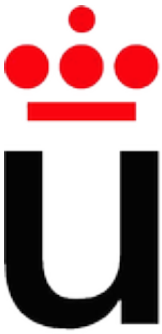


Universidad Rey Juan Carlos

Machine Learning clásico y Redes Parenclíticas. Un análisis comparativo

Trabajo Fin de Máster

Escuela Técnica Superior de Ingeniería Informática

Ingeniería de Sistemas de Decisión

Curso 2016–2017

José Ignacio Escribano Pablos

Tutores:
Ana Elizabeth García Sipols
Miguel Romance del Río

Índice general

Índice de figuras

Índice de tablas

1

Introducción

Almacenar información es un problema que se lleva desarrollando desde el siglo III a.C, cuando Ptolomeo II fundó la biblioteca de Alejandría que almacenaba todo el saber de la época, entre lo que se incluían obras de teatro, poemas, tratados de medicina y matemáticas. La biblioteca llegó a almacenar hasta 900 000 ejemplares, pero la información no era valiosa puesto que se guardaba sin ningún tipo de orden, por lo que era imposible explotar la información almacenada de la biblioteca, hasta que Ptolomeo II contactó con Zenodoto, y decidió ordenar los ejemplares de la biblioteca por orden alfabético, pudiendo consultar la información de forma sencilla y explotar el conocimiento que la biblioteca albergaba [20].

En la actualidad, el crecimiento exponencial de internet genera grandes volúmenes de información relevante que es necesario almacenar en grandes centros de datos. Como pasó con Ptolomeo II, es necesrio buscar nuevas formas de almacenar y procesar estos grandes volúmenes de información para generar valor de la información almacenada. Este nuevo paradigma se conoce como Big Data y tiene tres características fundamentales, también llamada la regla de "las tres uves": volumen, velocidad y variedad [46].
El volumen se refiere a la gran cantidad necesaria de información que se almacena y procesa; ya no se centra en una muestra, sino que se almacena y procesa todo el histórico de datos. La velocidad se refiere a la velocidad que se procesa la información, llegando en algunos casos al tiempo real o "real-time". Por último, la variedad se refiere a que la información no llega en un solo formato, sino que hay que procesar varios de ellos, tales como imágenes, documentos, vídeos, audios, etc.

Para dar valor a la información almacenada se hace uso de algoritmos de Machine Learning o Aprendizaje Automático, que consiste en diseñar e implementar algoritmos que permitan "aprender" a un ordenador a resolver problemas de forma automática, generando soluciones del problema, a partir de unos datos de prueba, y realimentarse para obtener mejores soluciones al problema.

El Machine Learning se ha aplicado en muchos casos de éxito como reconocimiento de texto manuscrito [26], reconocimento facial [44], la predicción de mercados financieros [13], ganando al campeón del mundo de Go [41], o mejorando a algunos humanos jugando a juegos clásicos (Figura 1.1) de Atari [25], entre otros.

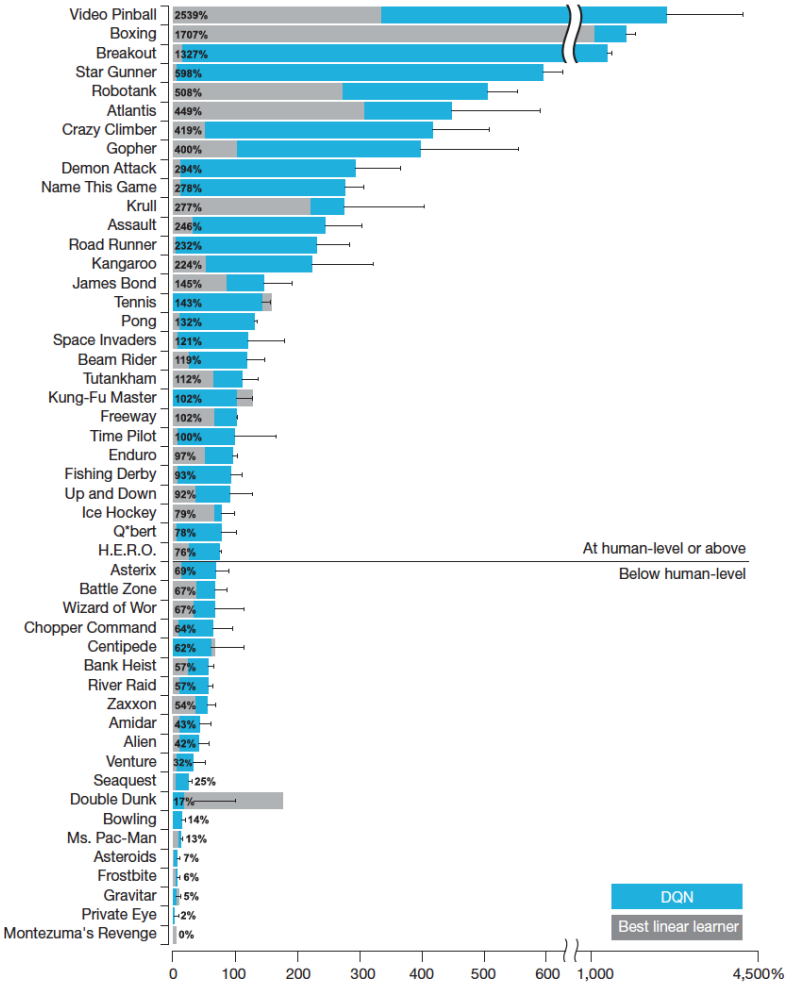


Figura 1.1: Juegos de Atari donde un algoritmo de Machine Learning supera a humanos. Fuente: (25)

En la memoria nos centraremos en el Machine Learning y pondremos énfasis en los tipos que existen y qué algoritmos hay disponibles. Además, describiremos un nuevo método basado en teoría de grafos, y lo compararemos con otros algoritmos clásicos, desarrollando una aplicación que permita analizar conjuntos de datos y comparar entre distintos algoritmos. Por último, usaremos la aplicación para analizar un conjunto de datos.

El proyecto tiene los siguientes objetivos:

1. Introducir qué es el Machine Learning, qué tipos de Machine Learning existen según el tipo de aprendizaje, qué problemas se pueden resolver aplicándolo y familiarizarse con algunos de los algoritmos clásicos.
2. Introducir los conceptos básicos de teoría de grafos, como algunas medidas de redes complejas.
3. Dar a conocer nuevos algoritmos de Machine Learning basados en teoría de grafos, conocidos como redes parenclíticas.
4. Desarrollar una aplicación que automatice el proceso del aprendizaje según distintos algoritmos de Machine Learning (clásicos y basados en teoría de grafos).
5. Analizar una base de datos y comparar los resultados de los distintos algoritmos de Machine Learning.

La memoria está estructurada de la siguiente forma:

En el Capítulo 2 se introduce el concepto de Machine Learning, qué tipos hay según el aprendizaje y qué tareas permite resolver, así como algunos de los algoritmos existentes según el tipo de aprendizaje.

En el Capítulo 3, se introducen brevemente la teoría de grafos, así como un nuevo algoritmo de Machine Learning basado en grafos, las redes parenclíticas.

En el Capítulo 4, se presenta el desarrollo de una aplicación que permita analizar una base de datos utilizando distintos algoritmos de Machine Learning, y compararlos entre ellos.

En el Capítulo 5, se analiza una base de datos usando distintos algoritmos de Machine Learning, y una comparación entre ellos.

En el Capítulo 6, se presentan las conclusiones y el trabajo futuro.

2

Introducción al Machine Learning

El Machine Learning o Aprendizaje Automático es la rama de las Ciencias de la Computación, y en particular, de la Inteligencia de la Artificial que se encarga del reconocimiento de patrones y de la teoría computacional del aprendizaje [22]. El machine learning se basa en la construcción de modelos que permitan aprender y hacer predicciones sobre unos datos, de forma automática, es decir, sin intervención humana [42].

El machine learning permite resolver infinidad de problemas que se pueden clasificar de la siguiente manera:

- **Clasificación:** las entradas son divididas entre dos o más clases y el modelo debe aprender a asignar a cada entrada una de las clases. [23, pág 9]

  Un ejemplo clásico de problema de clasificación es el filtro de spam: el modelo debe ser capaz de identificar un correo electrónico como "spam" o "no spam", es decir, éstas serán las clases en las que se deberán dividir las entradas (por ejemplo, número de apariciones o frecuencia de distintas palabras en el correo) para identificar el spam.

- **Regresión:** las salidas son continuas. En contraposición con la clasificación, la regresión tiene una salida continua, es decir, puede tomar todos los valores reales o un intervalo de ellos. [23, pág 8]

  Por ejemplo, el valor de las acciones de una determinada empresa a lo largo del tiempo es un ejemplo de regresión, ya que el valor de las acciones pueden tomar cualquier valor en el intervalo $[0, \infty)$.

- **Clustering:** un conjunto de entrada se divide en grupos (los grupos no están fijados de antemano). [23, pág 195]

Un ejemplo clásico es la segmentación del mercado, es decir, encontrar grupos con similares características de una población para ofrecer ofertas personalizadas.

- **Reducción de dimensionalidad:** simplifica las entradas por medio de una función a un espacio de dimensión inferior. [23, pág 221]

En el machine learning puede clasificarse por tipo de aprendizaje:

- **Aprendizaje supervisado:** consiste en construir un modelo a partir de un conjunto de entrenamiento que contiene los datos de entrada y la salida (o etiqueta) esperada. El algoritmo produce un modelo para inferir la salida de nuevos ejemplos [23, pág 7].

