Varios algoritmos de I.A. para análisis de datos

Desarrollo de un software que combina varios algoritmos de inteligencia artificial para detectar el patrón de comportamiento en series de datos.

AUTOR | *Rafael Alberto Moreno Parra*

Varios algoritmos de I.A. para análisis de datos

Desarrollo de un software que combina varios algoritmos de inteligencia artificial para detectar el patrón de comportamiento en series de datos.

Varios algoritmos de I.A. para análisis de datos

Desarrollo de un software que combina varios algoritmos de inteligencia artificial para detectar el patrón de comportamiento en series de datos.

AUTOR

Rafael Alberto Moreno Parra

Universidad Libre

2020

Moreno Parra, Rafael Alberto
Varios algoritmos de I.A. para análisis de datos: desarrollo de un software que combina varios algoritmos de inteligencia artificial para detectar el patrón de comportamiento en series de datos / Rafael Alberto Moreno Parra. -- 1a ed. -- Cali: Universidad Libre, 2021.

p.

Incluye datos del autor. -- Incluye bibliografía.

ISBN 978-958-5545-93-9 (digital)

1. Algoritmos (Computadores) 2. Análisis de datos - Programas para computador I. Título

CDD: 005.13 ed. 23 CO-BoBN- a1070938

Varios algoritmos de I.A. para análisis de datos

Desarrollo de un software que combina varios algoritmos de inteligencia artificial para detectar el patrón de comportamiento en series de datos.

1a. Edición
Cali, Colombia - 2020
ISBN: 978-958-5545-93-9

Diagramación e impresión
Artes Gráficas del Valle S.A.S.
Tel. 333 2742

©Editorial
Sello Editorial Universidad Libre Seccional Cali
Universidad Libre de Cali
Cra. 109 No. 22 -00 Sede Valle del Lili
Teléfono: 524 0007 Ext- 1200, 1201, 1208
Cali – Colombia
2020

CONTENIDO

OTROS LIBROS DEL AUTOR

Libro 16: "C#. Programación Orientada a Objetos". Págs. 90. Libro y código fuente descargable en:
https://github.com/ramsoftware/C-Sharp-POO

Libro 15: "C#. Estructuras básicas de memoria." Págs. 60. Libro y código fuente descargable en:
https://github.com/ramsoftware/EstructuraBasicaMemoriaCSharp

Libro 14: "Iniciando en C#". Págs. 72. Libro y código fuente descargable en:
https://github.com/ramsoftware/C-Sharp-Iniciando

Libro 13: "Algoritmos Genéticos". Págs. 62. Libro y código fuente descargable en:
https://github.com/ramsoftware/LibroAlgoritmoGenetico2020

Libro 12: "Redes Neuronales. Segunda Edición". Págs. 108. Libro y código fuente descargable en:
https://github.com/ramsoftware/LibroRedNeuronal2020

Libro 11: "Capacitándose en JavaScript". Págs. 317. Libro y código fuente descargable en:
https://github.com/ramsoftware/JavaScript

Libro 10: "Desarrollo de aplicaciones para Android usando MIT App Inventor 2". Págs. 102. Ubicado en:
https://openlibra.com/es/book/desarrollo-de-aplicaciones-para-android-usando-mit-app-inventor-2

Libro 9: "Redes Neuronales. Parte 1". Págs. 90. Libro descargable en:
https://openlibra.com/es/book/redes-neuronales-parte-1

Libro 8: "Segunda parte de uso de algoritmos genéticos para la búsqueda de patrones". Págs. 303. En publicación por la Universidad Libre – Cali.

Libro 7: "Desarrollo de un evaluador de expresiones algebraicas. **Versión 2.0**. C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". Págs. 308. Ubicado en:

https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas-ii

Libro 6: "Un uso de algoritmos genéticos para la búsqueda de patrones". En publicación por la Universidad Libre – Cali.

Libro 5: Desarrollo fácil y paso a paso de aplicaciones para Android usando MIT App Inventor. En Colombia 2013. Págs. 104. Estado: Obsoleto (No hay enlace).

Libro 4: "Desarrollo de un evaluador de expresiones algebraicas. C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". Págs. 308. Ubicado en:

https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas

Libro 3: "Simulación: Conceptos y Programación". Págs. 81. Ubicado en:

https://openlibra.com/es/book/simulacion-conceptos-y-programacion

Libro 2: "Desarrollo de videojuegos en 2D con Java y Microsoft XNA". ISBN: 978-958-8630-45-8. Págs. 260. Ubicado en:

https://openlibra.com/es/book/desarrollo-de-juegos-en-2d-usando-java-y-microsoft-xna

Libro 1: "Desarrollo de gráficos para PC, Web y dispositivos móviles". Ed.: Artes Gráficas Del Valle Editores Impresores Ltda. ISBN: 978-958-8308-95-1 v. 1 págs. 317

Artículo: "Programación Genética: La regresión simbólica". Entramado ISSN: 1900-3803 ed.: Universidad Libre Seccional Cali. v.3 fasc.1 p. 76 - 85, 2007

Página web del autor y canal en Youtube

Investigación sobre Vida Artificial: http://darwin.50webs.com

Canal en Youtube: http://www.youtube.com/user/RafaelMorenoP (dedicado al desarrollo en C#)

Sitio en GitHub

El código fuente se puede descargar en: **[1]** https://github.com/ramsoftware/

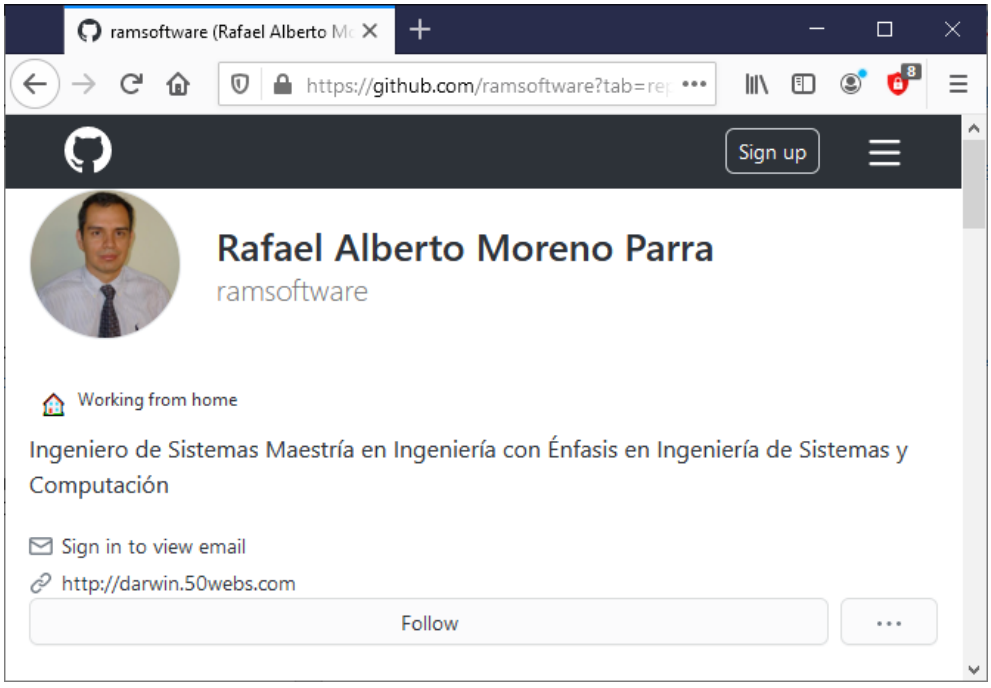


Ilustración 1: Sitio en GitHub

Licencia del software

Todo el software desarrollado aquí tiene licencia LGPL “Lesser General Public License” **[2]**

Marcas registradas

En este libro se hace uso de las siguientes tecnologías registradas:

Microsoft ® Windows ® **[3]**

Microsoft ® Visual Studio 2019 ® **[4]**

Microsoft ® Excel

Definición del problema

Las empresas y organizaciones poseen sistemas de información para la gestión de sus procesos. Una gran cantidad de información **[5]** es almacenada por esos sistemas y su análisis permitiría una mejor toma de decisiones **[6] [7] [8] [9]**. La complejidad de la realidad hace que muchos efectos no estén supeditados a tan solo una sola variable sino a varias, por lo tanto, sólo usar ecuaciones del tipo Y=F(X) donde "X" es la variable independiente y "Y" es la variable dependiente, a fenómenos de la vida real, no ofrecería resultados correctos. El comportamiento de cada proceso de una empresa, por lo general, es complejo **[10]**.

Luego es necesario trabajar con múltiples variables de entrada y encontrar la curva $Y=F(X_1, X_2, X_3, \ldots, X_n)$ que explique el patrón de datos recolectados. Sin embargo, encontrar un patrón con este nivel de complejidad es una tarea matemática muy difícil (usando métodos deterministas), por lo que se hace uso de métodos de aproximación. Uno de esos métodos del campo de la inteligencia artificial son las redes neuronales (por ejemplo, el perceptrón multicapa) que permite múltiples variables de entrada. Sin embargo, es un método aproximado, por lo tanto, no se debe confiar ciegamente en sólo ese método (que tiene sus puntos débiles como requerir una gran cantidad de datos para entrenar la red neuronal o que se sobreentrene y sea demasiado rígida y no pueda interpolar o extrapolar). Luego debe haber otros métodos que usando caminos distintos en el análisis de datos lleguen también a encontrar patrones. Si esos métodos llegan a resultados similares, habrá más confianza en la toma de decisiones.

Justificación

Haciendo una analogía con un examen médico, si ese examen señala un problema de salud ¿Se comenzaría con un tratamiento costoso, doloroso y con efectos secundarios indeseados en forma inmediata? La respuesta más justa sería no; lo más sensato es hacerse otros exámenes, inclusive repetir el primer examen en otro laboratorio clínico para verificar los resultados. Sucede igual con los análisis de los datos de una empresa: Si un tipo de red neuronal muestra un comportamiento que exija tomar difíciles decisiones en la empresa u organización ¿Se haría de forma inmediata? La respuesta sería NO. Hay que hacer otros análisis con otras herramientas distintas para verificar.

Otra analogía es el sitio VirusTotal **[11]** el cual permite a los usuarios subir un archivo sospechoso para que sea analizado por una gran cantidad de motores antivirus y anti-troyanos, obteniendo un mejor diagnóstico. No fiarse de un solo motor antivirus.

El proyecto que se plantea, buscar unir en una misma pantalla, varias técnicas distintas de análisis de datos englobados en el campo de la inteligencia artificial para mostrar al empresario si se han detectado patrones de comportamiento en los datos dados, con diversas variables de entrada. Estas técnicas se ejecutan localmente en el PC del usuario: redes neuronales (perceptrón multicapa), algoritmos genéticos (regresión simbólica), curvas de aproximación o un servicio ofrecido por terceros.

Objetivo General

Desarrollar una aplicación de software que reúna formas de análisis de datos en el campo de la inteligencia artificial (algoritmos genéticos, redes neuronales, curvas de aproximación y uno basado en un servicio de terceros) que permita al usuario encontrar patrones en una serie de datos con múltiples variables de entrada y una de salida.

Objetivos Específicos

1. Desarrollar un módulo parametrizable de procesamiento local basado en redes neuronales que encuentre patrones en una serie de datos con múltiples variables de entrada y una de salida.
2. Desarrollar un módulo parametrizable de procesamiento local basado en algoritmos genéticos que encuentre patrones en una serie de datos con múltiples variables de entrada y una de salida.
3. Desarrollar un módulo parametrizable de procesamiento local basado en curvas de aproximación que encuentre patrones en una serie de datos con múltiples variables de entrada y una de salida.
4. Desarrollar un módulo que facilite el acceso a procesamiento de un servicio de terceros que encuentre patrones en una serie de datos con múltiples variables de entrada y una de salida.
5. Implementar los módulos en una sola herramienta de software.
6. Implementar en esa herramienta la funcionalidad de interpolar o extrapolar.

Búsqueda de la mejor curva

Dada una serie de datos, usualmente históricos, el objetivo es hallar la mejor curva **[12] [13]**, es decir, la mejor ecuación que se identifique con el comportamiento de esa serie de datos para poder más adelante hacer predicciones del comportamiento de los datos (interpolar y extrapolar **[14]**).

Para simplificar este proceso, usualmente se inicia con una ecuación del tipo Y=F(X) donde X es la única variable independiente, en cambio, Y es la variable dependiente. Este tipo de ecuaciones se grafican en un plano cartesiano.

Por ejemplo:

X	Y
0	3,4
3	6,2
6	8
9	9,2
12	10,2
15	11,1
18	11,8
24	12,9
36	15,1
48	16,07
60	18,03
72	19,91
84	22
96	23,56

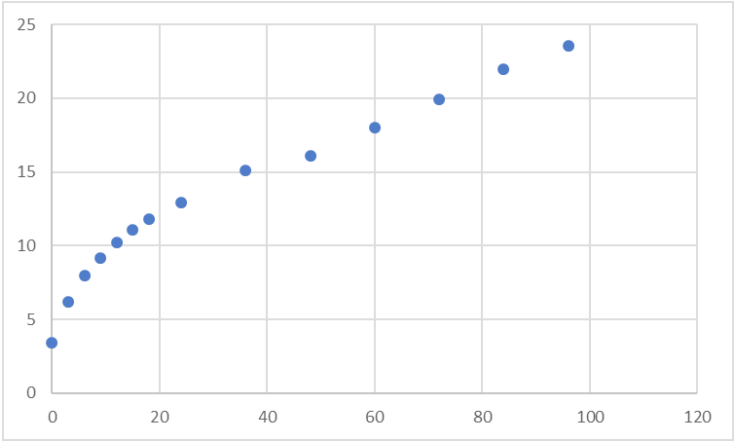


Ilustración 2: Gráfico resultante de unir los puntos de los datos X, Y

En el gráfico de la derecha se muestran los puntos que se unen con una línea.

El siguiente paso es encontrar la mejor curva que se acerque a ese comportamiento. Si se dispone de herramientas como Microsoft Excel es

dar clic con el botón derecho en la línea que une los puntos y seleccionar la opción de "Agregar Línea de Tendencia"

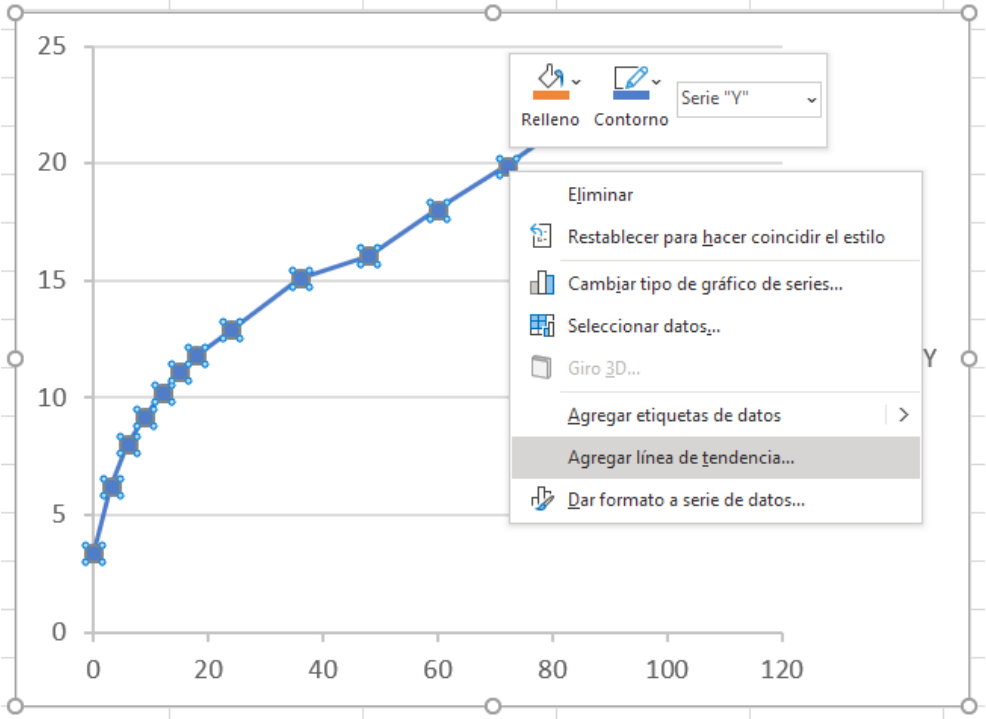


Ilustración 3: En Excel se da clic botón derecho para seleccionar "Agregar línea de tendencia"

En el menú resultante, a la derecha, aparecen varios tipos de curva para escoger: Exponencial, Lineal, Logarítmica, Polinómica, Potencial, Media Móvil. Para destacar están las opciones de "Presentar ecuación en el gráfico" y "Presentar el valor R cuadrado en el gráfico" **[15]**.

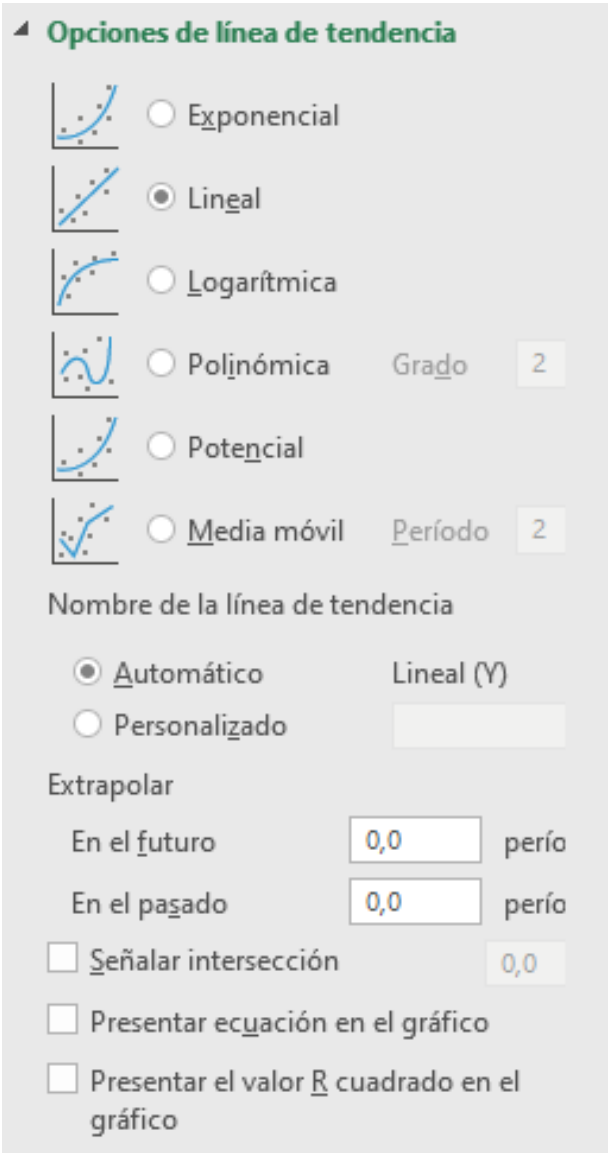


Ilustración 4: Opciones de línea de tendencia

Fuente: propia del autor

¿Cuál curva escoger? El criterio es que "el valor R cuadrado en el gráfico" esté lo más cercano a 1.

