.2% | 3 1.2% | 6 2.4% | 9 3.6% | 6 2.4% | 0 0.0% | 32.1% 67.9% |
| | 8 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 9 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 1 0.4% | 4 1.6% | 80.0% 20.0% |
| | | | 52.2% 47.8% | 27.8% 72.2% | 35.6% 64.4% | 27.9% 72.1% | 37.5% 62.5% | 6.9% 93.1% | 40.9% 59.1% | 0.0% 100% | 32.0% 68.0% |
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | | | Target Class | | | | | | | | |

Figura 12.26: Matriz de confusión. Conjunto de datos ERA. Clasificación ordinal.

en capa oculta calculado y el tiempo total de la ejecución.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.762295 | 0.245902 | 32 | 73.111055 |
| 2 | 0.762295 | 0.254098 | 32 | 73.954598 |
| 3 | 0.721311 | 0.295082 | 32 | 74.902812 |
| 4 | 0.704918 | 0.311475 | 32 | 72.731284 |
| 5 | 0.721311 | 0.295082 | 32 | 73.573849 |
| 6 | 0.713115 | 0.319672 | 32 | 74.395547 |
| 7 | 0.729508 | 0.278689 | 32 | 72.644063 |
| 8 | 0.614754 | 0.409836 | 32 | 73.415949 |
| 9 | 0.614754 | 0.385246 | 32 | 74.203063 |
| 10 | 0.729508 | 0.286885 | 32 | 72.923912 |
| 11 | 0.713115 | 0.303279 | 32 | 73.896835 |
| 12 | 0.721311 | 0.295082 | 32 | 74.752250 |
| 13 | 0.696721 | 0.319672 | 1 | 73.309400 |
| 14 | 0.680328 | 0.344262 | 1 | 74.430872 |
| 15 | 0.713115 | 0.311475 | 1 | 75.333860 |
| 16 | 0.729508 | 0.278689 | 32 | 68.860636 |
| 17 | 0.778689 | 0.221311 | 32 | 69.771369 |
| 18 | 0.754098 | 0.254098 | 32 | 70.685123 |
| 19 | 0.655738 | 0.368852 | 8 | 71.847641 |
| 20 | 0.688525 | 0.319672 | 8 | 72.982492 |
| 21 | 0.672131 | 0.336066 | 8 | 73.809356 |
| 22 | 0.663934 | 0.368852 | 16 | 75.645535 |
| 23 | 0.778689 | 0.254098 | 16 | 76.503528 |
| 24 | 0.696721 | 0.344262 | 16 | 77.310532 |
| 25 | 0.729508 | 0.278689 | 16 | 76.199186 |
| 26 | 0.762295 | 0.245902 | 16 | 77.482252 |
| 27 | 0.745902 | 0.262295 | 16 | 78.488366 |
| 28 | 0.713115 | 0.303279 | 16 | 74.576613 |
| 29 | 0.754098 | 0.270492 | 16 | 75.391821 |
| 30 | 0.713115 | 0.319672 | 16 | 76.566124 |

Tabla 12.27: Resultados individuales. Conjunto de datos ESL. Clasificación ordinal.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.27.

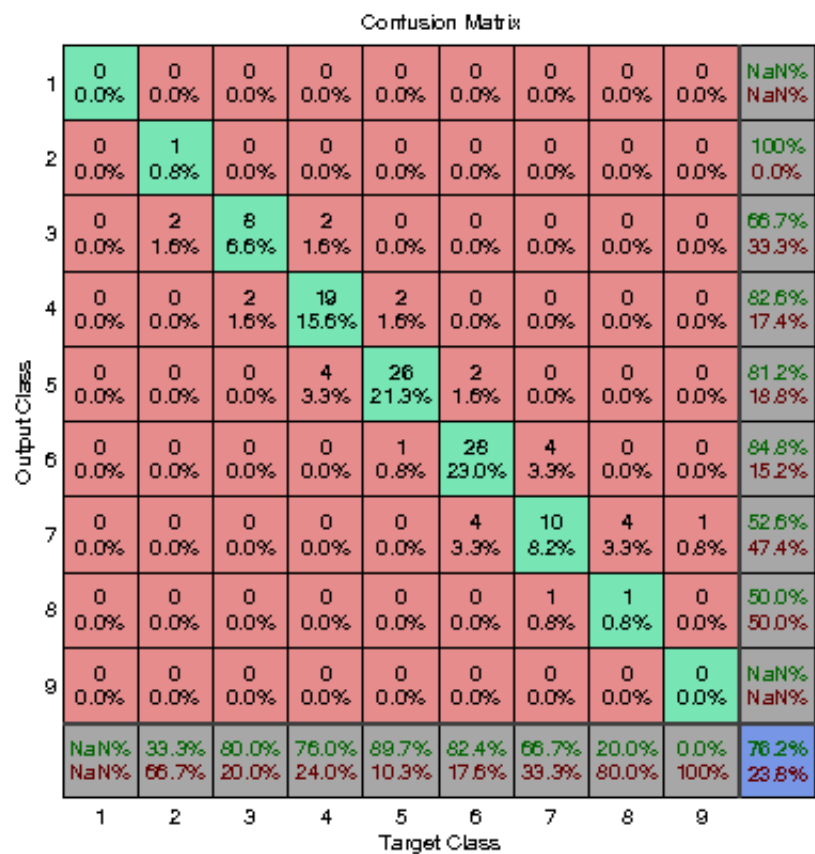


Figura 12.27: Matriz de confusión. Conjunto de datos ESL. Clasificación ordinal.

12.1.2.9. Conjunto de datos Eucalyptus

En la Tabla 12.28 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Eucalyptus, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.28.

Confusion Matrix

| | | | | | | |
|---|----------------|----------------|----------------|----------------|----------------|----------------|
| 1 | 39 21.2% | 6 3.3% | 0 0.0% | 1 0.5% | 0 0.0% | 84.8% 15.2% |
| 2 | 6 3.3% | 15 8.2% | 9 4.9% | 2 1.1% | 0 0.0% | 46.9% 53.1% |
| 3 | 0 0.0% | 4 2.2% | 18 9.8% | 9 4.9% | 1 0.5% | 56.2% 43.8% |
| 4 | 0 0.0% | 2 1.1% | 5 2.7% | 36 19.6% | 11 6.0% | 66.7% 33.3% |
| 5 | 0 0.0% | 0 0.0% | 0 0.0% | 6 3.3% | 14 7.6% | 70.0% 30.0% |
| | 86.7% 13.3% | 55.6% 44.4% | 56.2% 43.8% | 66.7% 33.3% | 53.8% 46.2% | 66.3% 33.7% |
| | 1 | 2 | 3 | 4 | 5 | |
| | Target Class | | | | | |

Output Class

Figura 12.28: Matriz de confusión. Conjunto de datos Eucalyptus. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.663043 | 0.375000 | 32 | 139.904202 |
| 2 | 0.695652 | 0.336957 | 32 | 141.017066 |
| 3 | 0.663043 | 0.375000 | 32 | 141.906766 |
| 4 | 0.614130 | 0.413043 | 32 | 133.851583 |
| 5 | 0.646739 | 0.375000 | 32 | 135.390441 |
| 6 | 0.603261 | 0.434783 | 32 | 136.199342 |
| 7 | 0.635870 | 0.402174 | 32 | 134.407011 |
| 8 | 0.663043 | 0.385870 | 32 | 135.440279 |
| 9 | 0.614130 | 0.429348 | 32 | 136.585156 |
| 10 | 0.711957 | 0.342391 | 16 | 131.246496 |
| 11 | 0.717391 | 0.336957 | 16 | 132.146056 |
| 12 | 0.679348 | 0.385870 | 16 | 133.291784 |
| 13 | 0.635870 | 0.413043 | 32 | 137.933579 |
| 14 | 0.652174 | 0.385870 | 32 | 138.898911 |
| 15 | 0.635870 | 0.396739 | 32 | 140.001763 |
| 16 | 0.559783 | 0.456522 | 32 | 138.885796 |
| 17 | 0.619565 | 0.402174 | 32 | 140.161285 |
| 18 | 0.635870 | 0.380435 | 32 | 141.715885 |
| 19 | 0.641304 | 0.413043 | 32 | 130.816867 |
| 20 | 0.625000 | 0.413043 | 32 | 132.149242 |
| 21 | 0.673913 | 0.364130 | 32 | 133.181465 |
| 22 | 0.641304 | 0.385870 | 32 | 132.055419 |
| 23 | 0.679348 | 0.353261 | 32 | 133.322255 |
| 24 | 0.625000 | 0.396739 | 32 | 134.621499 |
| 25 | 0.701087 | 0.342391 | 16 | 100.018357 |
| 26 | 0.711957 | 0.309783 | 16 | 101.103338 |
| 27 | 0.657609 | 0.364130 | 16 | 102.228588 |
| 28 | 0.625000 | 0.407609 | 16 | 97.540006 |
| 29 | 0.635870 | 0.407609 | 16 | 98.621027 |
| 30 | 0.663043 | 0.369565 | 16 | 99.599710 |

Tabla 12.28: Resultados individuales. Conjunto de datos Eucalyptus. Clasificación ordinal.

12.1.2.10. Conjunto de datos LEV

En la Tabla 12.29 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos LEV, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.29.

Confusion Matrix

| | | | | | | |
|---|----------------|----------------|----------------|----------------|--------------|----------------|
| 1 | 14 5.6% | 11 4.4% | 2 0.8% | 0 0.0% | 0 0.0% | 51.9% 48.1% |
| 2 | 6 2.4% | 42 16.8% | 19 7.6% | 0 0.0% | 0 0.0% | 62.7% 37.3% |
| 3 | 4 1.6% | 16 6.4% | 68 27.2% | 25 10.0% | 2 0.8% | 59.1% 40.9% |
| 4 | 0 0.0% | 1 0.4% | 11 4.4% | 25 10.0% | 4 1.6% | 61.0% 39.0% |
| 5 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 58.3% 41.7% | 60.0% 40.0% | 68.0% 32.0% | 50.0% 50.0% | 0.0% 100% | 59.6% 40.4% |
| | 1 | 2 | 3 | 4 | 5 | |
| | Target Class | | | | | |

Figura 12.29: Matriz de confusión. Conjunto de datos LEV. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.596000 | 0.440000 | 1 | 71.444229 |
| 2 | 0.600000 | 0.428000 | 1 | 72.153309 |
| 3 | 0.588000 | 0.440000 | 1 | 72.778424 |
| 4 | 0.548000 | 0.492000 | 16 | 73.158229 |
| 5 | 0.512000 | 0.516000 | 16 | 73.751552 |
| 6 | 0.556000 | 0.484000 | 16 | 74.402532 |
| 7 | 0.664000 | 0.356000 | 32 | 70.268842 |
| 8 | 0.668000 | 0.352000 | 32 | 70.951872 |
| 9 | 0.672000 | 0.356000 | 32 | 71.690842 |
| 10 | 0.620000 | 0.404000 | 1 | 68.617956 |
| 11 | 0.636000 | 0.392000 | 1 | 69.523429 |
| 12 | 0.616000 | 0.412000 | 1 | 70.526389 |
| 13 | 0.604000 | 0.420000 | 16 | 72.057636 |
| 14 | 0.612000 | 0.404000 | 16 | 72.689486 |
| 15 | 0.636000 | 0.380000 | 16 | 73.315681 |
| 16 | 0.620000 | 0.424000 | 16 | 72.732791 |
| 17 | 0.600000 | 0.444000 | 16 | 73.418169 |
| 18 | 0.604000 | 0.436000 | 16 | 74.054849 |
| 19 | 0.540000 | 0.480000 | 8 | 72.068269 |
| 20 | 0.568000 | 0.448000 | 8 | 72.674963 |
| 21 | 0.568000 | 0.444000 | 8 | 73.281402 |
| 22 | 0.652000 | 0.368000 | 32 | 67.702705 |
| 23 | 0.644000 | 0.372000 | 32 | 68.392009 |
| 24 | 0.648000 | 0.372000 | 32 | 69.224341 |
| 25 | 0.620000 | 0.400000 | 1 | 67.442029 |
| 26 | 0.600000 | 0.424000 | 1 | 68.160931 |
| 27 | 0.620000 | 0.400000 | 1 | 69.214297 |
| 28 | 0.580000 | 0.448000 | 1 | 66.942955 |
| 29 | 0.664000 | 0.372000 | 1 | 67.764071 |
| 30 | 0.556000 | 0.472000 | 1 | 68.477861 |

Tabla 12.29: Resultados individuales. Conjunto de datos LEV. Clasificación ordinal.

12.1.2.11. Conjunto de datos New Thyroid

En la Tabla 12.30 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos New Thyroid, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.30.

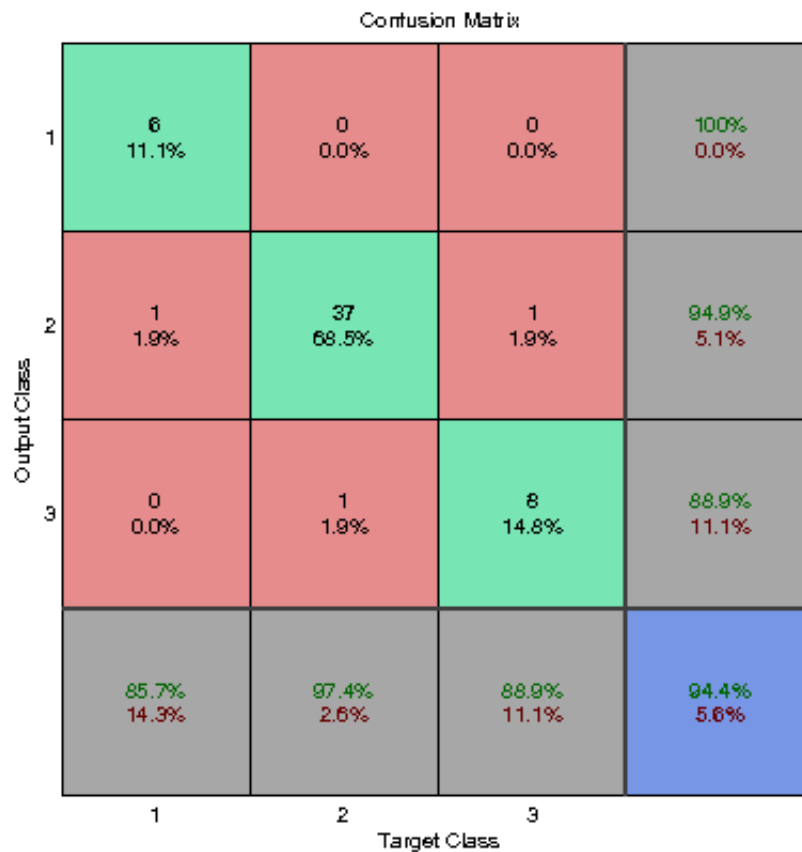


Figura 12.30: Matriz de confusión. Conjunto de datos New Thyroid. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.944444 | 0.055556 | 4 | 70.795148 |
| 2 | 0.962963 | 0.037037 | 4 | 71.550893 |
| 3 | 0.944444 | 0.055556 | 4 | 72.267727 |
| 4 | 0.981481 | 0.018519 | 4 | 68.680562 |
| 5 | 0.981481 | 0.018519 | 4 | 69.369708 |
| 6 | 0.981481 | 0.018519 | 4 | 70.154898 |
| 7 | 0.944444 | 0.055556 | 1 | 71.276647 |
| 8 | 0.925926 | 0.074074 | 1 | 72.438756 |
| 9 | 0.925926 | 0.074074 | 1 | 73.213927 |
| 10 | 0.981481 | 0.018519 | 1 | 69.448458 |
| 11 | 0.962963 | 0.037037 | 1 | 70.264830 |
| 12 | 0.962963 | 0.037037 | 1 | 71.021653 |
| 13 | 0.962963 | 0.037037 | 1 | 69.419543 |
| 14 | 0.981481 | 0.018519 | 1 | 70.237790 |
| 15 | 0.962963 | 0.037037 | 1 | 70.916475 |
| 16 | 0.944444 | 0.055556 | 1 | 68.390146 |
| 17 | 0.925926 | 0.074074 | 1 | 69.133175 |
| 18 | 1.000000 | 0.000000 | 1 | 69.801357 |
| 19 | 0.962963 | 0.037037 | 4 | 68.148403 |
| 20 | 0.925926 | 0.074074 | 4 | 68.829329 |
| 21 | 0.925926 | 0.074074 | 4 | 69.518724 |
| 22 | 0.981481 | 0.018519 | 4 | 70.864581 |
| 23 | 0.962963 | 0.037037 | 4 | 71.629677 |
| 24 | 0.944444 | 0.055556 | 4 | 72.436795 |
| 25 | 0.981481 | 0.018519 | 32 | 68.602528 |
| 26 | 0.981481 | 0.018519 | 32 | 69.185656 |
| 27 | 0.981481 | 0.018519 | 32 | 69.817260 |
| 28 | 1.000000 | 0.000000 | 1 | 68.967954 |
| 29 | 1.000000 | 0.000000 | 1 | 69.655350 |
| 30 | 1.000000 | 0.000000 | 1 | 70.437470 |

Tabla 12.30: Resultados individuales. Conjunto de datos New Thyroid. Clasificación ordinal.

12.1.2.12. Conjunto de datos Pasture

En la Tabla 12.31 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Pasture, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.31.

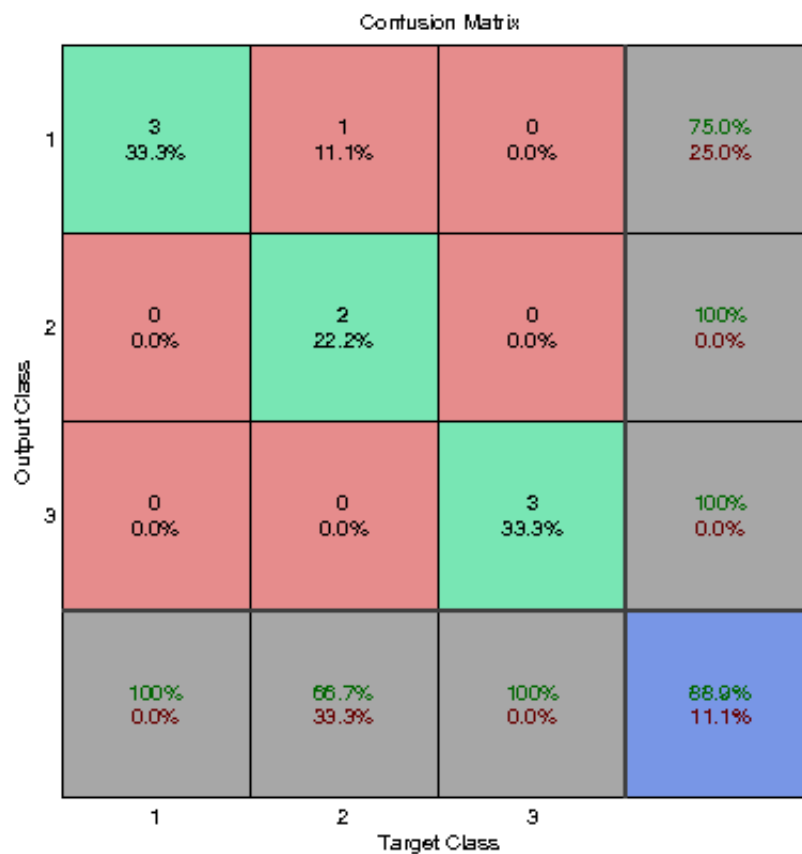


Figura 12.31: Matriz de confusión. Conjunto de datos Pasture. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.888889 | 0.111111 | 4 | 44.239209 |
| 2 | 0.666667 | 0.333333 | 4 | 44.810264 |
| 3 | 0.444444 | 0.555556 | 4 | 45.372224 |
| 4 | 0.777778 | 0.222222 | 2 | 43.439436 |
| 5 | 0.555556 | 0.444444 | 2 | 44.120185 |
| 6 | 0.666667 | 0.333333 | 2 | 44.701315 |
| 7 | 0.555556 | 0.444444 | 8 | 44.206138 |
| 8 | 0.777778 | 0.222222 | 8 | 44.793995 |
| 9 | 0.666667 | 0.333333 | 8 | 45.417036 |
| 10 | 0.666667 | 0.333333 | 2 | 44.245170 |
| 11 | 0.666667 | 0.333333 | 2 | 44.898652 |
| 12 | 0.777778 | 0.222222 | 2 | 45.473140 |
| 13 | 0.777778 | 0.222222 | 8 | 41.860688 |
| 14 | 0.777778 | 0.222222 | 8 | 42.431805 |
| 15 | 0.777778 | 0.222222 | 8 | 42.999920 |
| 16 | 0.555556 | 0.444444 | 4 | 46.186465 |
| 17 | 0.555556 | 0.444444 | 4 | 46.805314 |
| 18 | 0.555556 | 0.444444 | 4 | 47.440218 |
| 19 | 0.555556 | 0.444444 | 4 | 44.815651 |
| 20 | 0.555556 | 0.444444 | 4 | 45.430516 |
| 21 | 0.555556 | 0.444444 | 4 | 46.006458 |
| 22 | 0.666667 | 0.333333 | 32 | 44.332883 |
| 23 | 0.555556 | 0.444444 | 32 | 44.907047 |
| 24 | 0.555556 | 0.444444 | 32 | 45.486395 |
| 25 | 0.666667 | 0.333333 | 32 | 41.700159 |
| 26 | 0.777778 | 0.222222 | 32 | 42.267586 |
| 27 | 0.888889 | 0.111111 | 32 | 42.838798 |
| 28 | 0.555556 | 0.444444 | 1 | 42.496249 |
| 29 | 0.666667 | 0.333333 | 1 | 43.173644 |
| 30 | 0.333333 | 0.666667 | 1 | 43.750931 |

Tabla 12.31: Resultados individuales. Conjunto de datos Pasture. Clasificación ordinal.

12.1.2.13. Conjunto de datos Squash Stored

En la Tabla 12.32 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Squash Stored, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.32.

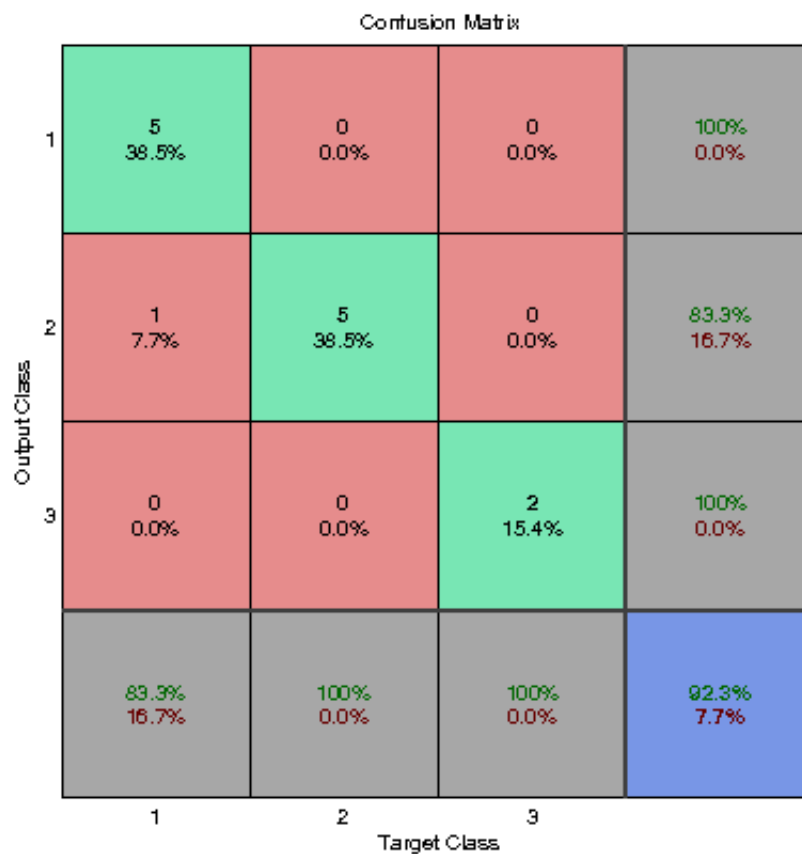


Figura 12.32: Matriz de confusión. Conjunto de datos Squash Stored. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.923077 | 0.076923 | 4 | 42.816743 |
| 2 | 0.769231 | 0.230769 | 4 | 43.426460 |
| 3 | 0.769231 | 0.230769 | 4 | 44.005752 |
| 4 | 0.461538 | 0.538462 | 16 | 43.012103 |
| 5 | 0.615385 | 0.461538 | 16 | 43.615296 |
| 6 | 0.692308 | 0.384615 | 16 | 44.213128 |
| 7 | 0.769231 | 0.230769 | 8 | 47.099631 |
| 8 | 0.769231 | 0.230769 | 8 | 47.738045 |
| 9 | 0.846154 | 0.153846 | 8 | 48.316245 |
| 10 | 0.615385 | 0.461538 | 8 | 48.566001 |
| 11 | 0.461538 | 0.538462 | 8 | 49.179877 |
| 12 | 0.538462 | 0.538462 | 8 | 49.756939 |
| 13 | 0.538462 | 0.461538 | 32 | 44.304642 |
| 14 | 0.538462 | 0.461538 | 32 | 44.915091 |
| 15 | 0.692308 | 0.307692 | 32 | 45.547607 |
| 16 | 0.692308 | 0.307692 | 16 | 41.935247 |
| 17 | 0.538462 | 0.461538 | 16 | 42.576700 |
| 18 | 0.769231 | 0.230769 | 16 | 43.172383 |
| 19 | 0.692308 | 0.307692 | 4 | 43.690080 |
| 20 | 0.692308 | 0.307692 | 4 | 44.345734 |
| 21 | 0.769231 | 0.230769 | 4 | 44.919982 |
| 22 | 0.692308 | 0.307692 | 16 | 41.849240 |
| 23 | 0.615385 | 0.384615 | 16 | 42.518294 |
| 24 | 0.692308 | 0.307692 | 16 | 43.095379 |
| 25 | 0.692308 | 0.384615 | 8 | 45.163836 |
| 26 | 0.615385 | 0.384615 | 8 | 45.820833 |
| 27 | 0.769231 | 0.230769 | 8 | 46.723140 |
| 28 | 0.538462 | 0.538462 | 8 | 45.369032 |
| 29 | 0.538462 | 0.461538 | 8 | 46.010182 |
| 30 | 0.538462 | 0.461538 | 8 | 46.630138 |

Tabla 12.32: Resultados individuales. Conjunto de datos Squash Stored. Clasificación ordinal.

12.1.2.14. Conjunto de datos Squash Unstored

En la Tabla 12.33 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Squash Unstored, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.33.