  En este tipo de aprendizaje se incluye la tarea de clasificación y la regresión.

- **Aprendizaje no supervisado:** en este caso no existen etiquetas predefinidas de antemano, y se trata de encontrar una función que describa la estructura oculta de los datos [23, pág 195].

  En este tipo de aprendizaje se incluye el clustering.

- **Aprendizaje por refuerzo:** un ordenador interactúa con un entorno dinámico para conseguir una determinada recompensa, sin que un nadie le diga como de lejos está de conseguirla [43, pág 4].

A continuación veremos cada uno de estos tipos con más profundidad.

2.1 Aprendizaje supervisado

En el aprendizaje supervisado existe un *conjunto de entrenamiento* que consiste en un conjunto de datos de entrada junto con su correspondiente salida, que es la respuesta que un algoritmo de machine learning debería producir para esa entrada. Normalmente se representa como un vector $(\mathbf{x}, \mathbf{y})$, donde $\mathbf{x} = (x_1, ..., x_n)$ son las entradas y $\mathbf{y} = (y_1, ..., y_n)$, las salidas.

Una característica importante de los algoritmos de machine learning es la capacidad de generalización: el algoritmo debería producir salidas *sensatas* para entradas que no se introdujeron en el entrenamiento. También es importante que el algoritmo pueda tratar con el ruido, es decir, con las imprecisiones que se obtienen al medir cualquier variable del mundo real.

Dentro de este tipo de aprendizaje tenemos varios tipos de problemas, entre los que se encuentran la clasificación y la regresión.

Clasificación

El problema de clasificación consiste en tomar las entradas y decidir a cuál de las n clases pertenece cada entrada, basado en el entrenamiento con ejemplares de cada clase. Un punto clave en el problema de clasificación es que es discreto, es decir, cada ejemplo pertenece a una sola de las clases y el conjunto de clases cubre por completo el espacio de salida.

Ejemplo 1. Consideremos la clasificación de un correo electrónico como "spam" o "no spam". En este caso el conjunto de clases vendría dado por $C = \{\text{spam}, \text{no spam}\}$. Las entradas podrían venir dadas, por ejemplo, por la frecuencia de aparición de distintas palabras claves del correo electrónico y la salida vendría dada por una de las clases (spam, no spam).

Regresión

El problema de regresión consiste en obtener un valor de salida a partir de las entradas. En contraposición con el problema de clasificación, las salidas toman valores sobre un intervalo continuo.

Ejemplo 2. Supongamos que queremos estimar el valor de las acciones de una determinada empresa a partir de una serie de variables como el número de empleados, los ingresos, etc. En este caso estamos ante un problema de regresión ya que la variable salida (valor de las acciones) toma un valor continuo en el intervalo $[0, \infty)$.

Existen diferentes tipos de regresión, aunque solo nos centraremos en la regresión lineal.

Regresión lineal

La regresión lineal viene dada por el modelo [24, pág 255]

$$y_i = \alpha_1 x_{i1} + \cdots + \alpha_p x_{ip} + \alpha_0, \quad i = 1, ..., n \tag{2.1}$$

donde y_i es la variable salida, x_i es la variable de entrada, n es el número de observaciones y p es el número de variables de entrada.

El método de mínimos cuadrados nos garantiza que los parámetros α_i que minimizan el error cuadrático[1] vienen dados por

$$\boldsymbol{\alpha} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.3}$$

[1] El error cuadrático viene dado por

$$\text{EC} = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.2}$$

donde $\hat{y}$ es el valor ajustado de la variable salida y y es el valor real de la variable salida

donde

$$\alpha = (\alpha_1, ..., \alpha_n)^T,$$

$$\mathbf{y} = (y_1, ..., y_n)^T,$$

$$\mathbf{X} = \begin{pmatrix} x_{11} & ... & x_{1p} \\ x_{21} & ... & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

En particular, en el caso bidimensional el modelo viene dado por

$$y = \alpha_1 x + \alpha_0. \tag{2.4}$$

Se tiene que α_0 y α_1 vienen dados por este sistema de ecuaciones lineales:

$$\begin{cases} \left(\sum_{i=1}^{n} x_i^2\right)\alpha_1 + \left(\sum_{i=1}^{n} x_i\right)\alpha_0 = \sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} x_i\right)\alpha_1 + n\alpha_0 = \sum_{i=1}^{n} y_i \end{cases} \tag{2.5}$$

Ejemplo 3. Supongamos que disponemos de los datos de la Tabla 2.1 y queremos ajustar un modelo lineal de la forma $y = \alpha_1 x + \alpha_0$.

De acuerdo a lo anterior, los valores α_0 y α_1 que minimizan el error cuadrático son las soluciones del sistema

$$\begin{cases} 92\alpha_1 + 20\alpha_0 = 25 \\ 20\alpha_1 + 8\alpha_0 = 37 \end{cases}$$

Resolviendo el sistema lineal, se tiene que,

$$\alpha_1 \approx -1.607$$
$$\alpha_0 \approx 8.642$$

Por tanto, $y = -1.607x + 8.642$.
Si quisiéramos predecir el valor para $x = 7$, tendríamos que $y = -1.607 \cdot 7 + 8.642 = -2.607$.

Es importante notar que este método permite ajustar modelos que, en principio, no son lineales como, por ejemplo,

$$y = ax^m$$

Tabla 2.1: Datos para regresión lineal

x	-1	0	1	2	3	4	5	6
y	10	9	7	5	4	3	0	-1

Este modelo no puede ajustarse como regresión lineal, pero se pueden linealizar las variables x, y para convertirlo en un modelo lineal [24, pág 257].

Tomando logaritmos a ambos lados de la igualdad,

$$\log y = \log(ax^m) = \log a + m \log x$$

Haciendo $Y = \log y$, $X = \log x$, $\alpha_1 = \log a$ y $m = \alpha_0$, tenemos un modelo lineal.

En la Tabla 2.2 se pueden encontrar cómo linealizar otros modelos de forma que la regresión lineal resuelve muchos otros casos.

Otros tipos de regresión se pueden encontrar en [12], como la regresión logística o los modelos generales lineales, entre otros.

Tabla 2.2: Linealización de distintos modelos

$y = f(x)$	Forma linealizada $y = \alpha_1 x + \alpha_0$	Cambio de variables y constantes
$y = \frac{\alpha_1}{x} + \alpha_0$	$y = \alpha_1 \frac{1}{x} + \alpha_0$	$X = \frac{1}{x}$; $Y = y$
$y = \frac{1}{\alpha_1 x + \alpha_0}$	$\frac{1}{y} = \alpha_1 x + \alpha_0$	$Y = \frac{1}{y}$; $X = x$
$y = \alpha_1 \log x + \alpha_0$	$y = \alpha_1 \log x + \alpha_0$	$Y = y$; $X = \log x$
$y = \alpha_1 e^{\alpha_0 x}$	$\log y = \log \alpha_1 + \alpha_2 \log x$	$Y = \log y$; $X = \log x$; $\alpha_1 = \log \alpha_1$
$y = (\alpha_0 + \alpha_1 x)^2$	$\sqrt{y} = \alpha_0 + \alpha_1 x$	$Y = \sqrt{y}$; $X = x$

2.1.1 El proceso de Machine Learning supervisado

El proceso general para resolver un problema usando aprendizaje supervisado consiste en los siguientes pasos [23]:

1. **Obtención de datos y preparación:** consiste en obtener y preparar los datos que se usarán para obtener un modelo de machine learning adecuado para los datos. Consiste en obtener unos datos que sean relevantes, tarea que es difícil cuando se dispone de una cantidad de datos muy grande y que contiene outliers y datos faltantes.

2. **Selección de características:** consiste en la identificación de características que sean más útiles para el problema en cuestión.

3. **Elección del algoritmo:** dado el conjunto de datos, consiste en elegir uno o varios algoritmos adecuados que permita resolver el problema de manera satisfactoria.

4. **Selección del modelo y sus parámetros:** la mayoría de los algoritmos de Machine Learning tienen parámetros que deben ser fijados manualmente, o que requieren experimentación para obtener valores adecuados.

5. **Entrenamiento:** dado el conjunto de datos, el algoritmo y los parámetros, el entrenamiento deberá construir un modelo a partir de los datos para predecir las salidas de nuevos datos.

6. **Evaluación:** antes de que el sistema sea desplegado, necesita ser probado y evaluado con datos con los que no ha sido entrenado. A veces, incluye una comparación con expertos humanos en el campo y la selección de métricas apropiadas para esa comparación.

De los 6 puntos anteriores, nos centraremos en la elección del algoritmo. De entre todos los algoritmos entraremos en detalle de las redes neuronales, los Support Vector Machine (SVM) y los árboles de decisión, por ser ampliamente estudiados y están basados en conceptos diferentes. Las redes neuronales se basan en el funcionamiento biológico de las neuronas, los SVM en transformar los datos a un espacio de dimensión superior y el uso de kernels, y los árboles de decisión que se basan en obtener reglas fácilmente entendibles por humanos.

2.1.2 Redes neuronales

Las redes neuronales están basadas en el modo que funcionan las neuronas en el cerebro. La operación general de la misma es transmitir químicos dentro del fluido del cerebro para aumentar o disminuir el potencial eléctrico dentro del cuerpo de la neurona. Si el potencial de la neurona alcanza algún determinado umbral, la neurona se activa y un pulso de duración fija se envía al axón. El axón se divide en conexiones a muchas otras neuronas, conectando a estas neuronas en una sinapsis [23, pág 11].