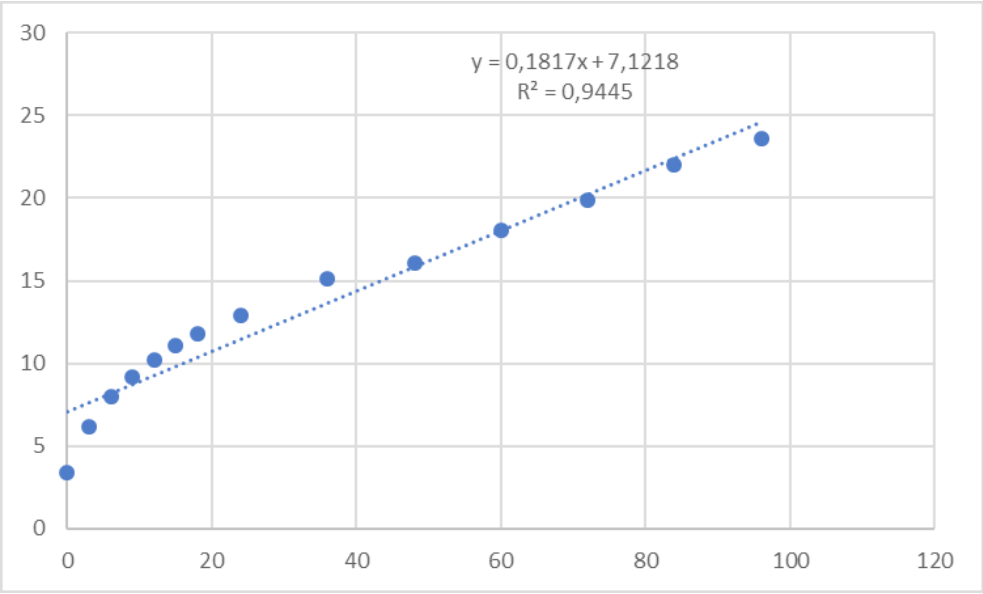


Ilustración 5: Gráfico lineal

Fuente: propia del autor

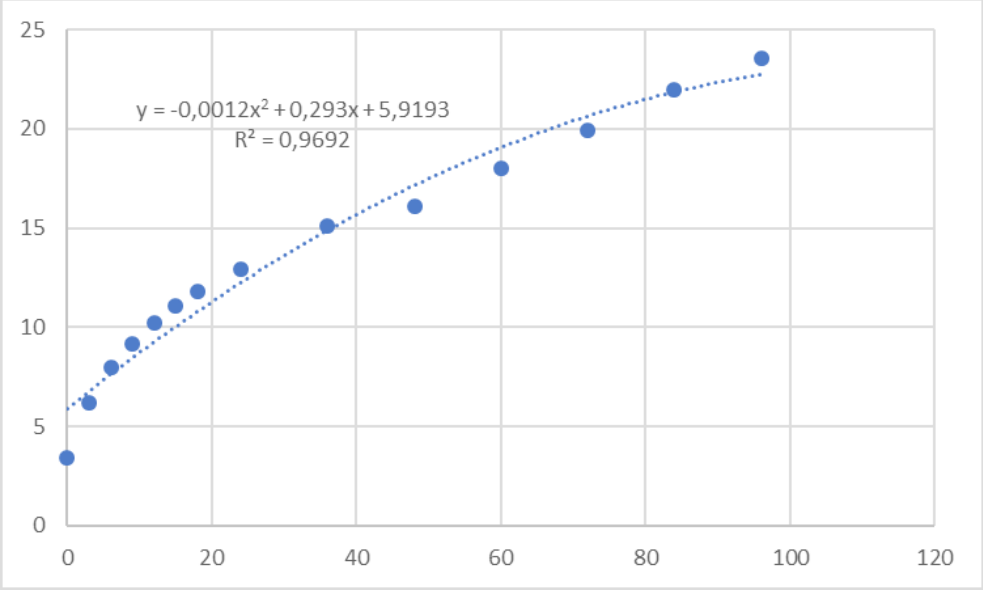


Ilustración 6: Gráfico polinómico de grado 2

Fuente: propia del autor

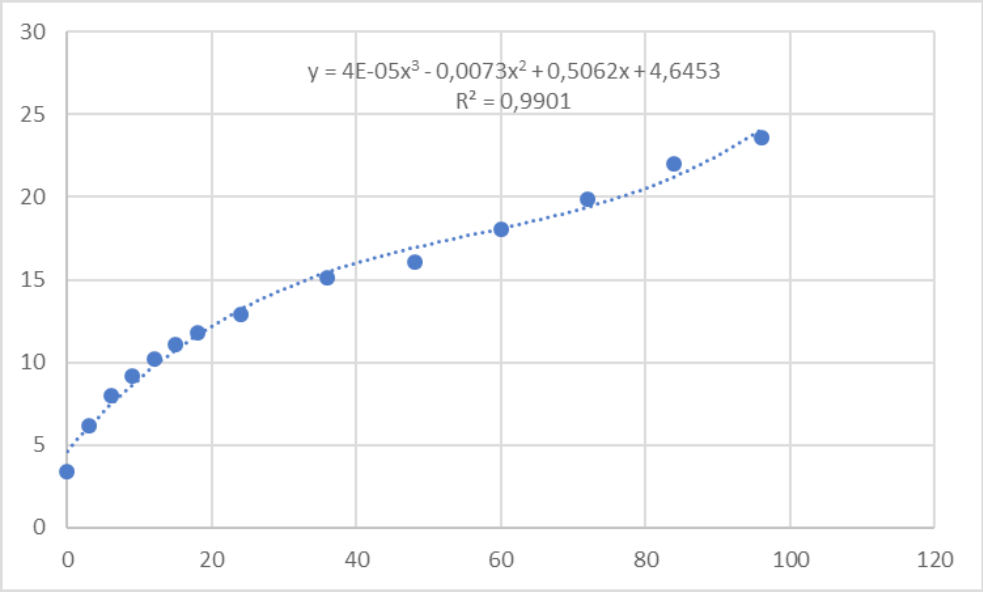


Ilustración 7: Gráfico polinómico de grado 3

Fuente: propia del autor

Sería probar todas las curvas y seleccionar la que tenga el R^2 más cercano a 1

Curva	R^2
$y = 7{,}2752e^{0{,}0144x}$	0,7517
$y = 0{,}1817x + 7{,}1218$	0,9445
$y = -0{,}0012x^2 + 0{,}293x + 5{,}9193$	0,9692
$y = 4E\text{-}05x^3 - 0{,}0073x^2 + 0{,}5062x + 4{,}6453$	0,9901
$y = -1E\text{-}06x^4 + 0{,}0002x^3 - 0{,}0187x^2 + 0{,}714x + 3{,}9184$	0,9981
$y = 1E\text{-}08x^5 - 4E\text{-}06x^4 + 0{,}0005x^3 - 0{,}0268x^2 + 0{,}8044x + 3{,}7314$	0,9987
$y = -5E\text{-}10x^6 + 2E\text{-}07x^5 - 2E\text{-}05x^4 + 0{,}0012x^3 - 0{,}0425x^2 + 0{,}9271x + 3{,}5682$	0,9992

Algunas ecuaciones como la logarítmica, en el caso del ejemplo, no se pueden hacer porque hay valores cero.

Observando la tabla, entonces la selección sería la ecuación de grado 6 porque tiene el valor más cercano a uno. Y si se pudiera llegar a polinomios de grados más altos habría mayor precisión. Finalmente, un polinomio de grado = número de datos – 1, sería el polinomio que acertaría a todos los puntos. Sin embargo, cuando acierta a todos los puntos, es muy probable que la curva no uniera suavemente los puntos, sino que existan enormes picos y profundos llaves entre los puntos por lo que no serviría ni para interpolar, ni extrapolar:

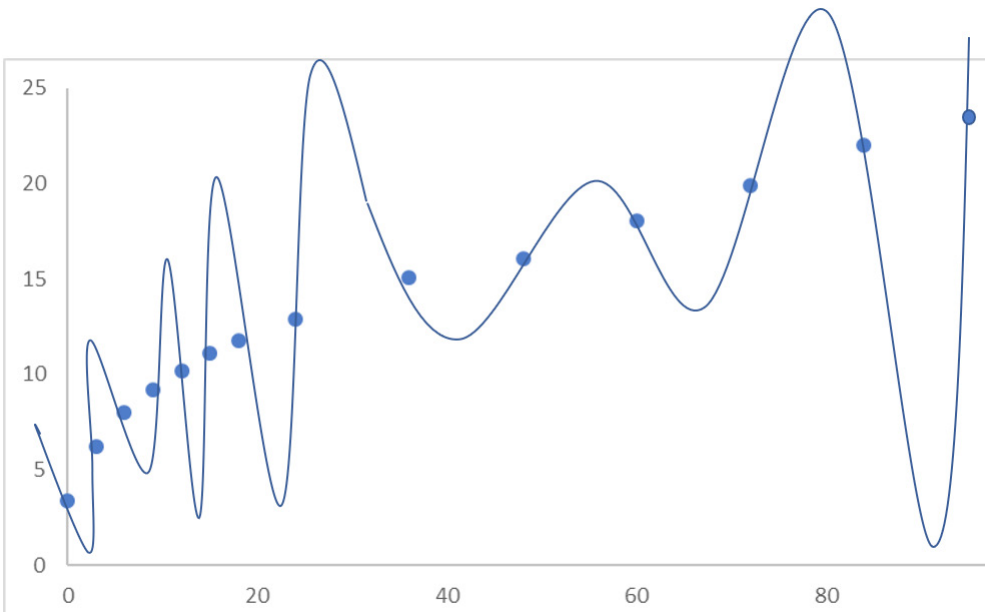


Ilustración 8: Una curva demasiado ajustada a los puntos, pero que realmente no muestra el comportamiento

Fuente: propia del autor

El sobreajuste

Cuando una curva muestra un ajuste tan preciso como el del gráfico anterior pero lleno de picos altos y profundos valles, estamos frente a un problema como el sobreajuste. Aunque matemáticamente se obtiene un

R^2 igual a 1, no serviría para hacer interpolaciones, muchísimo menos extrapolaciones. Este problema lo tienen métodos como las redes neuronales y los algoritmos genéticos, se le conoce como "overfitting" **[16] [17] [18]**.

Hay técnicas para minimizar ese problema **[19]**. Esta investigación, utiliza la combinación de redes neuronales, algoritmos genéticos, curvas de aproximación y regresión lineal de tal manera que los resultados de cada uno pueden compararse y el usuario del software puede deducir que los valores obtenidos en operaciones como interpolación son realistas.

Sobre la extrapolación

En el ejemplo anterior, los valores de X estaban entre 0 y 96, la curva se deduce con esos valores, pero ¿es posible, en ese ejemplo, saber los valores de Y en un rango X por debajo de 0 o por encima de 96? Este se conoce como extrapolación y es una operación en la cual debe haber mucho cuidado porque los valores ofrecidos por las curvas suelen ser muy dispares.

En el ejemplo, se aplican las curvas polinómicas de grado 1 hasta grado 6 y lo llevamos a un valor en X de 216, para ver qué valor Y se obtendría.

X	Y	Lineal	Grado 2	Grado 3	Grado 4	Grado 5	Grado 6
0	3,4	7,1218	5,9193	4,6453	3,9184	3,7314	3,5682
3	6,2	7,6669	6,7875	6,09928	5,897419	5,91657843	5,99782824
6	8	8,212	7,6341	7,42834	7,571104	7,69589376	7,83561187
9	9,2	8,7571	8,4591	8,63896	8,968939	9,13904649	9,22472408
12	10,2	9,3022	9,2625	9,73762	10,118464	10,3085443	10,2805534
15	11,1	9,8473	10,0443	10,7308	11,045275	11,2599938	11,0958797
18	11,8	10,3924	10,8045	11,62498	11,773024	12,0423917	11,7457875
24	12,9	11,4826	12,2601	13,14226	12,716224	13,2647222	12,7888533
36	15,1	13,663	14,9121	15,27394	13,038784	15,1711978	15,263524

X	Y	Lineal	Grado 2	Grado 3	Grado 4	Grado 5	Grado 6
48	16,07	15,8434	17,2185	16,54738	11,915584	17,2057757	21,5365784
60	18,03	18,0238	19,1793	17,3773	9,6784	19,4514	38,3862
72	19,91	20,2042	20,7945	18,17842	6,161344	21,1947523	77,7463716
84	22	22,3846	22,0641	19,36546	0,700864	21,2248502	157,841549
96	23,56	24,565	22,9881	21,35314	-7,864256	18,1316458	303,24638
216		46,369	13,2201	176,50354	-875,567936	-39,3304982	10035,5489

Fuente: propia del autor

Se observa que para X=216, los valores extrapolados de las distintas curvas son completamente disímiles. ¿Cuál acierta? Para este ejemplo, ninguna curva acertó. La razón de eso es que las dos variables (entrada y salida), son solo X y Y, pero en realidad, esos datos salieron de "Pesos y estatura del bebé, niño y niña" **[20]** y corresponde al peso en niños donde X son los meses y Y es el peso en kilogramos. El valor 216 correspondería cuando un ser humano varón cumple 18 años que debería estar entre 50,4kg y 76,2kg **[21]**. La curva más cercana fue la lineal, a pesar de tener un R^2 menor que de las otras curvas polinómicas.

Luego la extrapolación debe hacerse muy cerca de la frontera del rango X evaluado y además saber qué tipos de datos son los que se está trabajando.

Sobre la causalidad

En el ejemplo anterior, el valor X es el tiempo en meses y el valor Y es el peso en kilogramos. Se ha recogido el peso del individuo varón en cada mes cumplido. Luego se hace el análisis de datos. Esto se conoce como series temporales. Sin embargo, es importante saber que no es el tiempo en sí, el que causa que ese individuo varón tenga ese peso en particular. La causa o causas hay que buscarla en otros factores como son los genes (las distintas hormonas), la alimentación, el ambiente, etc. Luego un análisis de datos

debería enfocarse a encontrar la relación causal **[22]** de comportamiento entre unas variables y otras. Llegar a una ecuación del tipo:

$$Efecto = Función(Causa)$$

También hay que tener cuidado que correlación no implica causalidad **[23] [24] [25]**. Así dos variables tengan un comportamiento similar en una serie temporal, no significa que esas dos variables tengan alguna relación.

Cuando hay dos o más variables independientes

Hay eventos que se explican por la interrelación de dos o más variables independientes. Con solo dos variables independientes Z=F(X,Y) sería una representación en 3D, más complejo que una ecuación plana Y=F(X):

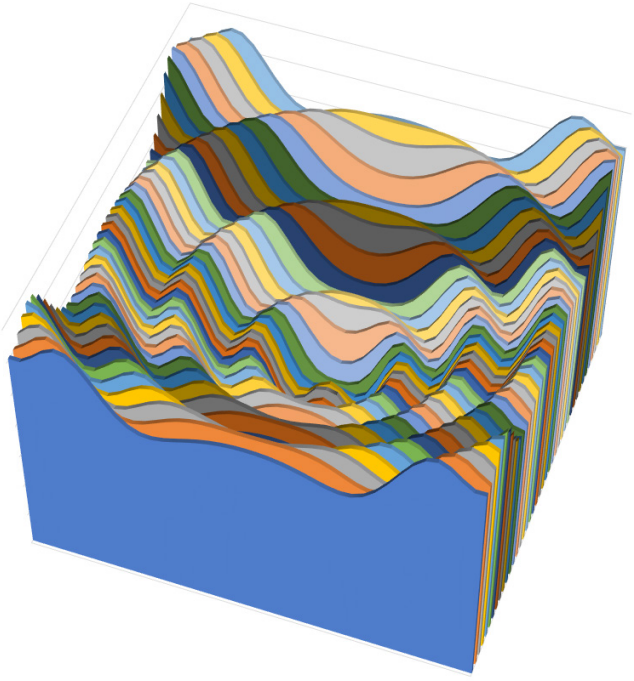


Ilustración 9: Ecuación en 3D

Fuente: propia del autor

¿Cuándo sean tres o más variables independientes? Se pierde la posibilidad de imaginarse el gráfico.

Y se requiere encontrar la curva que mejor ajuste al comportamiento de los datos pero que no caiga en el sobreajuste.

Encontrar un patrón para poder optimizar

Una de las razones para encontrar un patrón o ecuación que explique como una variable dependiente es el resultado de una expresión matemática de una, dos o más variables independientes, es el poder optimizar hallando el máximo o mínimo global.

Carga de datos

Archivos CSV

Los archivos CSV (Valores separados por comas) son una forma común de archivo para distribuir conjuntos de datos (también conocidos como "datasets"). Usualmente en Windows 10 tienen la siguiente representación:

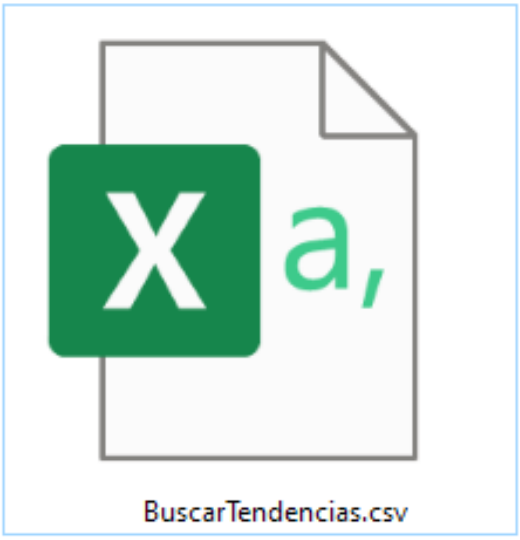


Ilustración 10: Icono representativo de archivos CSV

Son archivos planos, que pueden abrirse con un editor de texto, por ejemplo:

BuscarTendencias.csv

```
1 "valorA","valorB","valorC","valorD","valorE","valorF","valorG","valor
  H","valorI"
2 12,10,7,0.207911691,-0.77023591,4,1.192719444,65,93
3 20,26,13,0.342020143,-0.597672477,-276,1.24081419,-107,-107
4 28,40,20,0.469471563,-0.41347603,-816,1.235516006,-416,-416
5 42,46,24,0.669130606,-0.074014219,-444,1.363788977,176,224
```

Ilustración 11: Archivo CSV con 9 columnas de datos

Hay una línea inicial que es el encabezado de cada columna de datos, en la cual las palabras están separadas por comas y están encerradas con comillas dobles. Las siguientes líneas son los datos separados por comas y para los decimales se hace uso del punto. Para el software del proyecto, los datos deben ser numéricos, **no** se permiten datos alfanuméricos (cadenas o caracteres).

Carga de datos del CSV

En el software, se da clic en el menú Archivo y se escoge la opción "Abrir archivo de datos…"

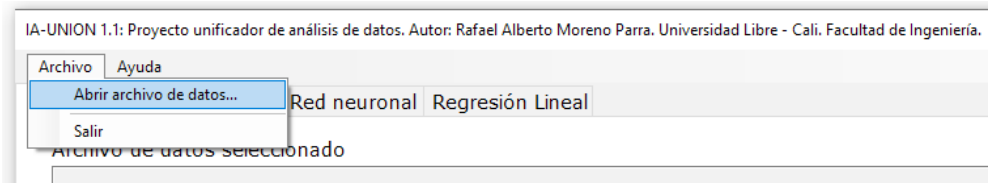


Ilustración 12: Dar clic en "Abrir archivo de datos..."

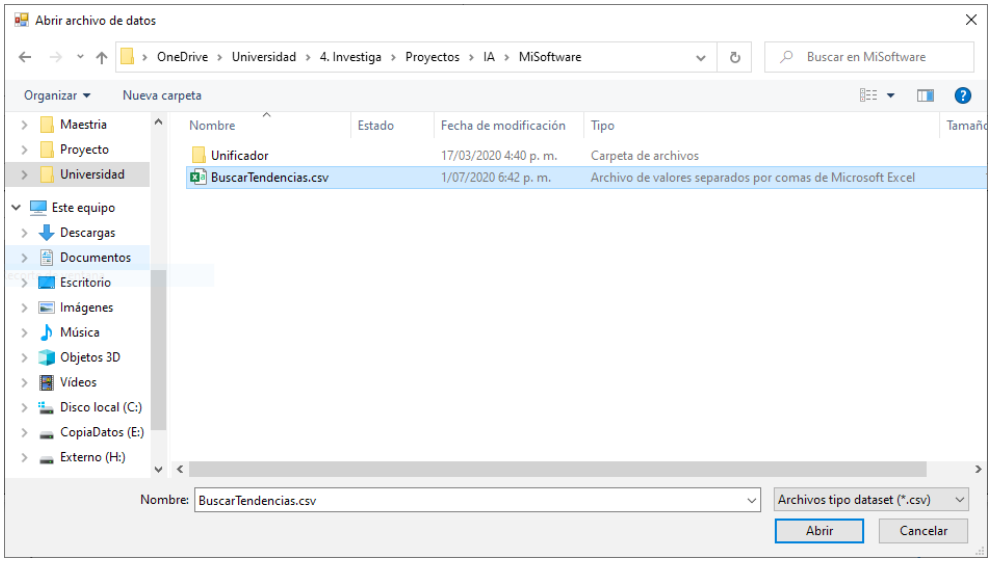


Ilustración 13: Seleccionando el archivo CSV

El software muestra los encabezados del archivo CSV pero **no** carga los datos aún. La razón es que un conjunto de datos puede ser muy extenso, que requiera una gran cantidad de memoria. Por ese motivo, es mejor sólo subir las columnas de datos que se van a analizar.

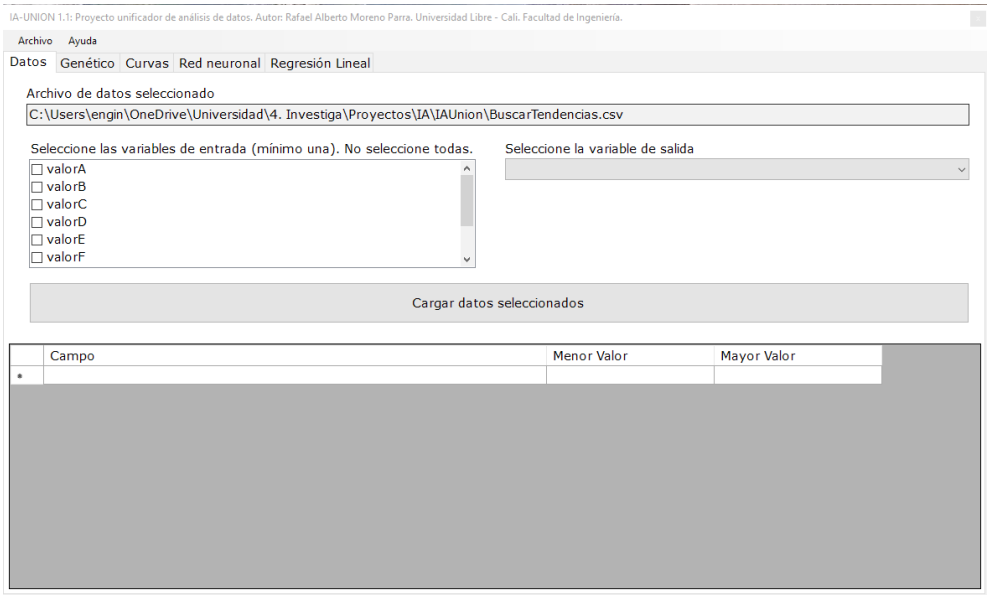


Ilustración 14: Se cargan sólo los encabezados de las columnas.