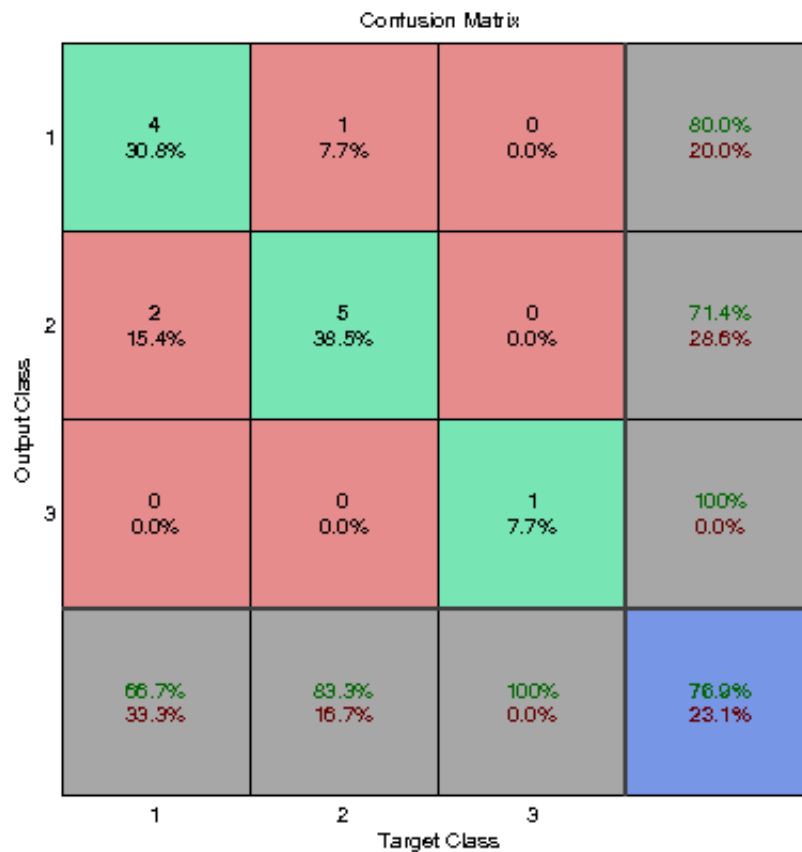


Figura 12.33: Matriz de confusión. Conjunto de datos Squash Unstored. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.769231 | 0.230769 | 4 | 46.315533 |
| 2 | 0.692308 | 0.307692 | 4 | 46.914511 |
| 3 | 0.769231 | 0.230769 | 4 | 47.496495 |
| 4 | 0.692308 | 0.307692 | 16 | 47.315828 |
| 5 | 0.615385 | 0.384615 | 16 | 48.018074 |
| 6 | 0.692308 | 0.307692 | 16 | 48.655599 |
| 7 | 0.538462 | 0.461538 | 4 | 45.934891 |
| 8 | 0.538462 | 0.461538 | 4 | 46.533829 |
| 9 | 0.692308 | 0.307692 | 4 | 47.168286 |
| 10 | 0.769231 | 0.230769 | 8 | 48.374302 |
| 11 | 0.769231 | 0.230769 | 8 | 49.395310 |
| 12 | 0.461538 | 0.538462 | 8 | 50.049380 |
| 13 | 0.923077 | 0.076923 | 16 | 47.316657 |
| 14 | 0.461538 | 0.538462 | 16 | 47.907707 |
| 15 | 0.692308 | 0.307692 | 16 | 48.478897 |
| 16 | 0.769231 | 0.230769 | 32 | 44.453136 |
| 17 | 0.461538 | 0.538462 | 32 | 45.163293 |
| 18 | 0.538462 | 0.461538 | 32 | 45.763247 |
| 19 | 1.000000 | 0.000000 | 32 | 43.847262 |
| 20 | 0.769231 | 0.230769 | 32 | 44.485351 |
| 21 | 0.923077 | 0.076923 | 32 | 45.117502 |
| 22 | 0.461538 | 0.538462 | 1 | 55.929857 |
| 23 | 0.769231 | 0.230769 | 1 | 56.599683 |
| 24 | 0.692308 | 0.307692 | 1 | 57.263187 |
| 25 | 0.692308 | 0.307692 | 4 | 59.913640 |
| 26 | 0.461538 | 0.538462 | 4 | 60.698151 |
| 27 | 0.615385 | 0.384615 | 4 | 61.770435 |
| 28 | 0.461538 | 0.615385 | 16 | 58.030398 |
| 29 | 0.461538 | 0.538462 | 16 | 58.927919 |
| 30 | 0.461538 | 0.538462 | 16 | 59.887714 |

Tabla 12.33: Resultados individuales. Conjunto de datos Squash Unstored. Clasificación ordinal.

12.1.2.15. Conjunto de datos SWD

En la Tabla 12.34 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos SWD, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.34.

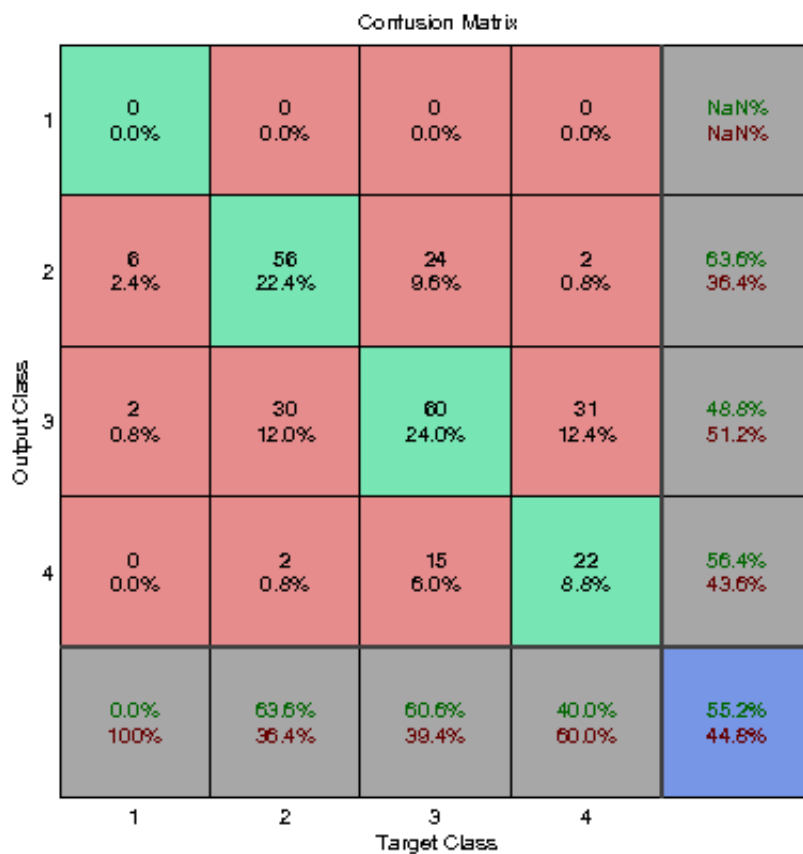


Figura 12.34: Matriz de confusión. Conjunto de datos SWD. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.552000 | 0.472000 | 8 | 63.418932 |
| 2 | 0.580000 | 0.440000 | 8 | 64.041602 |
| 3 | 0.568000 | 0.452000 | 8 | 64.599161 |
| 4 | 0.552000 | 0.456000 | 8 | 66.767482 |
| 5 | 0.584000 | 0.448000 | 8 | 67.456884 |
| 6 | 0.576000 | 0.448000 | 8 | 68.171364 |
| 7 | 0.592000 | 0.432000 | 16 | 83.609277 |
| 8 | 0.600000 | 0.416000 | 16 | 84.366476 |
| 9 | 0.560000 | 0.480000 | 16 | 85.291162 |
| 10 | 0.572000 | 0.436000 | 8 | 83.914320 |
| 11 | 0.624000 | 0.388000 | 8 | 84.618615 |
| 12 | 0.584000 | 0.432000 | 8 | 85.567129 |
| 13 | 0.624000 | 0.380000 | 16 | 68.208843 |
| 14 | 0.600000 | 0.404000 | 16 | 68.956250 |
| 15 | 0.600000 | 0.412000 | 16 | 70.159131 |
| 16 | 0.584000 | 0.428000 | 8 | 64.160502 |
| 17 | 0.536000 | 0.480000 | 8 | 64.732714 |
| 18 | 0.564000 | 0.464000 | 8 | 65.322255 |
| 19 | 0.576000 | 0.444000 | 8 | 67.910933 |
| 20 | 0.560000 | 0.444000 | 8 | 68.631902 |
| 21 | 0.584000 | 0.428000 | 8 | 69.548274 |
| 22 | 0.524000 | 0.504000 | 8 | 68.898980 |
| 23 | 0.560000 | 0.468000 | 8 | 69.592622 |
| 24 | 0.536000 | 0.492000 | 8 | 70.357339 |
| 25 | 0.548000 | 0.484000 | 32 | 71.535946 |
| 26 | 0.528000 | 0.524000 | 32 | 72.558646 |
| 27 | 0.532000 | 0.500000 | 32 | 73.465484 |
| 28 | 0.524000 | 0.484000 | 16 | 64.914710 |
| 29 | 0.552000 | 0.456000 | 16 | 65.506323 |
| 30 | 0.556000 | 0.448000 | 16 | 66.168129 |

Tabla 12.34: Resultados individuales. Conjunto de datos SWD. Clasificación ordinal.

12.1.2.16. Conjunto de datos TAE

En la Tabla 12.35 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos TAE, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.35.

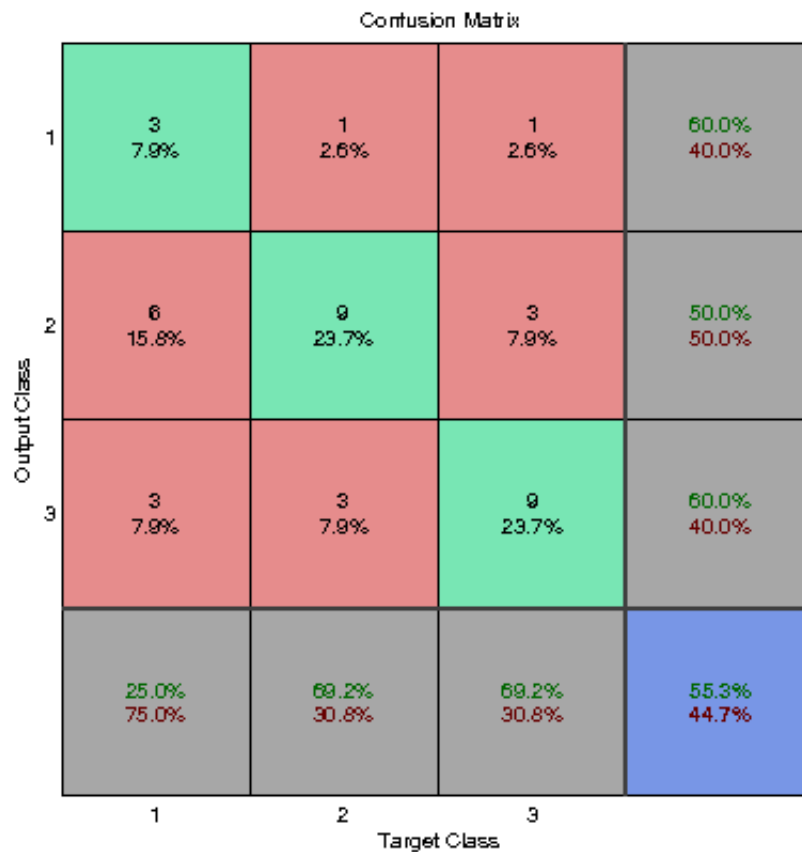


Figura 12.35: Matriz de confusión. Conjunto de datos TAE. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|-----------|
| 1 | 0.552632 | 0.552632 | 32 | 84.020168 |
| 2 | 0.473684 | 0.657895 | 32 | 85.241138 |
| 3 | 0.526316 | 0.552632 | 32 | 87.018726 |
| 4 | 0.368421 | 0.631579 | 16 | 84.899315 |
| 5 | 0.342105 | 0.657895 | 16 | 85.887896 |
| 6 | 0.500000 | 0.578947 | 16 | 87.390597 |
| 7 | 0.447368 | 0.631579 | 32 | 72.606129 |
| 8 | 0.421053 | 0.631579 | 32 | 73.322966 |
| 9 | 0.315789 | 0.736842 | 32 | 74.069536 |
| 10 | 0.578947 | 0.552632 | 4 | 49.114281 |
| 11 | 0.473684 | 0.657895 | 4 | 49.722888 |
| 12 | 0.473684 | 0.710526 | 4 | 50.366237 |
| 13 | 0.473684 | 0.605263 | 8 | 82.864739 |
| 14 | 0.394737 | 0.684211 | 8 | 83.717713 |
| 15 | 0.342105 | 0.815789 | 8 | 84.482640 |
| 16 | 0.368421 | 0.657895 | 16 | 77.765865 |
| 17 | 0.342105 | 0.657895 | 16 | 79.065734 |
| 18 | 0.394737 | 0.710526 | 16 | 80.250863 |
| 19 | 0.500000 | 0.578947 | 8 | 77.036648 |
| 20 | 0.315789 | 0.684211 | 8 | 77.706177 |
| 21 | 0.526316 | 0.552632 | 8 | 78.512316 |
| 22 | 0.421053 | 0.736842 | 32 | 83.434261 |
| 23 | 0.421053 | 0.684211 | 32 | 84.489000 |
| 24 | 0.473684 | 0.684211 | 32 | 86.192510 |
| 25 | 0.552632 | 0.526316 | 16 | 84.224727 |
| 26 | 0.421053 | 0.736842 | 16 | 85.839300 |
| 27 | 0.473684 | 0.657895 | 16 | 87.469962 |
| 28 | 0.447368 | 0.657895 | 8 | 79.569105 |
| 29 | 0.473684 | 0.578947 | 8 | 80.431481 |
| 30 | 0.421053 | 0.684211 | 8 | 81.278828 |

Tabla 12.35: Resultados individuales. Conjunto de datos TAE. Clasificación ordinal.

12.1.2.17. Conjunto de datos Thyroid

En la Tabla 12.36 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Thyroid, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.36.

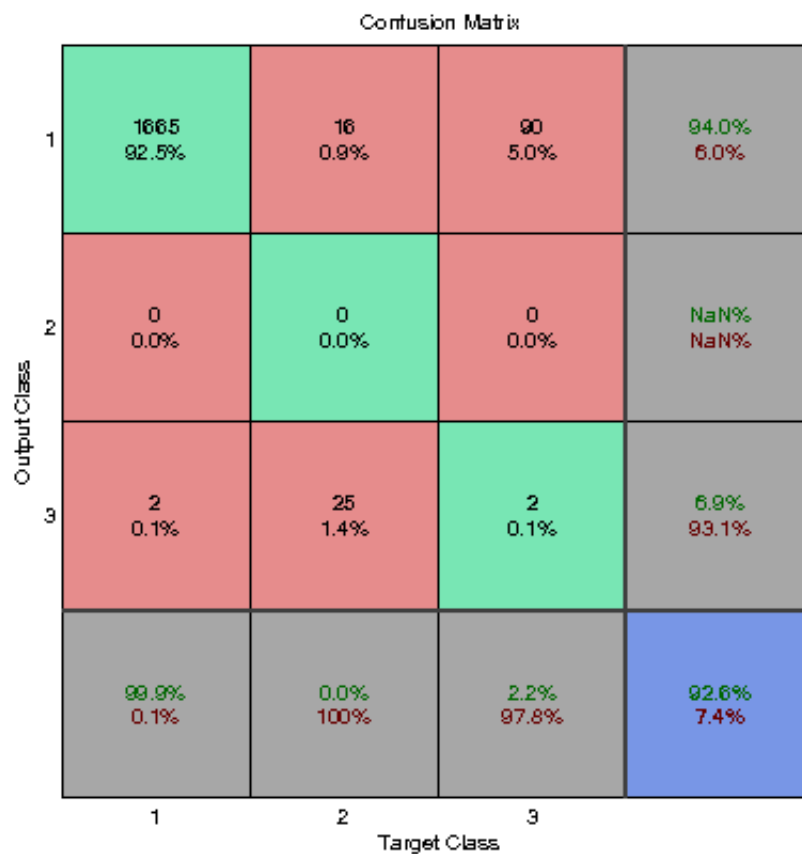


Figura 12.36: Matriz de confusión. Conjunto de datos Thyroid. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.926111 | 0.125000 | 32 | 546.704040 |
| 2 | 0.939444 | 0.098333 | 32 | 566.196460 |
| 3 | 0.937778 | 0.101667 | 32 | 588.249084 |
| 4 | 0.927778 | 0.121667 | 32 | 657.531746 |
| 5 | 0.930000 | 0.117222 | 32 | 677.320757 |
| 6 | 0.927778 | 0.121667 | 32 | 689.436802 |
| 7 | 0.927222 | 0.122778 | 16 | 524.396968 |
| 8 | 0.926667 | 0.123889 | 16 | 530.232181 |
| 9 | 0.926667 | 0.123889 | 16 | 538.010303 |
| 10 | 0.948889 | 0.079444 | 32 | 603.400977 |
| 11 | 0.926667 | 0.123889 | 32 | 609.786611 |
| 12 | 0.925556 | 0.126111 | 32 | 616.619637 |
| 13 | 0.926111 | 0.125000 | 16 | 464.242199 |
| 14 | 0.925556 | 0.126111 | 16 | 466.744543 |
| 15 | 0.925556 | 0.126111 | 16 | 474.353336 |
| 16 | 0.925556 | 0.126111 | 32 | 645.302691 |
| 17 | 0.925556 | 0.125556 | 32 | 647.292943 |
| 18 | 0.923889 | 0.129444 | 32 | 653.438097 |
| 19 | 0.926667 | 0.123889 | 32 | 541.553691 |
| 20 | 0.925556 | 0.126111 | 32 | 544.492318 |
| 21 | 0.926667 | 0.123889 | 32 | 549.985364 |
| 22 | 0.925556 | 0.126111 | 16 | 580.289160 |
| 23 | 0.925556 | 0.126111 | 16 | 583.232103 |
| 24 | 0.925556 | 0.126111 | 16 | 587.484288 |
| 25 | 0.940000 | 0.097222 | 32 | 472.892580 |
| 26 | 0.926667 | 0.123889 | 32 | 479.281923 |
| 27 | 0.927778 | 0.121667 | 32 | 489.366928 |
| 28 | 0.927222 | 0.122778 | 16 | 546.669293 |
| 29 | 0.934444 | 0.108333 | 16 | 553.731361 |
| 30 | 0.948333 | 0.080556 | 16 | 575.312857 |

Tabla 12.36: Resultados individuales. Conjunto de datos Thyroid. Clasificación ordinal.

12.1.2.18. Conjunto de datos Wine Quality Red

En la Tabla 12.37 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNNet mediante una red neuronal artificial ordinal para el conjunto de datos Wine Quality Red, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.37.

Confusion Matrix

| | | | | | | | |
|---|--------------|--------------|----------------|----------------|----------------|--------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 3 | 2 0.5% | 12 3.0% | 128 32.0% | 49 12.2% | 0 0.0% | 0 0.0% | 67.0% 33.0% |
| 4 | 0 0.0% | 1 0.2% | 41 10.2% | 96 24.0% | 34 8.5% | 2 0.5% | 55.2% 44.8% |
| 5 | 0 0.0% | 0 0.0% | 2 0.5% | 14 3.5% | 16 4.0% | 3 0.8% | 45.7% 54.3% |
| 6 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 0.0% 100% | 0.0% 100% | 74.9% 25.1% | 60.4% 39.6% | 32.0% 68.0% | 0.0% 100% | 60.0% 40.0% |
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| | Target Class | | | | | | |

Figura 12.37: Matriz de confusión. Conjunto de datos Wine Quality Red. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.600000 | 0.417500 | 8 | 178.738008 |
| 2 | 0.605000 | 0.420000 | 8 | 180.565251 |
| 3 | 0.600000 | 0.425000 | 8 | 182.573359 |
| 4 | 0.590000 | 0.432500 | 8 | 189.294793 |
| 5 | 0.597500 | 0.427500 | 8 | 191.244421 |
| 6 | 0.585000 | 0.445000 | 8 | 192.824698 |
| 7 | 0.550000 | 0.487500 | 16 | 195.254332 |
| 8 | 0.570000 | 0.462500 | 16 | 196.226974 |
| 9 | 0.580000 | 0.450000 | 16 | 197.247267 |
| 10 | 0.612500 | 0.420000 | 16 | 161.207331 |
| 11 | 0.537500 | 0.505000 | 16 | 162.744005 |
| 12 | 0.595000 | 0.440000 | 16 | 164.693878 |
| 13 | 0.560000 | 0.465000 | 32 | 160.657585 |
| 14 | 0.570000 | 0.460000 | 32 | 163.048084 |
| 15 | 0.590000 | 0.442500 | 32 | 164.832986 |
| 16 | 0.590000 | 0.442500 | 32 | 136.398673 |
| 17 | 0.600000 | 0.435000 | 32 | 140.812003 |
| 18 | 0.570000 | 0.467500 | 32 | 142.833603 |
| 19 | 0.540000 | 0.482500 | 16 | 135.031377 |
| 20 | 0.547500 | 0.482500 | 16 | 136.852452 |
| 21 | 0.560000 | 0.477500 | 16 | 138.812173 |
| 22 | 0.570000 | 0.467500 | 4 | 113.186516 |
| 23 | 0.575000 | 0.462500 | 4 | 114.280573 |
| 24 | 0.585000 | 0.450000 | 4 | 115.326427 |
| 25 | 0.595000 | 0.435000 | 32 | 141.680161 |
| 26 | 0.617500 | 0.412500 | 32 | 143.213124 |
| 27 | 0.600000 | 0.432500 | 32 | 144.646540 |
| 28 | 0.570000 | 0.457500 | 16 | 144.440529 |
| 29 | 0.585000 | 0.445000 | 16 | 145.359890 |
| 30 | 0.577500 | 0.447500 | 16 | 146.486127 |

Tabla 12.37: Resultados individuales. Conjunto de datos Wine Quality Red. Clasificación ordinal.

12.1.2.19. Conjunto de datos Wine Quality White

En la Tabla 12.38 se muestran los resultados individuales de las 30 ejecuciones del algoritmo de regresión ordinal ORNNet mediante una red neuronal artificial ordinal para el conjunto de datos Wine Quality White, donde se muestra el número de ejecuciones, el CCR, el MAE, el número de neuronas en capa oculta calculado y el tiempo total de la ejecución.

La matriz de confusión para el mejor resultado del CCR obtenido de las 30 ejecuciones es la que se muestra en la Figura 12.38.

Confusion Matrix

| | | | | | | | | |
|---|--------------|--------------|----------------|----------------|----------------|--------------|--------------|----------------|
| 1 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 2 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 3 | 3 0.2% | 24 2.0% | 202 16.5% | 117 9.6% | 9 0.7% | 1 0.1% | 0 0.0% | 56.7% 43.3% |
| 4 | 2 0.2% | 17 1.4% | 160 13.1% | 382 31.2% | 147 12.0% | 24 2.0% | 0 0.0% | 52.2% 47.8% |
| 5 | 0 0.0% | 0 0.0% | 2 0.2% | 51 4.2% | 64 5.2% | 19 1.6% | 1 0.1% | 46.7% 53.3% |
| 6 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| 7 | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN% |
| | 0.0% 100% | 0.0% 100% | 55.5% 44.5% | 69.5% 30.5% | 29.1% 70.9% | 0.0% 100% | 0.0% 100% | 52.9% 47.1% |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |

Target Class

Figura 12.38: Matriz de confusión. Conjunto de datos Wine Quality White. Clasificación ordinal.

| numIter | CCR | MAE | NH | CompTime |
|---------|----------|----------|----|------------|
| 1 | 0.528980 | 0.521633 | 8 | 353.182520 |
| 2 | 0.533061 | 0.514286 | 8 | 355.780622 |
| 3 | 0.532245 | 0.517551 | 8 | 358.935483 |
| 4 | 0.528163 | 0.519184 | 32 | 375.545415 |
| 5 | 0.535510 | 0.507755 | 32 | 378.841802 |
| 6 | 0.538776 | 0.506122 | 32 | 383.873993 |
| 7 | 0.533061 | 0.516735 | 32 | 425.594138 |
| 8 | 0.547755 | 0.495510 | 32 | 429.960576 |
| 9 | 0.535510 | 0.510204 | 32 | 433.624884 |
| 10 | 0.564898 | 0.484898 | 32 | 464.288439 |
| 11 | 0.568163 | 0.478367 | 32 | 474.502034 |
| 12 | 0.560000 | 0.489796 | 32 | 479.743118 |
| 13 | 0.550204 | 0.498776 | 32 | 435.353838 |
| 14 | 0.549388 | 0.503673 | 32 | 440.097851 |
| 15 | 0.544490 | 0.500408 | 32 | 450.914484 |
| 16 | 0.539592 | 0.506939 | 32 | 404.978188 |
| 17 | 0.527347 | 0.523265 | 32 | 410.949933 |
| 18 | 0.538776 | 0.510204 | 32 | 423.355420 |
| 19 | 0.548571 | 0.505306 | 32 | 390.164853 |
| 20 | 0.544490 | 0.506122 | 32 | 397.944788 |
| 21 | 0.553469 | 0.497143 | 32 | 402.287354 |
| 22 | 0.520816 | 0.532245 | 16 | 342.791505 |
| 23 | 0.537143 | 0.514286 | 16 | 345.332927 |
| 24 | 0.538776 | 0.515918 | 16 | 348.639862 |
| 25 | 0.537143 | 0.523265 | 32 | 351.273923 |
| 26 | 0.559184 | 0.488980 | 32 | 357.810833 |
| 27 | 0.551837 | 0.500408 | 32 | 366.244917 |
| 28 | 0.524898 | 0.520000 | 32 | 373.380071 |
| 29 | 0.525714 | 0.518367 | 32 | 378.442744 |
| 30 | 0.525714 | 0.524898 | 32 | 381.795932 |

Tabla 12.38: Resultados individuales. Conjunto de datos Wine Quality White. Clasificación ordinal.

12.2. Resultados generales

Estos resultados son los obtenidos a partir de los resultados individuales, es decir, de cada uno de los conjuntos de datos se ha realizado la media y la desviación típica del CCR, el MAE y el NH.

A partir de estos datos se podrá observar como de buenos son los resultados para una posterior comparativa.

12.2.1. Clasificación Nominal

La Tabla 12.39 muestra los resultados generales para los conjuntos de datos aplicando la clasificación nominal para la creación de la red neuronal y el entrenamiento y simulación.

| dataset | CCRMean | CCRSd | MAEMean | MAESd | NHMean | NHSD |
|-------------------|----------|----------|----------|----------|-----------|-----------|
| ERA | 0.249333 | 0.026623 | 1.360267 | 0.150086 | 15.200000 | 11.593101 |
| ESL | 0.694536 | 0.028794 | 0.329235 | 0.031200 | 25.600000 | 8.262364 |
| LEV | 0.598933 | 0.033564 | 0.431067 | 0.037665 | 20.000000 | 10.832051 |
| SWD | 0.561733 | 0.031839 | 0.467867 | 0.037294 | 13.200000 | 8.011103 |
| automobile | 0.600000 | 0.089548 | 0.580128 | 0.144536 | 30.400000 | 5.059644 |
| balance | 0.928662 | 0.026469 | 0.083864 | 0.034446 | 24.000000 | 8.432740 |
| bondrate | 0.560000 | 0.077509 | 0.606667 | 0.124229 | 20.400000 | 10.741405 |
| car | 0.977701 | 0.010900 | 0.025617 | 0.012634 | 30.400000 | 5.059644 |
| contact-lenses | 0.661111 | 0.202869 | 0.500000 | 0.327536 | 24.300000 | 13.013241 |
| depression | 0.653283 | 0.192474 | 0.509091 | 0.316757 | 24.300000 | 13.013241 |
| eucalyptus | 0.599638 | 0.036704 | 0.476993 | 0.055290 | 24.000000 | 8.432740 |
| newthyroid | 0.956790 | 0.032014 | 0.043210 | 0.032014 | 21.600000 | 11.345092 |
| pasture | 0.722222 | 0.119421 | 0.311111 | 0.138101 | 17.600000 | 8.262364 |
| squash-stored | 0.656410 | 0.139567 | 0.366667 | 0.154750 | 28.800000 | 6.746192 |
| squash-unstored | 0.669231 | 0.131151 | 0.330769 | 0.131151 | 20.800000 | 10.119289 |
| tae | 0.414912 | 0.085952 | 0.764912 | 0.145938 | 17.600000 | 11.027239 |
| thyroid | 0.944630 | 0.008216 | 0.101574 | 0.015156 | 28.800000 | 6.746192 |
| winequality-red | 0.578250 | 0.016856 | 0.459833 | 0.020128 | 28.000000 | 8.640988 |
| winequality-white | 0.543728 | 0.010689 | 0.515374 | 0.009745 | 30.400000 | 5.059644 |

Tabla 12.39: Resultados generales. Clasificación nominal.

12.2.2. Clasificación Ordinal

La Tabla 12.40 muestra los resultados generales para los conjuntos de datos aplicando la clasificación ordinal para la creación de la red neuronal y el entrenamiento y simulación.

| dataset | CCRMean | CCRSd | MAEMean | MAESd | NHMean | NHSD |
|-------------------|----------------|--------------|----------------|--------------|---------------|-------------|
| ERA | 0.267600 | 0.025415 | 1.240133 | 0.081750 | 16.500000 | 11.843892 |
| ESL | 0.714481 | 0.041744 | 0.302732 | 0.044685 | 21.700000 | 11.757267 |
| LEV | 0.607067 | 0.040936 | 0.419333 | 0.043490 | 12.400000 | 12.231108 |
| SWD | 0.567733 | 0.027270 | 0.451467 | 0.033855 | 12.800000 | 7.728734 |
| automobile | 0.641667 | 0.085480 | 0.443590 | 0.117023 | 24.800000 | 9.577752 |
| balance | 0.963907 | 0.018041 | 0.039066 | 0.019843 | 21.600000 | 9.276014 |
| bondrate | 0.568889 | 0.083475 | 0.584444 | 0.093765 | 12.400000 | 10.905656 |
| car | 0.968827 | 0.010955 | 0.031327 | 0.011039 | 30.400000 | 5.059644 |
| contact-lenses | 0.650000 | 0.153815 | 0.427778 | 0.184055 | 14.400000 | 10.362325 |
| depression | 0.647475 | 0.149711 | 0.430808 | 0.179999 | 13.800000 | 10.932114 |
| eucalyptus | 0.650906 | 0.035615 | 0.385145 | 0.032663 | 27.200000 | 7.728734 |
| newthyroid | 0.964198 | 0.024284 | 0.035802 | 0.024284 | 5.300000 | 9.499123 |
| pasture | 0.648148 | 0.127468 | 0.351852 | 0.127468 | 9.700000 | 11.981931 |
| squash-stored | 0.661538 | 0.115340 | 0.353846 | 0.125507 | 12.000000 | 8.432740 |
| squash-unstored | 0.653846 | 0.156150 | 0.348718 | 0.160001 | 13.300000 | 11.353414 |
| tae | 0.441228 | 0.072047 | 0.648246 | 0.068233 | 17.200000 | 11.003030 |
| thyroid | 0.929426 | 0.006682 | 0.118352 | 0.013353 | 25.600000 | 8.262364 |
| winequality-red | 0.580833 | 0.020754 | 0.449917 | 0.022993 | 18.000000 | 10.540926 |
| winequality-white | 0.540789 | 0.012448 | 0.508408 | 0.012904 | 28.000000 | 8.640988 |

Tabla 12.40: Resultados generales. Clasificación ordinal.