En 1943, McCulloch y Pitts propusieron un modelo matemático simplificado del funcionamiento de una neurona con las siguientes características (Figura 2.2):

- Un conjunto de pesos w_i que corresponden con la sinapsis.
- Un sumador Σ que suma las señales entrantes (equivalente a la membrana de la célula que recoge la carga eléctrica)
- Una función de activación g que decide si la membrana se activa o no para las entradas actuales.

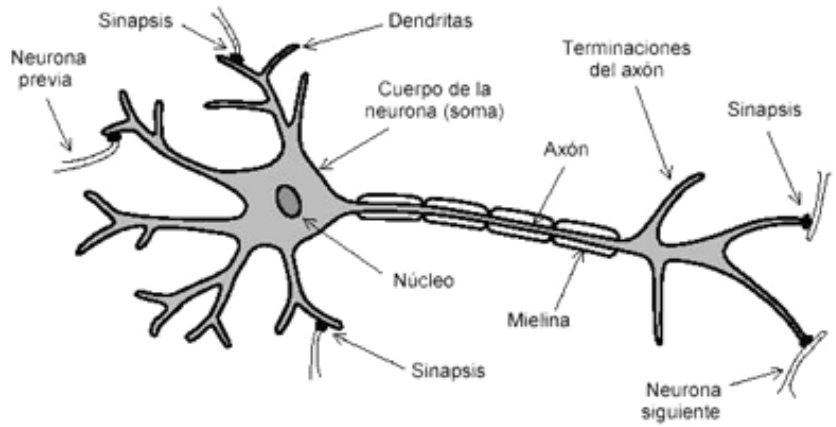


Figura 2.1: Estructura de una neurona

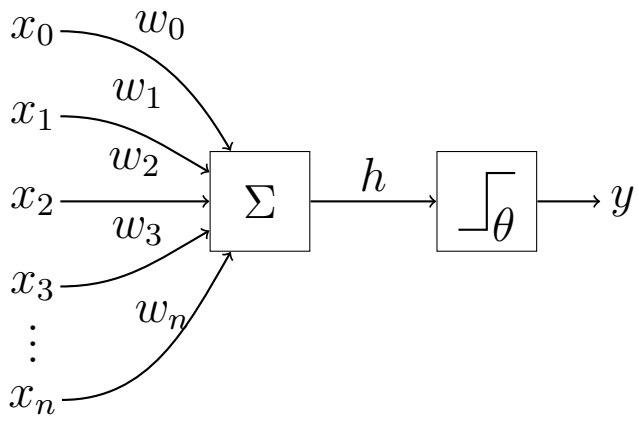


Figura 2.2: Neurona de McCulloch y Pitts

Llamaremos h a a la suma de las entradas multiplicadas por los pesos.

Es decir h es el valor,

$$h = \sum_{i=1}^{n} w_i x_i. \tag{2.6}$$

Si $h > \theta$ (valor fijado), la neurona se activará. Matemáticamente,

$$y = g(h) = \begin{cases} 1 & \text{si } h > \theta \\ 0 & \text{si } h \leq \theta \end{cases} \tag{2.7}$$

Un problema obvio de la neurona de McCulloch y Pitts es que sólo puede activarse o no hacerlo, por lo que no puede aprender. Para ello, necesitamos poner neuronas juntas formando una red neuronal [23, pág 11].

El **perceptrón** es la red neuronal más sencilla, ya que no es más que una colección de neuronas de MacCulloch y Pitts (Figura 2.3).

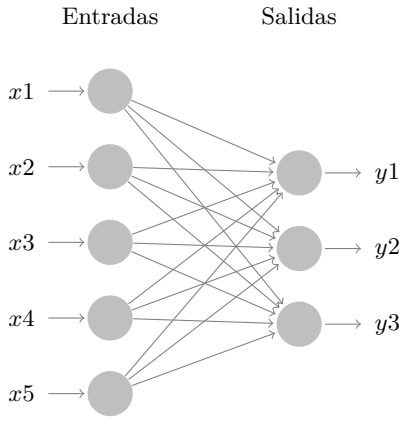


Figura 2.3: Perceptrón

A la izquierda se sitúan las entradas y a la derecha se muestran las neuronas. Éstas son completamente indepedendientes unas de otras: no importa qué estén haciendo las otras neuronas, la neurona se activará multiplicando sus pesos por las entradas, sumando el resultado y comparando el resultado con su umbral, sin importar lo que estén haciendo las demás neuronas.

Una limitación importante del perceptrón es que sólo es capaz de clasificar conjuntos linealmente separables[2]. Novikoff probó que el algoritmo de aprendizaje de un perceptrón converge en un número finito de iteraciones si los datos son linealmente separables [29].

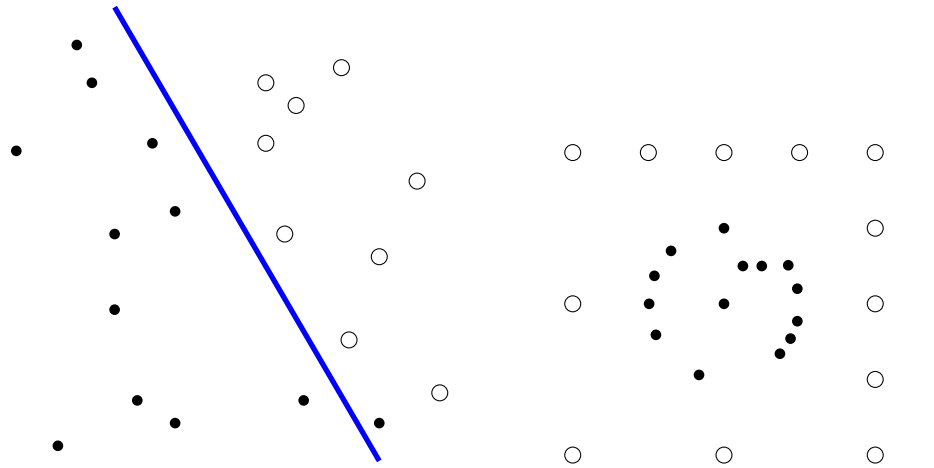
Un ejemplo clásico que no puede aprender un perceptrón es la función XOR: $\{0,1\} \times \{0,1\} \to \{0,1\}$, que se define como se ve en la Tabla 2.3.

Tabla 2.3: Definición de la función XOR

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Si representamos estos puntos en el plano xy (Figura 2.5), veremos que no podemos encontrar ningún hiperplano (en este caso, recta) que divida a las dos clases.

[2]Un conjunto es linealmente separable si existe un hiperplano que separe dos clases.



(a) Linealmente separable (b) No linealmente separable

Figura 2.4: Conjuntos linealmente separables y no separables

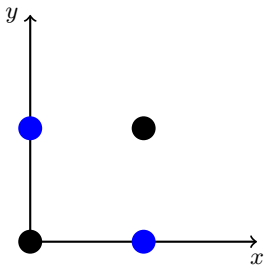


Figura 2.5: Función XOR

Para suplir la limitación anterior, nació el perceptrón multicapa, que consiste en múltiples capas de neuronas (Figura 2.6).

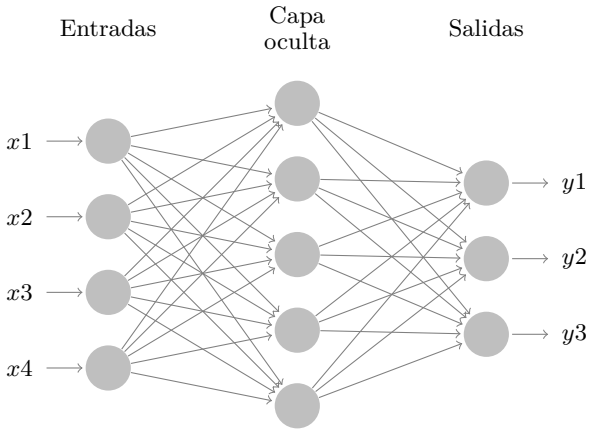


Figura 2.6: Perceptrón multicapa

El teorema de aproximación universal establece que un perceptrón multicapa con una capa oculta puede aproximar funciones continuas sobre conjuntos compactos de $\mathbb{R}^n$ [6].

Una descripción de los algoritmos del perceptrón multicapa se pueden encontrar en [1, 23, 26].

2.1.3 Support Vector Machine

Los Support Vector Machine (SVM) son uno de los algoritmos más populares en machine learning. Fueron introducidos en 1992 por Vapnik [4] y se han empleado en multitud de aplicaciones desde entonces, debido principalmente a que consigue una gran tasa de clasificación en conjuntos de datos con tamaño razonable. Los SVM no funcionan bien en conjuntos de datos muy grande, ya que los cálculos no escalan bien con el número de datos, y se vuelve computacionalmente muy costoso [23, pág 119].

La Figura 2.7 muestra una clasificación con tres posibles clasificadores lineales. Las tres rectas dividen de forma "correcta" ambas clases, por lo que el perceptrón pararía si encontrara cualquiera de ellas, teniendo varias soluciones.