Seleccionando las columnas de entrada y la de salida

El software trabaja con la siguiente fórmula

Variable Salida = funcion (variable Entra 1, variable Entra 2, variable Entra N)

Es decir, uno o varias columnas serán las variables de entrada y sólo una columna será la variable de salida. El usuario debe escoger una o más columnas que serán las variables de entrada y una variable de salida que será distinta a las de entrada.

Después de seleccionar las variables de entrada y la única variable de salida el usuario debe presionar el botón "Cargar datos seleccionados":

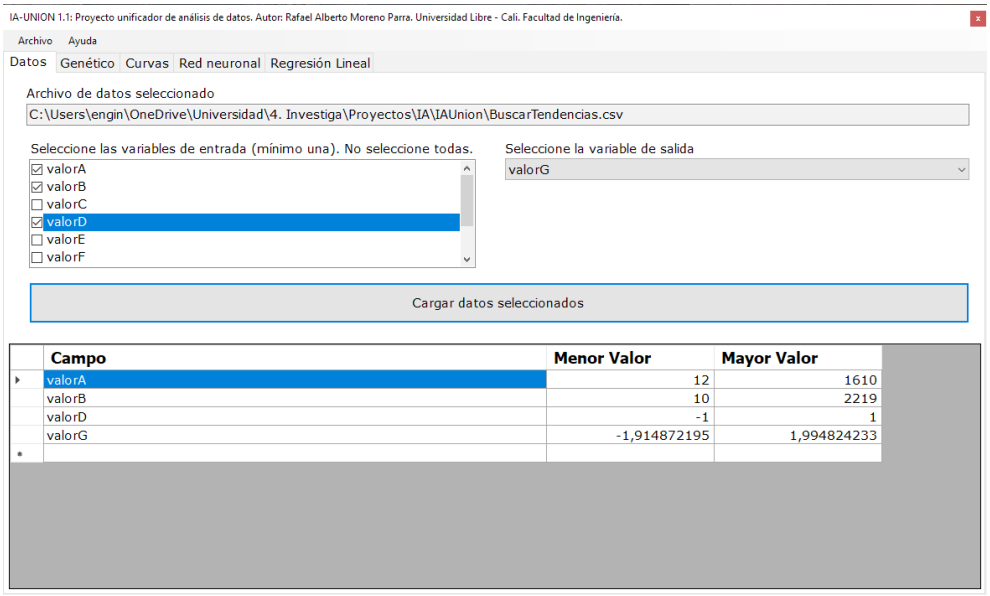


Ilustración 15: Datos de variables de entrada cargadas y datos de única variable de salida también cargados

Al cargar los datos, el software muestra un resumen del menor y mayor valor que encuentra de cada variable. Estos valores son importantes porque el software los utiliza para normalizar los datos cargados y, además, el usuario cuando quiera saber qué valor tendrá la variable de salida con ciertos valores de las variables de entrada, estos valores de entrada deben estar entre el máximo y mínimo para poder hacer una interpolación que es una operación con mayor probabilidad de ser más precisa que una extrapolación.

Implementación en C# de la carga de datos

Hay una clase llamada DatosArchivo.cs que tiene como función cargar los datos de un archivo CSV y ponerlos en una lista.

DatosArchivo.cs

```
/* Proyecto: Desarrollo de una herramienta de software unificadora de
análisis de datos para
 * la detección de patrones de comportamiento haciendo uso de varios métodos
de inteligencia
 * artificial.
 *
 * Autor: Rafael Alberto Moreno Parra
 * Correo: rafael-morenop@unilibre.edu.co
 * Semillero: Semillero en investigación, desarrollo y documentación de
herramientas para la construcción de sistemas de información en Web
 * Universidad Libre - Cali
 * Ingeniería
 *
 * Fecha: 2020
 * */
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;

namespace Unificador {
  //Clase que se encarga de leer los datos del archivo CSV
  class DatosArchivo {
    //Gestiona el error en la carga del archivo CSV
    public int ErrorCargaDatos { get; set; }

    //Los datos seleccionados para encontrar patrón de comportamiento
    public List<List<double>> ValorDatos;

    //Almacena los valores máximos y mínimos de los datos leídos
    public List<double> MayorValor;
    public List<double> MenorValor;

    //Constructor
    public DatosArchivo() {
      ValorDatos = new List<List<double>>();
      MayorValor = new List<double>();
      MenorValor = new List<double>();
    }

    //Retorna los nombres de las columnas
    public List<string> NombreDatos(string urlArchivo, char delimitador) {
      List<string> Nombres = new List<string>();

      //Empieza a leer el archivo
      ErrorCargaDatos = 0;
      var archivo = new StreamReader(urlArchivo);

      //Título de los datos
      string Titulo = archivo.ReadLine();

      //Primera línea de datos
      string posibleValor = archivo.ReadLine();
      double UnValor;
```

```
      //Genera la lista de títulos, siempre y cuando sean numéricos
      int totalCampos = NumeroCampos(Titulo, delimitador);
      for (int campo = 1; campo <= totalCampos; campo++) {
        string datotitulo = ExtraerCadena(Titulo, delimitador, campo);
        string posibleNumero = ExtraerCadena(posibleValor, delimitador, campo);
        datotitulo = datotitulo.Trim('"');
        if (Double.TryParse(posibleNumero, out UnValor))
          Nombres.Add(datotitulo);
        else {
          Nombres.Add("[Error: Dato no cuantitativo] " + datotitulo);
          ErrorCargaDatos = 1;
        }
      }

      return Nombres;
    }

    //Lee los valores de las columnas seleccionadas y los normaliza
    public void LeeDatosColumnas(string urlArchivo, char delimitador,
List<int> CamposLee) {
      //Empieza a leer el archivo
      var archivo = new StreamReader(urlArchivo);
      archivo.ReadLine(); //Pasa de la línea de título

      //Inicializa las listas
      ValorDatos.Clear();
      MayorValor.Clear();
      MenorValor.Clear();

      //Lee la linea de datos numéricos
      ErrorCargaDatos = 0; //Si todo sale bien, este valor no cambia
      string lineaDato;
      while ((lineaDato = archivo.ReadLine()) != null) {

        //Extrae sólo los campos requeridos
        List<double> valorReal = new List<double>();
        for (int num = 0; num < CamposLee.Count; num++) {
          double valor = ExtraerNumero(lineaDato, delimitador,
CamposLee[num]);
          valorReal.Add(valor);
        }

        //Guarda esos valores en la lista de valores
        ValorDatos.Add(valorReal);
      }

      //Normaliza los datos
      //Va de columna en columna de datos
      for (int columna = 0; columna < CamposLee.Count; columna++) {
        //Deduce el mínimo y máximo valor de esa columna en particular
        double minimo = ValorDatos[0][columna];
        double maximo = ValorDatos[0][columna];

        //Va de dato en dato de esa columna en particular
        //para deducir el menor y mayor valor
```

```
        for (int registro = 0; registro < ValorDatos.Count; registro++) {
          if (ValorDatos[registro][columna] > maximo) maximo =
ValorDatos[registro][columna];
          if (ValorDatos[registro][columna] < minimo) minimo =
ValorDatos[registro][columna];
        }

        //Almacena los valores máximos y mínimos
        MayorValor.Add(maximo);
        MenorValor.Add(minimo);

        //Normaliza los valores
        for (int registro = 0; registro < ValorDatos.Count; registro++) {
          ValorDatos[registro][columna] = (ValorDatos[registro][columna] -
minimo) / (maximo - minimo);
        }
      }
    }

    //Retorna dato normalizado dependiendo de la columna
    public double Normalizado(double valor, int columna) {
      return (valor - MenorValor[columna]) / (MayorValor[columna] -
MenorValor[columna]);
    }

    //Retorna dato desnormalizado dependiendo de la columna
    public double Desnormalizado(double valor, int columna) {
      return valor * (MayorValor[columna] - MenorValor[columna]) +
MenorValor[columna];
    }


    //Cuenta el número de datos por línea
    private int NumeroCampos(string linea, char delimitador) {
      int cuenta = 0;
      foreach (char caracter in linea)
        if (caracter == delimitador) cuenta++;
      return cuenta + 1;
    }

    //Dada una cadena con separaciones por algún delimitador
    //trae determinado ítem de tipo double
    private double ExtraerNumero(string linea, char delimitador, int
queItemTrae) {
      string numero = "";
      int itemTrae = 0;
      double valorSale;

      foreach (char caracter in linea) {
        if (caracter != delimitador)
          numero += caracter;
        else {
          itemTrae++;
          if (itemTrae == queItemTrae) {
            numero = numero.Trim();
            if (numero == "") return 0;
```

```
            if (double.TryParse(numero, NumberStyles.Any, CultureInfo.
InvariantCulture, out valorSale)) {
              return valorSale;
            }
            else {
              ErrorCargaDatos = 2; //Mala conversión
              return 0;
            }
          }
          numero = "";
        }
      }
      numero = numero.Trim();
      if (numero == "") return 0;
      if (double.TryParse(numero, NumberStyles.Any, CultureInfo.
InvariantCulture, out valorSale)) {
        return valorSale;
      }
      else {
        ErrorCargaDatos = 2; //Mala conversión
        return 0;
      }
    }

    //Dada una cadena con separaciones por algún delimitador
    //trae determinado ítem de tipo string
    private string ExtraerCadena(string linea, char delimitador, int
queItemTrae) {
      string texto = "";
      int itemTrae = 0;
      foreach (char caracter in linea) {
        if (caracter != delimitador)
          texto += caracter;
        else {
          itemTrae++;
          if (itemTrae == queItemTrae) {
            texto = texto.Trim();
            return texto;
          }
          texto = "";
        }
      }
      texto = texto.Trim();
      return texto;
    }
  }
}
```

Normalización de los datos

Una vez se han cargado los datos de las columnas seleccionadas por el usuario final, el software procede internamente a normalizarlos. Este proceso consiste en convertir esos datos a una escala para que estén en un rango de 0 a 1 ¿Por qué? El algoritmo de red neuronal requiere que los datos de entrada estén entre 0 y 1, además facilita mucho los cálculos para los algoritmos genéticos y curva de aproximación. Este es el proceso de normalización:

Se tienen los datos cargados y se busca el mínimo y máximo valor por columna

Valor A	Valor B	Valor C
1259	9	-57
2914	-62	-176
770	-287	-143
2431	-28	-174
1246	-48	-139
862	-91	-194
1150	748	-104
2195	-496	-184
1835	75	-116
813	289	-122
1501	493	-142
2250	-246	-131
1314	-446	-76
1393	333	-83
2890	-274	-147
1373	528	-99
2782	448	-84
2077	-428	-63

	Valor A	Valor B	Valor C
	1959	684	-155
	1660	658	-128
	2566	257	-182
	812	540	-77
	2060	-49	-56
	1198	329	-94
Mínimo	770	-496	-194
Máximo	2914	748	-56

Fuente: propia del autor

Se aplica la siguiente fórmula para normalizar

$$Valor\ Normalizado = \frac{Valor\ original - mínimo}{máximo - mínimo}$$

Luego se recalcula cada columna

Valor A	Valor B	Valor C
1259	9	-57
2914	-62	-176
770	-287	-143
2431	-28	-174
1246	-48	-139
862	-91	-194
1150	748	-104

Valor A normalizado	Valor B normalizado	Valor C normalizado
0,228078358	0,405948553	0,992753623
1	0,348874598	0,130434783
0	0,168006431	0,369565217
0,774720149	0,376205788	0,144927536
0,222014925	0,360128617	0,398550725
0,042910448	0,325562701	0
0,177238806	1	0,652173913

	Valor A	Valor B	Valor C
	2195	-496	-184
	1835	75	-116
	813	289	-122
	1501	493	-142
	2250	-246	-131
	1314	-446	-76
	1393	333	-83
	2890	-274	-147
	1373	528	-99
	2782	448	-84
	2077	-428	-63
	1959	684	-155
	1660	658	-128
	2566	257	-182
	812	540	-77
	2060	-49	-56
	1198	329	-94
Mínimo	770	-496	-194
Máximo	2914	748	-56

Valor A normalizado	Valor B normalizado	Valor C normalizado
0,664645522	0	0,072463768
0,496735075	0,459003215	0,565217391
0,02005597	0,631028939	0,52173913
0,340951493	0,795016077	0,376811594
0,690298507	0,20096463	0,456521739
0,253731343	0,040192926	0,855072464
0,290578358	0,666398714	0,804347826
0,98880597	0,178456592	0,34057971
0,28125	0,823151125	0,688405797
0,938432836	0,758842444	0,797101449
0,609608209	0,054662379	0,949275362
0,554570896	0,948553055	0,282608696
0,41511194	0,927652733	0,47826087
0,837686567	0,605305466	0,086956522
0,019589552	0,832797428	0,847826087
0,601679104	0,359324759	1
0,199626866	0,66318328	0,724637681

Fuente: propia del autor

El software almacena los valores máximos y mínimos de cada columna de datos para aplicar más adelante el proceso inverso de la normalización.

En la implementación en DatosArchivo.cs, al cargar los datos seleccionados por el usuario, estos son normalizados.

Algoritmo genético

Paso 1: Generar una población

Con los datos de las variables de entrada y la variable de salida normalizados, se procede a trabajar con el algoritmo genético para encontrar un patrón.

El algoritmo genético tiene una serie de pasos, el primero de ellos es generar una población de individuos, donde cada individuo es una ecuación.

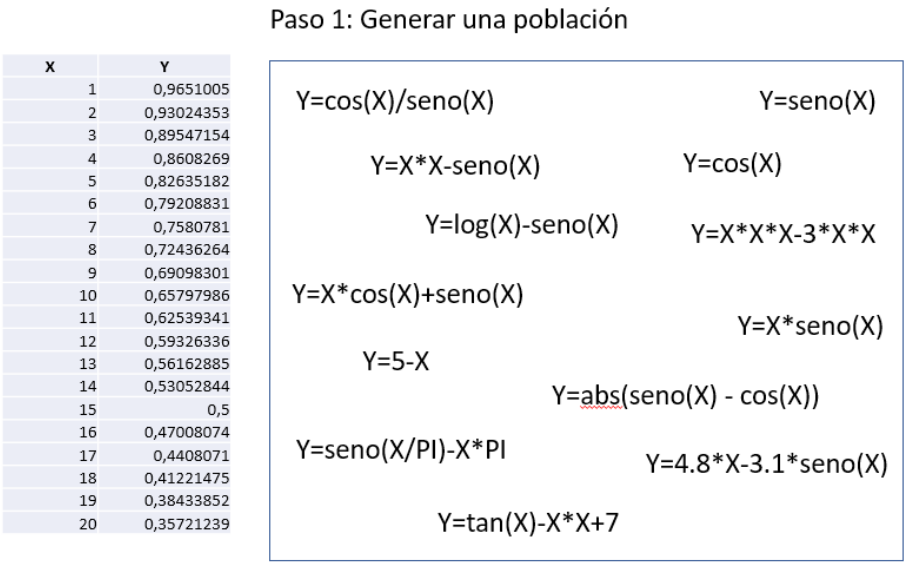


Ilustración 16: Paso 1: Generar una población

Fuente: propia del autor

Se plantean varias preguntas:

1. ¿Cuántos individuos va a tener esa población inicial? Si son muy pocos no habrá suficiente diversidad que es requerida para aumentar

las probabilidades de dar con esa ecuación. Demasiados y será lento dar con la ecuación más cercana.

2. ¿Cómo se genera cada individuo al azar? Se requiere un método que usando el azar genere ecuaciones sintácticamente correctas.
3. ¿Qué tan larga debe ser la ecuación? Una ecuación muy corta no permite muchas variaciones, una muy larga tomará tiempo evaluarla.

Los individuos

Un patrón entre las entradas y la salida numéricas significa que debe haber una ecuación donde las entradas son las variables independientes y la salida es la variable dependiente:

Variable Salida = función (variable Entra 1, variable Entra 2, variable Entra N)

Los individuos en el algoritmo genético son ecuaciones algebraicas. Generar ecuaciones algebraicas es fácil, lo que puede impactar negativamente el desempeño es la evaluación de esas ecuaciones, que requiere un evaluador de expresiones. Debe haber otra técnica para tener la flexibilidad de las ecuaciones pero que su evaluación sea muy rápida. En el libro **"Un uso de algoritmos genéticos para la búsqueda de patrones"**, se propone una técnica para representar ecuaciones. Este es un ejemplo:

$$k = \frac{a^f + \sqrt[3]{\frac{g}{h}}}{\cos(d-e)} + \sqrt[2]{7 * b * c} - j^{\sqrt[2]{l+m}}$$

Es posible representar una ecuación como esta, en una sucesión de piezas más simples que tienen esta representación:

Variable Acumula = [Función] Operando Operador Operando

Donde Operador puede ser suma(+), resta(-), multiplicación(*), división(/), potencia(^); [Función] puede ser raíz cuadrada, raíz cubica, seno, coseno, tangente, logaritmo, etc., si no hay función es ”no función”

Luego la ecuación anterior, se puede dividir en piezas así:

A = [raíz cuadrada] l + m
B = [no función] j ^ A
C = [no función] b * c
D = [raíz cuadrada] 7 * C
E = [coseno] d - e
F = [raíz cúbica] g / h
G = [no función] a ^ f
H = [no función] G + F
I = [no función] H / E
J = [no función] I + D
K = [no función] J - B

En el software, el usuario puede escoger cuantos individuos tendrá la población y cuantas piezas tendrá cada ecuación (entre más piezas, más compleja es la ecuación).

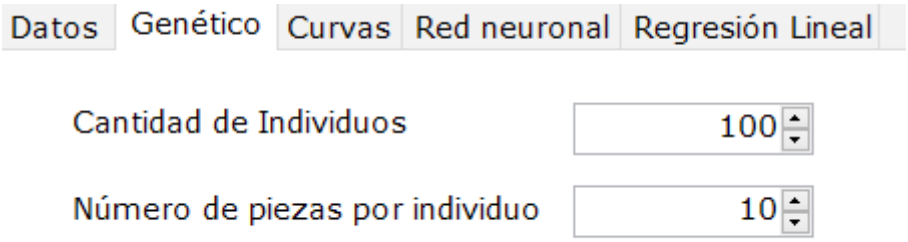


Ilustración 17: Se configura en el algoritmo genético el número de individuos tendrá la población y la cantidad de piezas que tendrá cada individuo (ecuación).

Paso 2: Seleccionar dos individuos al azar

X	Y
1	0,9651005
2	0,93024353
3	0,89547154
4	0,8608269
5	0,82635182
6	0,79208831
7	0,7580781
8	0,72436264
9	0,69098301
10	0,65797986
11	0,62539341
12	0,59326336
13	0,56162885
14	0,53052844
15	0,5
16	0,47008074
17	0,4408071
18	0,41221475
19	0,38433852
20	0,35721239

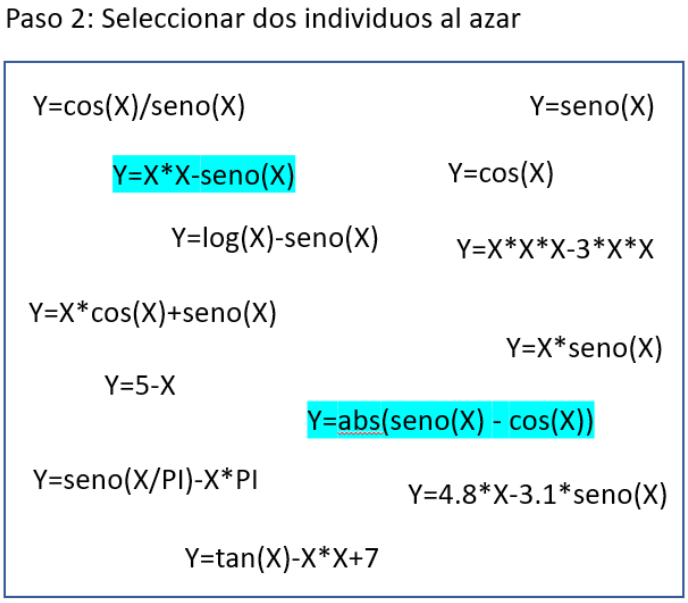


Ilustración 18: Paso 2: Seleccionar dos individuos al azar

Fuente: propia del autor

De la población definida, se seleccionan dos individuos al azar. Es notable entonces el uso intensivo de números aleatorios, por lo que es necesario tener un buen generador de números aleatorios. C# tiene uno confiable [26].