12.3. Comparativa entre los resultados

En esta sección se hará un estudio comparativo entre las dos metodologías empleadas, nominal y ordinal, y de las cuales se podrán obtener las conclusiones en un capítulo posterior.

Se hará una comparación entre los resultados generales de cada uno de las clasificaciones, observando y resaltando en cada caso cual ha sido mejor o peor para los conjuntos de datos evaluados. Para la comparativa

entre los resultados obtenidos nos fijaremos en los valores medios del CCR y del MAE .

Como se puede observar en la Tabla 12.41, los resultados obtenidos por el algoritmo ordinal ORNNet son mejores que para el algoritmo nominal, puesto que en 12 de los conjuntos de datos el CCR es mayor que los resultados del algoritmo nominal y en 15 es menor el valor del MAE para el ordinal. Además, en 11 la desviación típica es menor para el CCR utilizando la clasificación ordinal frente a la nominal y en 14 también es menor la desviación típica para el MAE utilizando la clasificación ordinal.

| dataset | Nominal | | | Ordinal | | |
|-------------------|----------|----------|----------|----------|----------|----------|
| | CCRMean | CCRSd | MAEMean | MAESD | CCRMean | MAESD |
| ERA | 0.249333 | 0.026623 | 1.360267 | 0.150086 | 0.267600 | 0.025415 |
| ESL | 0.694536 | 0.028794 | 0.329235 | 0.031200 | 0.714481 | 0.041744 |
| LEV | 0.598933 | 0.033564 | 0.431067 | 0.037665 | 0.607067 | 0.040936 |
| SWD | 0.561733 | 0.031839 | 0.467867 | 0.037294 | 0.567733 | 0.027270 |
| automobile | 0.600000 | 0.089548 | 0.580128 | 0.144536 | 0.641667 | 0.085480 |
| balance | 0.928662 | 0.026469 | 0.083864 | 0.034446 | 0.963907 | 0.018041 |
| bondrate | 0.560000 | 0.077509 | 0.606667 | 0.124229 | 0.568889 | 0.083475 |
| car | 0.977701 | 0.010900 | 0.025617 | 0.012634 | 0.968827 | 0.010955 |
| contact-lenses | 0.661111 | 0.202869 | 0.500000 | 0.327536 | 0.650000 | 0.153815 |
| depression | 0.653283 | 0.192474 | 0.509091 | 0.316757 | 0.647475 | 0.149711 |
| eucalyptus | 0.599638 | 0.036704 | 0.476993 | 0.055290 | 0.650906 | 0.035615 |
| newthyroid | 0.956790 | 0.032014 | 0.043210 | 0.032014 | 0.964198 | 0.024284 |
| pasture | 0.722222 | 0.119421 | 0.311111 | 0.138101 | 0.648148 | 0.127468 |
| squash-stored | 0.656410 | 0.139567 | 0.366667 | 0.154750 | 0.661538 | 0.115340 |
| squash-unstored | 0.669231 | 0.131151 | 0.330769 | 0.131151 | 0.653846 | 0.156150 |
| tae | 0.414912 | 0.085952 | 0.764912 | 0.145938 | 0.441228 | 0.072047 |
| thyroid | 0.944630 | 0.008216 | 0.101574 | 0.015156 | 0.929426 | 0.006682 |
| winequality-red | 0.578250 | 0.016856 | 0.459833 | 0.020128 | 0.580833 | 0.020754 |
| winequality-white | 0.543728 | 0.010689 | 0.515374 | 0.009745 | 0.540789 | 0.012448 |

Tabla 12.41: Comparativa entre resultados generales.



13 Conclusiones y Futuras Mejoras

Una vez concluido el desarrollo y la experimentación del proyecto, se realizará una exposición de las conclusiones que se han extraído. Estas conclusiones irán en relación con los objetivos planteados al principio del desarrollo especificados en el Capítulo de Objetivos y con los resultados obtenidos durante la fase de experimentación.

13.1. Conclusiones

En esta sección se va a considerar cuales de los objetivos se han alcanzado en la realización del Proyecto. Una vez finalizado el desarrollo de la implementación del algoritmo de regresión ordinal para redes neuronales y la aplicación gráfica, se puede concretar que se han cumplido los requisitos especificados al comienzo del Proyecto.

Los objetivos alcanzados han sido los siguientes:

- Se ha desarrollado la implementación del algoritmo propuesto de forma teórica para regresión ordinal basado en redes neuronales artificiales.

- Se ha realizado una comparativa de los resultados obtenidos basado en efectividad de la clasificación obtenida por el algoritmo nominal y el nuevo algoritmo basado en la ordinalidad de las clases.
- Se ha desarrollado una herramienta software para el tratado y la obtención de resultados a partir de un conjunto de datos, aplicando el algoritmo implementado basado en redes neuronales y haciendo uso del toolbox de Matlab, *nnet*. Dicha herramienta ha sido implementada optimizando al máximo tanto la precisión de los modelos generados como el tiempo de computación necesario para obtener los resultados y que es algo muy importante en algoritmos de este estilo.
- Se ha modularizado todas los módulos y funciones para que tenga una compatibilidad completa con el toolbox *nnet* de Matlab.
- Se ha realizado un diseño experimental para cada uno de los problemas considerados, ajustando los parámetros de los experimentos y extrayendo las conclusiones pertinentes. Los modelos obtenidos como solución a dichos problemas han obtenido unos resultados muy buenos en clasificación, mejorando en un porcentaje variable para cada caso específico los resultados que hasta ahora se habían obtenido con modelos de clasificación nominal.

Por todos estos motivos, considerando que el proyecto que se ha desarrollado ha conseguido abarcar todas las metas que se propusieron al inicio del mismo, es un orgullo y una satisfacción haber aportado al grupo de investigación AYRNA, en sus estudios sobre aprendizaje e inteligencia artificial con ayuda de las redes neuronales artificiales, la implementación y comprobación del algoritmo desarrollado de forma teórica y la aportación de una interfaz gráfica completa y compatible con la librería de Matlab.

13.2. Futuras Mejoras

En esta sección se expondrán varias ideas con posibles mejoras a alcanzar en futuras ampliaciones del trabajo desarrollado.

Como se ha expuesto a lo largo del desarrollo del presente proyecto, el objetivo principal era el desarrollo de la implementación del algoritmo de regresión ordinal y la comprobación de la efectividad del mismo en comparación con el método nominal.

Dicho esto, se podría ver como una posible mejora del proyecto el estudio continuo de nuevos algoritmos basados en regresión ordinal y a partir de estos la nueva implementación de otro algoritmo que mejorase la efectividad de los resultados.

Otra posible mejora sería el aumento de la funcionalidad de la herramienta software, por ejemplo, añadiendo la funcionalidad para la realización del método nominal o algún otro método basado en redes neuronales.

Por último, otra posible mejora que podría realizarse es la adaptación, en alguno de los sentidos, al toolbox nnet de Matlab del resto de algoritmos o desarrollos que el grupo de investigación AYRNA tiene de proyectos o estudios anteriores basados en redes neuronales artificiales, completando así dicho paquete de software computacionalmente muy potente.



Bibliografía

- [1] Bishop C.M., “Pattern recognition and machine learning”. Singapur, Springer, 676 p, ISBN-10: 0-387-31073-8.
- [2] W. Chu and S.S. Keerthi, “New Approaches to Support Vector Ordinal Regression”. Proc. 22nd Int. Conf. Machine Learning (ICML ’05), pp. 145-152, 2005.
- [3] L. Lin and H.-T. Lin, “Ordinal Regression by Extended Binary Classification”. Advances in Neural Infor. Processing Systems, vol. 19, pp. 865-872, MIT Press, 2007.
- [4] McCullagh, “P. Regression models for ordinal data”. Journal of the Royal Statistical Society, Series B (Methodological), 42, 109–142, 1980.
- [5] Shashua and A. Levin, “Ranking with Large Margin Principle: Two Approaches”, Advances in Neural Inform. Processing Systems, vol. 15, 961-968, MIT Press, 2003.
- [6] Riedmiller, Martin. “RPROP - Descripción and Implmentation Details”. University of Karlsruhe. Technical Report, 1994.

- [7] Igel, Christian; Hüsken, Michael. “Empirical evaluation of the improved RPROP learning algorithms”. Institut für Neuroinformatik, Ruhr-Universität Bochum. Germany, 2001.
- [8] “MATLAB version 2010a for Ubuntu”. Natick, Massachusetts. The MathWorks Inc., 2010.
- [9] “Documentación oficial de MATLAB”. Natick, Massachusetts. The MathWorks Inc., 2010. Enlace: <http://www.mathworks.com/help/techdoc/>
- [10] “Documentación oficial de MATLAB para el Toolbox NNET”. Natick, Massachusetts. The MathWorks Inc., 2010. Enlace: <http://www.mathworks.com/help/toolbox/nnet/>
- [11] “Documentación oficial de MATLAB para creación de GUIs (Graphical User Interfaces)”. Natick, Massachusetts. The MathWorks Inc., 2010. Enlace: http://www.mathworks.com/help/techdoc/creating_guis/bqz79mu.html
- [12] G. Booch, J. Rumbaugh, I. Jacobson, “El lenguaje Unificado de Modelado”. Ed. Addison Wesley Iberoamericana. Madrid, 1999.



A Manual de usuario

Este apéndice constituye el manual de usuario del producto software desarrollado por el presente proyecto fin de carrera e incluye la información necesaria para su ejecución.

A.1. Instalación y desinstalación

En esta sección se realizará una explicación de cómo se podrá instalar y desinstalar la nueva funcionalidad del toolbox *nnet* añadida por las nuevas funciones y la aplicación gráfica.

A.1.1. Instalación

Para la instalación se requerirá un descompresor, ya que todo el código fuente del toolbox se encontrará en archivo comprimido .zip como se muestra en la Figura A.1.

Al descomprimir el archivo comprimido se quedará la carpeta con el código fuente, como se muestra en la Figura A.2, lista para poder añadirla al *path* de Matlab.

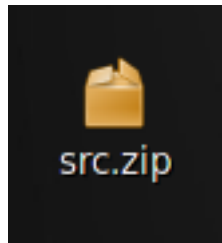


Figura A.1: Instalación. Archivo del código fuente comprimido.

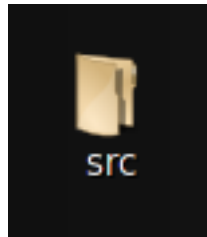


Figura A.2: Instalación. Carpeta del código fuente.

Para añadir el código al *path* de Matlab habrá que seguir los siguientes pasos:

Primero seleccione del menú *File* y pulse sobre la opción *Set Path* tal y como se muestra en la Figura A.3.

Aparecerá una pantalla como la que se muestra en la Figura A.4. En esta pantalla deberá seleccionar la opción *Add with Subfolders*.

En este momento aparecerá un cuadro de diálogo como el que se muestra en la Figura A.5 en el que deberá seleccionar la carpeta que se ha descomprimido y que contiene todo el código fuente.

Al pulsar *OK* del cuadro de diálogo se añadirá todo el contenido al *Path* de Matlab. En este momento habrá que pulsar sobre el botón *Save* y luego en *Close*, con lo que ya se podrá llamar a cualquiera de las funciones del nuevo *toolbox* como a cualquier otra función propia de Matlab.

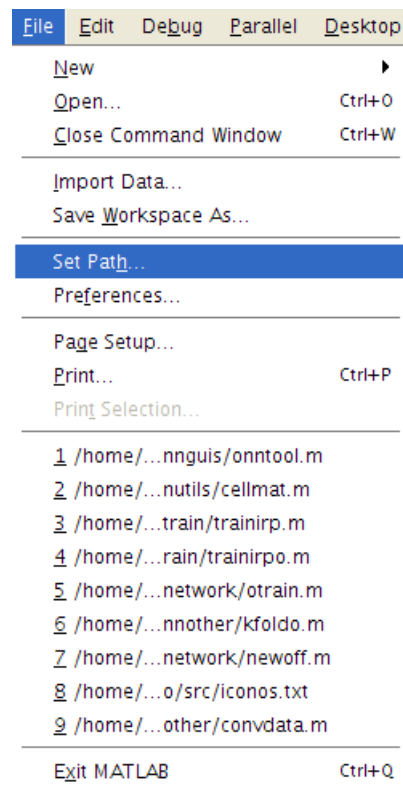


Figura A.3: Instalación. Menú File.

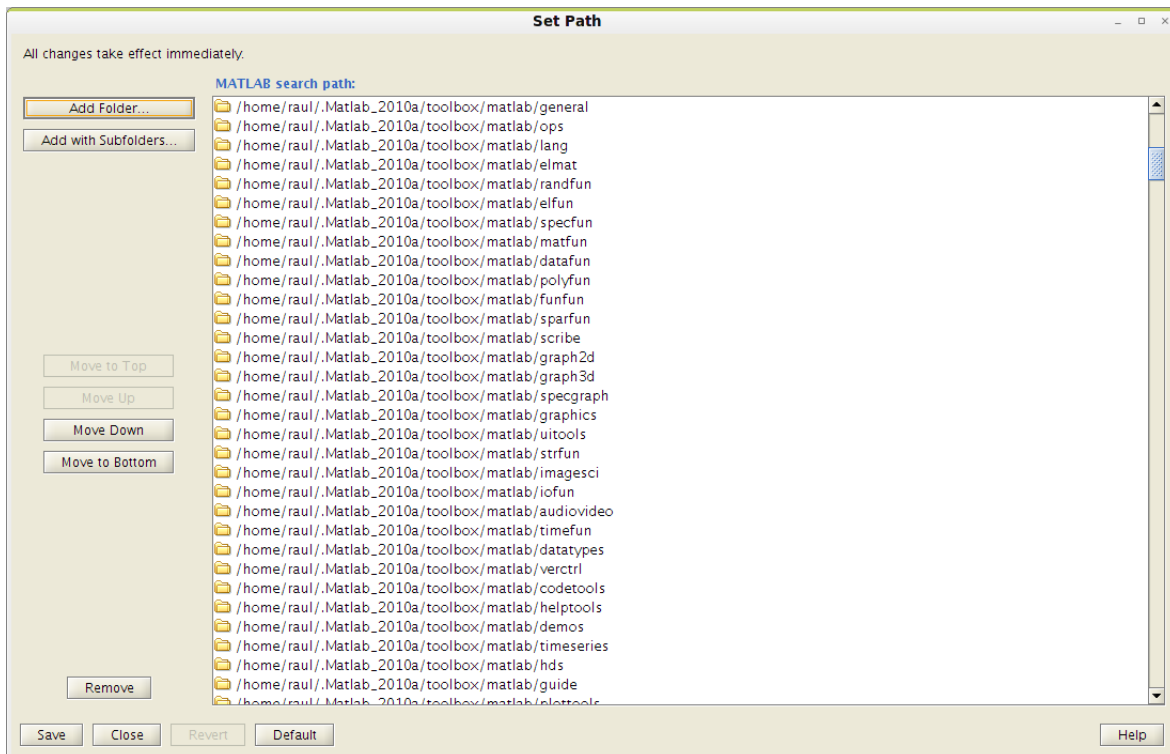


Figura A.4: Instalación. Añadir al Path.

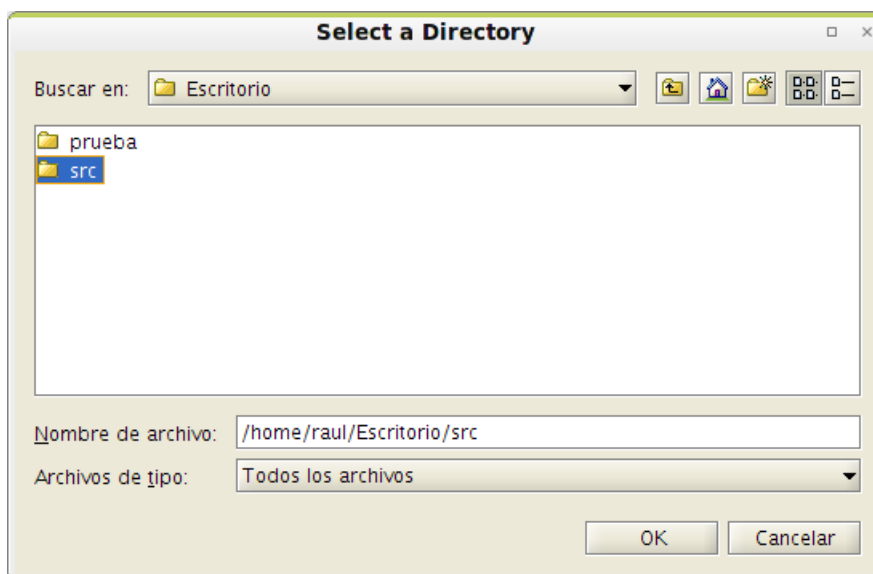


Figura A.5: Instalación. Cuadro de diálogo.

A.1.2. Desinstalación

Para la desinstalación de la nueva funcionalidad del *toolbox* para Matlab sólo habrá que realizar un paso. Como se puede observar en la Figura A.6, el *path* contendrá el árbol de contenidos del *toolbox*, con lo que habrá que seleccionar todos los elementos y seguidamente pulsar sobre el botón *Remove*. Una vez eliminado deberá pulsar sobre el botón *Save* y seguidamente sobre *Close* y ya estará todo desinstalado.

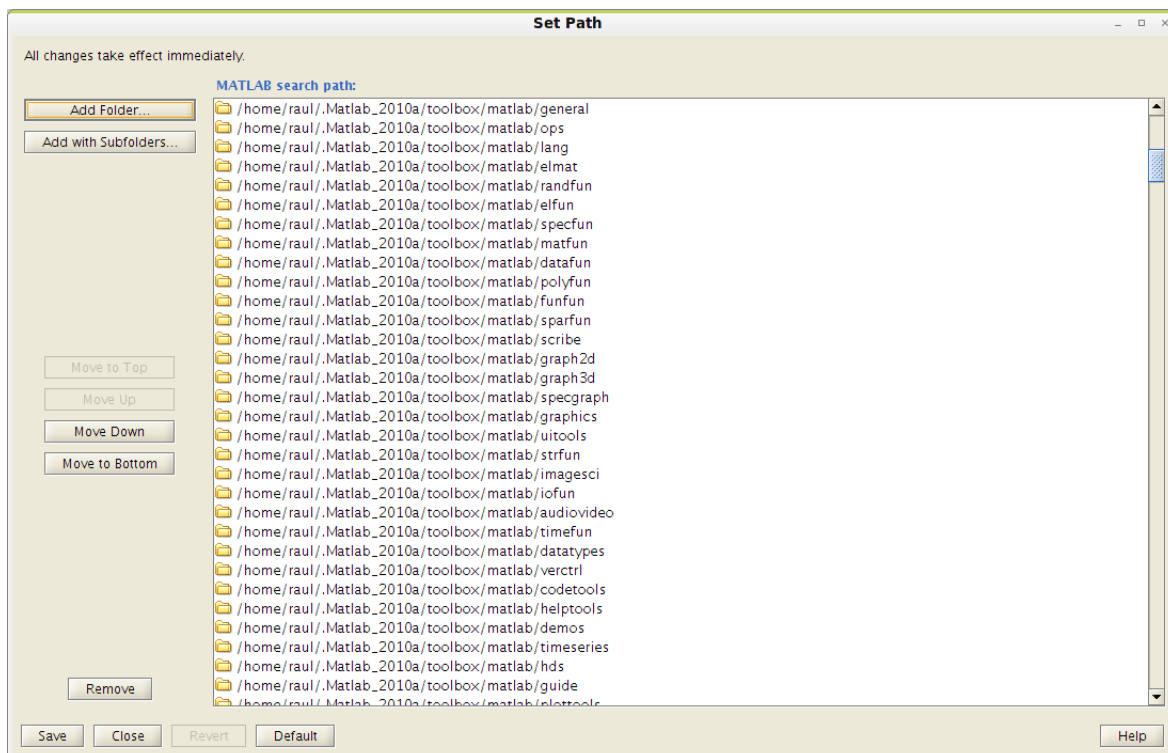


Figura A.6: Desinstalación. Eliminación de contenido del Path.

A.2. Uso de la aplicación

En esta sección se expondrá un ejemplo de la aplicación para que cualquiera pueda usarla. Se explicará detalladamente cada pantalla que proporciona la aplicación mostrando además una imagen de la misma.

La primera pantalla que se verá cuando se inicia la aplicación será la pantalla inicial de bienvenida, que se muestra en la Figura A.7. En esta pantalla se puede observar los siguientes elementos: una barra en la parte superior con el nombre de la aplicación “nntool”, seguidamente se muestra una cabecera en la que se ve el saludo de bienvenida a la aplicación junto a los escudos de la Universidad de Córdoba y de Ingeniería Informática. Después se muestran dos paneles, uno con información general sobre el tema que se trata y otro con información específica sobre redes neuronales ordinales. Dos de los botones que contiene esta pantalla y que contendrán el resto a partir de ahora a excepción de la última pantalla son los botones para pasar a la siguiente pantalla (Next) y para cancelar la aplicación y salirse (Cancel).

Al pulsar el botón siguiente (Next) se pasará a la siguiente pantalla, la pantalla de obtención de datos, la cual muestra dos paneles, uno para cargar los datos del conjunto de entrenamiento y otro para cargar los datos del conjunto de test. Estos paneles constan de un menú popup en el que se mostrarán de cada conjunto los datos de entrada (Inputs) y los datos objetivos (Targets).

Si se intenta pasar a la siguiente pantalla pulsando directamente el botón siguiente (Next) sin haber cargado los conjuntos de datos, se mostrará un mensaje de advertencia como el que se muestra en la Figura A.9.

En caso de que no haya datos cargados previamente, habrá que pulsar alguno de los botones para abrir un cuadro de diálogo y de este modo poder cargar un conjunto de datos desde un par de ficheros (train y test). Las Figuras de la A.10 a la A.14 muestran como se puede cargar un par de ficheros de un conjunto de datos paso a paso.

Una vez seleccionados los datos se mostrará una breve información sobre

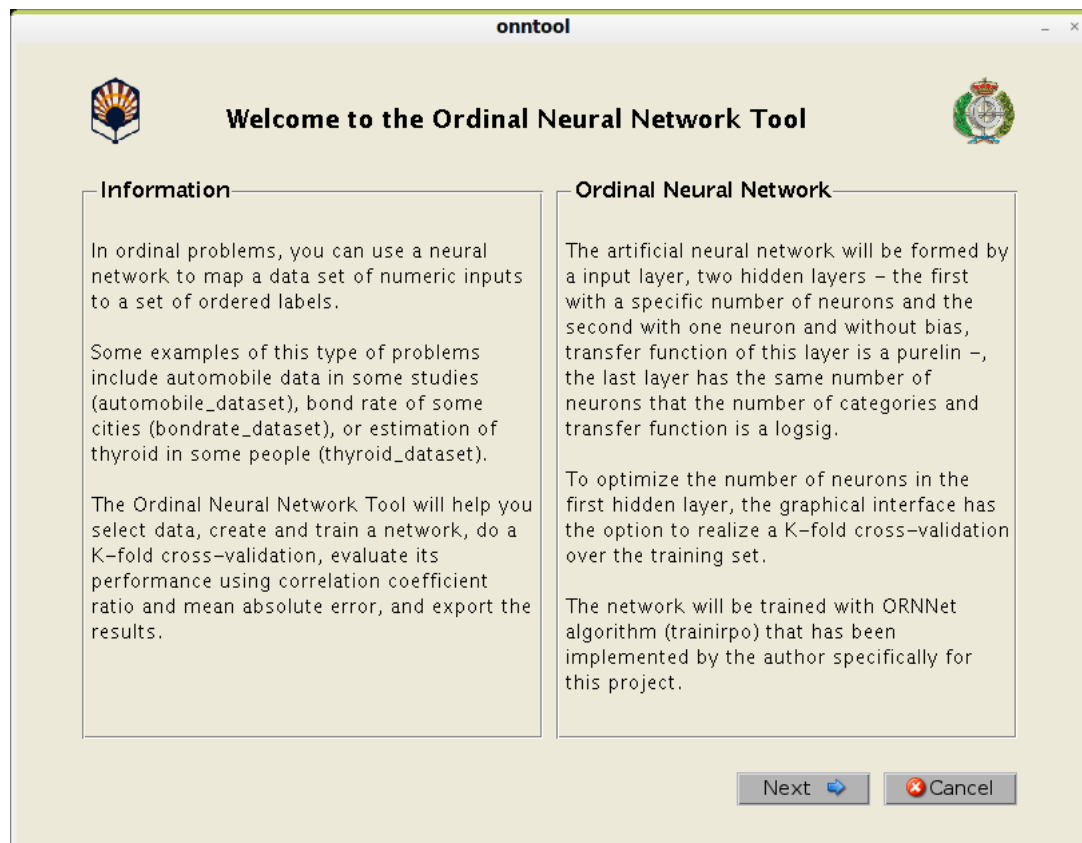


Figura A.7: Interfaz gráfica. Pantalla de bienvenida.

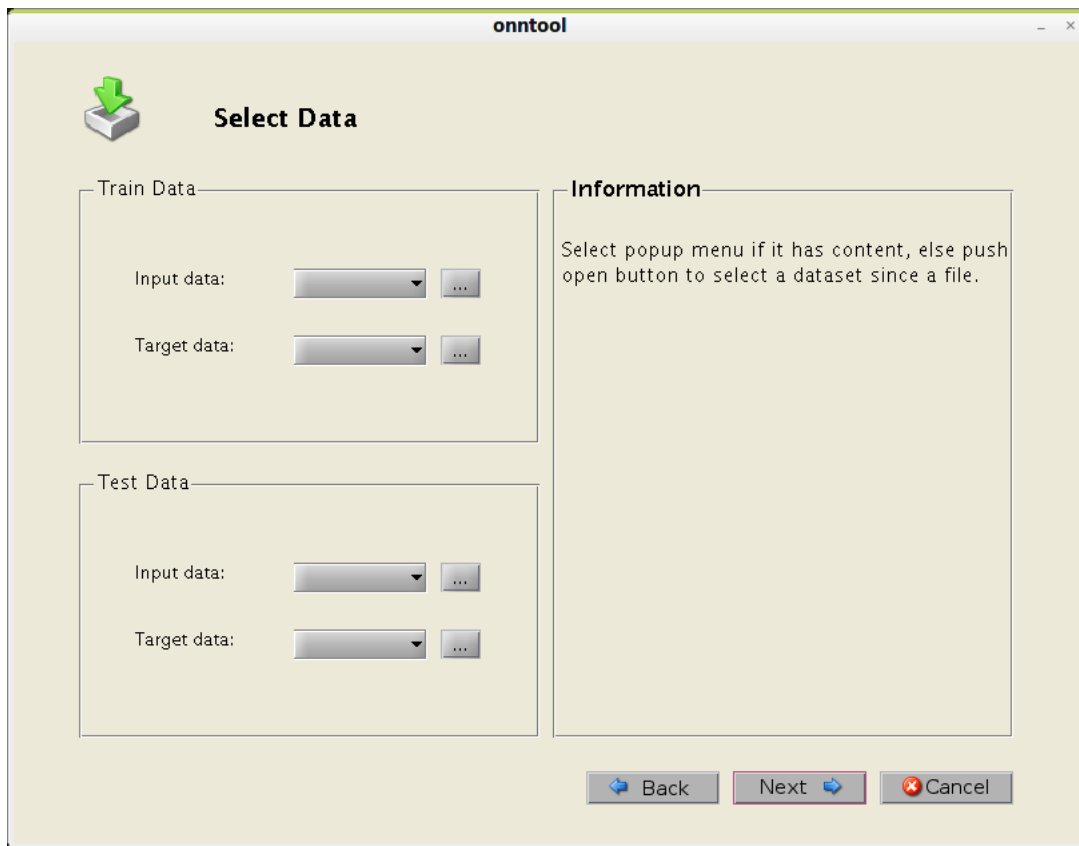


Figura A.8: Interfaz gráfica. Pantalla de carga de datos.