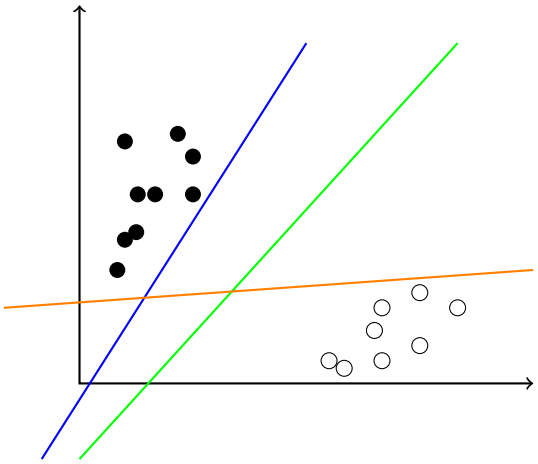


Figura 2.7: Distintos clasificadores lineales

Los SVM se basan el concepto de margen (Figura 2.8), que no es nada más que la región más grande que podemos separar las clases sin que haya puntos dentro, donde la caja se hace con dos líneas paralelas al clasificador lineal. El clasificador que tenga mayor margen se llamará clasificador con mayor margen. Los puntos de cada clase que están más cerca de la recta reciben el nombre de vectores de soporte (support vectors).

Se puede demostrar [1, 23] que el problema anterior se puede plantear como un problema de programación cuadrática con la siguiente estructura:

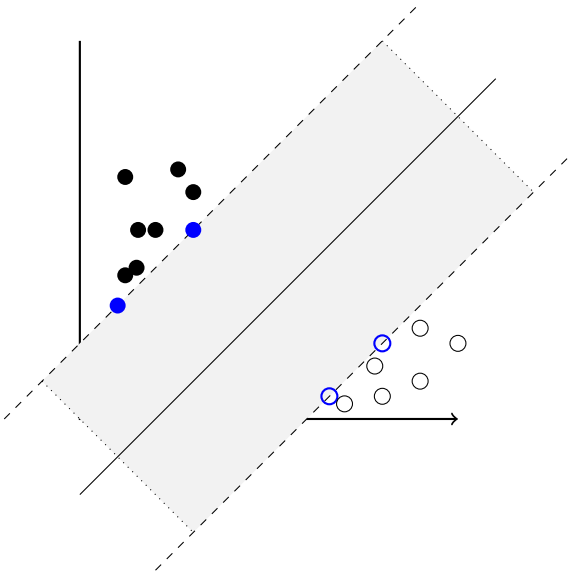


Figura 2.8: Margen en SVM

$$\min \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{2.8}$$

$$\text{s.a.} \quad y_i(\mathbf{w}^T x_i + b) \geq 1 \ \forall i = 1, \ldots, n \tag{2.9}$$

donde $\mathbf{x}$ es el vector de entradas, $\mathbf{w}$ es un vector perpendicular al hiperplano clasificador, b es una constante y y_i es el valor de la salida i-ésima que puede tomar dos valores $\{-1, 1\}$.

Otro concepto fundamental en SVM es el de **kernel**, que consiste en modificar las variables de alguna forma que se puedan separar los datos (Figura 2.9).

Un kernel K es una función que de define como $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T\phi(\mathbf{y})$, donde $\mathbf{x}$, $\mathbf{y}$ son vectores de entradas, ϕ es una función a un espacio de dimensión superior al de entrada [23, pág 117].

Los kernels más habituales son:

- Polinomiales de grado s

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^s \tag{2.10}$$

- Sigmoidales con parámetros κ y δ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} - \delta) \tag{2.11}$$

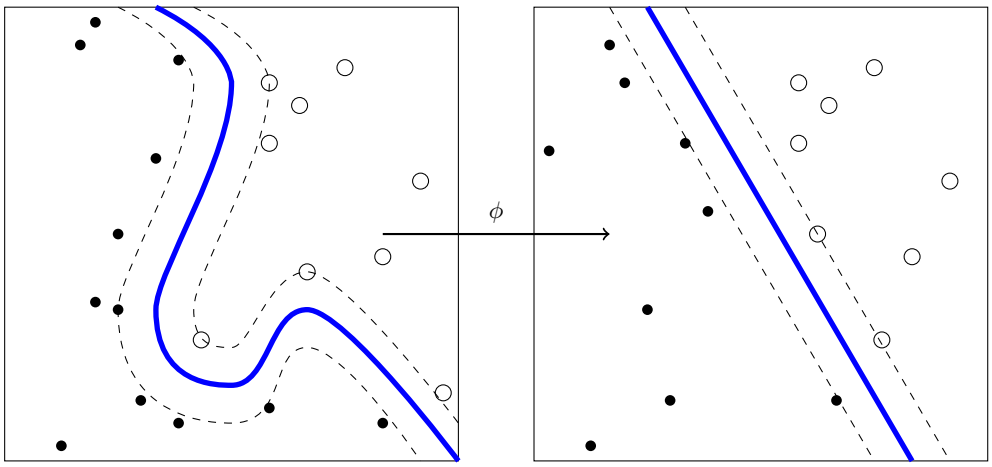


Figura 2.9: Kernel

- Funciones de base radial con parámetro σ

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\left(\frac{\mathbf{x}-\mathbf{y}}{2\sigma^2}\right)^2\right) \tag{2.12}$$

Una descripción de los algoritmos de los Support Vector Machine se puede encontrar en [5].

2.1.4 Árboles de decisión

La idea de los árboles de decisión es partir el conjunto de clasificación en un conjunto de opciones sobre cada variable comenzando por la raíz del árbol y bajando hasta las hojas, donde se reciben la decisión de clasificación.

Ejemplo 4. Supongamos que queremos decidir qué hacer en función del dinero que tengamos y el tiempo que haga. Supongamos que el tiempo sólo puede ser soleado y lluvioso y el dinero que tenemos es mucho o poco.

Queremos decidir si ir al parque, al cine o quedarse en casa.

Así, un posible árbol de decisión se puede ver en la Figura 2.10.

Una de las ventajas de los árboles de decisión es que pueden convertirse en una unión de conjunciones y programarse de la forma “Si [...], Entonces [...]“.

El problema consiste en encontrar un árbol de decisión “óptimo” en algún sentido. El algoritmo ID3 construye árboles de decisión maximizando la ganancia de entropía.

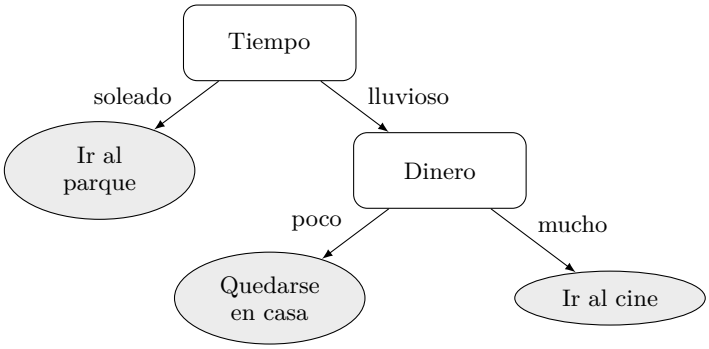


Figura 2.10: Árbol de decisión

2.1.4.1 Algoritmo ID3

El algoritmo ID3 se basa en el concepto de entropía, propuesto por Claude Shannon, padre de la Teoría de la Información. La entropía [37] se define como

$$E(\mathbf{p}) = -\sum_i p_i \log_2 p_i, \tag{2.13}$$

donde $\mathbf{p} = (p_1, \ldots, p_n)$ es un vector de probabilidad, asociado a una variable aleatoria.

Ejemplo 5. Supongamos que tenemos una variable aleatoria que toma dos posibles valores: $+$ y $-$, y la probabilidad de cada valor es 0.6 y 0.4, respectivamente.

Entonces, la entropía de la variable aleatoria viene dada por

$$\begin{aligned} E(p) &= -(0.6 \log_2 0.6 + 0.4 \log_2 0.4) && (2.14)\\ &= -(-0.44 - 0.52) && (2.15)\\ &= 0.96 && (2.16) \end{aligned}$$

La idea detrás de ID3 es calcular cuánta entropía del conjunto de entrenamiento completo disminuirá si elegimos una variable particular en el siguiente paso. Esto es lo que se conoce como ganancia de información y se define como la entropía del conjunto completo menos la entropía cuando una variable es elegida. Matemáticamente, se define como

$$G(S, F) = E(S) - \sum_{f \in \text{valores}(F)} \frac{|S_f|}{|S|} E(S_f), \tag{2.17}$$

donde S es el conjunto de entrenamiento, F es una posible variable fuera del conjunto de todas las variables posibles.

El algoritmo ID3 computa la ganancia de información de cada variable y elige la que produce un mayor valor.

El pseudocódigo del algoritmo se puede ver a continuación:

- ▶ Si todos los ejemplos tienen la misma clase,
  - Devuelve una hoja con esa etiqueta.
- ▶ Si no hay variables restantes para probar
  - Devuelve una hoja con la etiqueta más común.
- ▶ Si no
  - Elige la variable $\hat{F}$ que maximiza la información de S para ser el siguiente nodo del árbol.
  - Añade una rama del nodo para cada posible valor $f \in \hat{F}$.
  - Por cada rama,
    - Calcula S_f eliminando $\hat{F}$ del conjunto de variables.
    - Recursivamente llamar al algoritmo con S_f para calcular la ganancia relativa al conjunto actual de ejemplos.

Ejemplo 6. Supongamos que queremos construir un árbol de decisión con el algoritmo ID3 usando los datos de la Tabla 2.4 para decidir si conceder un crédito o no de acuerdo a distintas variables (morosidad, antigüedad, ingresos, etc).