Paso 3: Evaluar esos individuos seleccionados

Paso 3: Evaluar esos individuos seleccionados

X	Y	Y=X*X-seno(X)	Diferencia	Y=abs(seno(X)-cos(X))	Diferencia
1	0,9651005	0,158529015		0,301168679	
2	0,93024353	3,090702573		1,325444263	
3	0,89547154	8,858879992		1,131112505	
4	0,8608269	16,7568025		0,103158874	
5	0,82635182	25,95892427		1,24258646	
6	0,79208831	36,2794155		1,239585785	
7	0,7580781	48,3430134		0,096915656	
8	0,72436264	63,01064175		1,13485828	
9	0,69098301	80,58788151		1,323248747	
10	0,65797986	100,5440211		0,295050418	
11	0,62539341	121,9999902		1,004415905	
12	0,59326336	144,5365729		1,380426877	
13	0,56162885	168,579833		0,487279745	
14	0,53052844	195,0093926		0,853870137	
15	0,5	224,3497122		1,409975753	
16	0,47008074	256,2879033		0,669756164	
17	0,4408071	289,9613975		0,686234154	
18	0,41221475	324,7509872		1,411303955	
19	0,38433852	360,8501228		0,838827409	
20	0,35721239	399,0870547		0,504863189	

Ilustración 19: Paso 3: Evaluar esos individuos seleccionados

Fuente: propia del autor

Como se observa en la ilustración, cada ecuación genera un valor Y con los diferentes valores de X. La pregunta es ¿Cómo evaluar rápidamente cada ecuación con el X dado? La respuesta es evaluar las piezas en orden, con eso se obtiene buena velocidad.

Los modificadores

Se tiene por ejemplo esta pieza:

$$Z = [seno]\ A + B$$

Que significa

$$Z = seno\ (A + B)$$

Los modificadores son coeficientes (número reales que multiplican) a todas las partes de esa pieza así:

$$Z = m_0 * seno(m_1 * (m_2 * A + m_3 * B))$$

Si m_0, m_1, m_2 tienen valor 1, entonces se obtendrían los mismos valores que la ecuación anterior. Estos modificadores son números reales, pueden ser positivos, negativos o ser cero por lo que pueden cambiar notablemente el resultado de la pieza. Entonces hay modificador del resultado de la función (si existe), modificador de la operación matemática (+, -, *, /) y modificador por cada operando.

En una ecuación más compleja como:

$$w = seno\ (2 * x) - coseno\ (3 * y)$$

Estas serían las piezas:

A = [seno] 2 * x

B = [coseno] 3 * y

C = [no función] A – B

Se le añaden los modificadores:

A = m_o*[seno] (m_1*2*m_2*x)

B = m_3*[coseno] (m_4*3*m_5*y)

C = [no función] m_6*A –m_7*B

Luego una mutación es variar los modificadores y es así:

$$m_n \leftarrow m_n + 2 * aleatorio() - 1$$

La función aleatorio() retorna un número al azar entre 0 y 1, al multiplicarlo por 2 y restarle 1, se obtiene un número al azar entre -1 y 1. Ese resultado se le suma al valor anterior del modificador.

Para mejorar posiblemente el comportamiento de los individuos seleccionados, se hacen cambios a los modificadores.

Paso 4: Seleccionar el mejor de esos dos individuos

Paso 4: Seleccionar el mejor de esos dos

X	Y	Y=X*X-seno(X)	Diferencia	Y=abs(seno(X)-cos(X))	Diferencia
1	0,965100503	0,158529015	0,650557565	0,301168679	0,440805467
2	0,930243526	3,090702573	4,667583293	1,325444263	0,156183623
3	0,895471537	8,858879992	63,41587422	1,131112505	0,055526666
4	0,860826899	16,7568025	252,6820402	0,103158874	0,574060835
5	0,826351822	25,95892427	631,6461981	1,24258646	0,173251274
6	0,792088309	36,2794155	1259,350391	1,239585785	0,200253991
7	0,758078104	48,3430134	2264,326067	0,096915656	0,437135784
8	0,724362644	63,01064175	3879,580565	1,13485828	0,168506667
9	0,690983006	80,58788151	6383,514391	1,323248747	0,399759968
10	0,657979857	100,5440211	9977,221237	0,295050418	0,131717777
11	0,625393407	121,9999902	14731,79275	1,004415905	0,143658054
12	0,593263357	144,5365729	20719,67637	1,380426877	0,619626407
13	0,561628853	168,579833	28230,11691	0,487279745	0,00552779
14	0,530528437	195,0093926	37822,02862	0,853870137	0,104549855
15	0,5	224,3497122	50108,69363	1,409975753	0,828055871
16	0,470080736	256,2879033	65442,75835	0,669756164	0,039870277
17	0,440807097	289,9613975	83822,17226	0,686234154	0,06023444
18	0,412214748	324,7509872	105195,6393	1,411303955	0,998179244
19	0,384338525	360,8501228	129935,5816	0,838827409	0,206560146
20	0,35721239	399,0870547	158985,4872	0,504863189	0,021800758
		Sumatoria	**719711,002**		**5,765264894**

Menor diferencia

Ilustración 20: Paso 4: Seleccionar el mejor de esos dos individuos

Fuente: propia del autor

La segunda columna es el Y esperado, las ecuaciones generan sus propios Y. Luego se hace la siguiente operación

$$Diferencia = (Y\ esperado - Y\ ecuación)^2$$

Luego se suman las diferencias, el que obtenga el menor valor acumulado de diferencias, es el seleccionado, es el mejor individuo.

Paso 5: Eliminar el peor individuo de la población

X	Y
1	0,9651005
2	0,93024353
3	0,89547154
4	0,8608269
5	0,82635182
6	0,79208831
7	0,7580781
8	0,72436264
9	0,69098301
10	0,65797986
11	0,62539341
12	0,59326336
13	0,56162885
14	0,53052844
15	0,5
16	0,47008074
17	0,4408071
18	0,41221475
19	0,38433852
20	0,35721239

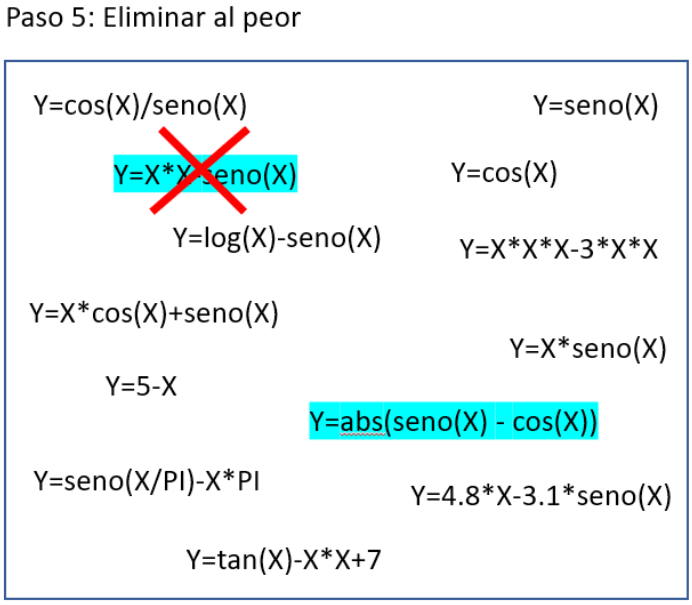


Ilustración 21: Paso 5: Eliminar el peor individuo de la población

Fuente: propia del autor

El peor individuo se elimina de la población, dejando una vacante que pronto será llenada.

Paso 6: Duplicar el mejor individuo

X	Y
1	0,9651005
2	0,93024353
3	0,89547154
4	0,8608269
5	0,82635182
6	0,79208831
7	0,7580781
8	0,72436264
9	0,69098301
10	0,65797986
11	0,62539341
12	0,59326336
13	0,56162885
14	0,53052844
15	0,5
16	0,47008074
17	0,4408071
18	0,41221475
19	0,38433852
20	0,35721239

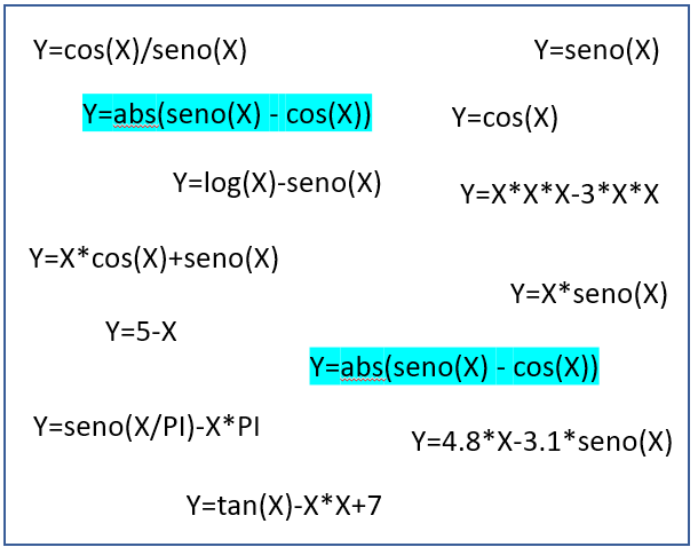


Ilustración 22: Paso 6: Duplicar el mejor individuo

Fuente: propia del autor

El mejor individuo, tiene una recompensa: reproducirse, en este caso, clonarse. Así ocupa el espacio dejado por el individuo eliminado previamente.

Paso 7: Modificar el duplicado

X	Y
1	0,9651005
2	0,93024353
3	0,89547154
4	0,8608269
5	0,82635182
6	0,79208831
7	0,7580781
8	0,72436264
9	0,69098301
10	0,65797986
11	0,62539341
12	0,59326336
13	0,56162885
14	0,53052844
15	0,5
16	0,47008074
17	0,4408071
18	0,41221475
19	0,38433852
20	0,35721239

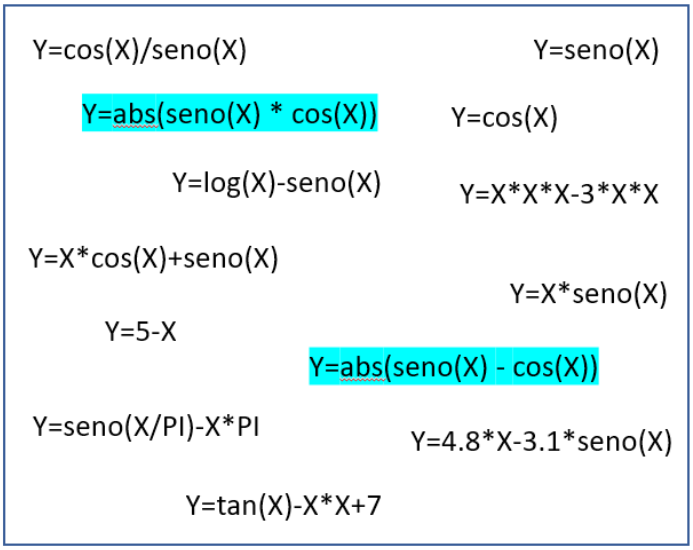


Ilustración 23: Paso 7: Modificar el duplicado

Fuente: propia del autor

En forma aleatoria, una parte del individuo clonado es modificada, sea cambiar un operador por otro, una variable por un número, un número por una variable, un número por otro número, o una función por otra. Por el uso de piezas, este cambio pasa por seleccionar una pieza al azar y luego que parte de esa pieza se cambia.

Paso 8: Repetir nuevamente desde el paso 2

X	Y
1	0,9651005
2	0,93024353
3	0,89547154
4	0,8608269
5	0,82635182
6	0,79208831
7	0,7580781
8	0,72436264
9	0,69098301
10	0,65797986
11	0,62539341
12	0,59326336
13	0,56162885
14	0,53052844
15	0,5
16	0,47008074
17	0,4408071
18	0,41221475
19	0,38433852
20	0,35721239

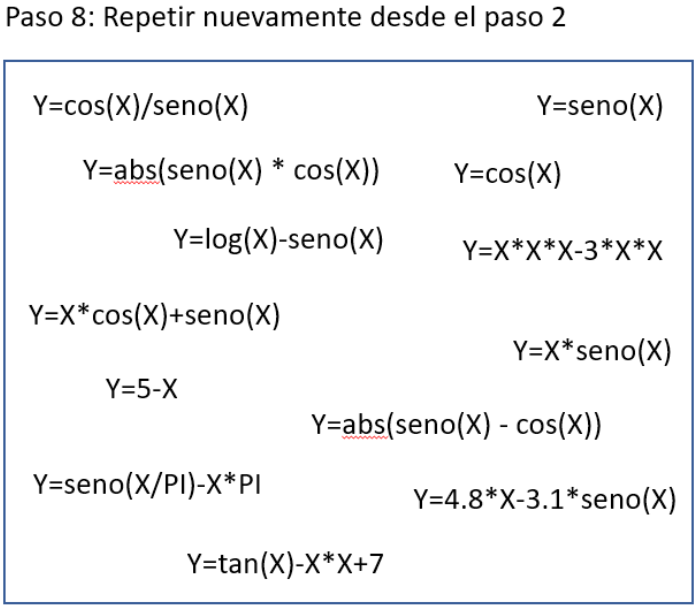


Ilustración 24: Paso 8: Repetir nuevamente desde el paso 2

Fuente: propia del autor

Se vuelve al estado inicial, pero con una población con individuos cuyas expresiones algebraicas son cada vez más cercanas al comportamiento de Y dado X.

El algoritmo

```
Algoritmo Genético
Inicio
 Generar población inicial de individuos al azar
 Desde ciclo=1 hasta N paso 1 hacer
  Tomar dos individuos al azar y evaluar su adaptación
  Modificar coeficientes de esos individuos seleccionados y evaluar si mejora adaptación
  Tomar el peor individuo adaptado y eliminarlo de la población
  Tomar el mejor individuo adaptado y sacarle una copia
  Modificar esa copia al azar y ponerla en la población
 Fin desde
Fin
```

El algoritmo puede cambiar, pero se mantiene el mismo concepto: seleccionar el mejor individuo, reproducirlo y modificar la copia.

```
Algoritmo Genético2
Inicio
 Generar población inicial de individuos al azar
 Desde ciclo=1 hasta Número_ciclos_reproducción paso 1 hacer
  Desde modificar=1 hasta Número_ciclos_ajuste_modificadores_pieza paso 1 hacer
   Hacer cambios a los modificadores de cada individuo de la población.
   Almacenar el mejor y peor individuo.
  Fin desde
  Tomar el peor individuo adaptado y eliminarlo de la población
  Tomar el mejor individuo adaptado y sacarle una copia
  Modificar esa copia al azar y ponerla en la población
 Fin desde
Fin
```

El segundo algoritmo es el que se implementó en el software porque requería menos uso de números aleatorios, además se le daba más oportunidades a cada individuo para que con cambios en los modificadores se aproxima más a las salidas esperadas y sobre todo mantenía por más tiempo la diversidad en la población.

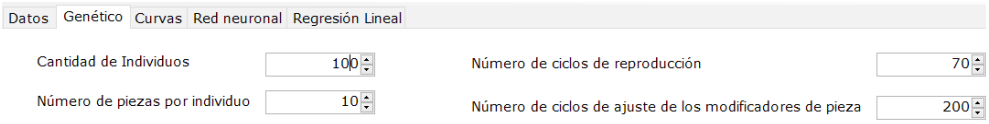


Ilustración 25: Ajustes en los ciclos de reproducción y ajustes de modificadores de pieza

Implementación en C#

Hay tres clases para implementar el algoritmo genético:

Ilustración 26: Clases para el algoritmo genético

Fuente: propia del autor

Una población tiene N individuos
Un individuo (que es una ecuación) tiene N piezas

La clase AG_Pieza tiene varias funciones:

1. Crear la pieza con valores aleatorios.
2. Almacenar los datos de cada pieza.
3. Evaluar la pieza (obtiene un resultado numérico).

4. Mutar los modificadores.
5. Mutar la pieza.
6. Imprimir la pieza.

AG_Pieza.cs

```
/* Universidad Libre - Cali
 * Facultad de Ingeniería
 * Programa de Ingeniería de Sistemas
 * Proyecto unificador de técnicas de análisis de datos para la
 * búsqueda de patrones de comportamiento basadas en
 * Inteligencia Artificial
 *
 * Autor: Rafael Alberto Moreno Parra
 * Fecha: 2020
 *
 * Cualquier ecuación puede generarse con piezas.
 * Luego, el algoritmo genético se basa en generar piezas al azar
 * que al interpretarse generarán cualquier ecuación.
 * Las ecuaciones serán los individuos del algoritmo genético.
 *
 * El problema de generar totalmente al azar las piezas son:
 * 1. Se forman individuos que solo funciona la última pieza
 *    ejemplo: [N] 7 * 8
 * 2. Hay piezas que generan división entre cero
 * 3. Hay piezas que generan siempre error matemático
 * 4. No se consideran todas las variables de entrada
 *
 * Esta clase se encarga de:
 * 1. Generar la pieza al azar
 * 2. Evaluar la pieza
 * 3. Mutar los modificadores
 * 4. Mutar la pieza
 *
 * Las piezas son creadas así, por ejemplo:
 * [0] seno( 3 + a )
 * [1] seno( b * 4 - [0] )
 * [2] seno( a * c + [1] )
 * [3] seno( 5 * [1] + [2] )
 * [4] seno( a + a + [3] )
 * [5] seno( b * c - [4] )
 *
```

```
 * Como se puede observar, a partir de la posición [1],
 * el tercer operando es la pieza anterior, de esa forma
 * obliga a hacer uso de todas las piezas.
 *
 * Los modificadores son constantes que afectan las partes de la pieza, ejemplo:
 * Pieza usual: [N] seno[ a + b - [N-1] ]
 * Pieza con modificadores: [N] mFuncion * seno [ mParteA*a + mParteB*b -
mParteC*[N-1] ]
 * Luego se cambian los modificadores al azar para encontrar el mejor individuo
 * */
using System;
using System.Collections.Generic;

namespace Unificador {
  class AG_Pieza {
    public int tipoA; //0. Variable, 1. Constante, 2. Pieza
    public double mParteA; //Modificador del valor de ParteA
    public int varA; //Usará variable de entrada
    public double constA; //Usará constante
    public int piezaA; //Se refiere a otra pieza (Que debe estar antes de esta)

    public int tipoB; //0. Variable, 1. Constante, 2. Pieza
    public double mParteB; //Modificador del valor de ParteB
    public int varB; //Usará variable de entrada
    public double constB; //Usará constante
    public int piezaB; //Se refiere a otra pieza (Que debe estar antes de esta)

    //Modificador del valor de ParteC
    public double mParteC;

    //Tipo de operación entre las partes
    //1: ParteA + ParteB + ParteC
    //2: ParteA + ParteB - ParteC
    public int tipoOperacion; //Operación entre parteA y parteB

    //Variables usadas cuando se cambia algún modificador
    //para poder restaurarlo
    private double valAnterior; //Guarda el valor anterior de un modificador
    private int QueCambio; //Guarda que modificador cambió

    //Crea la pieza, valida si la pieza está en una posición
    //por encima de cero para poder hacer uso de piezas anteriores
    public AG_Pieza(Random Azar, int PosPieza, int totalVariables) {
```

```
    //Si la pieza está por encima de la posición cero
    //entonces puede hacer uso de piezas anteriores

    //Parte A
    if (PosPieza > 0) tipoA = Azar.Next(3); else tipoA = Azar.Next(2);
    mParteA = Azar.NextDouble();
    varA = Azar.Next(totalVariables);
    constA = Azar.NextDouble();
    piezaA = Azar.Next(PosPieza);

    //Si la primera parte es constante, la segunda es variable o es pieza
    if (tipoA == 1)
      if (PosPieza == 0) {
        tipoB = 0;
      }
      else {
        switch (Azar.Next(2)) {
          case 0: tipoB = 0; break;
          case 1: tipoB = 2; break;
      }
    else { //Si la primera parte no es constante entonces tipoB puede ser 0, 1, 2
      if (PosPieza > 0)
        tipoB = Azar.Next(3);
      else
        tipoB = Azar.Next(2);
    }

    //Parte B
    mParteB = Azar.NextDouble();
    varB = Azar.Next(totalVariables);
    constB = Azar.NextDouble();
    piezaB = Azar.Next(PosPieza);

    //Parte C
    mParteB = Azar.NextDouble();

    //Operación entre las partes
    tipoOperacion = Azar.Next(9);
  }

  //Evalúa la pieza dándole el valor de las variables de entrada,
  //el valor obtenido de piezas anteriores (por eso
  //se usa valorPiezas y numeroPieza (para la pieza anterior))
```

```
    public double Evalua(List<List<double>> ValorDatos, int registro,
double[] valorPiezas, int numPieza) {
      double valA=0, valB=0, valC, total=0;

      switch(tipoA) { //0. Variable, 1. Constante, 2. Pieza
        case 0: valA = ValorDatos[registro][varA]; break;
        case 1: valA = constA; break;
        case 2: valA = valorPiezas[piezaA]; break;
      }

      switch (tipoB) { //0. Variable, 1. Constante, 2. Pieza
        case 0: valB = ValorDatos[registro][varB]; break;
        case 1: valB = constB; break;
        case 2: valB = valorPiezas[piezaB]; break;
      }

      valC = 1;
      if (numPieza > 0) valC = valorPiezas[numPieza - 1];

      //Hace la operación matemática
      switch (tipoOperacion) {
        case 0: total = mParteA * valA + mParteB * valB + mParteC * valC; break;
        case 1: total = mParteA * valA + mParteB * valB - mParteC * valC; break;
        case 2: total = mParteA * valA + mParteB * valB * mParteC * valC; break;
        case 3: total = mParteA * valA - mParteB * valB + mParteC * valC; break;
        case 4: total = mParteA * valA - mParteB * valB - mParteC * valC; break;
        case 5: total = mParteA * valA - mParteB * valB * mParteC * valC; break;
        case 6: total = mParteA * valA * mParteB * valB + mParteC * valC; break;
        case 7: total = mParteA * valA * mParteB * valB - mParteC * valC; break;
        case 8: total = mParteA * valA * mParteB * valB * mParteC * valC; break;
      }

      return Math.Abs(Math.Sin(total));
    }

    //Muta los modificadores de lo que protagoniza. Porque ahora puede
cambiar algo que no afecta
    public void MutaModificador(Random Azar) {
      double valCambio = (Azar.NextDouble()*2-1)/10;
      QueCambio = Azar.Next(4);
      switch (QueCambio) {
        case 0: valAnterior = mParteA; mParteA += valCambio; break;
        case 1: valAnterior = mParteB; mParteB += valCambio; break;
        case 2: valAnterior = mParteC; mParteC += valCambio; break;
```

```
      }
        }

        //Si el cambio al modificador desmejora la aproximación entonces
restaura el valor anterior
        public void RestauraModificador() {
      switch (QueCambio) {
        case 0: mParteA = valAnterior; break;
        case 1: mParteB = valAnterior; break;
        case 2: mParteC = valAnterior; break;
      }
    }

    //Cambia la pieza
    public void CambiaPieza() {
      tipoOperacion++;
      if (tipoOperacion > 8) tipoOperacion = 0;
    }

    //Retorna true si la pieza hace uso de una determinada variable de
entrada
    //Se utiliza para validar si todas las variables de entrada
    //son usadas por el individuo
    public bool UsaVariable(int variable) {
      if (tipoA == 0 && varA == variable) return true;
      if (tipoB == 0 && varB == variable) return true;
      return false;
    }

    //Imprime la pieza para ser interpretada por Excel
    public string ExportaExcel(int registro, int numeroPieza, int
numVarEntra) {
      string valA = "", valB = "", valC, total = "";
      char miVariableA = Convert.ToChar(varA + 'A');
      char miVariableB = Convert.ToChar(varB + 'A');
      char miPiezaA = Convert.ToChar(piezaA + numVarEntra + 1 + 'A');
      char miPiezaB = Convert.ToChar(piezaB + numVarEntra + 1 + 'A');
      char miPiezaC = Convert.ToChar((numeroPieza-1) + numVarEntra + 1 + 'A');

      //Suma 2 porque en el Excel empieza en 2
      registro+=2;

      switch (tipoA) {
        case 0: valA = miVariableA.ToString() + registro.ToString(); break;
        case 1: valA = constA.ToString(); break;
```

```
        case 2: valA = miPiezaA.ToString() + registro.ToString(); break;
      }

      switch (tipoB) {
        case 0: valB = miVariableB.ToString() + registro.ToString(); break;
        case 1: valB = constB.ToString(); break;
        case 2: valB = miPiezaB.ToString() + registro.ToString(); break;
      }

      valC = "1";
      if (numeroPieza > 0) valC = miPiezaC.ToString() + registro.ToString();

      //Hace la operación matemática entre parte 1 y parte 2
      switch (tipoOperacion) {
        case 0: total = mParteA.ToString() + "*" + valA + "+" + mParteB.
ToString() + "*" + valB + "+" + mParteC.ToString() + "*" + valC; break;

        case 1: total = mParteA.ToString() + "*" + valA + "+" + mParteB.
ToString() + "*" + valB + "-" + mParteC.ToString() + "*" + valC; break;

        case 2: total = mParteA.ToString() + "*" + valA + "+" + mParteB.
ToString() + "*" + valB + "*" + mParteC.ToString() + "*" + valC; break;

        case 3: total = mParteA.ToString() + "*" + valA + "-" + mParteB.
ToString() + "*" + valB + "+" + mParteC.ToString() + "*" + valC; break;

        case 4: total = mParteA.ToString() + "*" + valA + "-" + mParteB.
ToString() + "*" + valB + "-" + mParteC.ToString() + "*" + valC; break;

        case 5: total = mParteA.ToString() + "*" + valA + "-" + mParteB.
ToString() + "*" + valB + "*" + mParteC.ToString() + "*" + valC; break;

        case 6: total = mParteA.ToString() + "*" + valA + "*" + mParteB.
ToString() + "*" + valB + "+" + mParteC.ToString() + "*" + valC; break;

        case 7: total = mParteA.ToString() + "*" + valA + "*" + mParteB.
ToString() + "*" + valB + "-" + mParteC.ToString() + "*" + valC; break;

        case 8: total = mParteA.ToString() + "*" + valA + "*" + mParteB.
ToString() + "*" + valB + "*" + mParteC.ToString() + "*" + valC; break;

      }


      return "=abs(seno(" +total + "))";
    }
  }
}
```