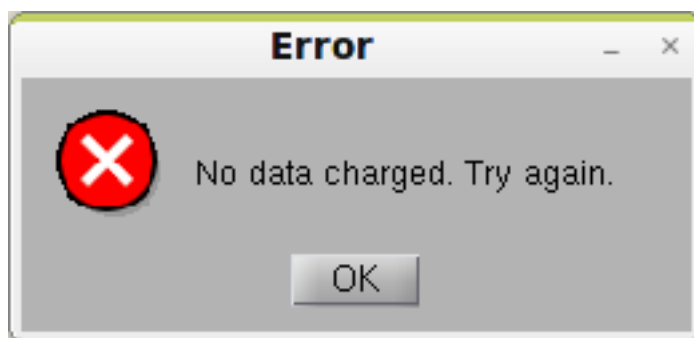


Figura A.9: Interfaz gráfica. Aviso de falta de carga de datos.

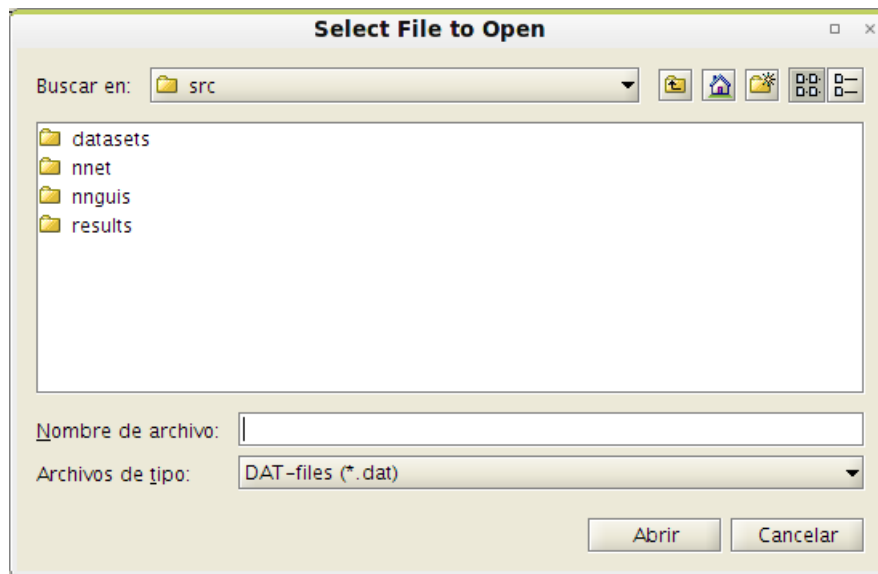


Figura A.10: Interfaz gráfica. Carga de datos.

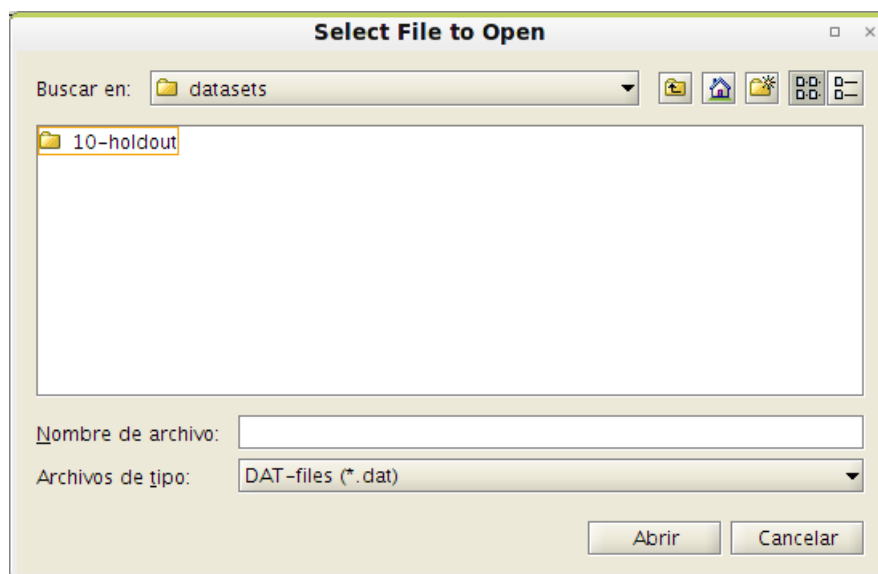


Figura A.11: Interfaz gráfica. Selección de la carpeta datasets.

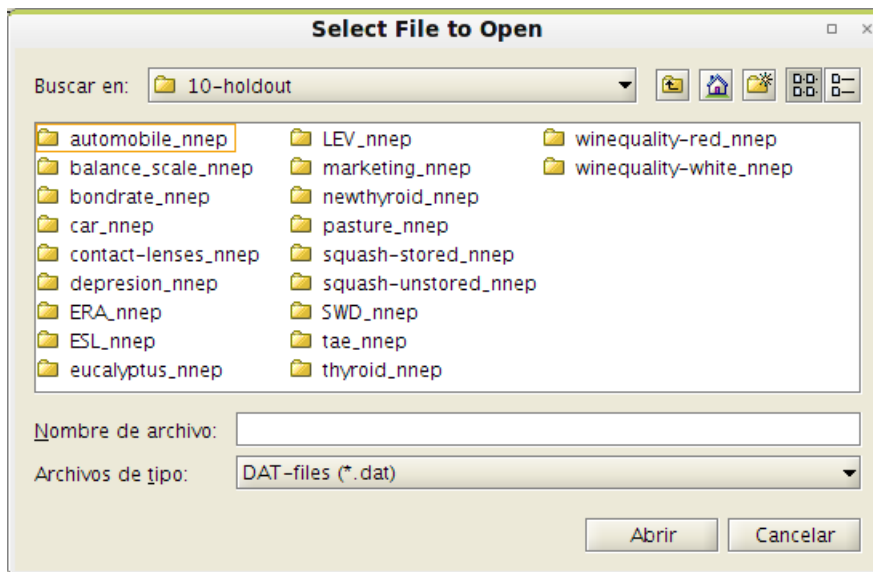


Figura A.12: Interfaz gráfica. Selección de la carpeta 10-holdout.

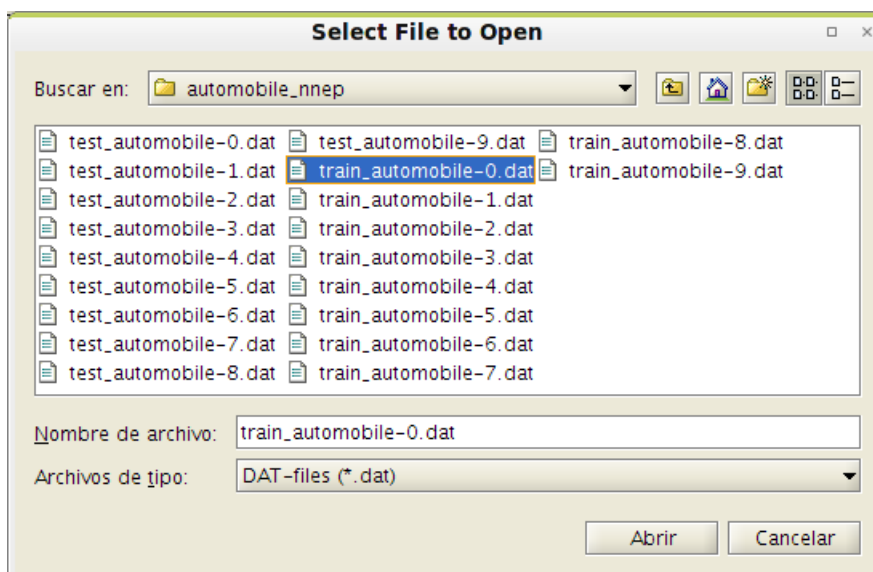


Figura A.13: Interfaz gráfica. Selección del fichero de entrenamiento.

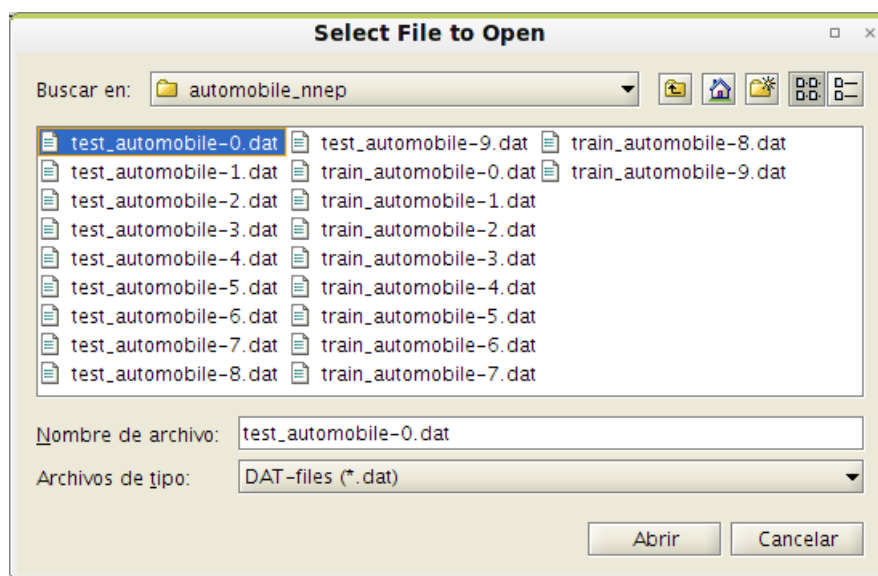


Figura A.14: Interfaz gráfica. Selección del fichero de test.

las dimensiones de los conjuntos de datos en el panel de información de la derecha. La Figura A.15 muestra dicha pantalla con la información del ejemplo cargado.

Si al intentar cargar los datos se produce una equivocación en el fichero de datos o se intenta cargar un fichero con una extensión incorrecta, se mostrará un aviso para que se carguen de nuevo los datos correctos y no se hará nada. Si todo el proceso se ha seguido correctamente se podrá pulsar el botón de siguiente (Next) o de atrás (Back) para navegar entre pantallas.

Una vez pulsado el botón para pasar a la siguiente pantalla (Next), se pasará a la pantalla de configuración de la red neuronal ordinal. Esta pantalla está constituida por dos paneles, uno para la configuración de la red propiamente en la parte izquierda y otro que muestra la información de las acciones disponibles en la parte derecha. La Figura A.16 muestra dicha pantalla.

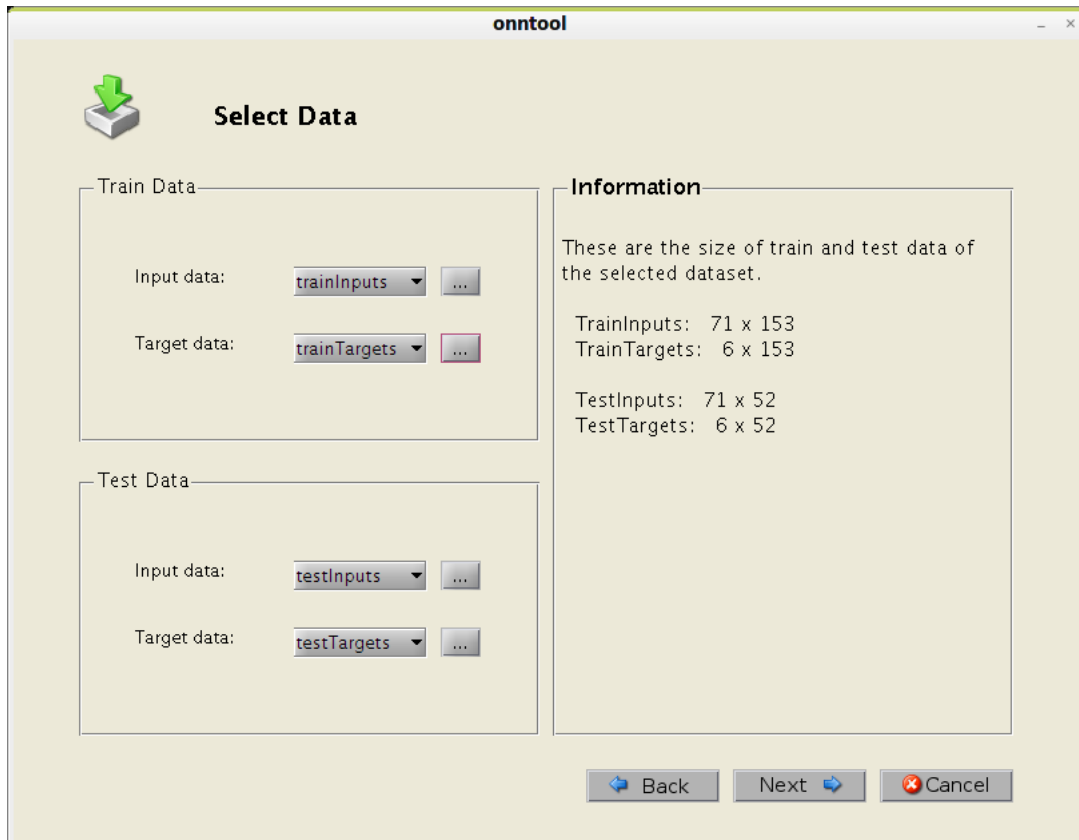


Figura A.15: Interfaz gráfica. Datos cargados.

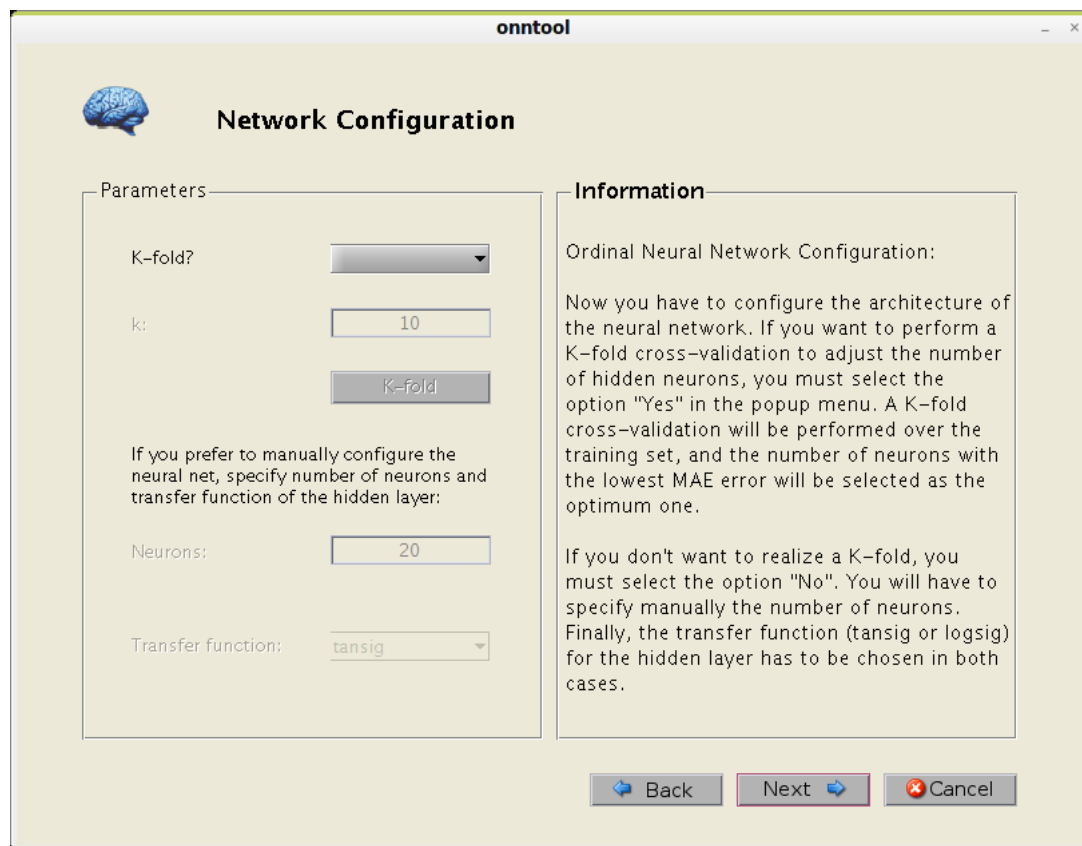


Figura A.16: Interfaz gráfica. Pantalla de configuración de la red neuronal.

En esta pantalla existen dos posibilidades para la configuración de la red neuronal ordinal, la posibilidad de realizar un K-fold cross-validation de los conjuntos de entrenamiento y test para obtener el valor del número de neuronas óptimo de la capa oculta o especificarlo manualmente (20 por defecto). También se tendrá que especificar en ambos casos la función de transferencia a usar, que podrá ser *logsig* o *tansig*. Las Figuras A.17 y A.18 muestran dichas pantallas.

Si se elige la opción de realizar un K-fold, se podrá especificar el valor de k (10 por defecto). Una vez seleccionado el valor de k habrá que pulsar el

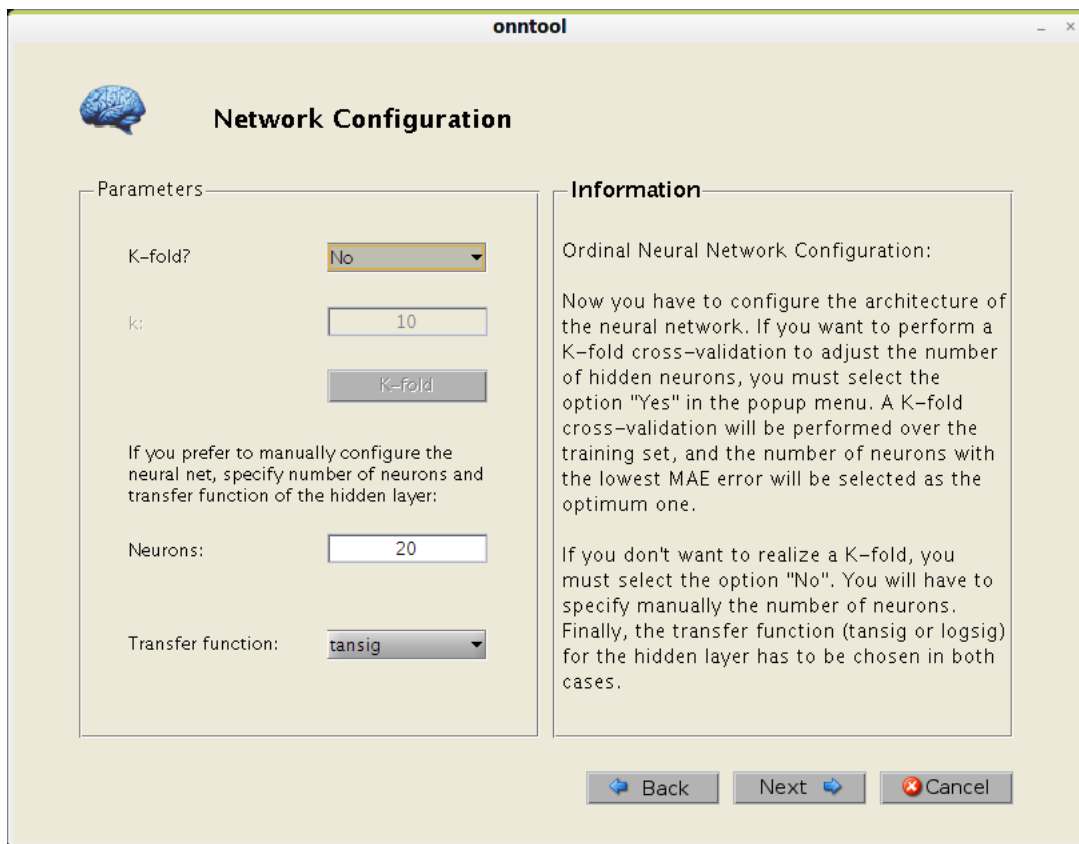


Figura A.17: Interfaz gráfica. Opción manual de configuración de la red neuronal.

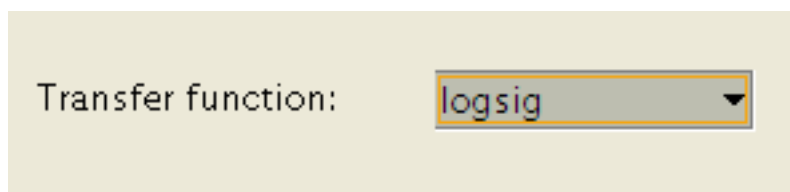


Figura A.18: Interfaz gráfica. Opción de función de transferencia.

botón de K-fold para realizar el proceso. Si se intenta pasar directamente a la siguiente pantalla sin pulsar dicho botón, se mostrará un aviso para que se pulse el botón y se pueda realizar el K-fold. La Figura A.19 muestra dicha pantalla y la Figura A.20 muestra el aviso.

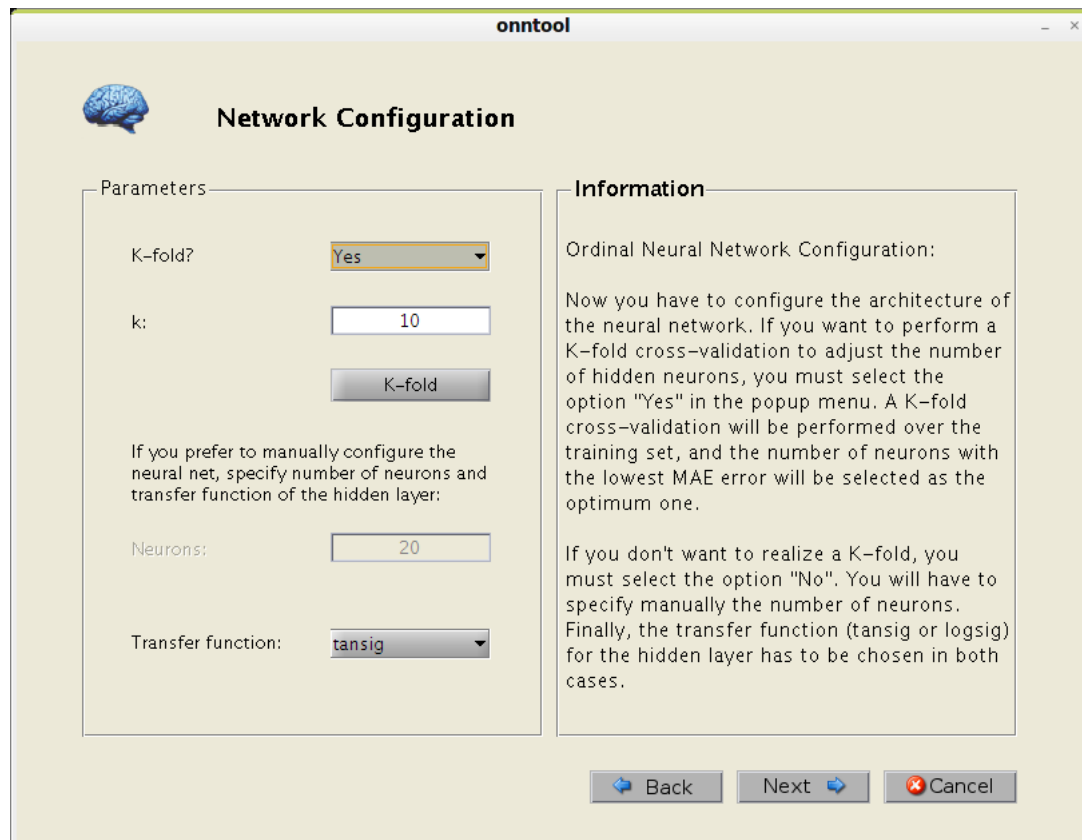


Figura A.19: Interfaz gráfica. Opción para realizar un K-fold.

Mientras se realiza el K-fold cross-validation se presentará un mensaje de espera para informar que el proceso está activo. Una vez terminado el proceso, se cerrará el mensaje y se podrá pasar a la siguiente pantalla.

Al pasar a la siguiente pantalla, que es la pantalla de entrenamiento, se mostrarán dos nuevos paneles. El primer panel mostrará el botón para

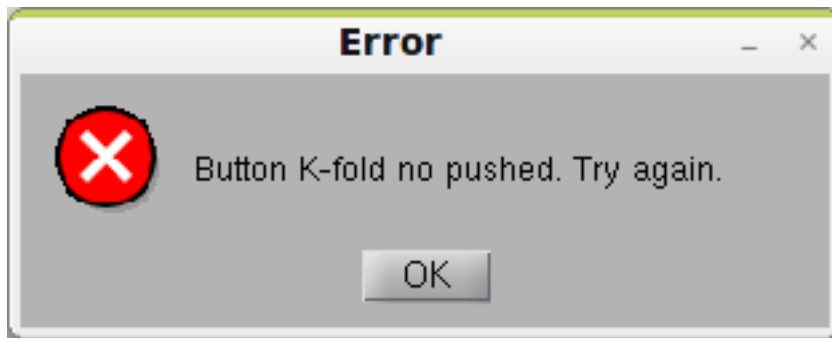


Figura A.20: Interfaz gráfica. Aviso de la opción K-fold.

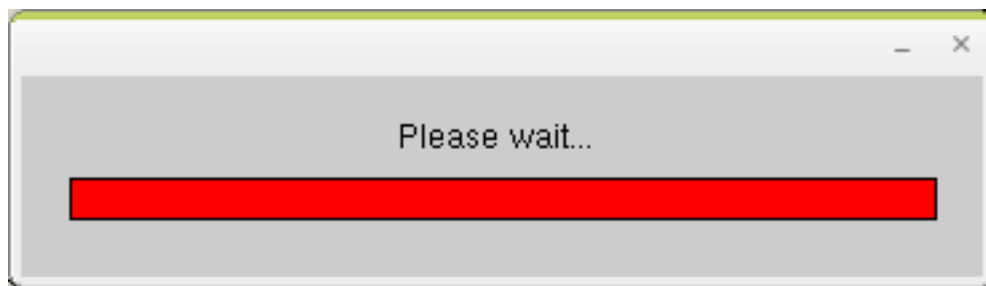


Figura A.21: Interfaz gráfica. Barra de espera de la opción K-fold.

realizar el entrenamiento de la red (Train). La Figura A.22 muestra dicha pantalla.

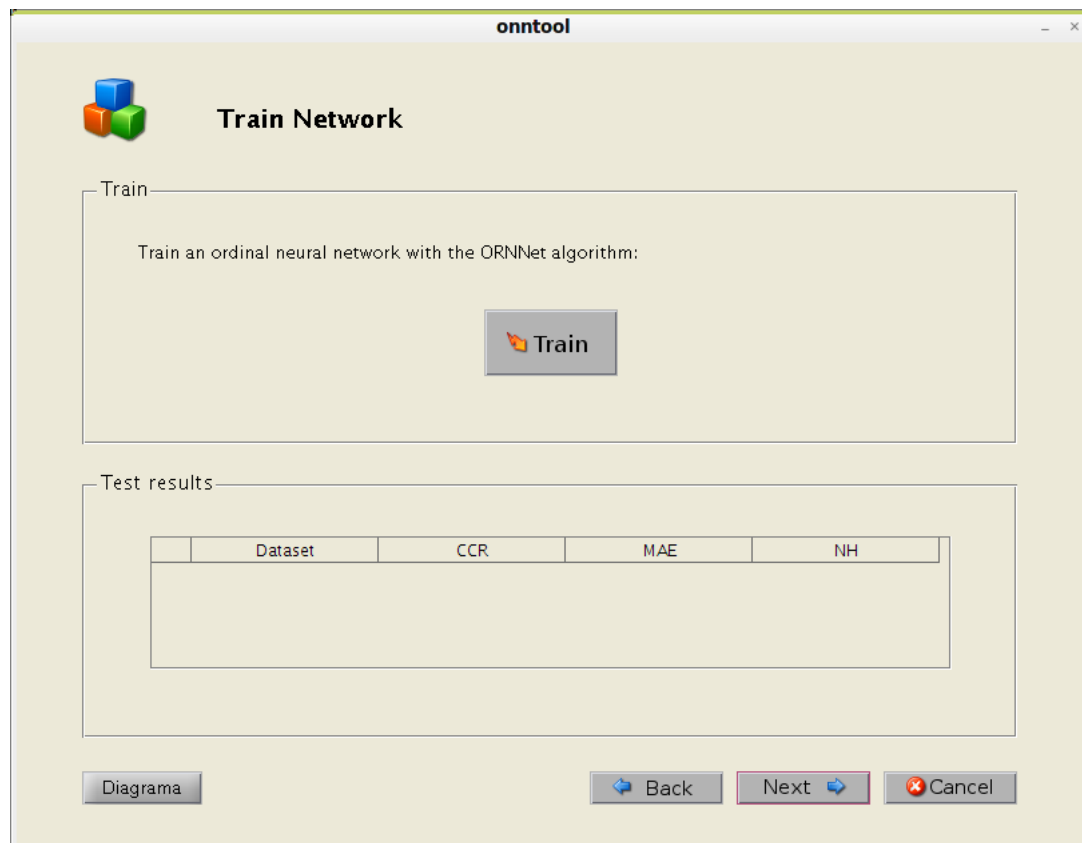


Figura A.22: Interfaz gráfica. Pantalla de entrenamiento.