Tabla 2.4: Datos para el ejemplo del algoritmo ID3

Cliente	Moroso	Antigüedad	Ingresos	Trabajo fijo	Conceder crédito
1	Sí	>5	600–1200	Sí	No
2	No	<1	600–1200	Sí	Sí
3	Sí	1–5	>1200	Sí	No
4	No	>5	>1200	No	Sí
5	No	<1	>1200	Sí	Sí
6	Sí	1–5	600–1200	Sí	No
7	No	1–5	>1200	Sí	Sí
8	No	<1	<600	Sí	No
9	No	>5	600–1200	No	No
10	Sí	1–5	<600	No	No

En primer lugar, calculamos la entropía de la variable Conceder crédito:

$$\begin{aligned} E(\text{Conceder crédito}) &= -p_{\text{sí}} \log_2 p_{\text{sí}} - p_{\text{no}} \log_2 p_{\text{no}} \\ &= -0.4 \log_2 0.4 - 0.6 \log_2 0.6 = 0.96 \end{aligned}$$

Ahora calculamos la ganancia de cada variable:

$$\begin{aligned} G(S, \text{Moroso}) &= 0.96 - \frac{|S_{\text{sí}}|}{10} E(S_{\text{sí}}) - \frac{|S_{\text{no}}|}{10} E(S_{\text{no}}) \\ &= 0.96 - 0.4\,(-\log_2 1) - \frac{6}{10}\left(-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right) = 0.42, \end{aligned}$$

$$G(S, \text{Antigüedad}) = 0.96 - \frac{|S_{>5}|}{10} E(S_{>5}) - \frac{|S_{<1}|}{10} E(S_{<1}) - \frac{|S_{1-5}|}{10} E(S_{1-5}) = 0.11,$$

$$G(S, \text{Ingresos}) = 0.96 - \frac{|S_{600-1200}|}{10} E(S_{600-1200}) - \frac{|S_{>1200}|}{10} E(S_{>1200}) - \frac{|S_{<600}|}{10} E(S_{<600}) = 0.31,$$

$$G(S, \text{Trabajo fijo}) = 0.96 - \frac{|S_{\text{sí}}|}{10} E(S_{\text{sí}}) - \frac{|S_{\text{no}}|}{10} E(S_{\text{no}}) = 0.12$$

Elegimos la variable Moroso por tener una mayor ganancia. Éste será nuestro nodo raíz del árbol.

Puesto que todos los ejemplos llevan a que si se es moroso, no se concede el crédito, añadimos una rama desde el nodo raíz hasta la hoja NO con la etiqueta Sí.

Calculamos la nueva entropía del conjunto, que llamaremos S'.

$$E(S') = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.90$$

Calculamos la ganancias de las variables restantes,

$$G(S', \text{Antigüedad}) = 0.90 - \frac{|S_{<1}|}{6} E(S_{<1}) - \frac{|S_{1-5}|}{6} E(S_{1-5}) - \frac{|S_{>5}|}{6} E(S_{>5}) = 0.12$$

$$G(S', \text{Ingresos}) = 0.90 - \frac{|S_{600-1200}|}{6} E(S_{600-1200}) - \frac{|S_{<600}|}{6} E(S_{<600}) - \frac{|S_{>1200}|}{6} E(S_{>1200}) = 0.56$$

$$G(S', \text{Trabajo fijo}) = 0.90 - \frac{|S_{\text{sí}}|}{6} E(S_{\text{sí}}) - \frac{|S_{\text{no}}|}{6} E(S_{\text{no}}) = 0$$

Así pues, la variable con mayor ganancia es Ingresos, por lo que se añade una rama desde Moroso hasta Ingresos con etiqueta NO.

Puesto que en todos los ejemplos llevan a que si los ingresos son <600 no se concede el crédito y si son >1200 sí se concede, se añaden dos ramas desde Ingresos y se añaden dos hojas con etiquetas NO y SÍ, respectivamente.

Calculamos la entropía del nuevo conjunto, que llamamos S''.

$$E(S'') = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

Calculamos la ganancia de las variables restantes,

$$G(S'', \text{Antigüedad}) = 1 - \frac{|S_{<1}|}{2}E(S_{<1}) - \frac{|S_{>5}|}{2}E(S_{>5}) = 1$$

$$G(S'', \text{Trabajo fijo}) = 1 - \frac{|S_{\text{sí}}|}{2}E(S_{\text{sí}}) - \frac{|S_{\text{no}}|}{2}E(S_{\text{no}}) = 1$$

En este caso, tenemos que la ganancia es la misma, por lo que elegimos una variable al azar, en este caso, Trabajo fijo.

Añadimos un nodo y una rama desde Ingresos. Puesto que todos los ejemplos restantes llevan a que si se tiene trabajo fijo se concede el crédito, y que en caso contrario, no se concede. Se añaden dos hojas procedentes de Trabajo fijo con valores SÍ y NO, respectivamente.

El árbol completo se puede ver en la Figura 2.11.

Otros algoritmos de árboles de decisión se pueden encontrar en [33] como C4.5, CART, CHAID o QUEST.

2.2 Aprendizaje no supervisado

Los algoritmos vistos anteriormente usaban un conjunto de entrenamiento que consistía en una colección de datos con la salida que se debía producir. En el aprendizaje no supervisado no se tiene conocimiento sobre los valores correctos de la salida. Esto hace que no se pueda resolver un problema de regresión con aprendizaje no supervisado.

El objetivo del aprendizaje no supervisado es encontrar clústers, es decir, conjuntos de ejemplares que se parezcan entre ellos. Una forma de medir la similitud es la distancia, normalmente la distancia euclídea.

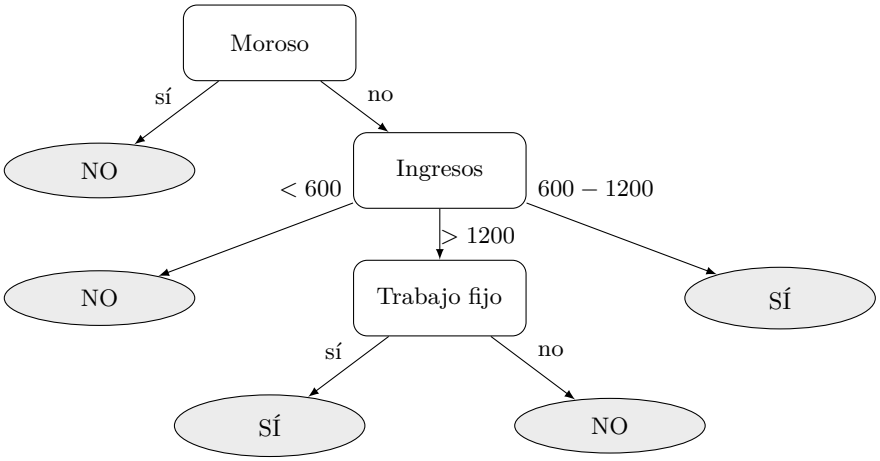


Figura 2.11: Árbol de decisión tras la ejecución del algoritmo ID3

2.2.1 Algoritmo de las k-medias

Supongamos que queremos dividir nuestros datos de entrada en k categorías (k es un valor fijado). La idea es situar los centros de los clústers en el espacio de entrada y situar los centros en el centro de los clústers.

Para medir la cercanía entre los puntos usamos alguna distancia como la euclídea. Una vez que tenemos la distancia, podemos calcular el centro como la media (aunque sólo es válido en el caso euclídeo).

El pseudocódigo de este algoritmo es el siguiente:

- ▶ Inicialización
  - Elegir k.
  - Elegir k posiciones aleatoria para el espacio de entrada.
  - Asignar el centro de los clústers μ_j a esas posiciones.
- ▶ Aprendizaje
  - Repetir
    - Para cada punto x_i
      - ◇ Computar la distancia a cada centro del clúster.
      - ◇ Asignar el punto al clúster más cercano con distancia

$$d_i = \min_j d(x_i, \mathbf{x_i}) \tag{2.18}$$

    - Para cada centro del clúster

◇ Mover la posición del centro a la media de los puntos en el clúster

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i \tag{2.19}$$

donde N_j es el número de puntos del clúster j.

- Hasta que el centro del clúster para de moverse.

Otros algoritmos de aprendizaje no supervisado se pueden encontrar en [16] e incluyen el clustering jerárquico, entre otros.

2.3 Aprendizaje por refuerzo

El aprendizaje por refuerzo rellena el hueco entre el aprendizaje supervisado, donde el algoritmo es entrenado con las respuestas correctas, y el aprendizaje no supervisado, donde el algoritmo sólo puede explotar similitudes en los datos para agruparlos. Este enfoque intermedio es que proporciona información acerca de si la respuesta es correcta o no, pero no cómo mejorarla, por lo que es necesario probar distintas estrategias y ver cuál funciona mejor.

Un algoritmo por refuerzo explora sobre un espacio de búsqueda de posibles entradas y salidas y trata de maximizar la recompensa.

El aprendizaje por refuerzo asigna estados a acciones para maximizar alguna recompensa numérica. Esto es, el agente (cosa que aprende) conoce la entrada actual (el estado) del entorno (lugar sobre el que actúa el agente), y las posibles acciones que puede realizar y su objetivo es maximizar la recompensa.

En [23, 43] se pueden encontrar algunos algoritmos sobre aprendizaje por refuerzo y algunas de sus aplicaciones.

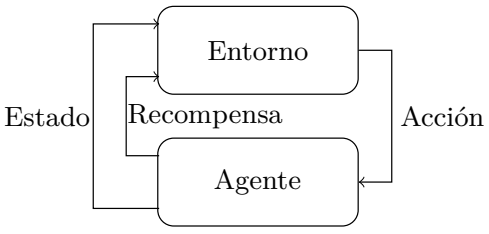


Figura 2.12: Esquema del aprendizaje por refuerzo

3

Redes parenclíticas

En el capítulo anterior hemos visto una serie de algoritmos de machine learning clásicos como lo son los árboles de decisión, las redes neuronales y los Support Vector Machine. En este capítulo veremos un nuevo método basado en teoría de grafos para el problema de clasificación: las redes parenclíticas.