Luego sigue la clase Individuo que tiene las siguientes funciones:

1. Crear el individuo con N piezas
2. Evaluar el individuo frente a la serie de datos
3. Mutar los modificadores de pieza (llamando los métodos de pieza)
4. Mutas la pieza (llamando los métodos de pieza)

AG_Individuo.cs

```
/* Proyecto: Desarrollo de una herramienta de software unificadora de análisis de datos para
 * la detección de patrones de comportamiento haciendo uso de varios métodos de inteligencia
 * artificial.
 *
 * Autor: Rafael Alberto Moreno Parra
 * Correo: rafael-morenop@unilibre.edu.co
 * Semillero: Semillero en investigación, desarrollo y documentación de herramientas para la construcción de sistemas de información en Web
 * Universidad Libre - Cali
 * Ingeniería
 *
 * Fecha: 2020
 * */
using System;
using System.Collections.Generic;

namespace Unificador {
  class AG_Individuo {
    public List<AG_Pieza> Piezas = new List<AG_Pieza>();
    //Valores que genera cada pieza al ser evaluada
    private double[] valorPiezas;
    //Almacena la aproximación del individuo
    //(como caché para evaluar la población después)
    public double Aproximacion;

    // Construir el individuo, teniendo especial cuidado
    //que haga uso de todas las variables de entrada
    public AG_Individuo(Random Azar, int numPiezas, int numVarEntra) {
      Aproximacion = double.MaxValue;
      valorPiezas = new double[numPiezas];

      bool usaVariable = false;
      do {
```

```
        // Crea las piezas en forma aleatoria
        Piezas.Clear();
        for (int num = 0; num < numPiezas; num++) {
          Piezas.Add(new AG_Pieza(Azar, num, numVarEntra));
        }

        //Valida que al final todas las piezas juntas hagan
        //uso de todas las variables de entrada
        for (int miVar = 0; miVar < numVarEntra; miVar++) {
          usaVariable = false;
          for (int num = 0; num < numPiezas; num++) {
            usaVariable = Piezas[num].UsaVariable(miVar);
            if (usaVariable) break;
          }
          if (usaVariable == false) break;
        }
      } while (usaVariable == false);
    }

    //Evalua el individuo, con las variables de entrada
    public double Evaluar(List<List<double>> ValorDatos, int registro) {
      for (int numPieza=0; numPieza<Piezas.Count; numPieza++) {
        valorPiezas[numPieza] = Piezas[numPieza].Evalua(ValorDatos,
registro, valorPiezas, numPieza);
      }
      return valorPiezas[Piezas.Count - 1];
    }

    //Evalúa la aproximación del individuo
    private double CalculaAproximacion(List<List<double>> ValorDatos, double
Aproximacion) {
      //La variable de salida siempre es la última
      int ultima = ValorDatos[0].Count - 1;

      //Calcula la aproximación
      double aproxima = 0;
      for (int registro=0; registro < ValorDatos.Count; registro++) {
        double Salida = Evaluar(ValorDatos, registro);
        aproxima += (Salida - ValorDatos[registro][ultima]) * (Salida -
ValorDatos[registro][ultima]);
        if (aproxima > Aproximacion) return double.MaxValue;
      }
      return aproxima;
    }
```

```
    //Muta los modificadores para ver si mejora la aproximación
    public void MutaModificador(Random Azar, List<List<double>> ValorDatos) {
      if (Aproximacion == double.MaxValue) Aproximacion =
CalculaAproximacion(ValorDatos, Aproximacion);

      //Selecciona una pieza al azar y muta alguno de sus modificadores
      int pieza = Azar.Next(Piezas.Count);
      Piezas[pieza].MutaModificador(Azar);

      //Calcula la nueva aproximación
      double nuevo = CalculaAproximacion(ValorDatos, Aproximacion);

      //Si es mejor, entonces se deja la mutación,
      //caso contrario se deja como estaba el individuo
      if (nuevo < Aproximacion)
        Aproximacion = nuevo;
      else
        Piezas[pieza].RestauraModificador();
    }

    //Muta una pieza al azar y pone la aproximación a -1 (no evaluado)
    public void MutaPieza(Random Azar) {
      int piezaMuta = Azar.Next(Piezas.Count);
      Piezas[piezaMuta].CambiaPieza();
      Aproximacion = -1;
    }
  }
}
```

Luego sigue la clase Población que tiene estas funciones:

1. Crear los individuos
2. Aplicar el algoritmo genético visto anteriormente

AG_Poblacion.cs

```
/* Proyecto: Desarrollo de una herramienta de software unificadora de
análisis de datos para
 * la detección de patrones de comportamiento haciendo uso de varios métodos
de inteligencia
 * artificial.
 *
 * Autor: Rafael Alberto Moreno Parra
 * Correo: rafael-morenop@unilibre.edu.co
```

```
 * Semillero: Semillero en investigación, desarrollo y documentación de
herramientas para la construcción de sistemas de información en Web
 * Universidad Libre - Cali
 * Ingeniería
 *
 * Fecha: 2020
 * */
using System;
using System.Collections.Generic;
using System.IO;

namespace Unificador {
  class AG_Poblacion {
    /* Generador de números aleatorios */
    private Random Azar;

    /* Mejor aproximación e individuo que mejor aproxima */
    public int mejorIndividuo;
    public double mejorAproxima;

    /* Una población tiene muchos individuos */
    public List<AG_Individuo> Individuos = new List<AG_Individuo>();

    /* Inicia el generador de números aleatorios */
    public AG_Poblacion() {
      Azar = new Random();
    }

    /* Crea la población de individuos */
    public void Configura(int numIndividuos, int numPiezas, int numVariables) {
      Individuos.Clear();
      for (int num = 1; num <= numIndividuos; num++) {
        Individuos.Add(new AG_Individuo(Azar, numPiezas, numVariables));
      }
    }

    // Los parámetros son el listado de datos a evaluar,
    //llamado ValorDatos, de esta forma:
    // varA, varB, varC, varD, varE
    //
    //La última variable es la única de salida,
    //las primeras son siempre de entrada
    public void Proceso(List<List<double>> ValorDatos, int CiclosModificador) {
      mejorAproxima = double.MaxValue; mejorIndividuo = -1;
```

```
      double peorAproxima = 0; int peorIndividuo = -1;

      // Los ciclos donde se ajustan los modificadores
      for (int Modificador = 1; Modificador <= CiclosModificador; Modificador++) {

        // Tomar individuo por individuo y evaluar su aproximación (mutando
los modificadores)
        for (int indiv = 0; indiv < Individuos.Count; indiv++) {

          //Evalúa el individuo y cambia los modificadores si mejora la
aproximación
          Individuos[indiv].MutaModificador(Azar, ValorDatos);

          //Se guarda el mejor individuo
          if (Individuos[indiv].Aproximacion < mejorAproxima) {
            mejorAproxima = Individuos[indiv].Aproximacion;
            mejorIndividuo = indiv;
          }

          //Se guarda el peor individuo
          if (Individuos[indiv].Aproximacion > peorAproxima) {
            peorAproxima = Individuos[indiv].Aproximacion;
            peorIndividuo = indiv;
          }
        }
      }

      //El mejor individuo sobreescribe al peor individuo
      for (int copia = 0; copia < Individuos[mejorIndividuo].Piezas.Count;
copia++) {
        AG_Pieza peor = Individuos[peorIndividuo].Piezas[copia];
        AG_Pieza mejor = Individuos[mejorIndividuo].Piezas[copia];

        peor.constA = mejor.constA;
        peor.constB = mejor.constB;
        peor.mParteA = mejor.mParteA;
        peor.mParteB = mejor.mParteB;
        peor.mParteC = mejor.mParteC;
        peor.tipoOperacion = mejor.tipoOperacion + 1;
        if (peor.tipoOperacion > 8) peor.tipoOperacion = 0;
        peor.piezaA = mejor.piezaA;
        peor.piezaB = mejor.piezaB;
        peor.tipoA = mejor.tipoA;
        peor.tipoB = mejor.tipoB;
        peor.varA = mejor.varA;
```

```
        peor.varB = mejor.varB;
      }
      Individuos[peorIndividuo].Aproximacion = double.MaxValue; //Nuevo
individuo queda a aproximación máxima
    }

    //Cuando el usuario haga interpolación
    public double SalidaCalculada(List<List<double>> ValorDatos) {
      return Individuos[mejorIndividuo].Evaluar(ValorDatos, 0);
    }

    //Imprime el mejor individuo
    public void ImprimeMejorIndividuo(StreamWriter archivo, int registro,
int numVariables) {
      for (int pieza=0; pieza < Individuos[mejorIndividuo].Piezas.Count;
pieza++) {
        string pedazo = Individuos[mejorIndividuo].Piezas[pieza].
ExportaExcel(registro, pieza, numVariables);
        archivo.Write(pedazo + ";");
      }
    }
  }
}
```

Por último, el código del programa principal que hace uso de las clases anteriores. Se hace uso de hilos de ejecución para que el programa no "paralice" el entorno gráfico mientras está haciendo los cálculos.

Parte de Form1.cs

```
    // ************************************************************************
    // PROCESO DEL ALGORITMO GENÉTICO
    // ************************************************************************
    private void BtnGenetico_Click(object sender, EventArgs e) {
      if (bgwAG.IsBusy != true) {
        btnAGCalculaSalida.Enabled = false;

        //Si el hilo no está trabajando
        bgwAG.RunWorkerAsync();
      }
              }

    private void ProcesoGenetico(object sender, DoWorkEventArgs e) {
      //Genera una población viable
```

```
        int cantidadIndividuos = (int)numIndividuos.Value;
        int numPiezasIndividuo = (int)numPiezas.Value;
        poblacion.Configura(cantidadIndividuos, numPiezasIndividuo, objDato.
ValorDatos[0].Count - 1);

        //Ejecuta el proceso de algoritmo genético
        int totalCiclosReproduce = (int)numReproduce.Value; //Veces que el
mejor individuo sobreescribe el peor individuo y luego lo muta
        int totalCiclosModificador = (int)numAjuste.Value; //Veces que se
cambian los modificadores de cada individuo

        //Ciclos de reproducción
        for (int Reproduce = 1; Reproduce <= totalCiclosReproduce; Reproduce++) {
          poblacion.Proceso(objDato.ValorDatos, totalCiclosModificador);
          bgwAG.ReportProgress(Reproduce * 100 / totalCiclosReproduce);
        }
    }

    private void bgwAG_ProgressChanged(object sender,
ProgressChangedEventArgs e) {
        ProgresoAG.Value = e.ProgressPercentage;
    }

    private void bgwAG_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e) {
        txtAproxGenetico.Text = poblacion.mejorAproxima.ToString(); //Imprime
la aproximación
        btnAGCalculaSalida.Enabled = true;
    }
```

El siguiente código es ejecutado cuando el usuario decide hacer interpolación o extrapolación. El programa lee los valores que el usuario ha digitado para las variables de entrada. Envía esos datos al algoritmo genético que ya ha calculado el mejor individuo que se ajusta a los valores leídos al inicio. Y con esos valores de entrada (digitados por el usuario) se calcula la salida.

Parte de Form1.cs

```
    private void BtnAGCalculaSalida_Click(object sender, EventArgs e) {
        //En esta lista envía las entradas al algoritmo genético escritas por
el usuario
        List<List<double>> NuevaEntrada = new List<List<double>>();
```

```
      //Lista provisional donde se arma el registro de los valores de
entrada del usuario
      List<double> valorProvisional = new List<double>();
      for (int num = 0; num < objDato.ValorDatos[0].Count - 1; num++) {
        double valor = Convert.ToDouble(dgAGinterpolar.Rows[num].Cells[1].
Value);
        valor = objDato.Normalizado(valor, num); //Debe normalizarlo
        valorProvisional.Add(valor);
      }
      valorProvisional.Add(0);

      //Las entradas se arman y se envían al algoritmo genético
      NuevaEntrada.Add(valorProvisional);
      double SalidaCalculada = poblacion.SalidaCalculada(NuevaEntrada);

      //Desnormaliza
      double valorSaleNormal = objDato.Desnormalizado(SalidaCalculada,
objDato.ValorDatos[0].Count - 1);
      txtAGCampoSalida.Text = valorSaleNormal.ToString();
    }
```

Uso en el unificador

Una vez cargados las variables de entrada y la variable de salida, el usuario se dirige a la pestaña "Genético"

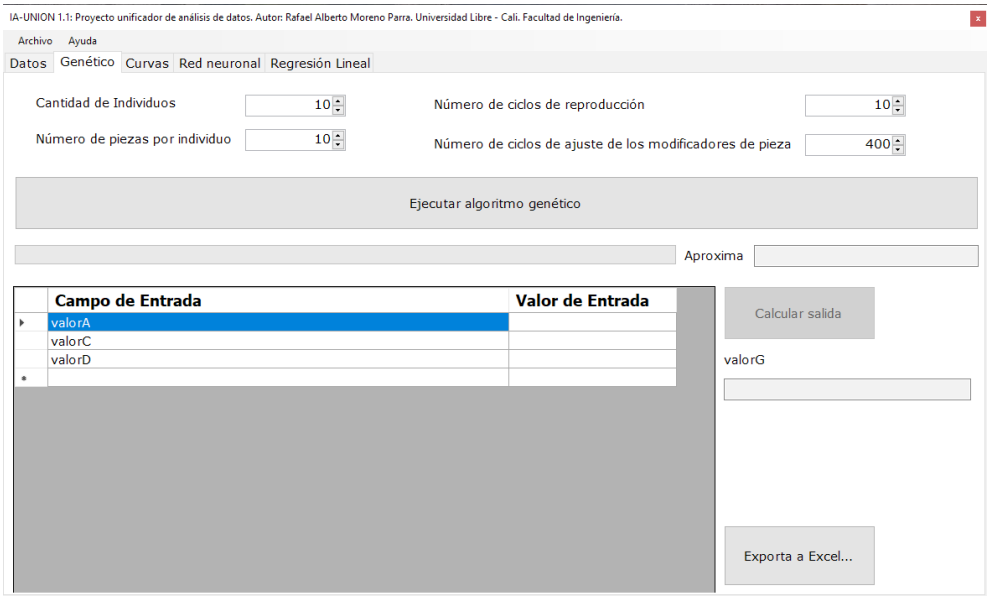


Ilustración 27: Pestaña para el proceso de Algoritmo Genético

Fuente: propia del autor

Las opciones que tiene este procedimiento son:

Cantidad de individuos: Número de individuos que tendrá la población. Un número alto añadirá variabilidad para encontrar soluciones, pero hará más lenta la búsqueda de la adaptación de los individuos. Un número bajo hará que la aproximación corra el riesgo que se atasque en individuos poco adaptados (problema de mínimos y máximos locales).

Número de piezas por individuo: Cuantas piezas formarán la ecuación algebraica que representa al individuo. Un número alto hará una ecuación más larga y compleja, pero hará más lenta su variación para encontrar la mejor aproximación. Un número bajo hará que la ecuación sea simple y no pueda aproximarse bien a la salida esperada.

Número de ciclos de reproducción: Número de veces que se selecciona el mejor individuo de la población, se elimina el peor individuo de la población, se hace una copia del mejor y se muta esa copia. Un número alto hace más lenta la operación, un número bajo haría una mala aproximación.

Número de ciclos de ajuste de los modificadores de pieza: Número de veces que se toma individuo por individuo, luego se selecciona al azar una pieza y luego un modificador, para que posteriormente se cambie aleatoriamente ese modificador y chequear si este cambio mejora la aproximación.

Se presiona el botón "Ejecutar algoritmo genético" y el proceso comienza a ejecutarse. El tiempo depende de los parámetros y el número de datos, se muestra una barra de progreso. Al finalizar, se muestra la aproximación alcanzada, entre más cercano a cero, más preciso es el individuo hallado que cumple con esas variables de entrada y esa variable de salida.