El segundo panel, en cambio, mostrará los resultados del proceso de entrenamiento y simulación. Adicionalmente, en esta pantalla se añade un nuevo botón para poder ver de manera gráfica la forma de la red neuronal ordinal (Diagrama) configurada. El resultado de este ejemplo se muestra en la Figura A.23.

Si se pulsa el botón de entrenamiento (Train), se realizará el entrenamiento y la simulación de la red neuronal ordinal con los datos seleccionados

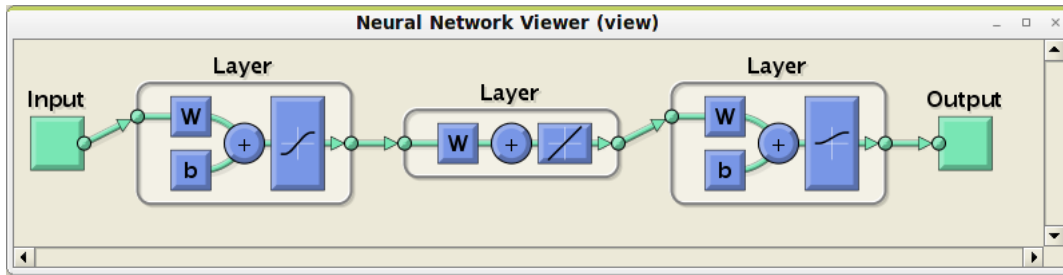


Figura A.23: Interfaz gráfica. Pantalla con red neuronal de forma gráfica.

y la configuración definida. Además, se mostrarán varias pantallas adicionales que se muestran en las Figuras A.24 y A.25. Éstas muestran la matriz de confusión proporcionada para las salidas obtenidas y la pantalla que proporciona el toolbox *nnet* cuando se entrena una red neuronal la cual muestra de forma gráfica, pero en pequeña escala, la red neuronal ordinal, la evolución de algunos parámetros para el entrenamiento y la posibilidad de generar alguna gráfica que mostrará estadísticas de los resultados de entrenamiento.

Una vez entrenada la red neuronal ordinal, se rellenará la tabla de resultados como se muestra en la Figura A.26, en la que se podrá observar los siguientes datos: el nombre del conjunto de datos evaluado, el valor del CCR obtenido, el valor del MAE y el número de neuronas en capa oculta, que será el especificado, en el caso de que la configuración haya sido manual, o un valor 2^n con $n = 0, \dots, 5$. Adicionalmente, se podrá observar que el botón de entrenamiento habrá cambiado (Retrain) para poder reentrenar la red neuronal.

Por último, al pulsar de nuevo el botón de siguiente (Next) se pasará a la siguiente pantalla, la pantalla de exportado de datos en el *workspace* de Matlab. La Figura A.27 muestra dicha pantalla, la cual contiene un panel en el que se podrá especificar el nombre de las variables a exportar. Los datos que se podrán exportar son: la red neuronal ordinal, las salidas

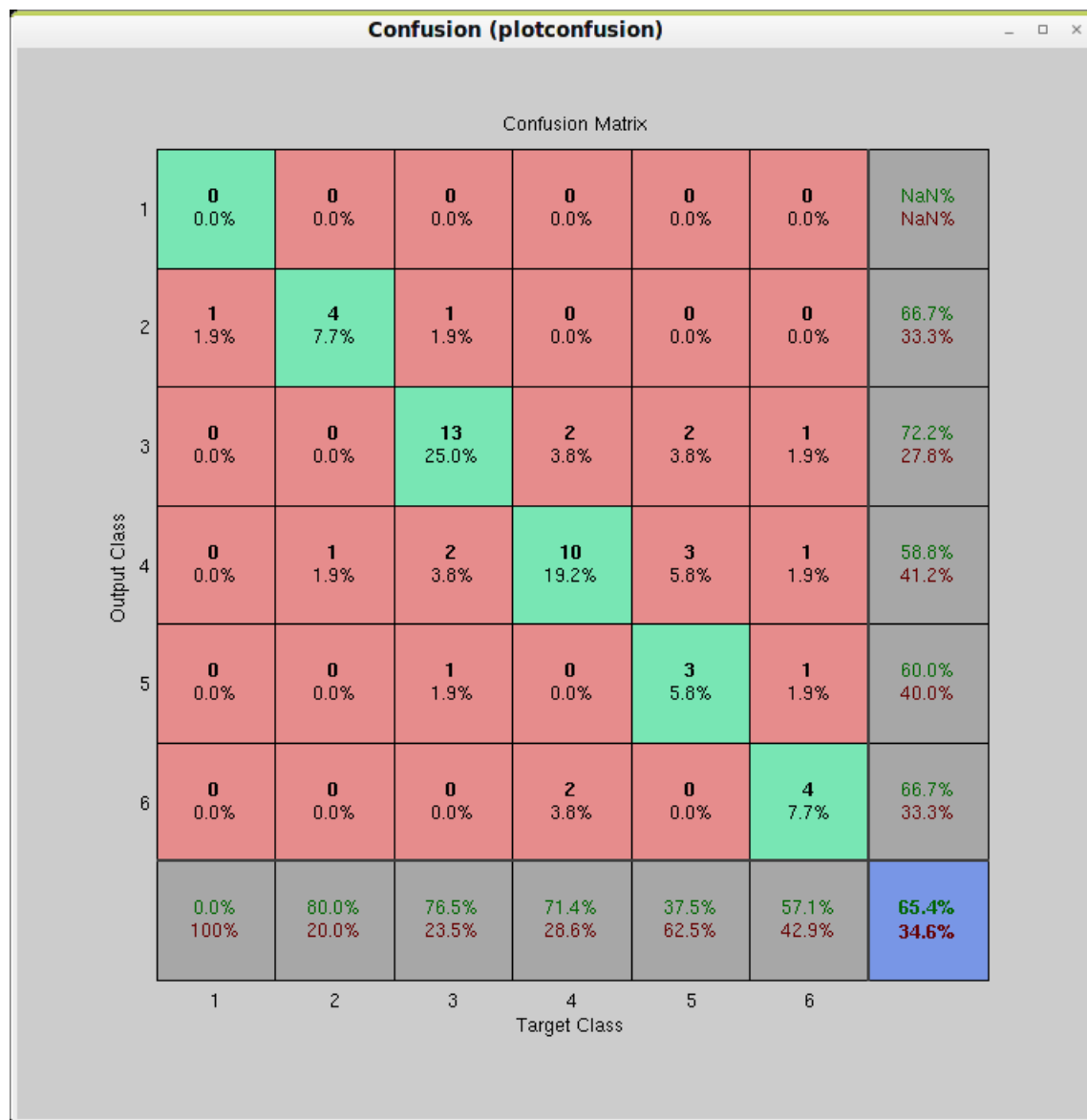


Figura A.24: Interfaz gráfica. Matriz de confusión.

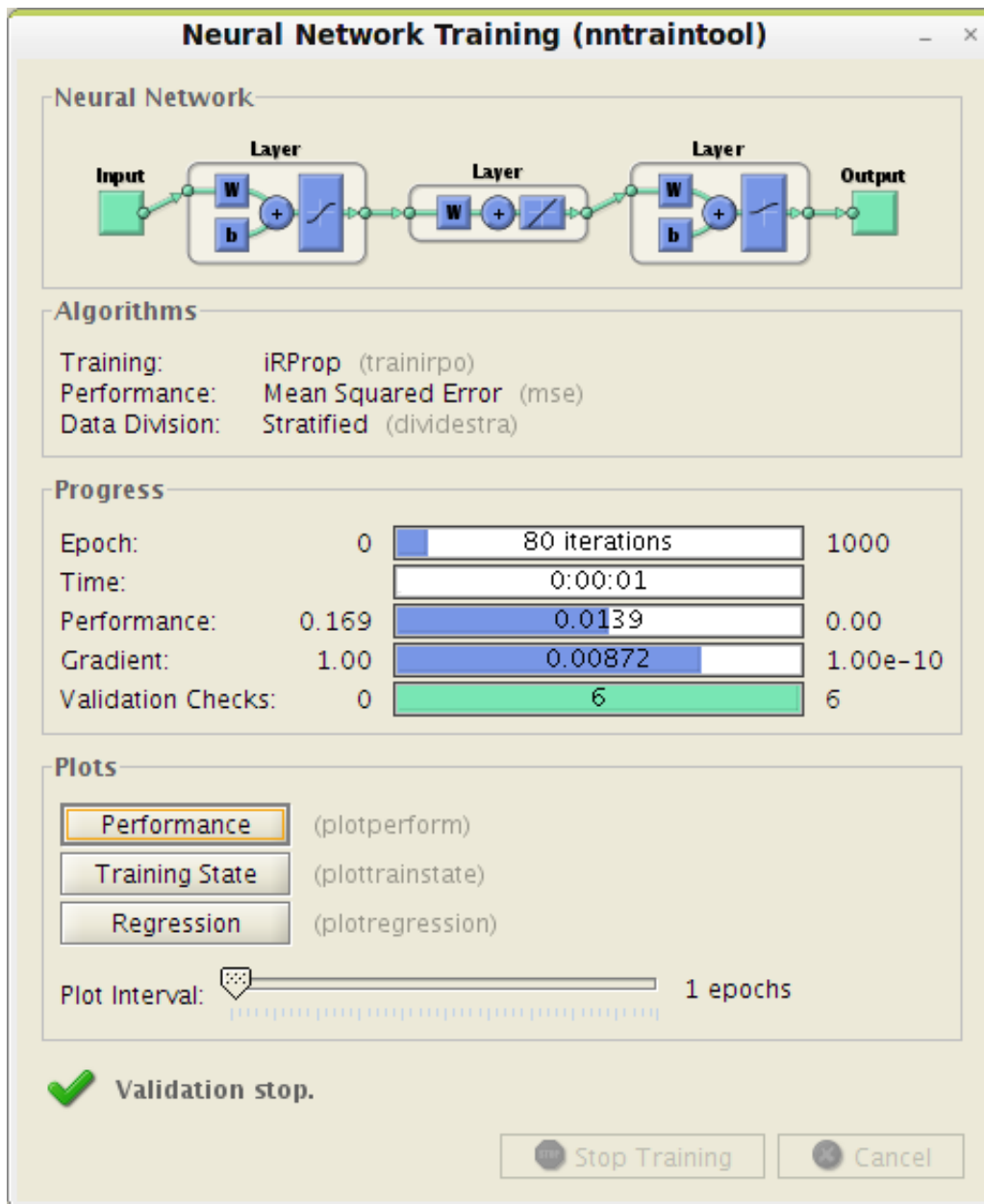


Figura A.25: Interfaz gráfica. Pantalla de entrenamiento de nnet.

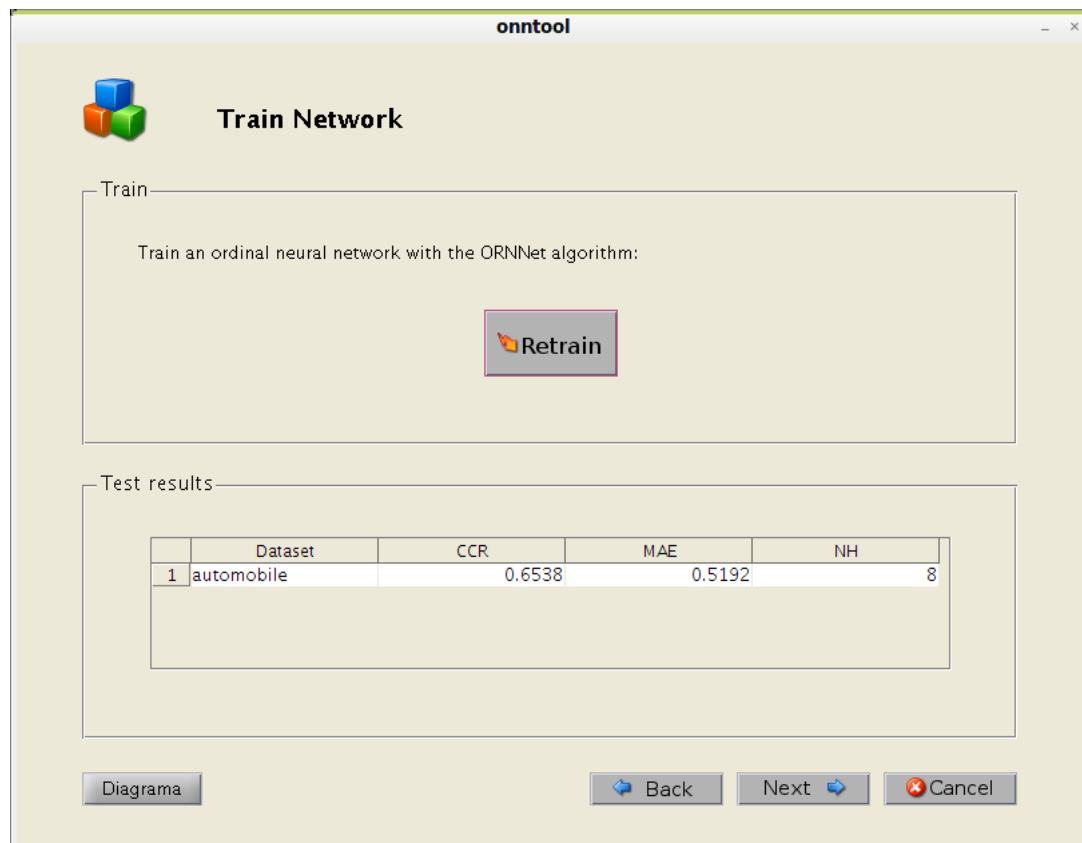


Figura A.26: Interfaz gráfica. Muestra de los resultados obtenidos.

obtenidas al aplicar el algoritmo ORNNNet y los resultados (CCR, MAE y NH). Tendrá la opción de exportar las variables o no según desee, en caso afirmativo habrá que pulsar el botón correspondiente. Por último, existirá un nuevo botón para poder finalizar y cerrar la aplicación (Finish).

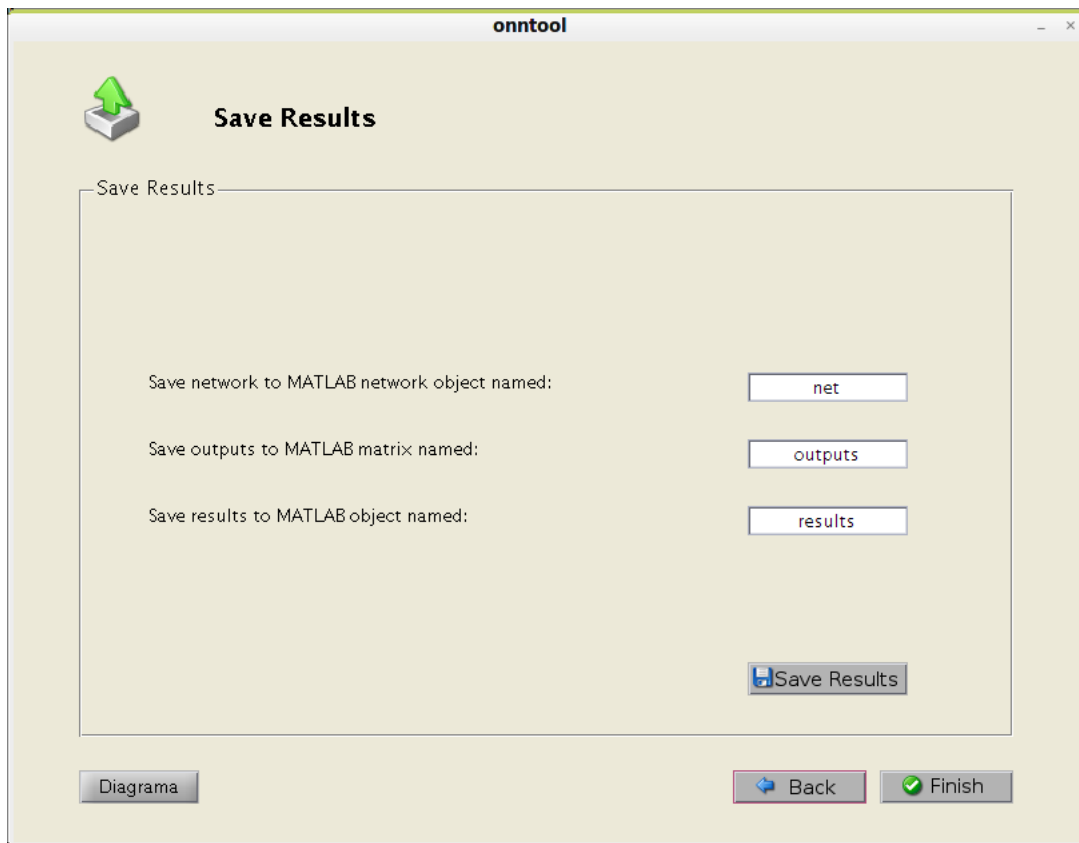


Figura A.27: Interfaz gráfica. Pantalla de guardado de datos.

Comentar que, como se ha observado en las distintas pantallas, se podrá navegar hacia adelante o hacia atrás por las pantallas en todo momento, mientras que todos los parámetros se encuentren especificados o no se pulse el botón de finalización (Finish), lo que facilita la modificación dinámica en todo momento de la aplicación, pudiendo obtener y visua-

lizar nuevos resultados de otros conjuntos de datos u otra configuración de la red neuronal ordinal.



B Manual de código

Este apéndice constituye el manual de código del proyecto fin de carrera. Para que se asemeje cuanto más a la organización seguida por el toolbox de Matlab nnet, se dispone de los siguientes ficheros organizados por carpetas. Las carpetas han sido nombradas con el mismo nombre que tendrían en el toolbox, añadiendo a cada una de dichas carpetas los ficheros que tienen un contenido similar en relación a la organización del toolbox.

■ Carpeta nnet

- *Carpeta @network*
 - Fichero osim.m
 - Fichero otrain.m
- *Carpeta nnformat*
 - Fichero dividestra.m
- *Carpeta nnnetwork*
 - Fichero newoff.m
- *Carpeta nnother*
 - Fichero convdata.m

- Fichero convoutputs.m
- Fichero getstra.m
- Fichero importfile.m
- Fichero kfold.m
- Fichero kfoldo.m
- Fichero transdata.m
- *Carpeta nnperformance*
 - Fichero ccrcalc.m
 - Fichero maecalc.m
- *Carpeta nntrain*
 - Fichero trainirp.m
 - Fichero trainirpo.m
- **Carpeta nnguis**
 - Fichero onntool.m

B.1. Carpeta nnet

B.1.1. Carpeta @network

B.1.1.1. Fichero osim.m

```

1 function testOutputs = simonet(net,testInputs)
2 %SIMONET Simulate an ordinal neural network.
3 %
4 % Syntax
5 %
6 %     simonet(net,test)
7 %
8 % Description
9 %
10 %     SIMONET(net,test) takes,
11 %         net           - Ordinal Neural Network.
12 %         test          - Test inputs of a dataset.
13 %     and returns:
14 %         testOutputs - Test outputs of simulation.

```

```

15 %
16 % Examples
17 %
18 %
19 % See also convoutputs.
20
21 % úRal éPrula íMartnez, 07-2011
22 % Copyright 2011 Universidad de óCrdoaba
23 % $Revision: 1.0 $
24
25
26 %%ERROR CHECKING
27 if (nargin < 2), error('NNET:Arguments','Not enough arguments. '),end
28
29 %%SIMULATION
30 % simulating the net to obtain the outputs (test set)
31 testOutputs = sim(net,testInputs);
32
33 % adding ones to the last output
34 testOutputs(size(testOutputs,1)+1,:) = ones(1,size(testOutputs,2));
35
36 % converting outputs values
37 testOutputs = convoutputs(testOutputs);

```

Listing B.1: Archivo osim.m

B.1.1.2. Fichero otrain.m

```

1 function [net,tr] = otrain(net,X,T)
2 %OTRAIN Train an ordinal neural network.
3 %
4 % Syntax
5 %
6 %     [net,tr] = otrain(NET,X,T)
7 %
8 % Description
9 %
10 %     OTRAIN trains an ordinal network NET according to NET.trainFcn and
11 %     NET.trainParam.
12 %
13 %     OTRAIN(NET,X,T) takes ,
14 %     NET - Network.
15 %     X   - Network inputs.
16 %     T   - Network targets.
17 %     and returns ,
18 %     NET - New network.

```

```

19 %      TR - Training record (epoch and perf).
20 %
21 % See also train.
22
23 % úRal éPrula íMartnez, 07-2011
24 % Copyright 2011 Universidad de óCrdoaba
25 % $Revision: 1.0 $
26
27
28 %%ARGUMENT CHECKS
29 if nargin < 3, error('NNET:Arguments','Not enough input arguments.');
```

Listing B.2: Archivo otrain.m

B.1.2. Carpeta nnformat

B.1.2.1. Fichero dividestra.m

```

1 function [trainV, valV, testV, trainInd, valInd, testInd] = dividestra(allV,
    trainRatio, valRatio, testRatio, targets)
2 %DIVIDESTRA Divide vectors into three sets using stratified indices.
3 %
4 % Syntax
5 %
6 % [trainV, valV, testV, trainInd, valInd, testInd] =
7 %     dividestra(allV, trainRatio, valRatio, testRatio)
8 %
9 % Description
10 %
11 % DIVIDESTRA is used to separate input and target vectors into three
12 % sets: training, validation and testing.
13 %
14 % DIVIDESTRA takes the following inputs,
15 %     allV      - RxQ matrix of Q R-element vectors.
16 %     trainRatio - Ratio of vectors for training, default = 0.8.
17 %     valRatio   - Ratio of vectors for validation, default = 0.2.
18 %     testRatio  - Ratio of vectors for testing, default = 0.
19 % and returns:
20 %     trainV    - Training vectors
21 %     valV      - Validation vectors
22 %     testV     - Test vectors
23 %     trainInd  - Training indices
24 %     valInd    - Validation indices
```



```

25 %      testInd - Test indices
26 %
27 % Examples
28 %
29 %      p = rand(3,1000);
30 %      t = [p(1,:) .* p(2,:); p(2,:) .* p(3,:)];
31 %      [trainP, valP, testV, trainInd, valInd, testInd] = dividestra(p
    ,0.8,0.2,0);
32 %      [trainT, valT, testT] = divideind(t, trainInd, valInd, testInd);
33 %
34 % Network Use
35 %
36 % Here are the network properties that defines which data division
    function
37 % to use, and what its parameters are, when TRAIN is called.
38 %
39 %      net.divideFcn
40 %      net.divideParam
41 %
42 % See also divideblock, divideind, divideint, dividerand.
43
44 % úRal éPrula íMartnez, 07-2011
45 % Copyright 2011 Universidad de óCrdoaba
46 % $Revision: 1.0 $
47
48 %%FUNCTION INFO
49 if ischar(allV)
50     switch (allV)
51     case 'info'
52         info.name = mfilename;
53         info.title = 'Stratified';
54         info.type = 'Data Division';
55         info.version = 6;
56         trainV = info;
57     case 'name'
58         trainV = 'Stratified';
59     case 'fpdefaults'
60         defaults = struct;
61         defaults.trainRatio = 0.8;
62         defaults.valRatio = 0.2;
63         defaults.testRatio = 0;
64         defaults.targets = [];
65         trainV = defaults;
66     otherwise
67         error('NNET:Arguments','Unrecognized string: %s',allV)
68 end
69 return

```

```

70 end
71
72 %%ERROR CHECKING AND DEFAULTS
73 if isstruct(trainRatio)
74     valRatio = trainRatio.valRatio;
75     testRatio = trainRatio.testRatio;
76     targets = trainRatio.targets;
77     trainRatio = trainRatio.trainRatio;
78 elseif (nargin < 4)
79     error('NNET:Arguments','Not enough arguments. ')
80 end
81
82 %%DIVIDE DATA
83 [allV,mode] = nnpackdata(allV);
84
85 % stratified divide
86 % [trainInd, valInd] = crossvalind('HoldOut', targets, trainRatio, 'classes', '1');
87 % for i=2:max(targets)
88 %     [tra, val] = crossvalind('HoldOut', targets, trainRatio, 'classes', num2str(i));
89 %
90 %     % apply logic OR
91 %     trainInd = trainInd | tra;
92 %     valInd = valInd | val;
93 % end
94 [trainInd, valInd] = crossvalind('HoldOut', targets, 1 - trainRatio);
95 testInd = [];
96
97 trainV = allV{1,1}(trainInd);
98 valV = allV{1,1}(valInd);
99 testV = [];

```

Listing B.3: Archivo dividestra.m

B.1.3. Carpeta nnnetwork

B.1.3.1. Fichero newoff.m

```

1 function net = newoff(varargin)
2 %NEWFFO Create an ordinal feed-forward backpropagation network.
3 %
4 % Syntax
5 %
6 %     net = newffo(P,T,S,TF)
7 %

```

```

 8 % Description
 9 %
10 %     NEWOFF(P,T,S,TF) takes ,
11 %         P - RxQ1 matrix of Q1 representative R-element input vectors.
12 %         T - SNxQ2 matrix of Q2 representative SN-element target vectors.
13 %         Si - Sizes of N-1 hidden layers , S1 to S(N-1), default = [20 1].
14 %             (Output layer size SN is determined from T.)
15 %         TFi - Transfer function of ith layer. Default is 'logsig' for
16 %             hidden layers , 'purelin' for the last hidden layer and '
logsig' for output layer.
17 %     and returns an N layer ordinal feed-forward backprop network.
18 %
19 %     The transfer functions TF{i} can be any differentiable transfer
20 %     function such as TANSIG, LOGSIG, or PURELIN.
21 %
22 %     *WARNING*: TRAINIRPO is the default training function because it
23 %     is very fast , but it requires a lot of memory to run.
24 %
25 %     The learning function can be either of the backpropagation
26 %     learning functions such as LEARNGD, or LEARNGDM.
27 %
28 %     The performance function can be any of the differentiable
performance
29 %     functions such as MSE or MSEREG.
30 %
31 % Examples
32 %
33 %     load iris_dataset
34 %     net = newoff(irisInputs,irisTargets,20,logsig);
35 %     net = train(net,irisInputs,irisTargets);
36 %     irisOutputs = osim(net,irisInputs);
37 %
38 % Algorithm
39 %
40 %     Ordinal feed-forward networks consist of Nl layers using the DOTPROD
41 %     weight function , NETSUM net input function , and the specified
42 %     transfer functions.
43 %
44 %     The first layer has weights coming from the input. Each subsequent
45 %     layer has a weight coming from the previous layer. All layers
46 %     have biases except the middle layer. The last layer is the network
output.
47 %
48 %     Each layer's weights and biases are initialized with INTNW.
49 %
50 %     Adaption is done with TRAINS which updates weights with the
51 %     specified learning function. Training is done with the specified

```

```

52 %   training function. Performance is measured according to the
    specified
53 %   performance function.
54 %
55 % See also NEWFF, OSIM, INIT, TRAINIRP, TRAINIRPO.
56
57 % úRal éPrula íMartnez, 07-2011
58 % Copyright 2011 Universidad de óCrdoba
59 % $Revision: 1.0 $
60
61 %%ERROR CHECKING
62 if nargin < 2, error('NNET:Arguments','Not enough input arguments'), end
63
64 %%CREATE NEURAL NETWORK
65 if (nargin == 2)
66     net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[20
        1],[ 'tansig','purelin','logsig'],'trainirpo',...
67         'learngdm','mse',{ 'fixunknowns','removeconstantrows','mapminmax'},{ },
        'dividestra');
68 elseif (nargin == 3)
69     net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[
        varargin{3} 1],[ 'tansig','purelin','logsig'],'trainirpo',...
70         'learngdm','mse',{ 'fixunknowns','removeconstantrows','mapminmax'},{ },
        'dividestra');
71 elseif (nargin == 4)
72     if isa(varargin{4},'cell') %transformation if it is a cell array
73         aux = varargin{4};
74         aux(:,length(aux)+1:length(aux)+2) = {'purelin','logsig'};
75         varargin{4} = aux;
76
77         net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[
        varargin{3} 1],varargin{4},'trainirpo',...
78         'learngdm','mse',{ 'fixunknowns','removeconstantrows','mapminmax'
        },{ },'dividestra');
79     else
80         net = newff(varargin{1},varargin{2}(1:(size(varargin{2},1)-1),:),[
        varargin{3} 1],[ varargin{4},'purelin','logsig'],'trainirpo',...
81         'learngdm','mse',{ 'fixunknowns','removeconstantrows','mapminmax'
        },{ },'dividestra');
82     end
83 else
84     error('NNET:Arguments','Input arguments incorrect');
85 end
86
87 %%ADJUST PARAMETERS
88
89 %set the name of de ANN