Este nuevo método se ha utilizado con éxito en la clasificación de sujetos con glomerulonefritis [49, 50] (enfermedad renal en la que se daña el sector de los riñones que ayuda a filtrar los desechos y los líquidos de la sangre), con magníficos resultados, llegando a clasificar al 100 % de los sujetos con esta enfermedad. Además, se señala que este método es menos sensible ruido que otros algoritmos clásicos de machine learning como el perceptrón multicapa, los árboles de decisión o Naïve Bayes [50]. En [48] se aplican las redes parenclíticas a los genes de la planta *Arabidopsis thaliana*. Este método permitió el descubrimiento de nuevos genes involucrados en respuestas a estrés osmótico. En [51] se usan las redes complejas para la optimización de series temporales multivariantes. En [21] se utilizan redes parenclíticas para la clasificación de los datos de metilación del ADN humano. Aplicado en 14 distintos tipos de cáncer, en 12 de ellos se obtiene una gran tasa de clasificación (clasificando como paciente con cáncer o sin cáncer), situándose en torno al 90–100 %.

El método de redes parenclíticas se puede representar de forma sintética en la Figura 3.1. Partiendo de unos datos estructurados (datos que constan de un número fijo de características o variables, representados normalmente en una matriz: cada una de las filas es una observación y las columnas son cada de las variables), se calcula la curva de regresión sobre cada par de variables, y se calcula el error de la estimación obtenida sobre un nuevo ejemplo (que no ha sido ajustado el modelo de regresión con él), que será el peso de la arista de entre las dos variables en el grafo. Con todas las redes parenclíticas (cada una de ellas asociada a una observación cuyos nodos corresponden a las variables) se obtienen

distintas medidas, que se pasan a un algoritmo de machine learning clásico, que obtiene la clasificación final.

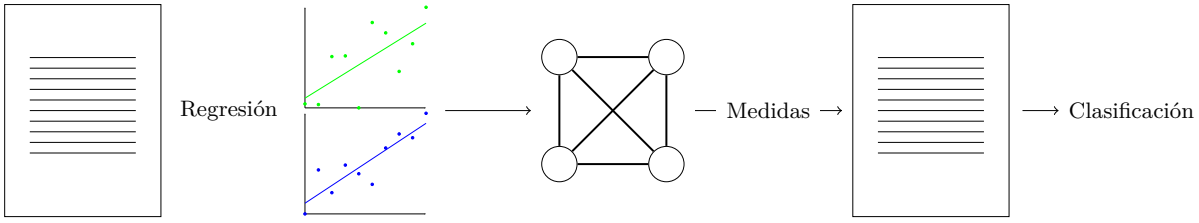


Figura 3.1: Esquema de redes parenclíticas

En este capítulo veremos una introducción a la teoría de grafos, y veremos cómo pasar de la teoría de grafos a las redes complejas, y por último veremos en qué consiste el método de redes parenclíticas.

3.1 Introducción a la teoría de grafos

La teoría de grafos es la rama de las matemáticas que estudia los grafos, objetos matemáticos que constan de dos elementos: los nodos o vértices y las aristas [8].

La teoría de grafos se inicia con un artículo de Leonhard Euler publicado en 1736 donde resolvía el problema de los puentes de Königsberg [11]. La ciudad de Königsberg es atravesada por el río Pregel, que se bifurca en dos para dividir la ciudad en cuatro zonas distintas, que estaban conectadas por siete puentes (Figura 3.2).

El problema consistía en encontrar un camino que recorriera la ciudad, pasando por cada puentes una única vez y regresara al punto de inicio. Euler probó que este problema no tiene solución, por lo que no es posible recorrer los siete puentes y volver al punto inicial, empleando un nuevo concepto: los grafos.

A continuación, introduciremos el concepto de grafo, algunas operaciones que se pueden realizar con ellos y alguna de las medidas sobre grafos que utilizaremos a lo largo de la memoria.

Definición 1. Un grafo (o grafo no dirigido) es un par $G = (V, E)$ de conjuntos que satisfacen que $E \subseteq V^2$ y $V \cap E = \emptyset$. Los elementos de V se denominan vértices (o nodos) del grafo G y los elementos de E se denominan arcos (o aristas). Una arista entre los vértices $x, y \in V$ se denota como xy o $yx \in E$.

La forma usual de representar un grafo es dibujar un punto (o círculo) por cada vértice y unir dos de estos dos puntos (o círculos) con una línea para formar un arco. Cómo estén

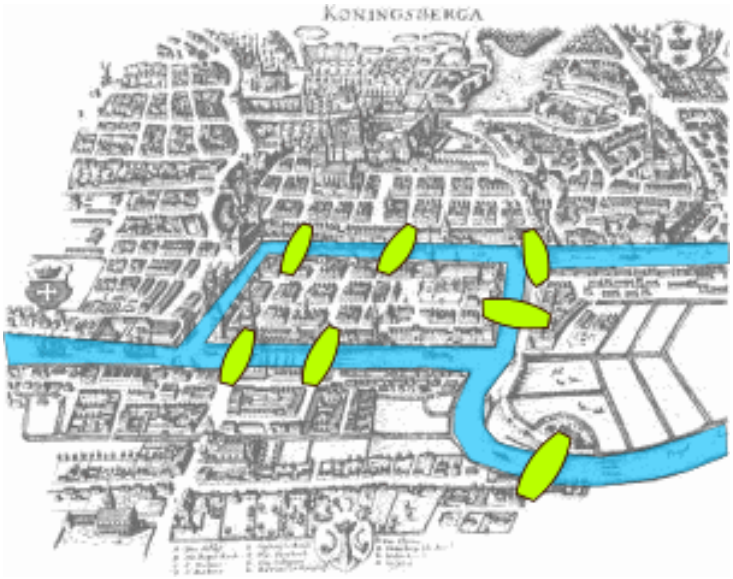


Figura 3.2: Situación de los siete puentes sobre la ciudad de Königsberg

dibujados los vértices y los arcos es irrelevante, sólo importa qué pares de nodos forman una arista y cuáles no.

Ejemplo 7. La Figura 3.3 muestra la representación gráfica de un grafo. Matemáticamente, el grafo es el par (V, E) donde

$$V = \{A, B, C, D\}$$

$$E = \{\{A, B\}, \{A, C\}, \{B, C\}, \{C, D\}\}$$

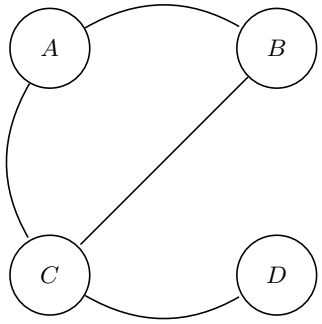


Figura 3.3: Ejemplo de grafo

Definición 2. Se llama orden de un grafo G al número de vértices de dicho grafo. Se denota como $|G|$.
Un grafo G se dice que es finito si $|G| < \infty$. Si $|G| = \infty$ se dice que el grafo G es infinito.

Ejemplo 8. El grafo de la Figura 3.3 es un grafo finito, puesto que el número de vértices del grafo es 4.

En esta memoria trabajaremos sólo con grafos finitos.

Definición 3. Dos vértices $x, y \in V$ del grafo $G = (V, E)$ se dicen adyacentes si existe una arista entre x e y (o $xy \in E$).

Definición 4. Un grafo se dice completo si todos sus vértices son adyacentes.

Ejemplo 9. El grafo de la Figura 3.4 es completo ya que todos sus vértices son adyacentes. En efecto, el vértice A tiene una arista que lo une con los nodos B, C y D. De la misma forma, se comprueba para los vértices B, C y D.

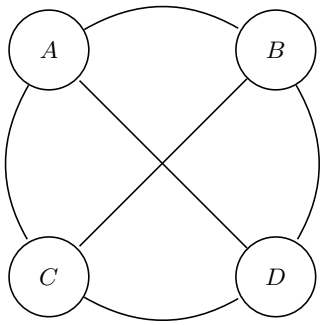


Figura 3.4: Ejemplo de grafo completo

Definición 5. Un grafo dirigido (o digrafo) es un par (V, E) de conjuntos disjuntos (de vértices y de aristas) junto con dos funciones init : $E \to V$ y ter : $E \to V$ que asigna a cada arista e un vértice inicial init(e) y un vértice terminal ter(e).

La arista e se dice dirigida desde init(e) hasta ter(e).

Si init(e) = ter(e), la arista e se dice que es un bucle.

Ejemplo 10. La Figura 3.5 muestra la representación gráfica un grafo dirigido. Matemáticamente, es el par (V, E) donde

$$V = \{A, B, C, D\}$$

$$E = \{\{A, B\}, \{A, C\}, \{C, C\}, \{B, C\}, \{C, D\}\}$$

junto con las funciones init : $E \to V$ y ter : $E \to V$ definidas de la siguiente manera:

$$\begin{array}{lccc|lccc} \text{init}: & E & \to & V & \text{ter}: & E & \to & V \\ & \{A,B\} & \mapsto & A & & \{A,B\} & \mapsto & B \\ & \{A,C\} & \mapsto & A & & \{A,C\} & \mapsto & C \\ & \{C,C\} & \mapsto & C & & \{C,C\} & \mapsto & C \\ & \{B,C\} & \mapsto & B & & \{B,C\} & \mapsto & C \\ & \{C,D\} & \mapsto & C & & \{C,D\} & \mapsto & D \end{array}$$

Además la arista $\{C,C\}$ es un bucle porque $\text{init}(\{C,C\}) = \text{ter}(\{C,C\}) = C$.