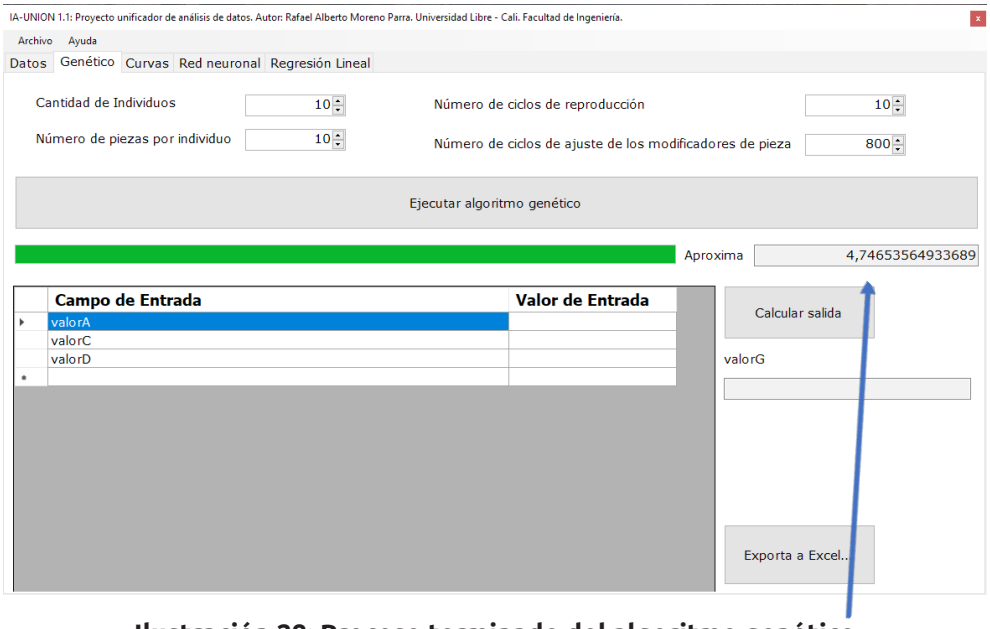


Ilustración 28: Proceso terminado del algoritmo genético

Fuente: propia del autor

Para saber qué valor tendrá la variable de salida, el usuario final escribirá valores numéricos para las variables de entrada en la columna "Valor de Entrada" y presiona el botón "Calcular salida", en la parte inferior aparecerá el valor que obtendría. La recomendación es que los valores que ingresen estén entre los valores máximo y mínimo de cada campo de entrada, eso haría una interpolación que sería más confiable que una extrapolación (cuando se digitan valores por fuera del rango de los valores mínimo y máximo).

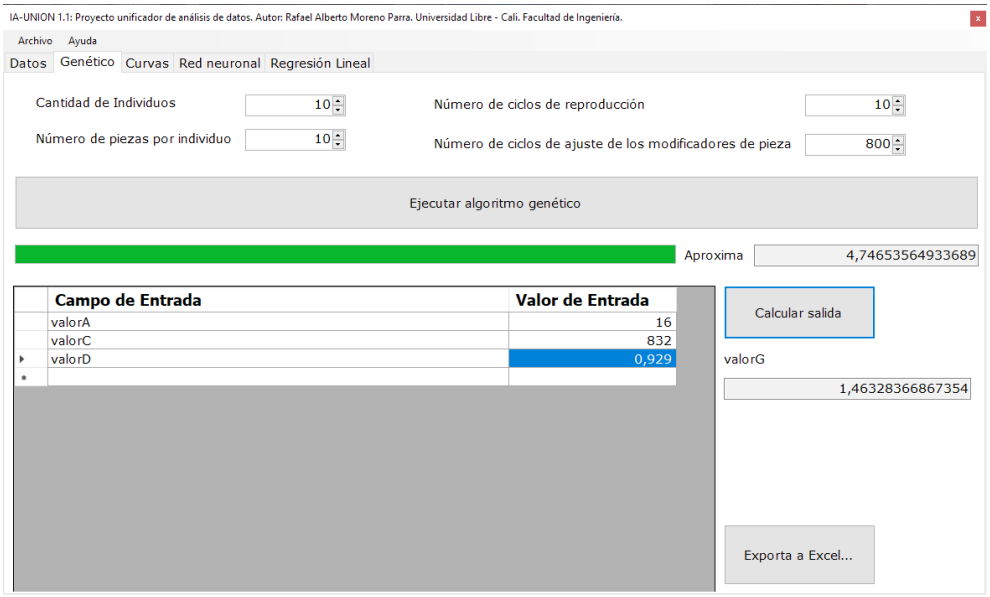


Ilustración 29: Ingreso de valores para variables de entrada y el resultado de la variable de salida

El botón "Exporta a Excel…" envía los datos de entrada y salida normalizados, y el mejor individuo (en forma de ecuación) que se ajusta a esos datos en un archivo TXT.

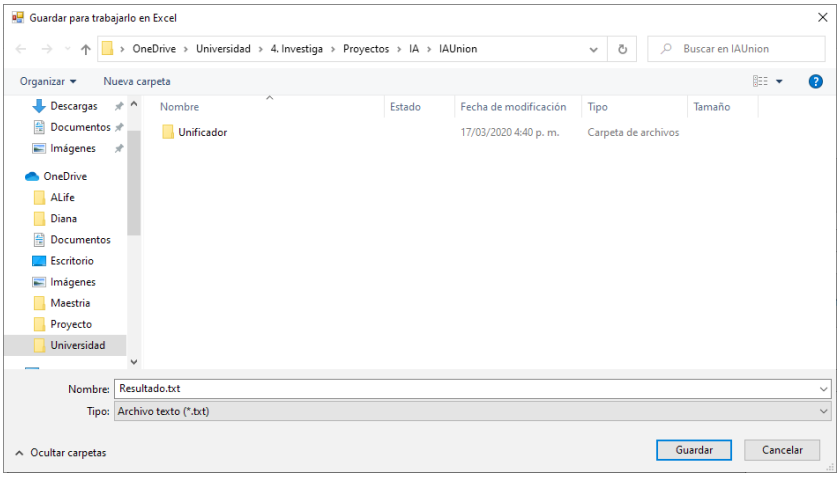


Ilustración 30: Exporta el resultado en un archivo texto para después pasarlo a Excel

Ilustración 31: Enviado a un archivo texto

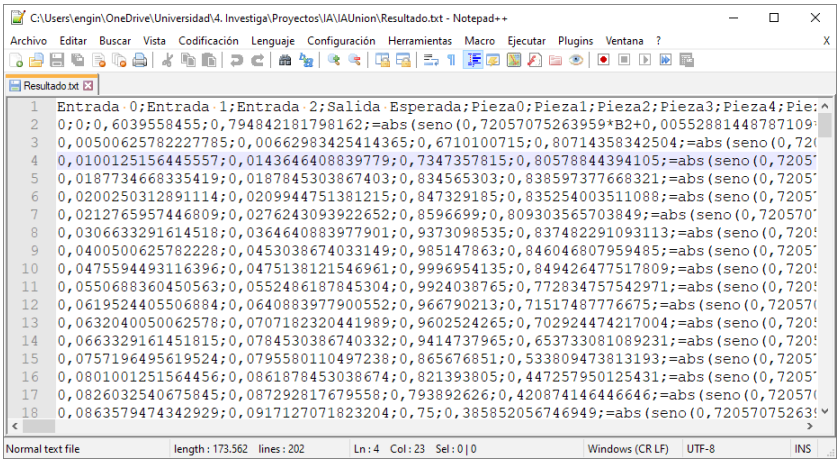


Ilustración 32: Interior de ese archivo texto

Se copia todo ese archivo a memoria y en Excel se usa la opción de "Usar el Asistente para importar texto…"

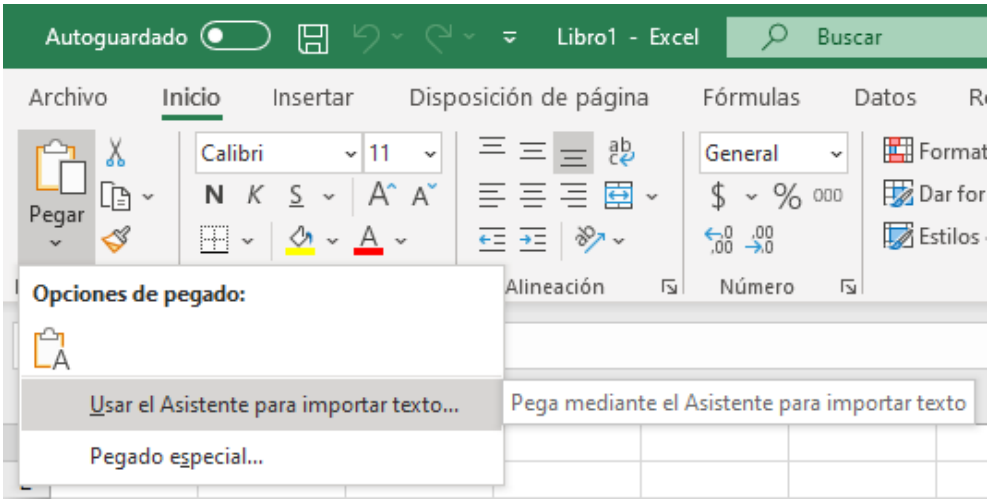


Ilustración 33: Seleccionar "Usar el Asistente para importar texto..."

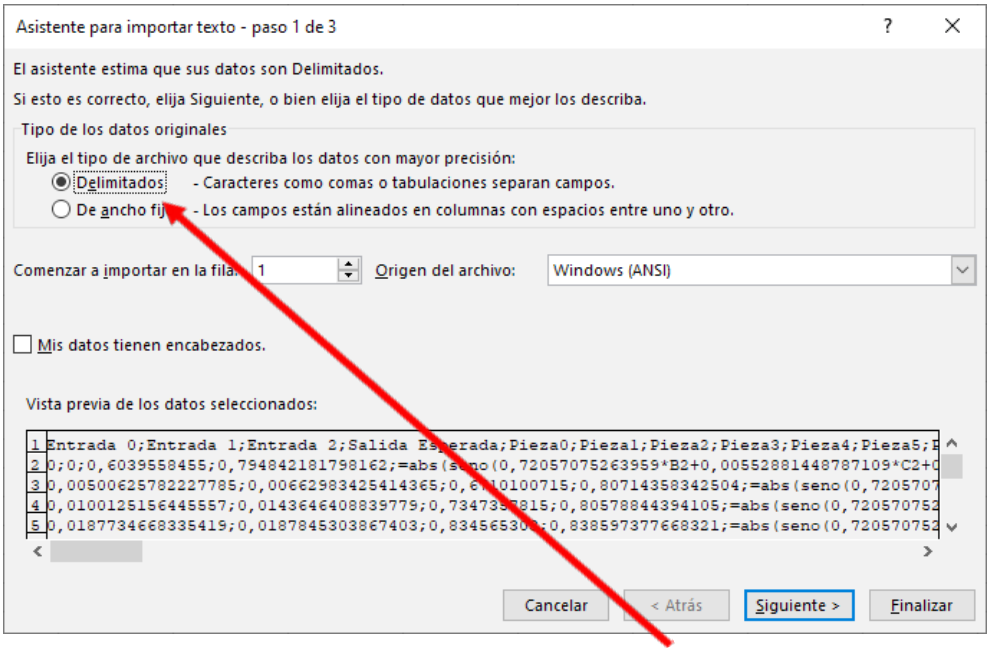


Ilustración 34: Escoger "Delimitados"

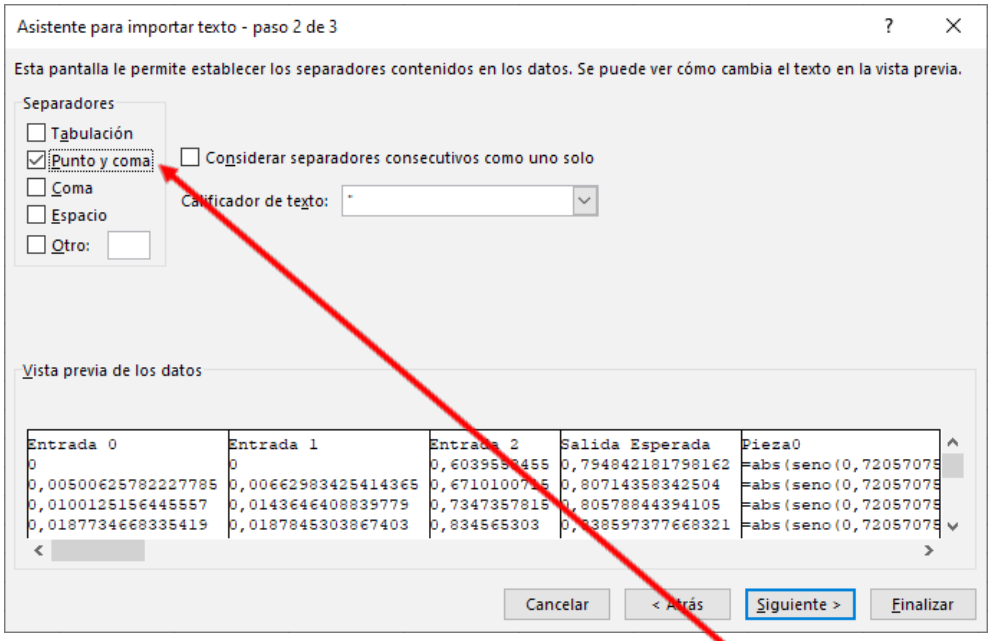


Ilustración 35: Y luego escoger "Punto y coma"

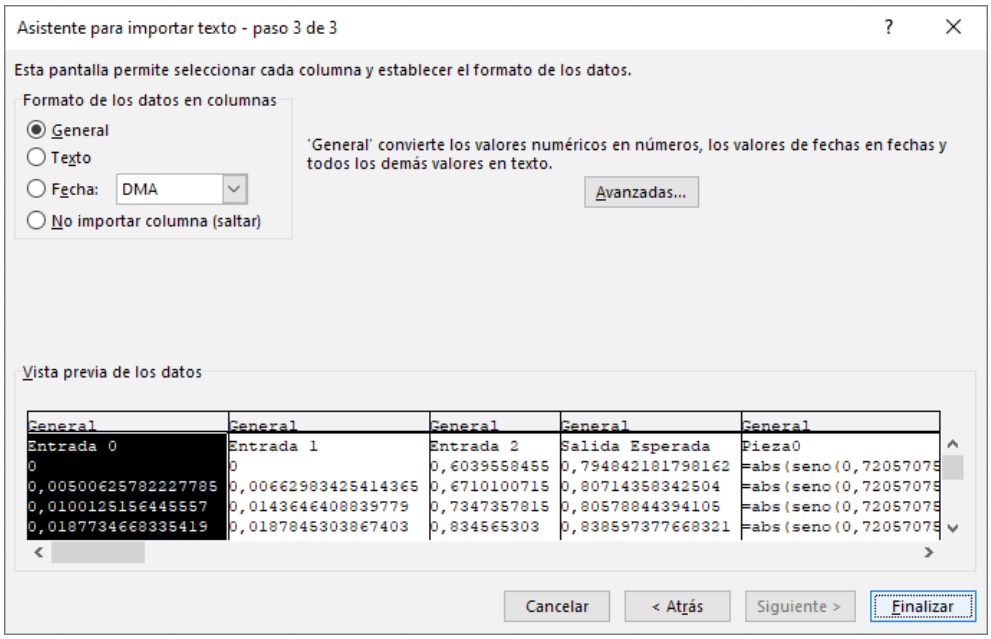
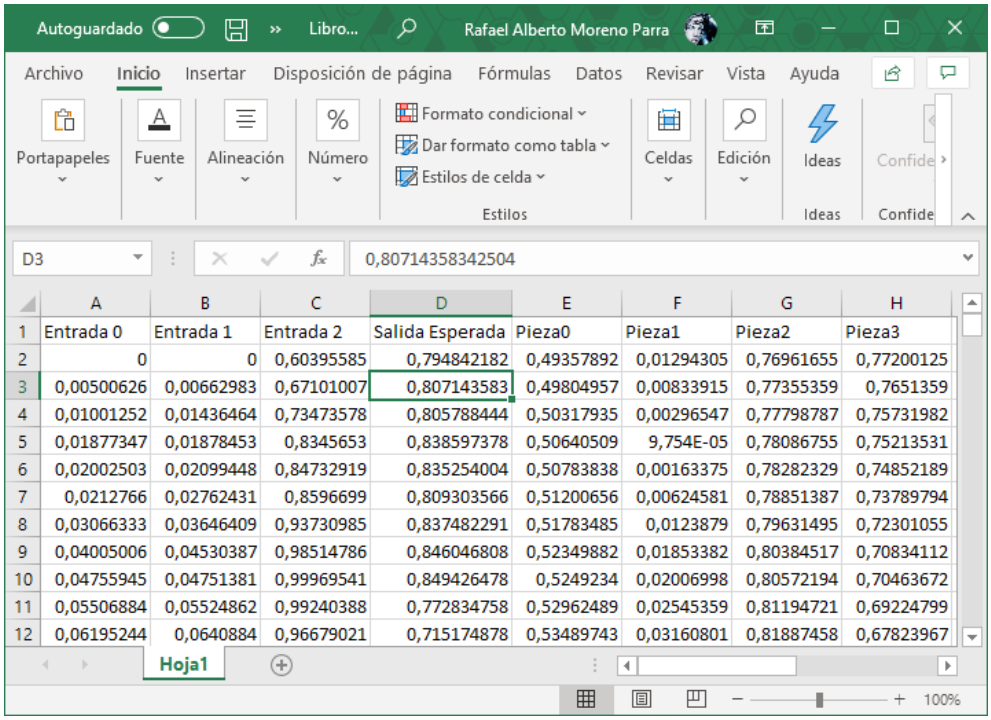


Ilustración 36: Luego clic en "Finalizar"

Finalizada la carga en Excel, se presenta la hoja de cálculo donde en las primeras columnas están los valores de entrada y luego el valor de "Salida Esperada". Esos valores están normalizados. En las columnas a la derecha de "Salida Esperada" están las diversas piezas o partes de la ecuación que generaría esa secuencia, por lo que se debe dirigir a la última columna llamada "Salida Calculada".

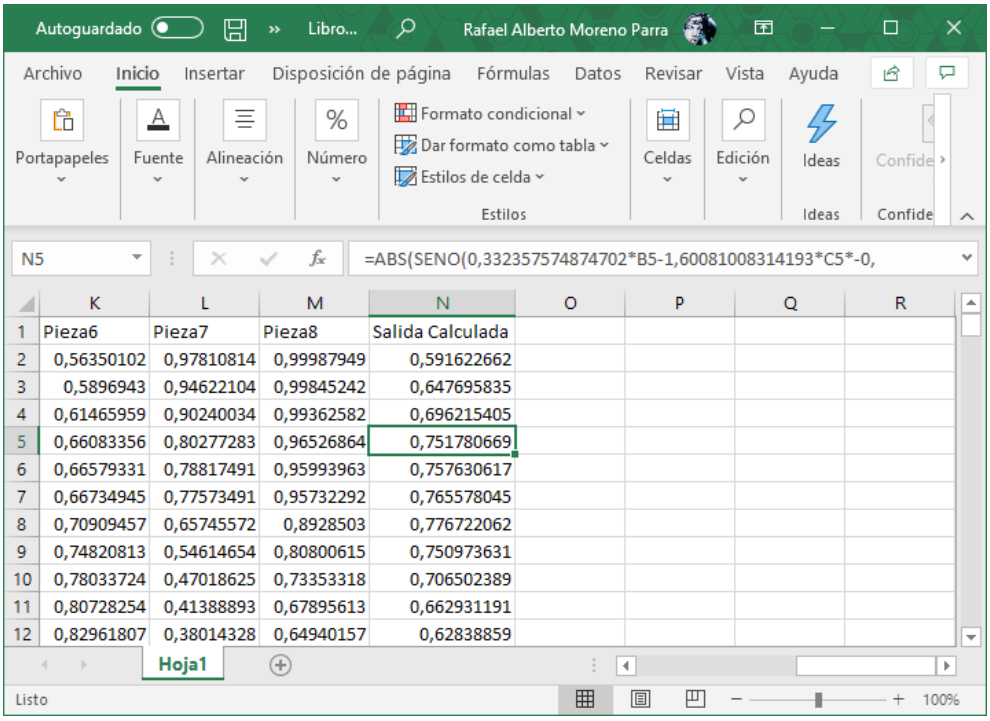


	A	B	C	D	E	F	G	H
1	Entrada 0	Entrada 1	Entrada 2	Salida Esperada	Pieza0	Pieza1	Pieza2	Pieza3
2	0	0	0,60395585	0,794842182	0,49357892	0,01294305	0,76961655	0,77200125
3	0,00500626	0,00662983	0,67101007	0,807143583	0,49804957	0,00833915	0,77355359	0,7651359
4	0,01001252	0,01436464	0,73473578	0,805788444	0,50317935	0,00296547	0,77798787	0,75731982
5	0,01877347	0,01878453	0,8345653	0,838597378	0,50640509	9,754E-05	0,78086755	0,75213531
6	0,02002503	0,02099448	0,84732919	0,835254004	0,50783838	0,00163375	0,78282329	0,74852189
7	0,0212766	0,02762431	0,8596699	0,809303566	0,51200656	0,00624581	0,78851387	0,73789794
8	0,03066333	0,03646409	0,93730985	0,837482291	0,51783485	0,0123879	0,79631495	0,72301055
9	0,04005006	0,04530387	0,98514786	0,846046808	0,52349882	0,01853382	0,80384517	0,70834112
10	0,04755945	0,04751381	0,99969541	0,849426478	0,5249234	0,02006998	0,80572194	0,70463672
11	0,05506884	0,05524862	0,99240388	0,772834758	0,52962489	0,02545359	0,81194721	0,69224799
12	0,06195244	0,0640884	0,96679021	0,715174878	0,53489743	0,03160801	0,81887458	0,67823967

Ilustración 37: Datos de entrada y el de salida esperada cargados (están normalizados)

En la columna de "Salida Calculada" está el valor encontrado por la expresión que mejor se ajustó a los datos de salida esperados.