```

```

90 net.name = 'Ordinal Neural Network';
91 % delete bias of the last hidden layer
92 net.biasConnect(net.numLayers-1) = 0;
93 % put weight of output layer to one fixed
94 net.LW{net.numLayers,net.numLayers-1} = ones(net.outputs{net.numLayers}.
    size,1);
95 % initialize with correct bias values
96 net.b{net.numLayers} = (-5*((size(net.b{net.numLayers},1)-1)/2):5:5*((
    size(net.b{net.numLayers},1)-1)/2))';
97
98 % adjust dataset divide
99 net.divideParam.trainRatio = 0.8;
100 net.divideParam.valRatio = 0.2;
101 net.divideParam.testRatio = 0;
102 net.divideParam.targets = getstra(varargin{2});

```

Listing B.4: Archivo newoff.m

B.1.4. Carpeta nnother

B.1.4.1. Fichero convdata.m

```

1 function convdata(name, data, C, O, S)
2 %CONVDATA Create a valid matlab inputs and targets datasets with a
   conversion of a invalid dataset.
3 %
4 % Syntax
5 %
6 %     convdata(name, data, C, O, S)
7 %
8 % Description
9 %
10 %     CONVDATA(name, data, C, O, S) takes,
11 %     name - Name of dataset.
12 %     data - Invalid dataset.
13 %     C    - Number of classes.
14 %     O    - Number of outputs.
15 %     S    - Save the inputs and targets in a mat file.
16 %
17 % Examples
18 %
19 %     % initialize values
20 %     name = 'train';
21 %     C = 4;
22 %     O = 3;

```

```

23 % data = [2,3,2,4,1,0,0; 4,1,2,5,1,0,0; 1,1,2,2,0,1,0; 2,1,5,5,0,1,0;
1,1,4,1,0,0,1];
24 %
25 % %convert invalid data in valid datasets
26 % convdata(name, data, C, O);
27 %
28 % See also TRANSDATA, CONVOUTPUTS.
29
30 % úRal éPrula íMartnez, 07–2011
31 % Copyright 2011 Universidad de óCrdoba
32 % $Revision: 1.0 $
33
34 %%ERROR CHECKING
35 if nargin < 4, error('NNET: Arguments', 'Not enough arguments. '), end
36
37 %%DATA CONVERSION
38
39 %data matrix transposition
40 data = data';
41
42 %creating variable with inputs
43 str = [name 'Inputs'];
44 var = genvarname(str);
45 assignin('base', var, data(1:C,:));
46
47 if (nargin == 5 && S == true), save([name '_dataset.mat'], var); end
48
49 %creating variable with targets
50 str = [name 'Targets'];
51 var = genvarname(str);
52 assignin('base', var, data(C+1:C+O,:));
53
54 if (nargin == 5 && S == true), save([name '_dataset.mat'], var, '-append'
); end

```

Listing B.5: Archivo convdata.m

B.1.4.2. Fichero convoutputs.m

```

1 function outputs = convoutputs(outputsOld)
2 %CONVOUTPUTS Transform old outputs dataset in a new outputs dataset.
3 %
4 % Syntax
5 %
6 % outputs = convoutputs(outputsOld)
7 %

```

```

 8 % Description
 9 %
10 %   CONVOUTPUTS(outputsOld) takes ,
11 %       outputsOld – Old outputs dataset.
12 %
13 % Examples
14 %
15 %   % outputs vector
16 %   outputs = [0.50071,0.60006,0.30003,0.4,0.5;
17 %       0.91916,0.90177,0.80103,0.90001,0.10024; 1,1,1,1,1];
18 %
19 %   % transform old outputs
20 %   Outputs = convoutputs(outputs);
21 % See also CONVDATA, TRANSDATA.
22
23 % úRal éPrula íMartnez, 07–2011
24 % Copyright 2011 Universidad de óCrdoaba
25 % $Revision: 1.0 $
26
27 %%ERROR CHECKING
28 if (nargin < 1), error('NNET:Arguments','Not enough arguments. '),end
29
30 %% obtain outputs dataset of no accumulated probabilities
31
32 %% %iterative mode (no valid)
33 % aux = zeros(size(outputsOld,1),1);
34 % for i=1:size(outputsOld,2)
35 %     aux(1) = outputsOld(1,i);
36 %     for j=2:size(outputsOld,1)
37 %         aux(j) = outputsOld(j,i)-outputsOld(j-1,i);
38 %     end
39 %     outputsOld(:,i) = aux;
40 % end
41
42 %% %matricial mode
43 outputsaux = circshift((circshift(outputsOld,-1).*[ones(size(outputsOld
44     ,1)-1,size(outputsOld,2)); zeros(1,size(outputsOld,2))]) - outputsOld,1)
45     ;
46 outputsaux(1,:) = outputsOld(1,:);
47
48 %% boolean transform of outputs dataset
49 aux = zeros(size(outputsaux));
50 [c,i] = max(outputsaux);
51 j = 1;
52 for k=i
53     aux(k,j) = 1;

```

```

52     j = j+1;
53 end
54
55 %%return value
56 outputs = aux;

```

Listing B.6: Archivo convoutputs.m

B.1.4.3. Fichero getstra.m

```

1  function classes = getstra(trainTargets)
2  %GETSTRA Get stratifiedclasses.
3  %
4  % Syntax
5  %
6  %     getstra(trainTargets)
7  %
8  % Description
9  %
10 %     GETSTRA(trainTargets) takes,
11 %     trainTargets – Train targets of a dataset.
12 %     and returns:
13 %     classes      – Vector with stratified classes.
14 %
15 % Examples
16 %
17 %
18 % See also kfold.
19
20 %úRal éPrula íMartnez, 07–2011
21 % Copyright 2011 Universidad de óCrdoba
22 % $Revision: 1.0 $
23
24
25 %%ERROR CHECKING
26 if (nargin < 1), error('NNET: Arguments', 'Not enough arguments. '), end
27
28 classes = zeros(1,max(size(trainTargets)));
29 for i=1:min(size(trainTargets))
30     classes = classes+((trainTargets(i,:))*i);
31 end

```

Listing B.7: Archivo getstra.m

B.1.4.4. Fichero importfile.m

```

1 function importfile(fileToRead1)
2 %IMPORTFILE(FILETOREAD1)
3 % Imports data from the specified file
4 % FILETOREAD1: file to read
5
6 % Auto-generated by MATLAB on 11-Jun-2011 12:40:53
7
8 DELIMITER = ' ';
9 HEADERLINES = 2;
10
11 % Import the file
12 newData1 = importdata(fileToRead1, DELIMITER, HEADERLINES);
13
14 % Create new variables in the base workspace from those fields.
15 vars = fieldnames(newData1);
16 for i = 1:length(vars)
17     assignin('base', vars{i}, newData1.(vars{i}));
18 end

```

Listing B.8: Archivo importfile.m

B.1.4.5. Fichero kfold.m

```

1 function [NHO,E] = kfold(trainInputs,trainTargets,NH,kFold,numIter)
2 %KFOLD Do k-Fold cross-validation.
3 %
4 % Syntax
5 %
6 %     kfold(trainInputs,trainTargets,NHmax,kFold,numIter)
7 %
8 % Description
9 %
10 %     KFOLD(trainInputs,trainTargets,NHmax,kFold,numIter) takes,
11 %     trainInputs - Train inputs of a dataset.
12 %     trainTargets - Train targets of a dataset.
13 %     NH           - Vector of hidden nodes.
14 %     kFold        - Number of folds.
15 %     numIter       - Mean number of calculations.
16 %     and returns:
17 %     NHO           - Number of optimal hidden nodes.
18 %     E             - Mean error.
19 %
20 % Examples

```

```

21 %
22 %
23 % See also transdata, newoff, simonet, maecalc.
24
25 % úRal éPrula íMartnez, 07–2011
26 % Copyright 2011 Universidad de óCrdoba
27 % $Revision: 1.0 $
28
29
30 %%ERROR CHECKING
31 if (nargin < 2), error('NNET: Arguments', 'Not enough arguments. '), end
32
33 %%DEFAULTS
34 if (nargin < 3), NH = 1:5:51; end
35 if (nargin < 4), kFold = 10; end
36 if (nargin < 5), numIter = 3; end
37
38 %%establecimiento de las clases
39 classes = getstra(trainTargets);
40
41 %%K-FOLD CALCULATING
42
43 % getting indices from stratified set of targets
44 indices = crossvalind('Kfold', classes, kFold);
45
46 v = zeros(length(NH), 1);
47
48 for i=NH
49     auxmae = 0;
50     for k=1:kFold
51         %calculating and dividing folds
52         testing = (indices == k)';
53         training = ~testing;
54
55         % Training set
56         trainX = trainInputs(:, training);
57         trainY = trainTargets(:, training);
58
59         % Test set
60         testX = trainInputs(:, testing);
61         testY = trainTargets(:, testing);
62
63         for j=1:numIter
64             %creating the neural network
65             net = newff(trainX, trainY, i, {'tansig', 'logsig'}, 'trainirp', '
                leargdm', 'mse', {'fixunknowns', 'removeconstantrows', 'mapminmax'
                }, {}, 'dividestra');

```

```

66     net.trainParam.showWindow = false; %don't show training interface
67
68     %adjust dataset divide
69         net.divideParam.trainRatio = 0.8;
70         net.divideParam.valRatio = 0.2;
71         net.divideParam.testRatio = 0;
72         net.divideParam.targets = getstra(trainY);
73
74     %training the net
75     net = train(net,trainX,trainY);
76
77     %simulating the net to obtain the outputs (test set)
78     testOutputs = sim(net,testX);
79
80     %tratamiento de los valores de salida
81     aux = zeros(size(testOutputs));
82     [c,index] = max(testOutputs);
83     l = 1;
84     for h=index
85         aux(h,l) = 1;
86         l = l+1;
87     end
88     testOutputs = aux;
89
90     %calculating confussion matrix and mae
91     [c,cm,ind,per] = confusion(testY,testOutputs);
92     auxmae = auxmae+maecalc(cm, size(testX,2));
93     end
94 end
95 %obtaining mean error
96 v(NH == i) = auxmae/(kFold*numIter);
97 end
98
99 %%return values
100 NHO = NH(find(v == min(v),1,'first'));
101 E = min(v);

```

Listing B.9: Archivo kfold.m

B.1.4.6. Fichero kfoldo.m

```

1 function [NHO,E] = kfoldo(trainInputs,trainTargets,NH,kFold,numIter)
2 %KFOLD Do ordinal k-Fold cross-validation.
3 %
4 % Syntax
5 %

```

```

6 %   kfoldo(trainInputs,trainTargets,NHmax,kFold,numIter)
7 %
8 % Description
9 %
10 %   KFOLDO(trainInputs,trainTargets,NHmax,kFold,numIter) takes,
11 %       trainInputs – Train inputs of a dataset.
12 %       trainTargets – Train targets of a dataset.
13 %       NH           – Vector of hidden nodes.
14 %       kFold        – Number of folds.
15 %       numIter      – Mean number of calculations.
16 %   and returns:
17 %       NHO          – Number of optimal hidden nodes.
18 %       E             – Mean error.
19 %
20 % Examples
21 %
22 %
23 % See also transdata, newoff, simonet, maecalc.
24
25 % úRal éPrula íMartnez, 07–2011
26 % Copyright 2011 Universidad de óCrdoba
27 % $Revision: 1.0 $
28
29
30 %%ERROR CHECKING
31 if (nargin < 2), error('NNET:Arguments','Not enough arguments. '),end
32
33 %%DEFAULTS
34 if (nargin < 3), NH = 2.^(0:1:3); end
35 if (nargin < 4), kFold = 10; end
36 if (nargin < 5), numIter = 3; end
37
38 %%establecimiento de las clases
39 classes = getstra(trainTargets);
40
41 %%K-FOLD CALCULATING
42
43 %geting indices from stratified set of targets
44 indices = crossvalind('Kfold',classes,kFold);
45
46 v = zeros(length(NH),1);
47
48 for i=1:NH
49     aux = 0;
50     for k=1:kFold
51         %calculating and dividing folds
52         testing = (indices == k)';

```

```

53     training = ~testing;
54
55     % Training set
56     trainX = trainInputs(:,training);
57     trainY = trainTargets(:,training);
58     trainYNew = transdata(trainY);
59
60     % Test set
61     testX = trainInputs(:,testing);
62     testY = trainTargets(:,testing);
63
64     for j=1:numIter
65         % creating the neural network
66         net = newoff(trainX,trainY,i,'tansig');
67         net.trainParam.showWindow = false; %don't show training interface
68
69         % training the net
70         net = otrain(net,trainX,trainYNew);
71
72         % simulating the net to obtain the outputs (test set)
73         testOutputs = osim(net,testX);
74
75         % calculating confussion matrix and mae
76         [c,cm,ind,per] = confusion(testY,testOutputs);
77         aux = aux+maecalc(cm, size(testX,2));
78     end
79 end
80 % obtaining mean error
81 v(NH == i) = aux/(kFold*numIter);
82 end
83
84 %% return values
85 NHO = NH(v == min(v));
86 E = min(v);

```

Listing B.10: Archivo kfold.m

B.1.4.7. Fichero transdata.m

```

1 function datasetNew = transdata(dataset)
2 %TRANSDATA Transform initial targets dataset in a new targets dataset.
3 %
4 % Syntax
5 %
6 %     datasetNew = transdata(dataset)
7 %

```

```

 8 % Description
 9 %
10 %   TRANSDATA(dataset) takes ,
11 %       dataset - Targets dataset.
12 %
13 % Examples
14 %
15 %   %targets vector
16 %   targets = [1 1 0 0 0 0; 0 0 1 1 0 0; 0 0 0 0 1 1];
17 %
18 %   %transform initial targets
19 %   TargetsNew = transdata(targets);
20 %
21 % See also CONVDATA, CONVOUTPUTS.
22
23 % úRal éPrula íMartnez , 07-2011
24 % Copyright 2011 Universidad de óCrdoaba
25 % $Revision: 1.0 $
26
27 %%ERROR CHECKING
28 if (nargin < 1), error('NNET: Arguments ', 'Not enough arguments. '),end
29
30 %%DATA TRANSFORMATION
31 datasetNew = dataset';
32
33 for j=1:size(datasetNew,2)-1
34     for i=1:size(datasetNew,1)
35         if datasetNew(i,j) == 1
36             datasetNew(i,j+1:size(datasetNew,2)) = ones(size(datasetNew(i,j+1:
37                 size(datasetNew,2)))));
38         end
39     end
40 end
41 %%return value
42 datasetNew = datasetNew';

```

Listing B.11: Archivo transdata.m

B.1.5. Carpeta nnperformance

B.1.5.1. Fichero ccrcalc.m

```

1 function CCR = ccrcalc(MC, TOTAL)
2 %%CCRCALC Calculate Correctly Classified Rate.
3 %

```

```

4 % Syntax
5 %
6 %   CCR = ccrcalc(MC, TOTAL)
7 %
8 % Description
9 %
10 %   CCRCALC(MC, TOTAL) takes ,
11 %       MC - Confusion matrix.
12 %       TOTAL - Total clases elements.
13 %
14 % Examples
15 %
16 % % targets and outputs vectors
17 % targets = [1 1 0 0 0 0; 0 0 1 1 0 0; 0 0 0 0 1 1];
18 % outputs = [0 1 0 0 1 0; 0 0 0 1 1 0; 0 0 0 0 1 1];
19 %
20 % % confussion matrix calc
21 % [c,cm,ind,per] = confusion(targets,outputs);
22 %
23 % %CCR calc
24 % ccrcalc(cm, size(targets,2))
25 %
26 % See also TRACE, MAE, MSE, MAECALC, CONFUSION, PLOTCONFUSION.
27
28 % úRal éPrula íMartnez , 07-2011
29 % Copyright 2011 Universidad de óCrdoaba
30 % $Revision: 1.0 $
31
32 %CCR calculation
33 CCR = trace(MC)/TOTAL;

```

Listing B.12: Archivo ccrcalc.m

B.1.5.2. Fichero maecalc.m

```

1 function MAE = maecalc(MC, TOTAL)
2 %MAECALC Calculate Mean Absolute Error.
3 %
4 % Syntax
5 %
6 %   MAE = maecalc(A)
7 %
8 % Description
9 %
10 %   MAECALC(MC, TOTAL) takes ,
11 %       MC - Confusion matrix.

```

```

12 %      TOTAL – Total clases elements.
13 %
14 % Examples
15 %
16 % %targets and outputs vectors
17 % targets = [1 1 0 0 0 0; 0 0 1 1 0 0; 0 0 0 0 1 1];
18 % outputs = [0 1 0 0 1 0; 0 0 0 1 1 0; 0 0 0 0 1 1];
19 %
20 % %confussion matrix calc
21 % [c,cm,ind,per] = confusion(targets,outputs);
22 %
23 % %MAE calc
24 % maecalc(cm, size(targets,2))
25 %
26 % See also TRACE, CCRCALC, MAE, MSE, CONFUSION, PLOTCONFUSION.
27
28 % úRal éPrula íMartnez, 07–2011
29 % Copyright 2011 Universidad de óCrdoaba
30 % $Revision: 1.0 $
31
32 % wieght matrix calculation
33 for i=1:size(MC, 1)
34     aux = i-1;
35     for j=1:size(MC, 2)
36         W(i,j) = abs(aux);
37         aux = aux-1;
38     end
39 end
40
41 % multiplication of confusion matrix by weight matrix
42 MCnew = MC.*W;
43
44 %MAE calculation
45 MAE = sum(sum(MCnew))/TOTAL;

```

Listing B.13: Archivo maecalc.m

B.1.6. Carpeta nntrain

B.1.6.1. Fichero trainirp.m

```

1 function [net,tr] = trainirp(net,tr,trainV,valV,testV,varargin)
2 %TRAINRP iRPROP+ backpropagation.
3 %
4 % Syntax
5 %

```



```

6 % [net,tr,Ac,El] = trainirp(net,tr,trainV,valV,testV)
7 % info = trainirp('info')
8 %
9 % Description
10 %
11 % TRAINIRP is a network training function that updates weight and
12 % bias values according to the resilient backpropagation algorithm
13 % (iRPROP+).
14 %
15 % TRAINIRP(NET,TR,TRAINV,VALV,TESTV) takes these inputs,
16 % NET - Neural network.
17 % TR - Initial training record created by TRAIN.
18 % TRAINV - Training data created by TRAIN.
19 % VALV - Validation data created by TRAIN.
20 % TESTV - Test data created by TRAIN.
21 % and returns,
22 % NET - Trained network.
23 % TR - Training record of various values over each epoch.
24 %
25 % Each argument TRAINV, VALV and TESTV is a structure of these fields:
26 % X - NxTS cell array of inputs for N inputs and TS timesteps.
27 % X{i,ts} is an RixQ matrix for ith input and ts timestep.
28 % Xi - NxNid cell array of input delay states for N inputs and Nid
29 % delays.
30 % Xi{i,j} is an RixQ matrix for ith input and jth state.
31 % Pd - NxSxNid cell array of delayed input states.
32 % T - NoxTS cell array of targets for No outputs and TS timesteps.
33 % T{i,ts} is an SixQ matrix for the ith output and ts timestep.
34 % Tl - NlxTS cell array of targets for Nl layers and TS timesteps.
35 % Tl{i,ts} is an SixQ matrix for the ith layer and ts timestep.
36 % Ai - NlxTS cell array of layer delays states for Nl layers, TS
37 % timesteps.
38 % Ai{i,j} is an SixQ matrix of delayed outputs for layer i,
39 % delay j.
40 %
41 % Training occurs according to training parameters, with default
42 % values:
43 % net.trainParam.show 25 Epochs between displays
44 % net.trainParam.showCommandLine 0 generate command line output
45 % net.trainParam.showWindow 1 show training GUI
46 % net.trainParam.epochs 100 Maximum number of epochs to train
47 % net.trainParam.goal 0 Performance goal
48 % net.trainParam.time inf Maximum time to train in seconds
49 % net.trainParam.min_grad 1e-6 Minimum performance gradient
50 % net.trainParam.max_fail 5 Maximum validation failures
51 % net.trainParam.lr 0.01 Learning rate
52 % net.trainParam.delt_inc 1.2 Increment to weight change

```

```

49 %      net.trainParam.delt_dec    0.5   Decrement to weight change
50 %      net.trainParam.delta0     0.07   Initial weight change
51 %      net.trainParam.deltamax   50.0   Maximum weight change
52 %
53 %      TRAINIRP('info') returns useful information about this function.
54 %
55 % Network Use
56 %
57 %      You can create a standard network that uses TRAINIRP with
58 %      NEWFF, NEWCF, or NEWELM.
59 %
60 %      To prepare a custom network to be trained with TRAINIRP:
61 %      1) Set NET.trainFcn to 'trainirp'.
62 %          This will set NET.trainParam to TRAINIRP's default parameters.
63 %      2) Set NET.trainParam properties to desired values.
64 %
65 %      In either case, calling TRAIN with the resulting network will
66 %      train the network with TRAINIRP.
67 %
68 % Examples
69 %
70 %      Here is a problem consisting of inputs P and targets T that we would
71 %      like to solve with a network.
72 %
73 %      p = [0 1 2 3 4 5];
74 %      t = [0 0 0 1 1 1];
75 %
76 %      Here a two-layer feed-forward network is created. The network's
77 %      input ranges from [0 to 10]. The first layer has two TANSIG
78 %      neurons, and the second layer has one LOGSIG neuron. The TRAINIRP
79 %      network training function is to be used.
80 %
81 %      % Create and Test a Network
82 %      net = newff([0 5],[2 1],{'tansig','logsig'},'trainirp');
83 %      a = sim(net,p)
84 %
85 %      % Train and Retest the Network
86 %      net.trainParam.epochs = 50;
87 %      net.trainParam.show = 10;
88 %      net.trainParam.goal = 0.1;
89 %      net = train(net,p,t);
90 %      a = sim(net,p)
91 %
92 %      See NEWFF, NEWCF, and NEWELM for other examples.
93 %
94 % Algorithm
95 %

```

```

96 % TRAINIRP can train any network as long as its weight, net input,
97 % and transfer functions have derivative functions.
98 %
99 % Backpropagation is used to calculate derivatives of performance
100 % PERF with respect to the weight and bias variables X. Each
101 % variable is adjusted according to the following:
102 %
103 %     dX = deltaX.*sign(gX);
104 %
105 % where the elements of deltaX are all initialized to delta0 and
106 % gX is the gradient. At each iteration the elements of deltaX
107 % are modified. If an element of gX changes sign from one
108 % iteration to the next, then the corresponding element of
109 % deltaX is decreased by delta_dec. If an element of gX
110 % maintains the same sign from one iteration to the next,
111 % then the corresponding element of deltaX is increased by
112 % delta_inc. See Reidmiller, Proceedings of the IEEE Int. Conf.
113 % on NN (ICNN) San Francisco, 1993, pp. 586–591.
114 %
115 % Training stops when any of these conditions occur:
116 % 1) The maximum number of EPOCHS (repetitions) is reached.
117 % 2) The maximum amount of TIME has been exceeded.
118 % 3) Performance has been minimized to the GOAL.
119 % 4) The performance gradient falls below MINGRAD.
120 % 5) Validation performance has increased more than MAX_FAIL times
121 % since the last time it decreased (when using validation).
122 %
123 % See also NEWFF, NEWCF, TRAINRP, TRAINGDM, TRAINGDA, TRAINGDX, TRAINLM,
124 % TRAINCGP, TRAINCGF, TRAINCGB, TRAINSCG, TRAINOSS,
125 % TRAINBFG.
126 %
127 % References
128 %
129 % Christian Igel, Michael üHsken, Empirical evaluation of the
    improved Rprop learning algorithms
130 % Institut üfr Neuroinformatik, Ruhr–Universität Bochum, 44780 Bochum
    , Germany
131
132 % Updated by éPrula íMartnez, úRal
133 % Copyright 2011
134 % $Revision: 1.0 $ $Date: 2011/mm/dd hh:mm:ss $
135
136 %% Info
137 if strcmp(net, 'info')
138     info.function = mfilename;
139     info.title = 'iRProp';
140     info.type = 'Training';

```

```

141  info.version = 6;
142  info.training_mode = 'Supervised';
143  info.gradient_mode = 'Gradient';
144  info.uses_validation = true;
145  info.param_defaults.show = 25;
146  info.param_defaults.showWindow = true;
147  info.param_defaults.showCommandLine = false;
148  info.param_defaults.epochs = 1000;
149  info.param_defaults.time = inf;
150  info.param_defaults.goal = 0;
151  info.param_defaults.max_fail = 6;
152  info.param_defaults.min_grad = 1e-10;
153  info.param_defaults.delt_inc = 1.2;
154  info.param_defaults.delt_dec = 0.5;
155  info.param_defaults.delta0 = 0.07;
156  info.param_defaults.deltamax = 50.0;
157
158  info.training_states = ...
159      [ ...
160        training_state_info('gradient','Gradient','continuous','log') ...
161        training_state_info('mu','Mu','continuous','log') ...
162        training_state_info('val_fail','Validation Checks','discrete','linear
163        '); ...
164    ];
165  net = info;
166  return
167 end
168 %%NNET 5.1 Backward Compatibility
169 if ischar(net)
170     switch (net)
171     case 'name', info = feval(mfilename,'info'); net = info.title;
172     case 'pnames', info = feval(mfilename,'info'); net = fieldnames(info.
173         param_defaults);
174     case 'pdefaults', info = feval(mfilename,'info'); net = info.
175         param_defaults;
176     case 'gdefaults', if (tr==0), 'calcgrad'; else net='calcgibt'; end
177     otherwise, error('NNET:Arguments','Unrecognized code.')
178     end
179     return
180 end
181 %% Parameters
182 epochs = net.trainParam.epochs;
183 show = net.trainParam.show;
184 goal = net.trainParam.goal;
185 time = net.trainParam.time;

```

```

185 min_grad = net.trainParam.min_grad;
186 max_fail = net.trainParam.max_fail;
187 delt_inc = net.trainParam.delt_inc;
188 delt_dec = net.trainParam.delt_dec;
189 delta0 = net.trainParam.delta0;
190 deltamax = net.trainParam.deltamax;
191 gradientFcn = net.gradientFcn;
192
193 %Parameter Checking
194 if (~isa(epochs, 'double')) || (~isreal(epochs)) || (any(size(epochs)) ~=
    1) || ...
195     (epochs < 1) || (round(epochs) ~= epochs)
196     error('NNET:Arguments', 'Epochs is not a positive integer.')
197 end
198 if (~isa(show, 'double')) || (~isreal(show)) || (any(size(show)) ~= 1) ||
    ...
199     (isfinite(show) && ((show < 1) || (round(show) ~= show)))
200     error('NNET:Arguments', 'Show is not ''NaN'' or a positive integer.')
201 end
202 if (~isa(goal, 'double')) || (~isreal(goal)) || (any(size(goal)) ~= 1) ||
    ...
203     (goal < 0)
204     error('NNET:Arguments', 'Goal is not zero or a positive real value.')
205 end
206 if (~isa(time, 'double')) || (~isreal(time)) || (any(size(time)) ~= 1) ||
    ...
207     (time < 0)
208     error('NNET:Arguments', 'Time is not zero or a positive real value.')
209 end
210 if (~isa(min_grad, 'double')) || (~isreal(min_grad)) || (any(size(min_grad
    )) ~= 1) || ...
211     (min_grad < 0)
212     error('NNET:Arguments', 'Min_grad is not zero or a positive real value.'
    )
213 end
214 if (~isa(max_fail, 'double')) || (~isreal(max_fail)) || (any(size(max_fail
    )) ~= 1) || ...
215     (max_fail < 1) || (round(max_fail) ~= max_fail)
216     error('NNET:Arguments', 'Max_fail is not a positive integer.')
217 end
218 if (~isa(delt_inc, 'double')) || (~isreal(delt_inc)) || (any(size(delt_inc
    )) ~= 1) || ...
219     (delt_inc < 1)
220     error('NNET:Arguments', 'Delt_inc is not a real value greater than 1.')
221 end
222 if (~isa(delt_dec, 'double')) || (~isreal(delt_dec)) || (any(size(delt_dec
    )) ~= 1) || ...