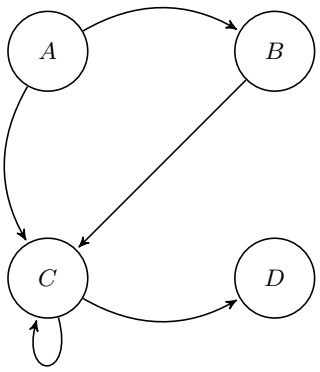


Figura 3.5: Ejemplo de grafo dirigido con bucle

Definición 6. Un grafo ponderado o pesado es un grafo con una función $\text{w} : E \to \mathbb{R}$, es decir, que w asocia un número real a cada arista. Esta función recibe el nombre de función peso.

Ejemplo 11. El grafo $G = (V, E)$ con

$$V = \{A, B, C\} \tag{3.1}$$
$$E = \{\{A,B\}, \{A,C\}, \{B,C\}\} \tag{3.2}$$

es un grafo. Si le añadimos la función $\text{w} : E \to \mathbb{R}$ definida de la siguiente forma, G es un grafo ponderado (ver Figura 3.6).

$$\begin{array}{lccc} \text{w}: & E & \to & \mathbb{R} \\ & \{A,B\} & \mapsto & e \\ & \{A,C\} & \mapsto & 5 \\ & \{B,C\} & \mapsto & \pi \end{array}$$

Definición 7. Dado un grafo ponderado $G = (V, E)$ y $\text{w} : E \to \mathbb{R}$, decimos que una arista $e \in E$ incide en el vértice $v \in V$, si $\text{ter}(e) = v$.

Definición 8. La fuerza de un vértice en un grafo ponderado se define como la suma de todos los pesos de sus aristas incidentes.

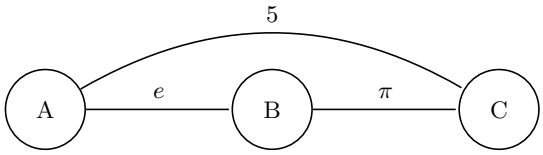


Figura 3.6: Ejemplo de grafo ponderado

3.2 De la teoría de grafos a las redes complejas

La teoría de grafos permite caracterizar un sistema una vez que sus partes constituyentes y relaciones están definidas, una vez que han eliminando todos los detalles innecesarios para extraer sus partes constituyentes y las relaciones entre ellas. La teoría de grafos no permite establecer cómo aislar las partes constituyentes de un sistema. La estructura creada por las interacciones se conoce como topología de red. La mayoría de redes sociales, biológicas y tecnológicas tienen propiedades topológicas no triviales, es decir, que existen patrones de conexiones entre sus elementos que no son totalmente regulares ni totalmente aleaorios [51].
La estructura topológica de la red puede ser representada como un conjunto de *métricas topológicas*, que se pueden clasificar en tres tipos atendiendo al tipo de escala: *microescala* (afecta a nodos individuales o relaciones), *mesoescala* (afecta a un grupo de pocos nodos) y *macroescala* (afecta a la red por completo).

Las medidas que usaremos a lo largo de la memoria serán las siguientes:

Definición 9. Se define densidad de enlaces[1] del grafo G como el número de enlaces dividido entre el número de enlaces que pueden estar presentes en el grafo.

$$d(G) = \frac{1}{n(n-1)} \sum_{i,j} a_{i,j} \tag{3.3}$$

donde n es el número de nodos del grafo y $a_{i,j}$ es la entrada ij-ésima de la matriz de adyacencia (matriz que en su posición (i, j) indica si existe una arista entre el nodo i y j).

Definición 10. Se define coeficiente de clustering C del grafo G como

$$C(G) = \frac{3N_{\triangle}}{N_3} \tag{3.4}$$

donde $N_{\triangle}$ es el número de triángulos del grafo y N_3 es el número de tripletas conectadas.

Definición 11. Se define eficiencia de un grafo G como

$$\mathrm{E}(\mathcal{G}) = \frac{1}{\binom{n}{2}} \sum_{i \neq j} \frac{1}{d(i,j)} \tag{3.5}$$

donde $d(i, j)$ es la distancia entre el nodo i y el nodo j y n es el número de nodos de G.

[1] Traducción de link density

La eficiencia es el inverso de la media armónica de todos los caminos del grafo.

Para calcular la distancia entre los nodos del grafo se pueden aplicar los algoritmos de Dijkstra o Floyd (Ver [3]).

Definición 12. Se define longitud del camino característico[2] del grafo G de la siguiente forma:

$$\mathrm{L}(\mathcal{G}) = \frac{1}{\binom{n}{2}} \sum_{i \neq j} d(i,j) \tag{3.6}$$

donde $d(i,j)$ es la distancia entre el nodo i y el nodo j y n es el número de nodos de G.

La longitud del camino característico es la media aritmética de todos los caminos del grafo.

Notar que la densidad de enlaces es una medida de microescala, el coeficiente de clustering es de mesoescala, y la eficiencia y la longitud del camino característico son medidas de macroescala.

En [7] se pueden encontrar una extensa lista con distintas medidas sobre grafos.

En este contexto, las redes parenclíticas son un tipo de red compleja que se construye a partir de la desviación del "comportamiento normal".

A continuación describimos el método de construcción de redes parenclíticas propuesto en [50].

Supongamos que tenemos un conjunto de datos con n variables para cada una de las m observaciones, y cada una de éstas vienen clasificadas en la clase c o d. Estas dos clases actuarán como datos de entrenamiento. Además, se supone que existe una observación, que llamaremos X, para la que la clase se desconoce.

El objetivo es construir un grafo, extraer de una serie de medidas y conseguir su clasificación mediante algoritmos clásicos de machine learning.

La creación del grafo requiere analizar todos los pares de nodos. Para ello, se proyectan cada par de variables sobre el plano xy y se aplica un modelo. En [48–50] se aplican modelos de regresión lineal.

$$y = \alpha_1 x + \alpha_2 + \varepsilon \tag{3.7}$$

donde α_1 es la pendiente del modelo, α_2 es el término independiente y ε es el término del error del modelo.

[2] Traducción de characteristic path length

Esto se hace por cada clase, dando lugar a dos modelos distintos. Se puede representar la salida de la variable proyecta de la observación X como dos distribuciones normales (una por cada clase) con parámetros μ como el valor esperado de la variable salida y σ como la desviación estándar del vector ϵ.

Se puede asignar una probabilidad a cada una de las clases de la siguiente forma.

$$p_d = \frac{\tilde{p}_d}{\tilde{p}_d + \tilde{p}_c} \tag{3.8}$$

$$p_c = \frac{\tilde{p}_c}{\tilde{p}_d + \tilde{p}_c} \tag{3.9}$$

donde $\tilde{p}_d$, $\tilde{p}_d$ son las probabilidades de estar en la clase d o c dadas por las distribuciones normales mencionadas anteriormente.

Para formar el grafo, asociamos un peso a la arista entre esas dos variables dado por el error entre el valor estimado y el valor real, es decir,

$$\hat{\varepsilon} = |\hat{y} - y|, \tag{3.10}$$

donde $\hat{y}$ es el valor estimado y y es el valor real de la variable salida, de la clase con mayor probabilidad de las calculadas anteriormente.

Ejemplo 12. Supongamos que disponemos de los datos de la Tabla 3.1, y queremos construir la red parenclítica del individuo $X = (3, 3)$, y obtener su clasificación.

En primer lugar calculamos la rectas de regresión de cada una de las clases A y B (Figura 3.7).

Las rectas para las clases A y B son respectivamente:

$$y_A = 0.82x + 1.69,$$
$$y_B = 0.46x - 1.25,$$

y las desviaciones típicas de los residuos son:

$$\sigma_A = 0.5135,$$
$$\sigma_B = 0.2801$$

Por tanto, imponemos que la probabilidad de estar en cada una de las clases es una variable aleatoria normal con media, el valor esperado de x sobre la recta de regresión del individuo X, y desviación típica, la obtenida de los residuos del modelo:

Tabla 3.1: Datos del ejemplo

x	y	Clase
0.5	1.5	A
1.5	2.5	A
1.2	2.5	A
0.75	2	A
0.6	1.9	A
0.77	2.5	A
1.5	3	A
1.3	3.3	A
0.6	3.2	A
4	1	B
3.3	0.3	B
4.5	1.2	B
4.5	0.5	B
3.9	0.7	B
5	1	B
3.5	0.2	B
4	0.3	B

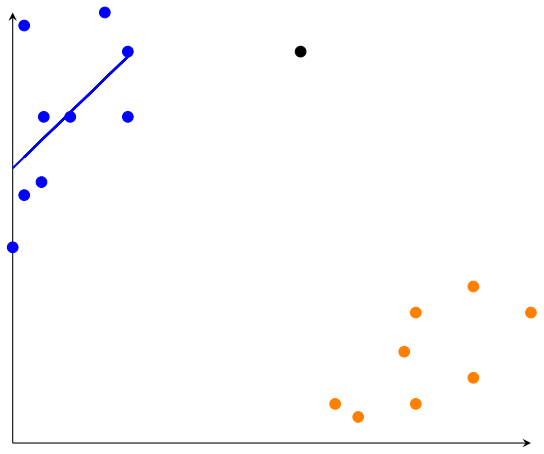


Figura 3.7: Recta de regresión de cada una de las clases. La clase A se muestra en azul y la clase B en naranja

$$Y_A = \mathcal{N}(0.82 \cdot 3 + 1.69, 0.5135) = \mathcal{N}(4.15, 0.5135)$$
$$Y_B = \mathcal{N}(0.46 \cdot 3 - 1.25, 0.2801) = \mathcal{N}(0.13, 0.2801)$$