Hay que recordar que la expresión algebraica puede ser muy larga por lo que se divide en piezas.



	K	L	M	N	O	P	Q	R
1	Pieza6	Pieza7	Pieza8	Salida Calculada				
2	0,56350102	0,97810814	0,99987949	0,591622662				
3	0,5896943	0,94622104	0,99845242	0,647695835				
4	0,61465959	0,90240034	0,99362582	0,696215405				
5	0,66083356	0,80277283	0,96526864	0,751780669				
6	0,66579331	0,78817491	0,95993963	0,757630617				
7	0,66734945	0,77573491	0,95732292	0,765578045				
8	0,70909457	0,65745572	0,8928503	0,776722062				
9	0,74820813	0,54614654	0,80800615	0,750973631				
10	0,78033724	0,47018625	0,73353318	0,706502389				
11	0,80728254	0,41388893	0,67895613	0,662931191				
12	0,82961807	0,38014328	0,64940157	0,62838859				

Ilustración 38: Salida Calculada es el resultado de la mejor expresión que se ajustó a la salida esperada

Con esta tabla en Excel, puede entonces hacer pruebas de interpolación y extrapolación.

Curvas de aproximación

Este método es experimental. Dada la siguiente ecuación fija:

$$Salida = seno(m_0 * E_0) * (m_1 + m_2 * E_0) + seno(m_3 * E_1) * (m_4 + m_5 * E_1) + seno(m_6 * E_2) * (m_7 + m_8 * E_2) + .. + seno(m_k * E_n) * (m_{k+1} + m_{k+2} * E_n)$$

Donde E_0, E_1, E_2, E_n son las diferentes variables de entrada seleccionadas por el usuario y m_0, m_1, m_2, m_3, m_4, m_5 son los coeficientes. El proceso consiste en deducir los valores de esos coeficientes que logren el mejor ajuste. Este sería el algoritmo:

```
Algoritmo Curva de Aproximación
Inicio
 Dar valores al azar a los coeficientes
 Desde ciclo=1 hasta Número de ciclos para aproximar a paso 1 hacer
  Evaluar ajuste de la ecuación
  Tomar un coeficiente al azar y modificarlo al azar
  Si el nuevo valor mejora el ajuste se conserva, caso contrario se retorna al valor anterior
 Fin desde
Fin
```

Este es el código:

Parte de Form1.cs

```
// ************************************************************************
// PROCESO DE USO DE CURVAS DE APROXIMACIÓN
// ************************************************************************
private void btnCurva_Click(object sender, EventArgs e) {
  if (bgwCurva.IsBusy != true) {
    BtnCurvaSalida.Enabled = false;
```

```
        //Si el hilo no está trabajando
        bgwCurva.RunWorkerAsync();
      }
    }
    private void ProcesoCurva(object sender, DoWorkEventArgs e) {
      //Numero de entradas
      int numEntrada = objDato.ValorDatos[0].Count - 1;
      int totalCiclos = (int)numCiclosCurva.Value;

      //Detecta el número de coeficientes a usar
      int totalCoef = (objDato.ValorDatos[0].Count - 1) * 3;

      //Crea la lista de coeficientes
      Random azar = new Random();
      coeficientes = new List<double>();
      coeficientes.Clear();
      for (int num = 0; num < totalCoef; num++) {
        coeficientes.Add(azar.NextDouble() * 2 - 1);
      }

      AproximaCurva = double.MaxValue;

      //Proceso como tal con el algoritmo similar al de búsqueda del menor
costo en ruta
      for (int ciclo = 1; ciclo <= totalCiclos; ciclo++) {
        //Para reportar al hilo
        bgwCurva.ReportProgress(ciclo * 100 / totalCiclos);

        int pos = azar.Next(coeficientes.Count);
        double tmp = coeficientes[pos];
        coeficientes[pos] += 2 * azar.NextDouble() - 1;

        //Va por cada registro del dataset
        double aproxima = 0;
        for (int Registro = 0; Registro < objDato.ValorDatos.Count;
Registro++) {
          double calculado = 0;

          //Va por cada variable de entrada y le aplica la plantilla
          for (int varEntra = 0, punto = 0; varEntra < numEntrada;
varEntra++, punto += 3) {
            double A = objDato.ValorDatos[Registro][varEntra];
            calculado += Math.Sin(coeficientes[0 + punto] * A) *
(coeficientes[1 + punto] + coeficientes[2 + punto] * A);
          }
```

```
            //Calcula la aproximación
            double esperado = objDato.ValorDatos[Registro][numEntrada];
            aproxima += (calculado - esperado) * (calculado - esperado);
            if (aproxima > AproximaCurva) {
              coeficientes[pos] = tmp;
              break;
            }
          }
          if (aproxima < AproximaCurva) AproximaCurva = aproxima;
        }
      }

      private void bgwCurva_ProgressChanged(object sender,
ProgressChangedEventArgs e) {
        ProgresoCurva.Value = e.ProgressPercentage;
      }

      private void bgwCurva_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e) {
        BtnCurvaSalida.Enabled = true;
        txtAproxCurva.Text = Convert.ToString(AproximaCurva);
      }

      private void BtnCurvaSalida_Click(object sender, EventArgs e) {
        //Numero de entradas
        int numEntrada = objDato.ValorDatos[0].Count - 1;

        //Va por cada variable de entrada y le aplica la plantilla
        double calculado = 0;
        for (int varEntra = 0, punto = 0; varEntra < numEntrada; varEntra++,
punto += 3) {
          double A = Convert.ToDouble(dgCurvaInterpolar.Rows[varEntra].
Cells[1].Value);
          A = objDato.Normalizado(A, varEntra); //Debe normalizarlo
          calculado += Math.Sin(coeficientes[0 + punto] * A) * (coeficientes[1 +
punto] + coeficientes[2 + punto] * A);
        }

        double SalidaCalculada = objDato.Desnormalizado(calculado, objDato.
ValorDatos[0].Count - 1);

        //Escribe la salida desnormalizada
        txtCurvaSalida.Text = SalidaCalculada.ToString();
      }
```

Uso en el unificador

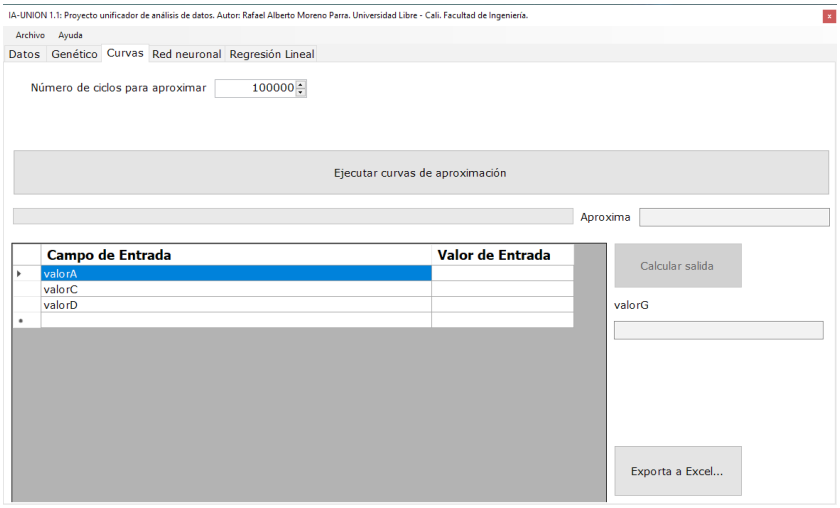


Ilustración 39: Pestaña del proceso de curva de aproximación

Sólo hay un parámetro y es el número de ciclos para aproximar, luego el usuario debe presionar el botón ”Ejecutar curvas de aproximación”. Similar al proceso anterior el usuario digita los valores de entrada.

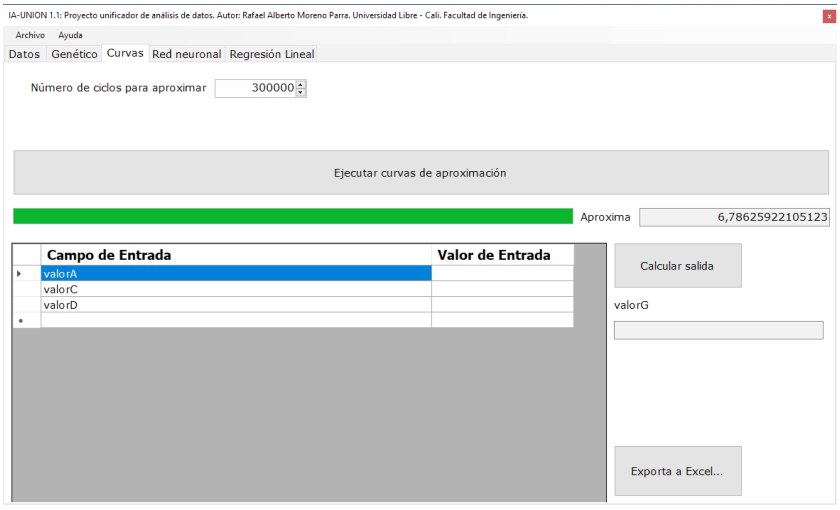


Ilustración 40: Proceso de curva de aproximación terminado

Para saber qué valor tendrá la variable de salida, el usuario final escribirá valores numéricos para las variables de entrada en la columna "Valor de Entrada" y presiona el botón "Calcular salida", en la parte inferior aparecerá el valor que obtendría. La recomendación es que los valores que ingresen estén entre los valores máximo y mínimo de cada campo de entrada, eso haría una interpolación que sería más confiable que una extrapolación (cuando se digitan valores por fuera del rango de los valores mínimo y máximo).

El botón "Exporta a Excel…" envía los datos de entrada y salida normalizados, y la curva de aproximación que se ajusta a esos datos en un archivo TXT:

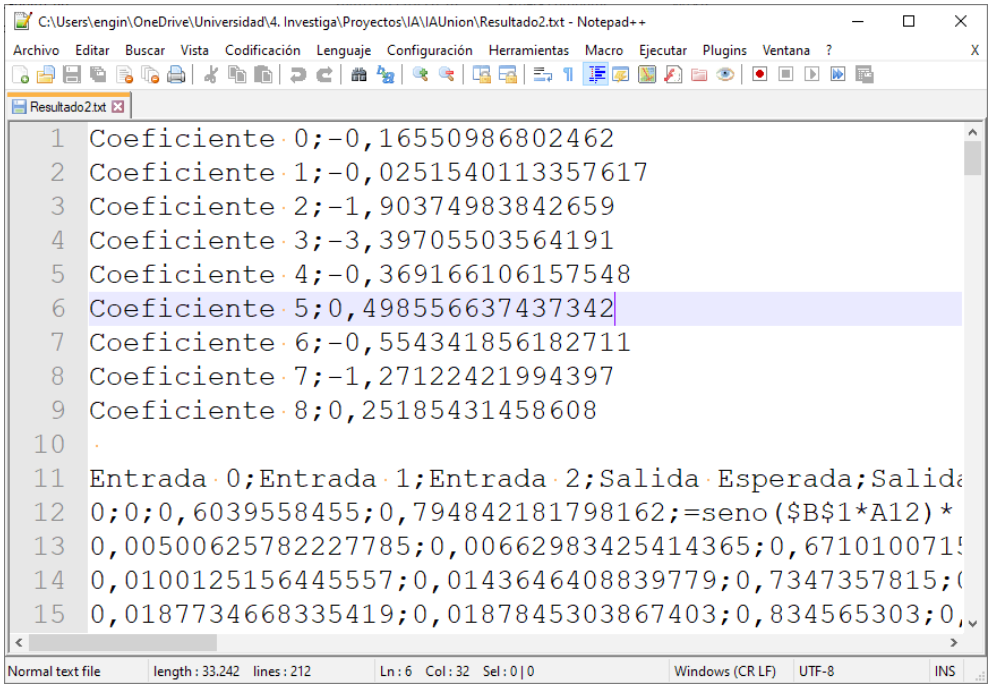
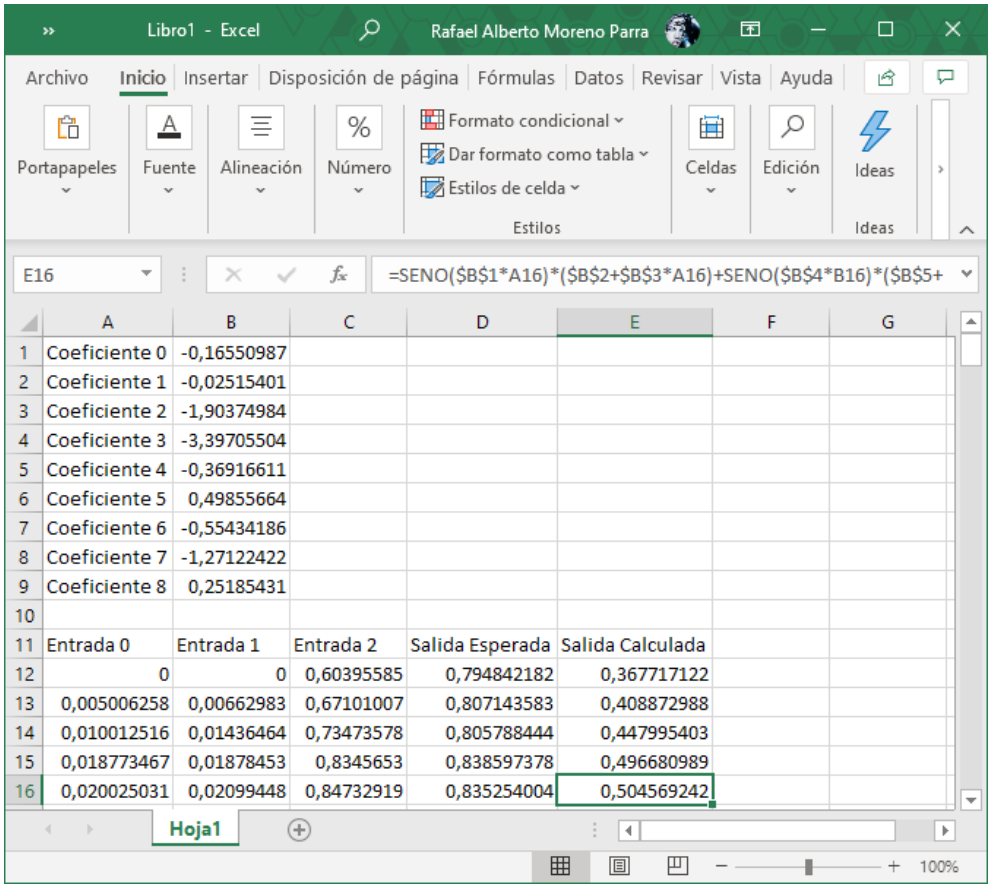


Ilustración 41: Datos de entrada y salida, curva de aproximación para llevarlo a Excel



	A	B	C	D	E	F	G
1	Coeficiente 0	-0,16550987					
2	Coeficiente 1	-0,02515401					
3	Coeficiente 2	-1,90374984					
4	Coeficiente 3	-3,39705504					
5	Coeficiente 4	-0,36916611					
6	Coeficiente 5	0,49855664					
7	Coeficiente 6	-0,55434186					
8	Coeficiente 7	-1,27122422					
9	Coeficiente 8	0,25185431					
10							
11	Entrada 0	Entrada 1	Entrada 2	Salida Esperada	Salida Calculada		
12	0	0	0,60395585	0,794842182	0,367717122		
13	0,005006258	0,00662983	0,67101007	0,807143583	0,408872988		
14	0,010012516	0,01436464	0,73473578	0,805788444	0,447995403		
15	0,018773467	0,01878453	0,8345653	0,838597378	0,496680989		
16	0,020025031	0,02099448	0,84732919	0,835254004	0,504569242		

Ilustración 42: Importación en Excel

La red neuronal

Concepto de red neuronal tipo perceptrón multicapa y el algoritmo "backpropagation"

En el libro "Redes Neuronales. Segunda Edición" [27], se explica paso a paso como funciona una red neuronal tipo perceptrón multicapa y su implementación en C#. El esquema usado es el siguiente:

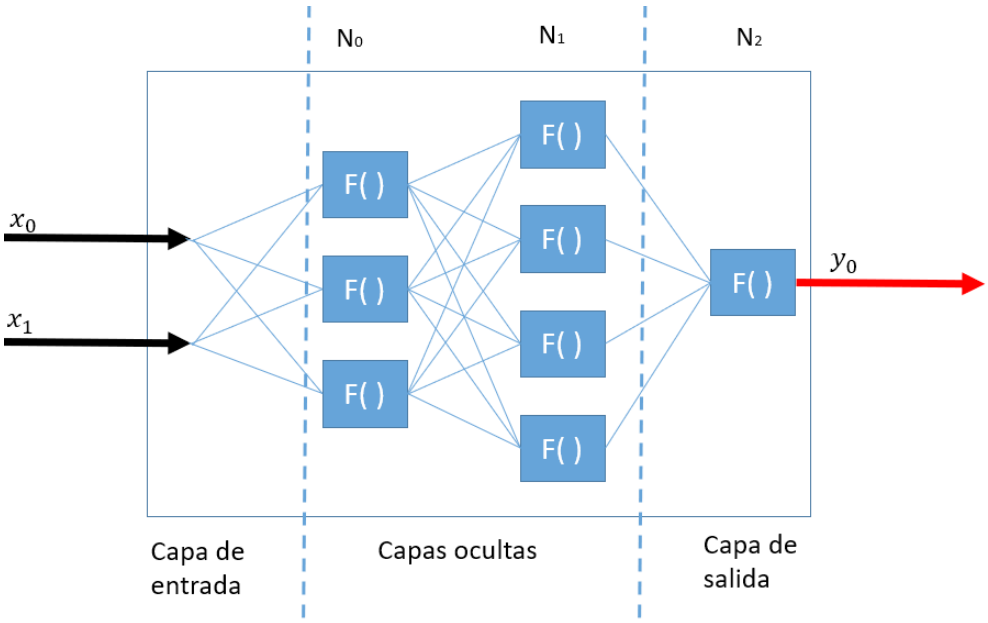


Ilustración 43: Esquema de una red neuronal tipo perceptrón multicapa

El perceptrón multicapa, puede tener de 1 a N entradas, dos capas ocultas y sólo una neurona en la capa de salida. El usuario puede definir cuantas neuronas tendrá cada capa. Si el número es bajo, no habrá buena aproximación, pero si es alto tardará mucho el algoritmo "backpropagation" en hacer un buen entrenamiento de la red neuronal.

Uso en el unificador

En la pestaña "Red neuronal", el usuario decide cuantas neuronas habrá en la capa oculta 1 y cuantas en la capa oculta 2 (mínimo van a haber 2 por capa) y el número de ciclos para entrenar. Presiona el botón "Ejecutar red neuronal (perceptrón multicapa)" y el programa comienza a calcular.

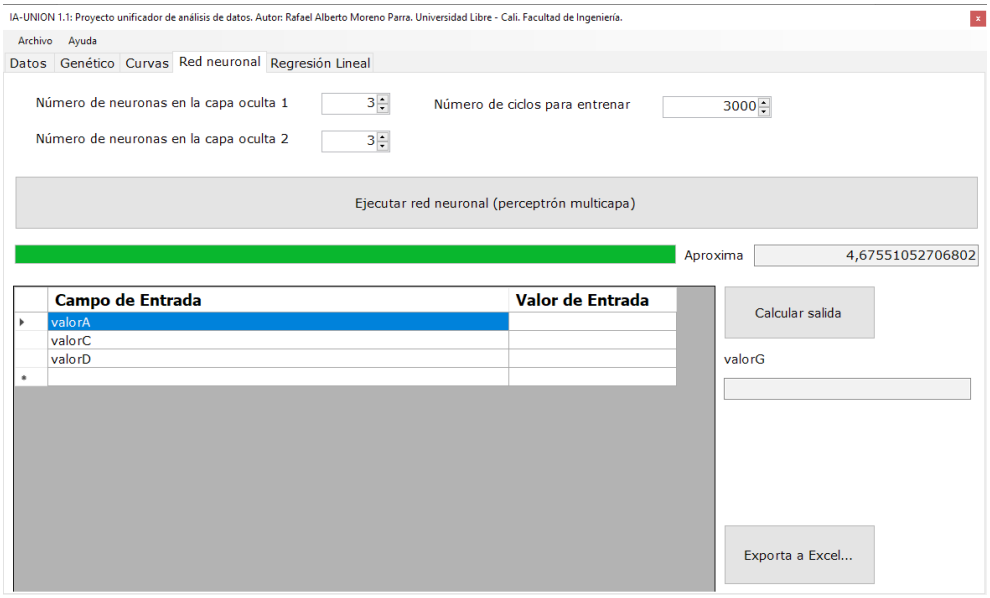


Ilustración 44: Ejecución de la red neuronal

Similar a los anteriores métodos, el usuario final puede ingresar valores a las variables de entrada y calcular un valor estimado de la variable de salida con este método.

También se puede exportar el resultado a Excel, en este caso, se mostrarían los pesos y umbrales de cada neurona de cada capa.