```

```

223 (delt_dec < 0) || (delt_dec > 1)
224 error('NNET:Arguments','Delt_dec is not a real value between 0 and 1.')
225 end
226 if (~isa(delta0,'double')) || (~isreal(delta0)) || (any(size(delta0)) ~=
227     1) || ...
228     (delta0 <= 0)
229 error('NNET:Arguments','Delta0 is not a positive real value.')
230 end
231 if (~isa(deltamax,'double')) || (~isreal(deltamax)) || (any(size(deltamax
232     )) ~= 1) || ...
233     (deltamax <= 0)
234 error('NNET:Arguments','Deltamax is not a positive real value.')
235 end
236
237 %% Initialize
238 Q = trainV.Q;
239 TS = trainV.TS;
240 vperf = NaN;
241 tperf = NaN;
242 val_fail = 0;
243 startTime = clock;
244 X = getx(net);
245 num_X = length(X);
246
247 % Initialize Performance
248 original_net = net;
249 best_net = net;
250 doValidation = ~isempty(valV.indices);
251 doTest = ~isempty(testV.indices);
252 [perf,El,trainV.Y,Ac,N,Zb,Zi,Zl] = calcperf2(net,X,trainV.Pd,trainV.Tl,
253     trainV.Ai,Q,TS);
254 best_perf = perf;
255
256 if (doValidation)
257     [vperf,ignore,valV.Y] = calcperf2(net,X,valV.Pd,valV.Tl,valV.Ai,valV.Q,
258         valV.TS);
259     best_vperf = vperf;
260 end
261
262 %% Training Record
263 tr.best_epoch = 0;
264 tr.goal = goal;
265 tr.states = {'epoch','time','perf','vperf','tperf','gradient','val_fail'
266     };
267
268 %% Status
269 status = ...

```

```

265     [ ...
266     training_status('Epoch','iterations','linear','discrete',0,epochs,0),
        ...
267     training_status('Time','seconds','linear','discrete',0,time,0), ...
268     training_status('Performance','','log','continuous',best_perf,goal,
        best_perf) ...
269     training_status('Gradient','','log','continuous',1,min_grad,1) ...
270     training_status('Validation Checks','','linear','discrete',0,max_fail
        ,0) ...
271 ];
272 nn_train_feedback('start',net,status);
273
274 %% Train
275
276 %— Value of delta0 = 0.07
277 deltaX = delta0*ones(size(X));
278 %— Value of deltamax = 50
279 deltaMAX = deltamax*ones(size(X));
280 deltaMIN = 0*ones(size(X));
281 gX = zeros(size(X));
282 dX = zeros(size(X));
283
284 %— Value of epochs = 1000
285 for epoch=0:epochs
286
287     % Performance and Gradient
288     gX_old = gX;
289     %— Calculate network outputs, signals, and performance
290     [perf,El,trainV.Y,Ac,N,Zb,Zi,Zl] = calcperf2(net,X,trainV.Pd,trainV.Tl,
        trainV.Ai,Q,TS);
291     %— Save old perf to future use to compare
292     perf_old = perf;
293     %— gX calculated by dPerf/dx
294     [gX,gradient] = calcgx(net,X,trainV.Pd,Zb,Zi,Zl,N,Ac,El,perf,Q,TS);
295     if (epoch == 0)
296         gX_old = gX;
297     end
298
299     % Stopping Criteria
300     current_time = etime(clock,startTime);
301     [userStop,userCancel] = nntraintool('check');
302     if userStop, tr.stop = 'User stop.'; net = best_net;
303     elseif userCancel, tr.stop = 'User cancel.'; net = original_net;
304     elseif (perf <= goal), tr.stop = 'Performance goal met.'; net =
        best_net;
305     elseif (epoch == epochs), tr.stop = 'Maximum epoch reached.'; net =
        best_net;

```

```

306     elseif (current_time >= time), tr.stop = 'Maximum time elapsed.'; net =
        best_net;
307     elseif (gradient <= min_grad), tr.stop = 'Minimum gradient reached.';
        net = best_net;
308     elseif (doValidation) && (val_fail >= max_fail), tr.stop = 'Validation
        stop.'; net = best_net;
309     end
310
311     % Training record
312     if doTest
313         [tperf, ignore, testV.Y] = calcperf2(net, X, testV.Pd, testV.Tl, testV.Ai,
            testV.Q, testV.TS);
314     end
315     tr = tr_update(tr, [epoch current_time perf vperf tperf gradient
        val_fail]);
316
317     % Feedback
318     nn_train_feedback('update', net, status, tr, {trainV valV testV}, ...
        [epoch, current_time, best_perf, gradient, val_fail]);
319
320
321     % Stop
322     if ~isempty(tr.stop), break, end
323
324     %%%APPLY iRPROP+ UPDATE %%%
325     ggX = gX.*gX_old;
326
327     %— valor delt_inc = 1.2, delt_dec = 0.5
328     %— deltaX = matriz del ñtamao de los pesos con valor delta0
329     deltaX = ((ggX>0)*delt_inc + (ggX<0)*delt_dec + (ggX==0)).*deltaX;
330     deltaX = (ggX==0).*deltaX + (ggX>0).*min(deltaX, deltaMAX)+ (ggX<0).*max
        (deltaX, deltaMIN);
331     dX = (ggX>0 | ggX==0).*deltaX.*sign(gX) + (ggX<0 & perf>perf_old).*dX;
332
333     %— actualizacion de pesos y bias
334     X = X+(ggX>0 | ggX==0).*dX - (ggX<0 & perf>perf_old).*dX;
335
336     gX = (~(ggX<0)).*gX;
337
338     net = setx(net, X);
339
340     % Validation
341     if (doValidation)
342         [vperf, ignore, valV.Y] = calcperf2(net, X, valV.Pd, valV.Tl, valV.Ai, valV.
            Q, valV.TS);
343         if (vperf < best_vperf)
344             best_net = net;
345             best_perf = perf;

```



```

346     best_vperf = vperf;
347     tr.best_epoch = epoch+1;
348     val_fail = 0;
349     elseif (vperf > best_vperf)
350         val_fail = val_fail + 1;
351     end
352     elseif (perf < best_perf)
353         best_net = net;
354         best_perf = perf;
355         tr.best_epoch = epoch+1;
356     end
357 end
358
359 %% Finish
360 tr = tr_clip(tr);

```

Listing B.14: Archivo trainirp.m

B.1.6.2. Fichero trainirpo.m

```

1 function [net,tr] = trainirpo(net,tr,trainV,valV,testV,varargin)
2 %TRAINRP iRPROP+ ordinal backpropagation.
3 %
4 % Syntax
5 %
6 %     [net,tr,Ac,El] = trainirpo(net,tr,trainV,valV,testV)
7 %     info = trainirpo('info')
8 %
9 % Description
10 %
11 %     TRAINIRPO is a network training function that updates weight and
12 %     bias values according to the resilient backpropagation algorithm
13 %     (iRPROP+).
14 %
15 %     TRAINIRPO(NET,TR,TRAINV,VALV,TESTV) takes these inputs,
16 %     NET - Neural network.
17 %     TR - Initial training record created by TRAIN.
18 %     TRAINV - Training data created by TRAIN.
19 %     VALV - Validation data created by TRAIN.
20 %     TESTV - Test data created by TRAIN.
21 %     and returns,
22 %     NET - Trained network.
23 %     TR - Training record of various values over each epoch.
24 %
25 %     Each argument TRAINV, VALV and TESTV is a structure of these fields:
26 %     X - NxTS cell array of inputs for N inputs and TS timesteps.

```

```

27 %      X{i,ts} is an RixQ matrix for ith input and ts timestep.
28 %      Xi - NxNid cell array of input delay states for N inputs and Nid
delays.
29 %      Xi{i,j} is an RixQ matrix for ith input and jth state.
30 %      Pd - NxSxNid cell array of delayed input states.
31 %      T - NoxTS cell array of targets for No outputs and TS timesteps.
32 %      T{i,ts} is an SixQ matrix for the ith output and ts timestep.
33 %      Tl - NlxTS cell array of targets for Nl layers and TS timesteps.
34 %      Tl{i,ts} is an SixQ matrix for the ith layer and ts timestep.
35 %      Ai - NlxTS cell array of layer delays states for Nl layers, TS
timesteps.
36 %      Ai{i,j} is an SixQ matrix of delayed outputs for layer i,
delay j.
37 %
38 %      Training occurs according to training parameters, with default
values:
39 %      net.trainParam.show          25  Epochs between displays
40 %      net.trainParam.showCommandLine 0 generate command line output
41 %      net.trainParam.showWindow    1 show training GUI
42 %      net.trainParam.epochs        100 Maximum number of epochs to train
43 %      net.trainParam.goal           0  Performance goal
44 %      net.trainParam.time           inf Maximum time to train in seconds
45 %      net.trainParam.min_grad       1e-6 Minimum performance gradient
46 %      net.trainParam.max_fail        5  Maximum validation failures
47 %      net.trainParam.lr             0.01 Learning rate
48 %      net.trainParam.delt_inc        1.2 Increment to weight change
49 %      net.trainParam.delt_dec        0.5 Decrement to weight change
50 %      net.trainParam.delta0          0.07 Initial weight change
51 %      net.trainParam.deltamax       50.0 Maximum weight change
52 %
53 %      TRAINRPO('info ') returns useful information about this function.
54 %
55 % Network Use
56 %
57 %      You can create a standard network that uses TRAINRPO with
58 %      NEWFF, NEWCF, or NEWELM.
59 %
60 %      To prepare a custom network to be trained with TRAINRPO:
61 %      1) Set NET.trainFcn to 'trainirpo'.
62 %      This will set NET.trainParam to TRAINRPO's default parameters.
63 %      2) Set NET.trainParam properties to desired values.
64 %
65 %      In either case, calling TRAIN with the resulting network will
66 %      train the network with TRAINRPO.
67 %
68 % Examples
69 %

```

```

70 % Here is a problem consisting of inputs P and targets T that we would
71 % like to solve with a network.
72 %
73 %     p = [0 1 2 3 4 5];
74 %     t = [0 0 0 1 1 1];
75 %
76 % Here a two-layer feed-forward network is created. The network's
77 % input ranges from [0 to 10]. The first layer has two TANSIG
78 % neurons, and the second layer has one LOGSIG neuron. The TRAINIRPO
79 % network training function is to be used.
80 %
81 %     % Create and Test a Network
82 %     net = newoff(p,t,20,'logsig','trainirpo');
83 %     a = sim(net,p)
84 %
85 %     % Train and Retest the Network
86 %     net.trainParam.epochs = 50;
87 %     net.trainParam.show = 10;
88 %     net.trainParam.goal = 0.1;
89 %     net = train(net,p,t);
90 %     a = sim(net,p)
91 %
92 % See NEWOFF, NEWFF, and TRAINIRP for other examples.
93 %
94 % Algorithm
95 %
96 % TRAINIRPO can train any network as long as its weight, net input,
97 % and transfer functions have derivative functions.
98 %
99 % Ordinal backpropagation is used to calculate derivatives of
100 % performance
101 % PERF with respect to the weight and bias variables X. Each
102 % variable is adjusted according to the following:
103 %
104 %     dX = deltaX.*sign(gX);
105 %
106 % where the elements of deltaX are all initialized to delta0 and
107 % gX is the gradient. At each iteration the elements of deltaX
108 % are modified. If an element of gX changes sign from one
109 % iteration to the next, then the corresponding element of
110 % deltaX is decreased by delta_dec. If an element of gX
111 % maintains the same sign from one iteration to the next,
112 % then the corresponding element of deltaX is increased by
113 % delta_inc. See Reidmiller, Proceedings of the IEEE Int. Conf.
114 % on NN (ICNN) San Francisco, 1993, pp. 586-591.
115 %
116 % Training stops when any of these conditions occur:

```

```

116 % 1) The maximum number of EPOCHS (repetitions) is reached.
117 % 2) The maximum amount of TIME has been exceeded.
118 % 3) Performance has been minimized to the GOAL.
119 % 4) The performance gradient falls below MINGRAD.
120 % 5) Validation performance has increased more than MAX_FAIL times
121 %     since the last time it decreased (when using validation).
122 %
123 % See also NEWOFF, NEWFF, TRAINRP, TRAINIRP.
124 %
125 % References
126 %
127 % Christian Igel, Michael üHskén, Empirical evaluation of the
    improved Rprop learning algorithms
128 % Institut üfr Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum
    , Germany
129
130 % Updated by úRal éPrula íMartnez
131 % Copyright 2011
132 % $Revision: 1.0 $
133
134 %% Info
135 if strcmp(net, 'info')
136     info.function = mfilename;
137     info.title = 'iRProp';
138     info.type = 'Training';
139     info.version = 6;
140     info.training_mode = 'Supervised';
141     info.gradient_mode = 'Gradient';
142     info.uses_validation = true;
143     info.param_defaults.show = 25;
144     info.param_defaults.showWindow = true;
145     info.param_defaults.showCommandLine = false;
146     info.param_defaults.epochs = 1000;
147     info.param_defaults.time = inf;
148     info.param_defaults.goal = 0;
149     info.param_defaults.max_fail = 6;
150     info.param_defaults.min_grad = 1e-10;
151     info.param_defaults.delt_inc = 1.2;
152     info.param_defaults.delt_dec = 0.5;
153     info.param_defaults.delta0 = 0.07;
154     info.param_defaults.deltamax = 50.0;
155
156     info.training_states = ...
157     [ ...
158         training_state_info('gradient', 'Gradient', 'continuous', 'log') ...
159         training_state_info('mu', 'Mu', 'continuous', 'log') ...

```

```

160     training_state_info('val_fail','Validation Checks','discrete','linear
161     '); ...
161 ];
162 net = info;
163 return
164 end
165
166 %%NNET 5.1 Backward Compatibility
167 if ischar(net)
168     switch (net)
169         case 'name', info = feval(mfilename,'info'); net = info.title;
170         case 'pnames', info = feval(mfilename,'info'); net = fieldnames(info.
171             param_defaults);
172         case 'pdefaults', info = feval(mfilename,'info'); net = info.
173             param_defaults;
174         case 'gdefaults', if (tr==0), 'calcgrad'; else net='calcgbtt'; end
175         otherwise, error('NNET:Arguments','Unrecognized code.')
176     end
177 return
178 end
179
180 %% Parameters
181 epochs = net.trainParam.epochs;
182 show = net.trainParam.show;
183 goal = net.trainParam.goal;
184 time = net.trainParam.time;
185 min_grad = net.trainParam.min_grad;
186 max_fail = net.trainParam.max_fail;
187 delt_inc = net.trainParam.delt_inc;
188 delt_dec = net.trainParam.delt_dec;
189 delta0 = net.trainParam.delta0;
190 deltamax = net.trainParam.deltamax;
191 gradientFcn = net.gradientFcn;
192
193 % Parameter Checking
194 if (~isa(epochs,'double')) || (~isreal(epochs)) || (any(size(epochs)) ~=
195     1) || ...
196     (epochs < 1) || (round(epochs) ~= epochs)
197     error('NNET:Arguments','Epochs is not a positive integer.')
198 end
199 if (~isa(show,'double')) || (~isreal(show)) || (any(size(show)) ~= 1) ||
200     ...
201     (isfinite(show) && ((show < 1) || (round(show) ~= show)))
202     error('NNET:Arguments','Show is not ''NaN'' or a positive integer.')
203 end
204 if (~isa(goal,'double')) || (~isreal(goal)) || (any(size(goal)) ~= 1) ||
205     ...

```

```

201 (goal < 0)
202 error('NNET:Arguments','Goal is not zero or a positive real value.')
203 end
204 if (~isa(time,'double')) || (~isreal(time)) || (any(size(time)) ~= 1) ||
    ...
205 (time < 0)
206 error('NNET:Arguments','Time is not zero or a positive real value.')
207 end
208 if (~isa(min_grad,'double')) || (~isreal(min_grad)) || (any(size(min_grad
    )) ~= 1) || ...
209 (min_grad < 0)
210 error('NNET:Arguments','Min_grad is not zero or a positive real value.'
    )
211 end
212 if (~isa(max_fail,'double')) || (~isreal(max_fail)) || (any(size(max_fail
    )) ~= 1) || ...
213 (max_fail < 1) || (round(max_fail) ~= max_fail)
214 error('NNET:Arguments','Max_fail is not a positive integer.')
215 end
216 if (~isa(delt_inc,'double')) || (~isreal(delt_inc)) || (any(size(delt_inc
    )) ~= 1) || ...
217 (delt_inc < 1)
218 error('NNET:Arguments','Delt_inc is not a real value greater than 1.')
219 end
220 if (~isa(delt_dec,'double')) || (~isreal(delt_dec)) || (any(size(delt_dec
    )) ~= 1) || ...
221 (delt_dec < 0) || (delt_dec > 1)
222 error('NNET:Arguments','Delt_dec is not a real value between 0 and 1.')
223 end
224 if (~isa(delta0,'double')) || (~isreal(delta0)) || (any(size(delta0)) ~=
    1) || ...
225 (delta0 <= 0)
226 error('NNET:Arguments','Delta0 is not a positive real value.')
227 end
228 if (~isa(deltamax,'double')) || (~isreal(deltamax)) || (any(size(deltamax
    )) ~= 1) || ...
229 (deltamax <= 0)
230 error('NNET:Arguments','Deltamax is not a positive real value.')
231 end
232
233 %% Initialize
234 Q = trainV.Q;
235 TS = trainV.TS;
236 vperf = NaN;
237 tperf = NaN;
238 val_fail = 0;
239 startTime = clock;

```

```

240 X = getx(net);
241 num_X = length(X);
242
243 % Initialize Performance
244 original_net = net;
245 best_net = net;
246 doValidation = ~isempty(valV.indices);
247 doTest = ~isempty(testV.indices);
248 [perf,El,trainV.Y,Ac,N,Zb,Zi,Zl] = calcperf2(net,X,trainV.Pd,trainV.Tl,
    trainV.Ai,Q,TS);
249 best_perf = perf;
250
251 if (doValidation)
252     [vperf,ignore,valV.Y] = calcperf2(net,X,valV.Pd,valV.Tl,valV.Ai,valV.Q,
    valV.TS);
253     best_vperf = vperf;
254 end
255
256 %% Training Record
257 tr.best_epoch = 0;
258 tr.goal = goal;
259 tr.states = {'epoch','time','perf','vperf','tperf','gradient','val_fail'
    };
260
261 %% Status
262 status = ...
263     [ ...
264         training_status('Epoch','iterations','linear','discrete',0,epochs,0),
    ...
265         training_status('Time','seconds','linear','discrete',0,time,0), ...
266         training_status('Performance','','log','continuous',best_perf,goal,
    best_perf) ...
267         training_status('Gradient','','log','continuous',1,min_grad,1) ...
268         training_status('Validation Checks','','linear','discrete',0,max_fail
    ,0) ...
269     ];
270 nn_train_feedback('start',net,status);
271
272 %% Train
273
274 %— Value of delta0 = 0.07
275 deltaX = delta0*ones(size(X));
276 %— Value of deltamax = 50
277 deltaMAX = deltamax*ones(size(X));
278 deltaMIN = 0*ones(size(X));
279 gX = zeros(size(X));
280 dX = zeros(size(X));

```

```

281
282 %— Value of epochs = 1000
283 for epoch=0:epochs
284
285 %Performance and Gradient
286 gX_old = gX;
287 %— Calculate network outputs, signals, and performance
288 [perf,El,trainV.Y,Ac,N,Zb,Zi,Zl] = calcperf2(net,X,trainV.Pd,trainV.Tl,
    trainV.Ai,Q,TS);
289 %— Save old perf to future use to compare
290 perf_old = perf;
291 %— gX calculated by dPerf/dx
292 [gX,gradient] = calcgx(net,X,trainV.Pd,Zb,Zi,Zl,N,Ac,El,perf,Q,TS);
293 if (epoch == 0)
294     gX_old = gX;
295 end
296
297 %Stopping Criteria
298 current_time = etime(clock,startTime);
299 [userStop,userCancel] = nntraintool('check');
300 if userStop, tr.stop = 'User stop.'; net = best_net;
301 elseif userCancel, tr.stop = 'User cancel.'; net = original_net;
302 elseif (perf <= goal), tr.stop = 'Performance goal met.'; net =
    best_net;
303 elseif (epoch == epochs), tr.stop = 'Maximum epoch reached.'; net =
    best_net;
304 elseif (current_time >= time), tr.stop = 'Maximum time elapsed.'; net =
    best_net;
305 elseif (gradient <= min_grad), tr.stop = 'Minimum gradient reached.';
    net = best_net;
306 elseif (doValidation) && (val_fail >= max_fail), tr.stop = 'Validation
    stop.'; net = best_net;
307 end
308
309 %Training record
310 if doTest
311     [tperf,ignore,testV.Y] = calcperf2(net,X,testV.Pd,testV.Tl,testV.Ai,
        testV.Q,testV.TS);
312 end
313 tr = tr_update(tr,[epoch current_time perf vperf tperf gradient
    val_fail]);
314
315 %Feedback
316 nn_train_feedback('update',net,status,tr,{trainV valV testV}, ...
    [epoch,current_time,best_perf,gradient,val_fail]);
317
318
319 %Stop

```



```

320     if ~isempty(tr.stop), break, end
321
322     %%%APPLY iRPROP+ UPDATE %%%
323     ggX = gX.*gX_old;
324
325     %— valor delt_inc = 1.2, delt_dec = 0.5
326     %— deltaX = matriz del tamaño de los pesos con valor delta0
327     deltaX = ((ggX>0)*delt_inc + (ggX<0)*delt_dec + (ggX==0)).*deltaX;
328     deltaX = (ggX==0).*deltaX + (ggX>0).*min(deltaX,deltaMAX)+ (ggX<0).*max
        (deltaX,deltaMIN);
329     dX = (ggX>0 | ggX==0).*deltaX.*sign(gX) + (ggX<0 & perf>perf_old).*dX;
330     ddX = (((ggX>0 | ggX==0).*dX)-((ggX<0 & perf>perf_old).*dX));
331
332     %— actualizacion de pesos, modificacion para que los pesos de la
        salida
333     %— siempre valgan uno
334     len = net.outputs{net.numLayers}.size;
335     X = X+ddX.*[ones(length(X)-2*len,1); zeros(2*len,1)];
336
337     %— actualizacion de bias
338     %— controlar la condicion de beta1 < beta2 < ... < betaN,
339     %— siendo las betas el valor de las bias de la capa de salida, que en
        este
340     %— caso
341     for i=(length(X)-len+1):length(X)
342         %— comprobar que el siguiente elemento es mayor
343         if i == length(X)
344             if X(i-1) < (X(i)+ddX(i))
345                 X(i) = X(i)+ddX(i);
346             else
347                 gX(i) = 0;
348             end
349         else
350             if (X(i)+ddX(i)) < X(i+1)
351                 X(i) = X(i)+ddX(i);
352             else
353                 gX(i) = 0;
354             end
355         end
356     end
357
358     gX = (~(ggX<0)).*gX;
359
360     net = setx(net,X);
361
362     % Validation
363     if (doValidation)

```

```

364     [vperf, ignore, valV.Y] = calcperf2(net, X, valV.Pd, valV.Tl, valV.Ai, valV.
        Q, valV.TS);
365     if (vperf < best_vperf)
366         best_net = net;
367         best_perf = perf;
368         best_vperf = vperf;
369         tr.best_epoch = epoch+1;
370         val_fail = 0;
371     elseif (vperf > best_vperf)
372         val_fail = val_fail + 1;
373     end
374     elseif (perf < best_perf)
375         best_net = net;
376         best_perf = perf;
377         tr.best_epoch = epoch+1;
378     end
379 end
380
381 %% Finish
382 tr = tr_clip(tr);

```

Listing B.15: Archivo trainirpo.m

B.2. Carpeta nnguis

B.2.1. Fichero onntool.m

```

1 function onntool()
2     %get screen size
3     screenSize = get(0, 'ScreenSize');
4
5     bgcolor = [236/255 233/255 216/255]; %background color
6     bgbtncolor = [0.7 0.7 0.7]; %background color of buttons
7
8     %check window size
9     if (screenSize(3) < 800 && screenSize(4) <= 600)
10         errordlg('Screen size incompatible.', 'Error', 'modal');
11     else
12         close(gcf);
13     end
14
15     %% global variables
16
17     global paso;
18     global guidata;

```

```

19  global text_data;
20  global method_selected;
21  global kfold_pushed;
22  global train_pushed;
23
24  paso = 0;
25  method_selected = false;
26  kfold_pushed = false;
27  train_pushed = false;
28
29  %% main figure
30  main_id = figure(...
31      'Position',[screenSize(3)/5 screenSize(4)/8 800 600],...
32      'Color',bgcolor,...
33      'Name','onntool',...
34      'NumberTitle','off',...
35      'Resize','off',...
36      'BusyAction','queue',...
37      'Interruptible','off',...
38      'DoubleBuffer','on',...
39      'Menubar','none');
40
41  %% header
42  txt_title = uicontrol(main_id,...
43      'BackgroundColor',bgcolor,...
44      'Style','text',...
45      'FontSize',15,...
46      'FontWeight','bold',...
47      'HorizontalAlignment','center',...
48      'String','Welcome to the Ordinal Neural Network Tool',...
49      'Units','pixels',...
50      'Position',[150 525 450 30]);
51
52  uco = imread('img/uco.png','BackgroundColor',bgcolor);
53  logo_left_id = axes(...
54      'Parent',main_id,...
55      'Units','pixels',...
56      'Position',[50 525 50 50]);
57  set(logo_left_id,'UserData',imshow(uco));
58
59  inf = imread('img/inf.png','BackgroundColor',bgcolor);
60  logo_right_id = axes(...
61      'Parent',main_id,...
62      'Units','pixels',...
63      'Position',[700 525 50 50]);
64  set(logo_right_id,'UserData',imshow(inf));
65

```

```

66  %%main buttons
67
68  imgs = imread(fullfile('img','next.png'),'BackgroundColor',bgbtncolor);
69  btn_next = uicontrol(main_id,...
70      'Position',[650-110 30 100 25],...
71      'CData',imgs,...
72      'Callback',@next_Callback);
73
74  imgs = imread(fullfile('img','back.png'),'BackgroundColor',bgbtncolor);
75  btn_back = uicontrol(main_id,...
76      'Position',[650-220 30 100 25],...
77      'CData',imgs,...
78      'Callback',@back_Callback);
79  set(btn_back,'Visible','off');
80
81  imgs = imread(fullfile('img','cancel.png'),'BackgroundColor',bgbtncolor);
82  btn_close = uicontrol(main_id,...
83      'Position',[650 30 100 25],...
84      'CData',imgs,...
85      'Callback','clear;close(gcf)');
86
87  %%main window content
88  ipanel_id = uipanel(...
89      'Title','Information',...
90      'FontSize',13,...
91      'FontWeight','bold',...
92      'BackgroundColor',bgcolor,...
93      'Units','pixels',...
94      'Position',[50 80 345 420]);
95
96  text_prin = sprintf(['\n'...
97      'In ordinal problems, you can use a neural network to map
98      '...
99      'a data set of numeric inputs to a set of ordered labels
100      '\n\n'...
101      'Some examples of this type of problems include
102      'automobile data '...
103      'in some studies (automobile_dataset), bond rate of some
104      'cities '...
105      '(bondrate_dataset), or estimation of thyroid in some
106      'people (thyroid_dataset).\n\n'...
107      'The Ordinal Neural Network Tool will help you select
108      'data, create '...
109      'and train a network, do a K-fold cross-validation,
110      'evaluate its '...

```

```

104         'performance using correlation coefficient ratio and mean
105         absolute '...
106     itxt_text = uicontrol(...
107         'Parent',ipanel_id,...
108         'Style','text',...
109         'BackgroundColor',bgcolor,...
110         'HorizontalAlignment','left',...
111         'Clipping','on',...
112         'FontSize',12,...
113         'String',text_prin,...
114         'Units','pixels',...
115         'Position',[5 2 335 390]);
116
117     npanel_id = uipanel(...
118         'Title','Ordinal Neural Network',...
119         'FontSize',13,...
120         'FontWeight','bold',...
121         'BackgroundColor',bgcolor,...
122         'Units','pixels',...
123         'Position',[405 80 345 420]);
124
125     text_nnet = sprintf(['\n'...
126         'The artificial neural network will be formed by a input
127         ',...
128         'layer, two hidden layers – the first with a specific
129         number of '...
130         'neurons and the second with one neuron and without bias,
131         transfer '...
132         'function of this layer is a purelin –, the last layer
133         has the same '...
134         'number of neurons that the number of categories and
135         transfer '...
136         'function is a logsig.\n\n'...
137         'To optimize the number of neurons in the first hidden
138         layer, '...
139         'the graphical interface has the option to realize a K-
140         fold '...
141         'cross-validation over the training set.\n\n'...
142         'The network will be trained with ORNNet algorithm (
143         trainirpo) '...
144         'that has been implemented by the author specifically for
145         this project.']);
146
147     ntxt_text = uicontrol(...
148         'Parent',npanel_id,...
149         'Style','text',...
150         'BackgroundColor',bgcolor,...

```

```

141         'HorizontalAlignment', 'left', ...
142         'Clipping', 'on', ...
143         'FontSize', 12, ...
144         'String', text_nnet, ...
145         'Units', 'pixels', ...
146         'Position', [5 2 335 390]);
147
148 %% data window content
149
150 % train zone
151 train_panel_id = uipanel(...
152     'Title', 'Train Data', ...
153     'FontSize', 12, ...
154     'BackgroundColor', bgcolor, ...
155     'Units', 'pixels', ...
156     'Position', [50 300 345 200]);
157 set(train_panel_id, 'Visible', 'off');
158
159 traini_text_id = uicontrol(train_panel_id, ...
160     'Style', 'text', ...
161     'String', 'Input data:', ...
162     'BackgroundColor', bgcolor, ...
163     'HorizontalAlignment', 'left', ...
164     'Position', [40 108 100 22]);
165
166 traini_pop_id = uicontrol(train_panel_id, ...
167     'Style', 'popupmenu', ...
168     'String', { '' }, ...
169     'Value', 1, ...
170     'Position', [160 110 100 20]);
171
172 btn_open = uicontrol(train_panel_id, ...
173     'Position', [270 108 30 22], ...
174     'String', '...', ...
175     'Callback', @open_Callback);
176
177 traint_text_id = uicontrol(train_panel_id, ...
178     'Style', 'text', ...
179     'String', 'Target data:', ...
180     'BackgroundColor', bgcolor, ...
181     'HorizontalAlignment', 'left', ...
182     'Position', [40 58 100 22]);
183
184 traint_pop_id = uicontrol(train_panel_id, ...
185     'Style', 'popupmenu', ...
186     'String', { '' }, ...
187     'Value', 1, ...

```

```

188         'Position',[160 60 100 20]);
189
190 btn_open = uicontrol(train_panel_id,...
191     'Position',[270 58 30 22],...
192     'String','... ',...
193     'Callback',@open_Callback);
194
195 % test zone
196 test_panel_id = uipanel(...
197     'Title','Test Data',...
198     'FontSize',12,...
199     'BackgroundColor',bgcolor,...
200     'Units','pixels',...
201     'Position',[50 80 345 200]);
202 set(test_panel_id,'Visible','off');
203
204 testi_text_id = uicontrol(test_panel_id,...
205     'Style','text',...
206     'String','Input data:',...
207     'BackgroundColor',bgcolor,...
208     'HorizontalAlignment','left',...
209     'Position',[40 108 100 22]);
210
211 testi_pop_id = uicontrol(test_panel_id,...
212     'Style','popupmenu',...
213     'String',{' '},...
214     'Value',1,...
215     'HorizontalAlignment','center',...
216     'Position',[160 110 100 20]);
217
218 btn_open = uicontrol(test_panel_id,...
219     'Position',[270 108 30 22],...
220     'String','... ',...
221     'Callback',@open_Callback);
222
223 testt_text_id = uicontrol(test_panel_id,...
224     'Style','text',...
225     'String','Target data:',...
226     'BackgroundColor',bgcolor,...
227     'HorizontalAlignment','left',...
228     'Position',[40 58 100 22]);
229
230 testt_pop_id = uicontrol(test_panel_id,...
231     'Style','popupmenu',...
232     'String',{' '},...
233     'Value',1,...
234     'HorizontalAlignment','center',...

```

```

235         'Position',[160 60 100 20]);
236
237 btn_open = uicontrol(test_panel_id,...
238     'Position',[270 58 30 22],...
239     'String','... ',...
240     'Callback',@open_Callback);
241
242 %% net window content
243
244 param_panel_id = uipanel(...
245     'Title','Parameters',...
246     'FontSize',12,...
247     'BackgroundColor',bgcolor,...
248     'Units','pixels',...
249     'Position',[50 80 345 420]);
250 set(param_panel_id,'Visible','off');
251
252 kfold_text_id = uicontrol(param_panel_id,...
253     'Style','text',...
254     'String','K-fold?',...
255     'BackgroundColor',bgcolor,...
256     'HorizontalAlignment','left',...
257     'Position',[35 345 120 22]);
258
259 kfold_pop_id = uicontrol(param_panel_id,...
260     'Style','popupmenu',...
261     'String',{' ','Yes','No'},...
262     'Value',1,...
263     'HorizontalAlignment','center',...
264     'Callback',@update_kfold,...
265     'Position',[185 350 120 20]);
266
267 k_text_id = uicontrol(param_panel_id,...
268     'Style','text',...
269     'String','k:',...
270     'BackgroundColor',bgcolor,...
271     'HorizontalAlignment','left',...
272     'Position',[35 295 120 22]);
273
274 k_edit_id = uicontrol(param_panel_id,...
275     'Style','edit',...
276     'String','10',...
277     'BackgroundColor','white',...
278     'Callback',@update_k,...
279     'Position',[185 300 120 22]);
280
281 btn_kfold = uicontrol(param_panel_id,...

```



```

282         'Position',[185 250 120 25],...
283         'String','K-fold',...
284         'Callback',@kfold_Callback);
285
286 ikfold_text_id = uicontrol(param_panel_id,...
287         'Style','text',...
288         'String',['If you prefer to manually configure '...
289         'the neural net, specify number of neurons '...
290         'and transfer function of the hidden layer:'],...
291         'HorizontalAlignment','left',...
292         'BackgroundColor',bgcolor,...
293         'Position',[35 170 270 50]);
294
295 neu_text_id = uicontrol(param_panel_id,...
296         'Style','text',...
297         'String','Neurons:',...
298         'BackgroundColor',bgcolor,...
299         'HorizontalAlignment','left',...
300         'Position',[35 125 120 22]);
301
302 neu_edit_id = uicontrol(param_panel_id,...
303         'Style','edit',...
304         'String','20',...
305         'Callback',@update_nh,...
306         'Position',[185 130 120 22]);
307
308 tra_text_id = uicontrol(param_panel_id,...
309         'Style','text',...
310         'String','Transfer function:',...
311         'BackgroundColor',bgcolor,...
312         'HorizontalAlignment','left',...
313         'Position',[35 55 120 22]);
314
315 tra_pop_id = uicontrol(param_panel_id,...
316         'Style','popupmenu',...
317         'String',{'tansig','logsig'},...
318         'Value',1,...
319         'HorizontalAlignment','center',...
320         'Callback',@update_trfunc,...
321         'Position',[185 60 120 20]);
322
323 btn_diagram = uicontrol(main_id,...
324         'Position',[50 30 90 25],...
325         'String','Diagram',...
326         'Callback',@diagram_Callback);
327 set(btn_diagram,'Visible','off');
328

```

```

329 %%train wondow content
330
331 tr_panel_id = uipanel(...
332     'Title','Train',...
333     'FontSize',12,...
334     'BackgroundColor',bgcolor,...
335     'Units','pixels',...
336     'Position',[50 300 700 200]);
337 set(tr_panel_id,'Visible','off');
338
339 train_text_id = uicontrol(tr_panel_id,...
340     'Style','text',...
341     'String','Train an ordinal neural network with the
342         ORNNet algorithm:',...
343     'BackgroundColor',bgcolor,...
344     'HorizontalAlignment','left',...
345     'Position',[40 120 600 30]);
346
347 imgs = imread(fullfile('img','train.png'),'BackgroundColor',bgbtncolor)
348 ;
349 btn_train = uicontrol(tr_panel_id,...
350     'Position',[300 50 100 50],...
351     'CData',imgs,...
352     'Callback',@train_Callback);
353
354 res_panel_id = uipanel(...
355     'Title','Test results',...
356     'FontSize',12,...
357     'BackgroundColor',bgcolor,...
358     'Units','pixels',...
359     'Position',[50 80 700 200]);
360 set(res_panel_id,'Visible','off');
361
362 res_table_id = uitable(res_panel_id,...
363     'ColumnName',{'Dataset','CCR','MAE','NH'},...
364     'ColumnWidth',{140},...
365     'Position',[50 50 600 100]);
366
367 %% export window content
368
369 save_panel_id = uipanel(...
370     'Title','Save Results',...
371     'FontSize',12,...
372     'BackgroundColor',bgcolor,...
373     'Units','pixels',...
374     'Position',[50 80 700 420]);
375 set(save_panel_id,'Visible','off');

```

```

374
375     savn_text_id = uicontrol(save_panel_id ,...
376                             'Style','text' ,...
377                             'String','Save network to MATLAB network object named
378                             ':' ,...
379                             'BackgroundColor',bgcolor ,...
380                             'HorizontalAlignment','left' ,...
381                             'Position',[50 250 400 22]);
382
383     savn_edit_id = uicontrol(save_panel_id ,...
384                             'Style','edit' ,...
385                             'String','net' ,...
386                             'BackgroundColor','white' ,...
387                             'Position',[500 250 120 22]);
388
389     savo_text_id = uicontrol(save_panel_id ,...
390                             'Style','text' ,...
391                             'String','Save outputs to MATLAB matrix named:' ,...
392                             'BackgroundColor',bgcolor ,...
393                             'HorizontalAlignment','left' ,...
394                             'Position',[50 250-50 400 22]);
395
396     savo_edit_id = uicontrol(save_panel_id ,...
397                             'Style','edit' ,...
398                             'String','outputs' ,...
399                             'BackgroundColor','white' ,...
400                             'Position',[500 250-50 120 22]);
401
402     savr_text_id = uicontrol(save_panel_id ,...
403                             'Style','text' ,...
404                             'String','Save results to MATLAB object named:' ,...
405                             'BackgroundColor',bgcolor ,...
406                             'HorizontalAlignment','left' ,...
407                             'Position',[50 250-100 400 22]);
408
409     savr_edit_id = uicontrol(save_panel_id ,...
410                             'Style','edit' ,...
411                             'String','results' ,...
412                             'BackgroundColor','white' ,...
413                             'Position',[500 250-100 120 22]);
414
415     imgs = imread(fullfile('img','save.png'),'BackgroundColor',bgbtncolor);
416     btn_sav = uicontrol(save_panel_id ,...
417                         'Position',[500 30 120 25] ,...
418                         'CData',imgs ,...
419                         'Callback',@save_Callback);

```

```

420 %% functions
421
422 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
423 %function to get the next interface
424 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
425 function next_Callback(hObject,eventdata)
426     switch paso
427     case 0 %data interface
428         % header
429         set(txt_title,'HorizontalAlignment','left','String','Select Data'
430             );
431
432         set(logo_left_id,'UserData',imshow(''));
433         set(logo_right_id,'UserData',imshow(''));
434
435         imgs = imread(fullfile(matlabroot,'toolbox','nnet','nnresource','
436             icons',...
437             'import.png'),'BackgroundColor',bgcolor);
438         logo_left_id = axes(...
439             'Parent',main_id,...
440             'Units','pixels',...
441             'Position',[50 525 50 50]);
442         set(logo_left_id,'UserData',imshow(imgs));
443
444         %objects deactivation
445         set(npanel_id,'Visible','off');
446
447         %objects activation
448         set(train_panel_id,'Visible','on');
449         set(test_panel_id,'Visible','on');
450
451         %objects change
452         set(ipanel_id,'Position',[405 80 345 420]);
453
454         text_aux = sprintf(['\nSelect popup menu if it has content, else
455             push open '...
456             'button to use a dataset from a file.']);
457         set(itxt_text,'String',text_aux);
458
459         set(btn_back,'Visible','on');
460
461         %prevision of next interface
462         update_popup_data();
463
464         paso = paso+1;

```

```

462 case 1 %net interface
463     if ~isempty(evalin('base','who('-regexp','train.')')) &&
464         ...
465         ~isempty(evalin('base','who('-regexp','test.')'))
466
467     % header
468     set(txt_title,'HorizontalAlignment','left','String','Network
        Configuration');
469
470     imgs = imread(fullfile(matlabroot,'toolbox','nnet','nnresource'
        , 'nnet6',...
471         'icon','general','brain_left','48.png'),'BackgroundColor',
        bgcolor);
472     set(logo_left_id,'UserData',imshow(imgs));
473
474     % objects deactivation
475     set(train_panel_id,'Visible','off');
476     set(test_panel_id,'Visible','off');
477
478     % objects activation
479     set(param_panel_id,'Visible','on');
480
481     % objects change
482     text_net = sprintf(['\n'...
483         'Ordinal Neural Network Configuration:\n\nNow you
        have to '...
484         'configure the architecture of the neural network.
        If you want to '...
485         'perform a K-fold cross-validation to adjust the
        number of hidden '...
486         'neurons, you must select the option "Yes" in the
        popup menu. '...
487         'A K-fold cross-validation will be performed over
        the training set, and the number '...
488         'of neurons with the lowest MAE error will be
        selected as the '...
489         'optimum one.\n\n'...
490         'If you don't want to realize a K-fold, you must
        select the '...
491         'option "No". You will have to specify manually the
        number '...
492         'of neurons.\nFinally, the transfer function (
        tansig or logsig) '...
493         'for the hidden layer has to be chosen in both
        cases.']);
494     set(itxt_text,'String',text_net);

```

```

495         update_kfold();
496
497     paso = paso+1;
498 else
499     errordlg('No data charged. Try again.','Error','modal');
500 end
501
502
503 case 2 %train interface
504     if method_selected == true
505         %check if method is selected
506         string_list = get(kfold_pop_id,'String');
507         selected_string = string_list{get(kfold_pop_id,'Value')};
508
509         if strcmp(selected_string,'No')
510             %creating the neural network
511             guidata.net = newoff(guidata.trainInputs, guidata.trainTargets,
512                                 , guidata.NHO, guidata.trfunc);
513         elseif strcmp(selected_string,'Yes') && kfold_pushed
514             %creating the neural network
515             guidata.net = newoff(guidata.trainInputs, guidata.trainTargets,
516                                 , guidata.NHO, guidata.trfunc);
517         else
518             errordlg('Button K-fold no pushed. Try again.','Error','modal');
519         end
520         return
521     end
522
523     % header
524     set(txt_title,'HorizontalAlignment','left','String','Train Network');
525
526     imgs = imread(fullfile(matlabroot,'toolbox','nnet','nnresource',
527                             'nnet6',...
528                             'icon','general','data_division','48.png'),'BackgroundColor',
529                             bgcolor);
530     set(logo_left_id,'UserData',imshow(imgs));
531
532     %objects deactivation
533     set(param_panel_id,'Visible','off');
534     set(ipanel_id,'Visible','off');
535
536     set(btn_diagram,'Visible','off');
537
538     %objects activation
539     set(tr_panel_id,'Visible','on');
540     set(res_panel_id,'Visible','on');

```

```

536
537         set(btn_diagram, 'Visible', 'on');
538
539         paso = paso+1;
540         method_selected = false;
541     else
542         errordlg('No method selected. Please select one.', 'Error', '
                    modal');
543     end
544
545 case 3 %export interface
546     if train_pushed == true
547         %header
548         set(txt_title, 'HorizontalAlignment', 'left', 'String', 'Save
                    Results');
549
550         imgs = imread(fullfile(matlabroot, 'toolbox', 'nnet', 'nnresource',
                    'icons', ...
551                        'export.png'), 'BackgroundColor', bgcolor);
552         set(logo_left_id, 'UserData', imshow(imgs));
553
554         %objects deactivation
555         set(tr_panel_id, 'Visible', 'off');
556         set(res_panel_id, 'Visible', 'off');
557         set(btn_next, 'Visible', 'off');
558
559         %objects activation
560         set(save_panel_id, 'Visible', 'on');
561
562         %objects change
563         set(btn_back, 'Position', [650-110 30 100 25]);
564         imgs = imread(fullfile('img', 'finish.png'), 'BackgroundColor',
                    bgbtncolor);
565         set(btn_close, 'CData', imgs);
566
567         paso = paso+1;
568     else
569         errordlg('Push Train button to obtain the results. Try again.',
                    'Error', 'modal');
570     end
571 end
572 end
573
574 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
575 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
576 %function to get the previous interface

```

```

576 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
577 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
578 function back_Callback(hObject,eventdata)
579     switch paso
580     case 1 %main interface
581         %header
582         set(txt_title,'HorizontalAlignment','center','String','Welcome to
583             the Ordinal Neural Network Tool');
584
585         set(logo_left_id,'UserData',imshow(uco));
586         logo_right_id = axes(...
587             'Parent',main_id,...
588             'Units','pixels',...
589             'Position',[700 525 50 50]);
590         set(logo_right_id,'UserData',imshow(inf));
591
592         %objects activation
593         set(npanel_id,'Visible','on');
594
595         set(txt_title,'Visible','on');
596
597         %objects deactivation
598         set(train_panel_id,'Visible','off');
599         set(test_panel_id,'Visible','off');
600
601         set(btn_back,'Visible','off');
602
603         %objects change
604         set(ipanel_id,'Position',[50 80 345 420]);
605         set(itxt_text,'String',text_prin);
606
607         paso = paso-1;
608
609     case 2 %data interface
610         %header
611         set(txt_title,'HorizontalAlignment','left','String','Select Data'
612             );
613
614         imgs = imread(fullfile(matlabroot, 'toolbox','nnet','nnresource','
615             icons',...
616             'import.png'),'BackgroundColor',bgcolor);
617         set(logo_left_id,'UserData',imshow(imgs));
618
619         %objects activation
620         set(train_panel_id,'Visible','on');
621         set(test_panel_id,'Visible','on');
622

```



```

619         % objects deactivation
620         set(param_panel_id, 'Visible', 'off');
621
622         % objects change
623         set(itxt_text, 'String', text_data);
624
625         paso = paso - 1;
626
627     case 3 % net interface
628         % header
629         set(txt_title, 'HorizontalAlignment', 'left', 'String', 'Network
        Configuration');
630
631         imgs = imread(fullfile(matlabroot, 'toolbox', 'nnet', 'nnresource', '
        nnet6', ...
632             'icon', 'general', 'brain_left', '48.png'), 'BackgroundColor',
            bgcolor);
633         set(logo_left_id, 'UserData', imshow(imgs));
634
635         % objects activation
636         set(param_panel_id, 'Visible', 'on');
637         set(ipanel_id, 'Visible', 'on');
638
639         set(btn_diagram, 'Visible', 'off');
640
641         % objects deactivation
642         set(tr_panel_id, 'Visible', 'off');
643         set(res_panel_id, 'Visible', 'off');
644
645         paso = paso - 1;
646         method_selected = true;
647
648     case 4 % interfaz train
649         % header
650         set(txt_title, 'HorizontalAlignment', 'left', 'String', 'Train
        Network');
651
652         imgs = imread(fullfile(matlabroot, 'toolbox', 'nnet', 'nnresource', '
        nnet6', ...
653             'icon', 'general', 'data_division', '48.png'), 'BackgroundColor',
            bgcolor);
654         set(logo_left_id, 'UserData', imshow(imgs));
655
656         % objects activation
657         set(tr_panel_id, 'Visible', 'on');
658         set(res_panel_id, 'Visible', 'on');
659         set(btn_next, 'Visible', 'on');

```

```

660         set(btn_diagram, 'Visible', 'on');
661
662         % objects deactivation
663         set(save_panel_id, 'Visible', 'off');
664
665         % objects change
666         set(btn_back, 'Position', [650-220 30 100 25]);
667         imgs = imread(fullfile('img', 'cancel.png'), 'BackgroundColor',
668             bgbtncolor);
669         set(btn_close, 'CData', imgs);
670
671         paso = paso-1;
672     end
673 end
674
675 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
676 %function to charge the datasets
677 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
678 function open_Callback(hObject, eventdata)
679     [filename, pathname] = uigetfile({'*.dat', 'DAT-files (*.dat)'; '*..*',
680         'All Files (*.*)'}, ...
681         'Select File to Open');
682     if filename ~= 0
683         if isempty(findstr('.dat', lower(filename)))
684             errordlg('File extension invalid. Data file extension is *.dat.',
685                 'Error', 'modal');
686             return
687         else
688             datpath = fullfile(pathname, filename);
689             return
690         end
691     else
692         warndlg('No file selected. Please select a correct data file.');
```

```

701 data = evalin('base','data');
702 v = sscanf(textdata{1}, '%d %d %d');
703
704 % put parameters correctly
705 name = strtok(filename, '-');
706 C = v(2);
707 O = v(3);
708
709 % converting data
710 convdata(name, data, C, O);
711
712 % check datasets size
713 if ~isempty(evalin('base','who('-regexp','train.')')) && ~
    isempty(evalin('base','who('-regexp','test.')'))
714     if (evalin('base','size(trainInputs,1)') ~= evalin('base','size(
        testInputs,1)')) || ...
715         (evalin('base','size(trainTargets,1)') ~= evalin('base','size(
            testTargets,1)'))
716         errordlg('Fail in size of datasets. Please select a correct
            dataset.','Error','modal');
717     return
718 end
719 end
720
721 update_popup_data();
722 end
723
724 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
725 % function to show the neural network diagram
726 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
727 function diagram_Callback(hObject,eventdata)
728     persistent JAVA_TOOLS;
729     if isempty(JAVA_TOOLS)
730         JAVA_TOOLS = javaObjectEDT('com.mathworks.toolbox.nnet.matlab.
            nnTools');
731         mroot = matlabroot;
732         nroot = nnetroot;
733         sLicense = ~isempty(ver('simulink')) && license('test','simulink');
734         JAVA_TOOLS.initialize(mroot,nroot,sLicense);
735     end
736
737     diagram = JAVA_TOOLS.newDiagram;
738
739     net = guidata.net;
740

```

```

741 inputs = cell(1,net.numInputs);
742 for i=1:net.numInputs
743     inputs{i} = diagram.newInput;
744 end
745 layers = cell(1,net.numLayers);
746 for i=1:net.numLayers
747     layers{i} = diagram.newLayer;
748     if ~isfield(net.layers{i},'name'), layerName = net.layers{i}.name;
749     else layerName = 'Layer'; end
750     layers{i}.layerProperties.title.set(layerName);
751     if net.biasConnect(i)
752         layers{i}.layerProperties.hasBias.set(true);
753     end
754 end
755 outputs = cell(1,net.numOutputs);
756 for i=1:net.numOutputs
757     outputs{i} = diagram.newOutput;
758 end
759 weightGroups = cell(1,net.numLayers);
760 numWeights = zeros(1,net.numLayers);
761 outputIndex = 1;
762 for i=1:net.numLayers
763     for j=1:net.numInputs
764         if net.inputConnect(i,j)
765             weightGroup = layers{i}.newWeightGroup;
766             weightGroups{i} = [weightGroups{i} {weightGroup}];
767             diagram.newInputToLayerConnection(i-1,j-1,numWeights(i));
768             numWeights(i) = numWeights(i) + 1;
769         end
770     end
771     for j=1:net.numLayers
772         jTransferFunction = nnjava('get-java-transfer-function',net.
773             layers{i}.transferFcn);
774         layers{i}.layerProperties.transferFunction.set(jTransferFunction)
775         ;
776         if net.layerConnect(i,j)
777             weightGroup = layers{i}.newWeightGroup;
778             weightGroups{i} = [weightGroups{i} {weightGroup}];
779             diagram.newLayerToLayerConnection(i-1,j-1,numWeights(i));
780             numWeights(i) = numWeights(i) + 1;
781         end
782     end
783     if net.outputConnect(i)
784         diagram.newLayerToOutputConnection(outputIndex-1,i-1);
785         outputIndex = outputIndex + 1;
786     end
787 end

```

```

786     diagram.layoutChildren;
787
788     JAVA_TOOLS.newView( diagram );
789 end
790
791 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
792 %function to update the values of popup menu in data interface
793 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
794 function update_popup_data()
795     %show variable names in popup menu
796     str_train = evalin( 'base', 'who( '-regexp', 'train.' )' );
797     if ~isempty( str_train )
798         set( traini_pop_id, 'String', str_train );
799         set( traint_pop_id, 'String', str_train, 'Value', 2 );
800     end
801
802     str_test = evalin( 'base', 'who( '-regexp', 'test.' )' );
803     if ~isempty( str_test )
804         set( testi_pop_id, 'String', str_test );
805         set( testt_pop_id, 'String', str_test, 'Value', 2 );
806     end
807
808     %show data information
809     if ~isempty( str_train ) && ~isempty( str_test )
810         guidata.trainInputs = evalin( 'base', 'trainInputs' );
811         guidata.trainTargets = evalin( 'base', 'trainTargets' );
812         guidata.testInputs = evalin( 'base', 'testInputs' );
813         guidata.testTargets = evalin( 'base', 'testTargets' );
814
815         %transform original train targets to final train targets
816         guidata.trainTargetsNew = transdata( guidata.trainTargets );
817
818         text_data = sprintf( ...
819             [ '\n' ...
820               'These are the size of train and test data of the
821                 selected dataset.\n\n' ...
822               '  TrainInputs:   %d x %d\n' ...
823               '  TrainTargets:  %d x %d\n' ...
824               '\n  TestInputs:   %d x %d\n' ...
825               '  TestTargets:   %d x %d' ], ...
826             size( guidata.trainInputs, 1 ), size( guidata.trainInputs,
827               2 ), ...
828             size( guidata.trainTargets, 1 ), size( guidata.
829               trainTargets, 2 ), ...

```

```

827         size(guidata.testInputs,1),size(guidata.testInputs,2)
828         ,...
829         size(guidata.testTargets,1),size(guidata.testTargets
830         ,2));
831     set(itxt_text,'String',text_data);
832 end
833 end
834 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
835 %function to get the kfold popup menu value
836 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
837 function update_kfold(hObject,eventdata)
838     string_list = get(kfold_pop_id,'String');
839     selected_string = string_list{get(kfold_pop_id,'Value')};
840
841     switch selected_string
842     case 'Yes'
843         method_selected = true;
844
845         set(neu_text_id,'Enable','off');
846         set(neu_edit_id,'Enable','off');
847
848         set(k_text_id,'Enable','on');
849         set(k_edit_id,'Enable','on');
850
851         set(btn_kfold,'Enable','on');
852
853         set(tra_text_id,'Enable','on');
854         set(tra_pop_id,'Enable','on');
855     case 'No'
856         method_selected = true;
857
858         set(neu_text_id,'Enable','on');
859         set(neu_edit_id,'Enable','on');
860
861         set(k_text_id,'Enable','off');
862         set(k_edit_id,'Enable','off');
863
864         set(tra_text_id,'Enable','on');
865         set(tra_pop_id,'Enable','on');
866
867         %update the values
868         update_nh();
869         update_trfunc();
870     otherwise

```

[illegible]

```

910 function update_nh(hObject,eventdata)
911     guidata.NHO = str2num(get(neu_edit_id,'String'));
912 end
913
914 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
915 %function to update the value of transfer function
916 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
917 function update_trfunc(hObject,eventdata)
918     string_list = get(tra_pop_id,'String');
919     guidata.trfunc = string_list{get(tra_pop_id,'Value')};
920 end
921
922 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
923 %train and simulation function
924 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
925 function train_Callback(hObject,eventdata)
926     train_pushed = true;
927
928     %training the net
929     net = otrain(guidata.net, guidata.trainInputs, guidata.trainTargetsNew)
930     ;
931
932     %simulating the net to obtain the outputs
933     guidata.testOutputs = osim(net, guidata.testInputs);
934
935     %calculating and plotting confussion matrix
936     [c,cm,ind,per] = confusion(guidata.testTargets, guidata.testOutputs);
937     plotconfusion(guidata.testTargets, guidata.testOutputs);
938
939     %calculating CCR and MAE
940     guidata.ccr = ccrcalc(cm, size(guidata.testInputs,2));
941     guidata.mae = maecalc(cm, size(guidata.testInputs,2));
942
943     %show results in table
944     set(res_table_id,'Data',{guidata.dataset_name, guidata.ccr, guidata.mae
945         , guidata.NHO});
946
947     %objects change
948     imgs = imread(fullfile('img','retrain.png'),'BackgroundColor',
949         bgbtncolor);
950     set(btn_train,'CData',imgs);
951 end
952
953

```



```

950  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
951  % function to export the results
952  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
953  function save_Callback(hObject,eventdata)
954      str = get(savn_edit_id,'String');
955      var = genvarname(str);
956      assignin('base',var,guidata.net);
957
958      str = get(savo_edit_id,'String');
959      var = genvarname(str);
960      assignin('base',var,guidata.testOutputs);
961
962      str = get(savr_edit_id,'String');
963      var = genvarname(str);
964      struct.ccr = guidata.ccr;
965      struct.mae = guidata.mae;
966      struct.NHO = guidata.NHO;
967      assignin('base',var,struct);
968  end
969
970 end

```

Listing B.16: Archivo onntool.m