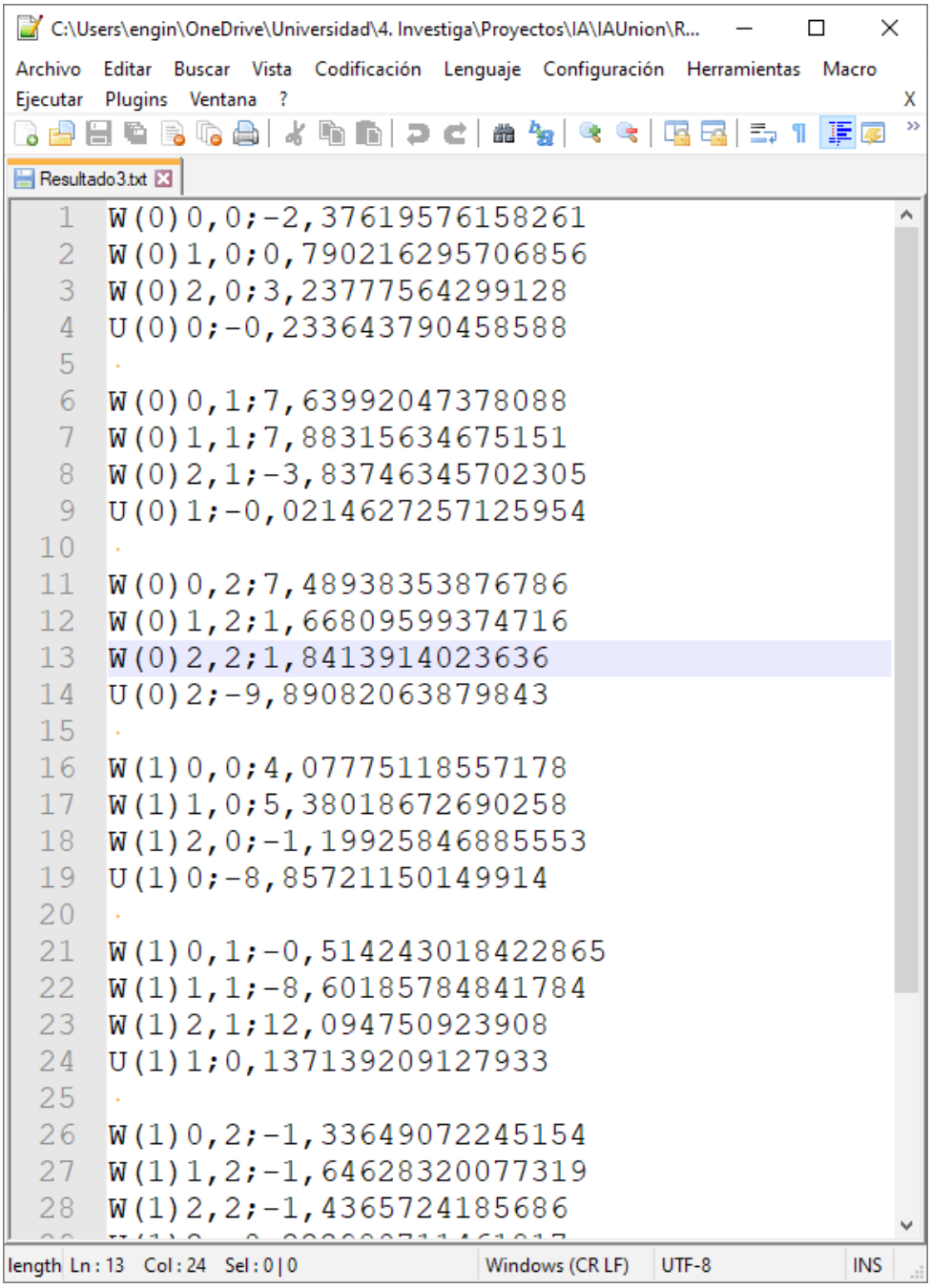


Ilustración 45: Exporta a Excel los pesos y los umbrales de cada neurona de cada capa

Regresión Lineal

El último método es el de regresión lineal, en el que el programa hace uso de un software de terceros, en este caso es Anaconda Individual Edition [28], que es un software de fuente abierta (Open Source). El unificador lo que hace es generar el script en Python que se copia y pega en Anaconda.

Uso en el unificador

En la pestaña" Regresión Lineal", se llenan los valores de entrada y se da clic en "Generar Script de Python y copiarlo al portapapeles (clipboard)", esto genera un script como este:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
datos = pd.read_csv('C:/Users/engin/OneDrive/Universidad/4.
Investiga/Proyectos/IA/MiSoftware/BuscarTendencias.csv')
valor0 = datos['valorA'].values
valor1 = datos['valorB'].values
valor2 = datos['valorC'].values
X = np.array([valor0,valor1]).T
Y = np.array(valor2)
reg=LinearRegression()
reg=reg.fit(X,Y)
Y_pred=reg.predict(X)
aproxima=np.sqrt(mean_squared_error(Y,Y_pred))
print("La aproximación es: ", aproxima)
valor0 = 150
valor1 = 210
print("valorC: ", reg.predict([[valor0,valor1]]))
```

Ese script se pega en Anaconda

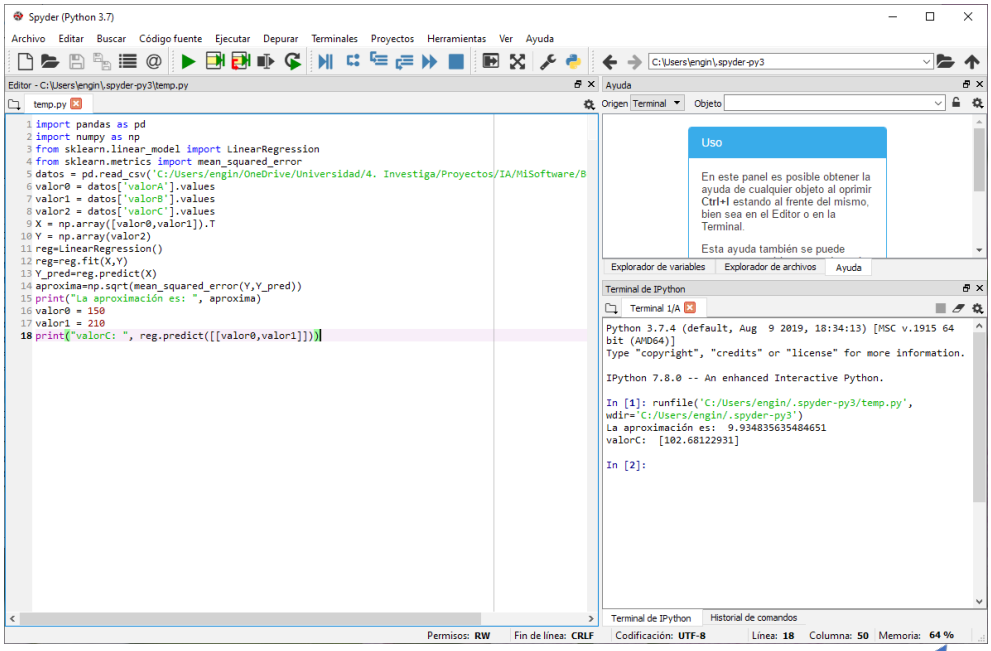


Ilustración 46: Pega el script, se ejecuta y se muestra el resultado de la variable de salida

Se ejecuta y se muestra el resultado de la variable de salida.

Este es el código fuente:

Parte de Form1.cs

```
// ********************************************************************
// PROCESO DE LA REGRESIÓN LINEAL, GENERA EL SCRIPT EN PYTHON
// ********************************************************************

private void btnScript_Click(object sender, EventArgs e) {
  string URLDatos = txtURLDatos.Text;
  URLDatos = URLDatos.Replace('\\', '/');

  txtScript.Text = "import pandas as pd\r\n";
```

```
      txtScript.Text += "import numpy as np\r\n";
      txtScript.Text += "from sklearn.linear_model import LinearRegression\r\n";
      txtScript.Text += "from sklearn.metrics import mean_squared_error\r\n";
      txtScript.Text += "datos = pd.read_csv('" + URLDatos + "')\r\n";

      //Los datos a leer
      int num;
      string nombrecampo, campos = "";
      for (num = 0; num < objDato.ValorDatos[0].Count - 1; num++) {
        nombrecampo = Convert.ToString(dgPYinterpolar.Rows[num].Cells[0].
Value);
        campos += "valor" + num.ToString() + ",";
        txtScript.Text += "valor" + num.ToString() + " = datos['" +
nombrecampo + "'].values\r\n";
      }
      campos = campos.Remove(campos.Length - 1);
      nombrecampo = cboSalida.Text;
      txtScript.Text += "valor" + num.ToString() + " = datos['" +
nombrecampo + "'].values\r\n";

      txtScript.Text += "X = np.array([" + campos + "]).T\r\n";
      txtScript.Text += "Y = np.array(valor" + num.ToString() + ")\r\n";

      txtScript.Text += "reg=LinearRegression()\r\n";
      txtScript.Text += "reg=reg.fit(X,Y)\r\n";
      txtScript.Text += "Y_pred=reg.predict(X)\r\n";
      txtScript.Text += "aproxima=np.sqrt(mean_squared_error(Y,Y_pred))\r\n";
      txtScript.Text += "print(\"La aproximación es: \", aproxima)\r\n";

      for (num = 0; num < objDato.ValorDatos[0].Count - 1; num++) {
        double valorcampo = Convert.ToDouble(dgPYinterpolar.Rows[num].
Cells[1].Value);
        txtScript.Text += "valor" + num.ToString() + " = " + valorcampo.
ToString() + "\r\n";
      }

      txtScript.Text += "print(\"" + nombrecampo + ": \", reg.predict([[" +
campos + "]]))\r\n";
      Clipboard.SetText(txtScript.Text);
    }
```

Implementación en C#

El uso de entorno gráfico

El software es una aplicación gráfica Windows usando el .NET Framework 4.7.2 con el lenguaje de programación C#.

El uso de hilos

Para que el programa no quede paralizado cuando se hacen los procesos de inteligencia artificial, se hacen uso de hilos y para esto se utiliza el control BackGroundWorker, hay uno por cada proceso (algoritmo genético, red neuronal y curvas de aproximación).

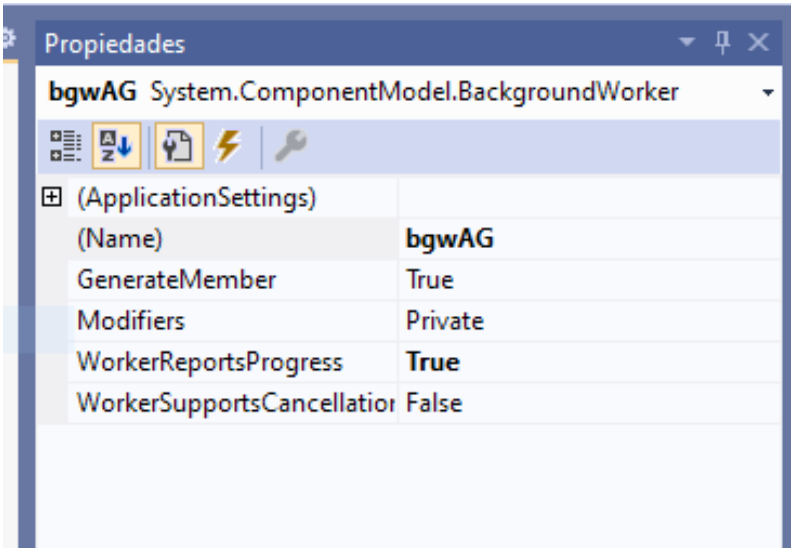


Ilustración 47: Propiedades del componente para uso de hilos

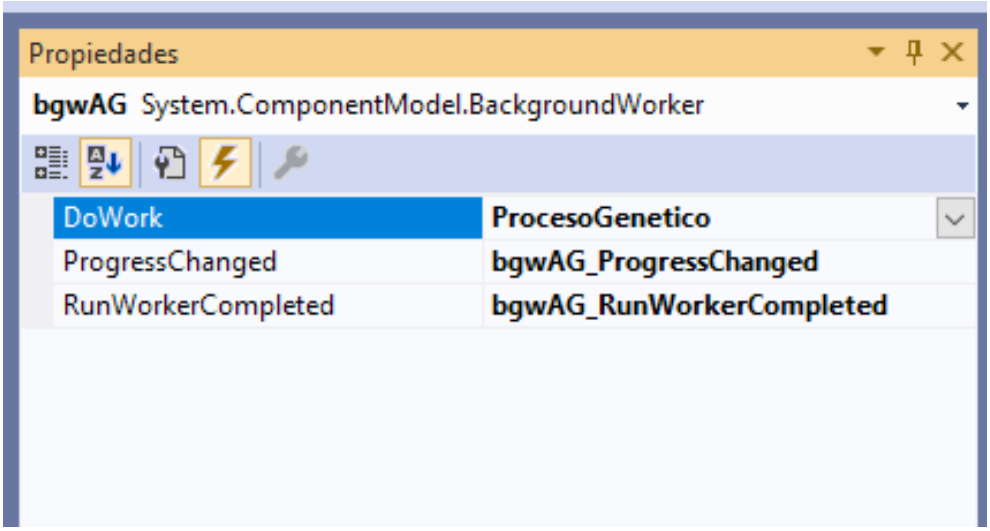


Ilustración 48: Eventos usando por el componente para uso de hilos

Bibliografía

[1] Creative Commons, «Sobre las licencias,» 2020. [En línea]. Available: https://creativecommons.org/licenses/?lang=es. [Último acceso: 14 octubre 2020].

[2] GNU Lesser General Public License, «LICENSES,» 29 junio 2007. [En línea]. Available: https://www.gnu.org/licenses/lgpl-3.0.en.html. [Último acceso: 14 octubre 2020].

[3] Microsoft, «Microsoft Windows,» 2020. [En línea]. Available: http://www.microsoft.com/es-mx/windows. [Último acceso: 14 octubre 2020].

[4] Microsoft, «Visual Studio 2019,» 2020. [En línea]. Available: https://visualstudio.microsoft.com/es/vs/. [Último acceso: 14 octubre 2020].

[5] J. Jiménez, «Científico de datos: así es y así se forma uno en esta profesión cada vez más demandada,» 09 mayo 2020. [En línea]. Available: https://www.xataka.com/otros/cientifico-datos-asi-profesion-demandada. [Último acceso: 14 octubre 2020].

[6] Portafolio, «'Las empresas apuntarán al análisis de datos',» 05 octubre 2020. [En línea]. Available: https://www.portafolio.co/economia/las-empresas-en-colombia-apuntaran-al-analisis-de-datos-545350. [Último acceso: 14 octubre 2020].

[7] Emprendedores, «Las siete mejores herramientas de análisis de datos para conocer mejor tu negocio,» 06 octubre 2020. [En línea]. Available: https://www.emprendedores.es/gestion/las-siete-mejores-herramientas-de-analisis-de-datos/. [Último acceso: 14 octubre 2020].

[8] Grupo BIT, «¿Qué es el análisis de datos y cómo funciona?,» 2020. [En línea]. Available: https://business-intelligence.grupobit.net/blog/que-es-el-analisis-de-datos-y-como-funciona. [Último acceso: 14 octubre 2020].

[9] Marketing Analítico, «Tipos de análisis de datos en la empresa,» 15 octubre 2019. [En línea]. Available: https://www.marketing-analitico.

com/analitica-web/tipos-de-analisis-de-datos-en-la-empresa/. [Último acceso: 14 octubre 2020].

[10] Job Wizards, «Gestión de la complejidad: Simplificar los flujos de trabajo y gestionar los datos,» 2020. [En línea]. Available: https://job-wizards.com/es/gestion-de-la-complejidad-simplificar-los-flujos-de-trabajo-y-gestionar-los-datos/. [Último acceso: 14 octubre 2020].

[11] VirusTotal, «Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community,» octubre 2020. [En línea]. Available: https://www.virustotal.com/gui/home/upload. [Último acceso: 14 octubre 2020].

[12] AddLink, «Ajuste de curvas con regresión lineal y no lineal con Minitab,» 12 marzo 2020. [En línea]. Available: https://www.addlink.es/noticias/minitab/2981-ajuste-de-curvas-con-regresion-lineal-y-no-lineal-con-minitab. [Último acceso: 14 octubre 2020].

[13] MundoWin, «7 mejor software de ajuste de curvas para Windows 10,» 2020. [En línea]. Available: https://mundowin.com/7-mejor-software-de-ajuste-de-curvas-para-windows-10/. [Último acceso: 14 octubre 2020].

[14] Disfruta las matemáticas, «Interpolación y Extrapolación,» 2020. [En línea]. Available: http://www.disfrutalasmatematicas.com/datos/grafica-dispersion-xy.html. [Último acceso: 14 octubre 2020].

[15] Univrsidad de Valencia. Estadística Descriptiva., «Coeficiente de Pearson,» 2020. [En línea]. Available: https://www.uv.es/webgid/Descriptiva/31_coeficiente_de_pearson.html. [Último acceso: 14 octubre 2020].

[16] Machine Learning para todos, «¿Qué es el sobreajuste u overfitting y por qué debemos evitarlo?,» 25 mayo 2020. [En línea]. Available: https://machinelearningparatodos.com/que-es-el-sobreajuste-u-overfitting-y-por-que-debemos-evitarlo/. [Último acceso: 14 octubre 2020].

[17] Sitio Big Data, «Regularización y sobreajuste en Aprendizaje automático,» 2020. [En línea]. Available: https://sitiobigdata.com/2019/12/24/regularizacion-y-sobreajuste-en-aprendizaje-automatico/#. [Último acceso: 14 octube 2020].

[18] Google, «Generalización: Riesgos del sobreajuste,» febrero 2020. [En línea]. Available: https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting?hl=es. [Último acceso: 14 mayo 2020].

[19] Microsoft, «Evitar el sobreajuste y los datos desequilibrados con el aprendizaje automático automatizado,» 09 abril 2020. [En línea]. Available: https://docs.microsoft.com/es-es/azure/machine-learning/concept-manage-ml-pitfalls. [Último acceso: 14 octubre 2020].

[20] Guiainfantil.com, «Pesos y estatura del bebé, niño y niña,» 23 agosto 2019. [En línea]. Available: https://www.guiainfantil.com/salud/embarazo/tabla_pesos.htm. [Último acceso: 14 octubre 2020].

[21] Calculadoras.uno, «Peso Ideal 18 Años Hombres,» 2020. [En línea]. Available: https://www.calculadoras.uno/peso-ideal/Peso_ideal---18-A%C3%B1os-hombre. [Último acceso: 14 octubre 2020].

[22] D. Heaven, «Nace la primera IA capaz de identificar las relaciones causa-efecto,» 12 febrero 2020. [En línea]. Available: https://www.technologyreview.es/s/11849/nace-la-primera-ia-capaz-de-identificar-las-relaciones-causa-efecto. [Último acceso: 14 octubre 2020].

[23] E. M. Trula, «A más margarina, más divorcios: 11 divertidos ejemplos de que correlación no implica causalidad,» 19 octubre 2019. [En línea]. Available: https://magnet.xataka.com/un-mundo-fascinante/a-margarina-divorcios-11-divertidos-ejemplos-que-correlacion-no-implica-causalidad. [Último acceso: 14 octubre 2020].

[24] R. Ferrero, «Correlación Vs Causalidad,» 2020. [En línea]. Available: https://www.maximaformacion.es/blog-dat/correlacion-vs-causalidad/. [Último acceso: 14 octubre 2020].

[25] V. Chandrasekaran, «Correlation or Causation?,» diciembre 2011. [En línea]. Available: https://www.bloomberg.com/news/articles/2011-12-01/correlation-or-causation. [Último acceso: 14 octubre 2020].

[26] Microsoft, «Random Class,» 2020. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netcore-3.1. [Último acceso: 15 octubre 2020].

[27] R. A. Moreno Parra, «Redes Neuronales. Segunda Edición,» 2020. [En línea]. Available: https://github.com/ramsoftware/LibroRedNeuronal2020. [Último acceso: 18 octubre 2020].

[28] Anaconda, «Anaconda Individual Edition,» 2020. [En línea]. Available: https://www.anaconda.com/products/individual. [Último acceso: 18 octubre 2020].

Haciendo una analogía con un examen médico, si ese examen señala un problema de salud ¿Se comenzaría con un tratamiento costoso, doloroso y con indeseados efectos secundarios en forma inmediata? La respuesta más justa sería no; lo más sensato es hacerse otros exámenes, inclusive repetir el primer examen en otro laboratorio clínico para verificar los resultados. Sucede igual con los análisis de los datos de una empresa: Si un tipo de red neuronal muestra un comportamiento que exija tomar difíciles decisiones en la empresa u organización ¿Se haría de forma inmediata? La respuesta sería NO. Hay que hacer otros análisis con otras herramientas distintas para verificar.

El proyecto que se plantea, buscar unir en una misma pantalla, varias técnicas distintas de análisis de datos englobados en el campo de la inteligencia artificial para mostrar al empresario si se han detectado patrones de comportamiento en los datos dados, con diversas variables de entrada. Estas técnicas se ejecutan localmente en el PC del usuario: redes neuronales (perceptrón multicapa), algoritmos genéticos (regresión simbólica), curvas de aproximación o un servicio ofrecido por terceros.

9 789585 545939