La Figura 3.8 muestra este hecho. Por tanto, la probabilidad de estar en cada una de las clase viene dado por

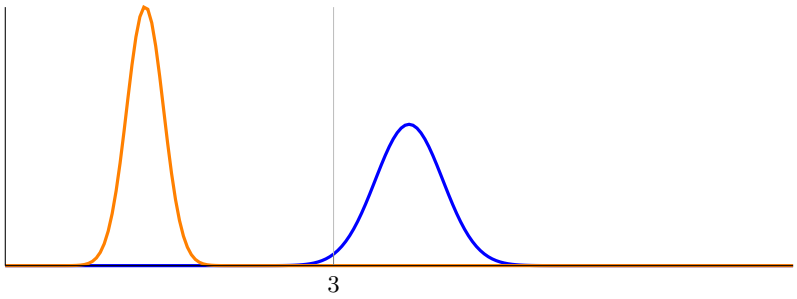


Figura 3.8: Distribuciones normales

$$\tilde{p}_A = \frac{\tilde{p}_A}{\tilde{p}_A + \tilde{p}_B} = \frac{0.063}{0.063 + 0} = 1$$
$$\tilde{p}_B = \frac{\tilde{p}_A}{\tilde{p}_A + \tilde{p}_B} = \frac{0}{0.063 + 0} = 0$$

Por tanto, la probabilidad mayor es de que el individuo X que pertenezca a la clase A.

Así pues, el error cometido en esta aproximación es

$$\epsilon = |4.15 - 3| = 1.15$$

Éste es el peso entre los nodos x,y, que da lugar a la red parenclítica del individuo X (Figura 3.9).

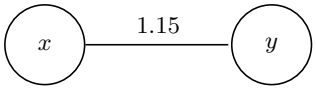


Figura 3.9: Red parenclítica del ejemplo

En caso de tener más variables repetiríamos este proceso con cada par de variables. Una vez obtenidas todas las redes parenclíticas de los individuos desconocidos, se extraerían medidas de los grafos como las vistas en la Sección 3.2, que actuarían como entrada a un algoritmo de machine learning visto en la Sección 2, que obtendría la clasificación.

Este método tiene algunas limitaciones:

1. Las variables no tienen por qué comportarse de forma lineal, por lo que este modelo podría ajustar de forma incorrecta.
2. Se limita a una clasificación binaria.

La primera limitación se puede solventar aplicando otros métodos distintos al lineal.

Proponemos aplicar tres métodos distintos:

1. Exponencial

$$y = \exp(\alpha_1 + \alpha_2 x) \quad \alpha_1, \alpha_2 \in \mathbb{R} \tag{3.11}$$

2. Cuadrático

$$y = (\alpha_1 + \alpha_2 x)^2 \quad \alpha_1, \alpha_2 \in \mathbb{R} \tag{3.12}$$

3. Recíproco

$$y = \frac{1}{\alpha_1 + \alpha_2 x} \quad \alpha_1, \alpha_2 \in \mathbb{R} \tag{3.13}$$

Estos métodos pueden ser linealizados como hemos visto en la Tabla 2.2.

La segunda limitación se puede solventar generalizando la Fórmula 3.8 a lo siguiente:

$$p_i = \frac{\tilde{p}_i}{\sum_i \tilde{p}_i} \tag{3.14}$$

donde p_i es la probabilidad de pertenecer a la clase i.

Una vez hecho ésto se puede aplicar el método descrito. Seguimos suponiendo normalidad de las variables.

En el capítulo siguiente veremos el diseño y la tecnología utilizada para implementar una aplicación que permita analizar un conjunto de datos utilizando las redes parenclíticas.

4

Diseño de la aplicación

En este capítulo veremos cómo construir una aplicación para realizar el problema de clasificación utilizando redes parenclíticas y otras técnicas clásicas de machine learning como redes neuronales, SVM o árboles de decisión, así como la comparación entre ellas.

En primer lugar veremos cuál es la arquitectura de la aplicación, posteriormente, qué tecnologías hemos utilizado para desarrollar esta aplicación y, por último describiremos las funcionalidades de la aplicación.

4.1 Arquitectura de la aplicación

La arquitectura de la aplicación está basada en una arquitectura clásica cliente-servidor, en la que en el lado del servidor se utiliza R como lenguaje de programación, como servidor web actúa Shiny Server, y Shiny como framework web para R. En el lado del cliente se usa Shiny Dashboard.

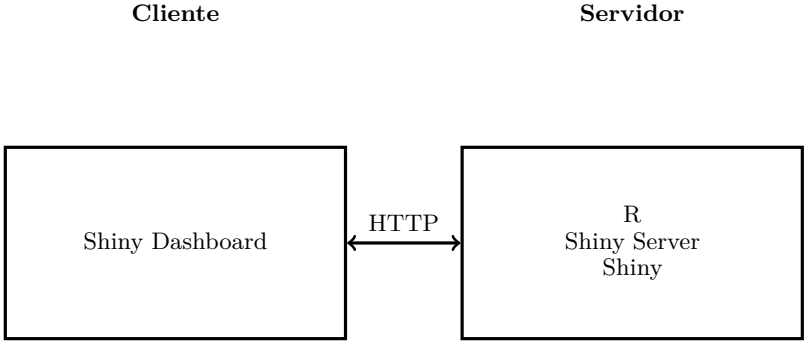


Figura 4.1: Arquitectura de la aplicación

4.2 Tecnología utilizada

Como lenguaje de programación se ha utilizado R [31], un lenguaje desarrollado para cálculos científicos y la realización de gráficas de alta calidad.

Figura 4.2: Logo de R

R incorpora la mayoría los test estadísticos más comunes, gran cantidad de funciones estadísticas y permite el manejo de datos de forma sencilla. R es gratis y software open-source y tiene licencia GNU General Public License. R no tiene restricciones, por lo que puede ser usado cuando y donde se quiera e incluso es posible venderlo bajo las condiciones de la licencia. Dispone de más de 8696 paquetes [32][1] disponibles de muchos repositorios especializados en diversas temáticas como minería de datos, econometría o bioestadística, entre otros muchos. R es multipltaforma, por lo que está disponible en Windows, Mac y Linux, y dispone tanto de versiones de 32 como de 64 bits.

Como servidor web se ha utilizado Shiny Server [40] que permite desplegar de forma rápida y sencilla aplicaciones Shiny en la nube. Dispone de dos licencias: Open Source Edition y Commercial. La primera permite acceso a través del navegador, y la segunda dispone de herramientas administrativas, autenticación y, métricas y sistemas de mantenimiento y gestión de recursos avanzados.

Como framework web se ha utilizado Shiny [38]. Permite desarrollar aplicaciones web sobre R sin necesidad de conocer HTML, CSS o JavaScript. Shiny combina la interactividad (usando programación reactiva) con la potencia de cálculo de R.

Shiny Dashboard [39] permite crear dashboards sobre aplicaciones Shiny de manera sencilla y rápida.

Además, se han utilizado distintos paquetes de R:

1. **igraph** [18]: análisis de grafos.
2. **e1071** [10]: clasificación con Support Vector Machine.

[1] A la fecha de escribir esta memoria

Figura 4.3: Logo de Shiny

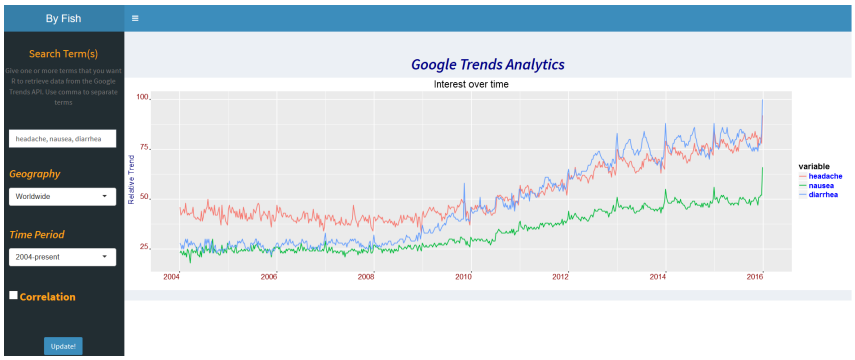


Figura 4.4: Dashborad realizado con Shiny Dashboard

3. **nnet** [28]: clasificación con redes neuronales (con una sola capa oculta).
4. **rpart** [35]: clasificación con árboles de decisión
5. **ggplot2** [14]: librería para visualización.

La aplicación está dividida en una serie de scripts, entre los que se encuentran:

- *calculatePrediction*: realiza las predicciones a partir del cálculo de las redes parenclíticas, calculando las medidas dadas por las Ecuaciones 3.3 , 3.4 , 3.5 y 3.6 y aplica una red neuronal para realizar la clasificación.
- *calculateML*: aplica redes neuronales, Support Vector Machine y árboles de decisión para realizar la clasificación de unos datos dados.
- *drawParencliticNetwork*: representa una red parenclítica con el uso del paquete *igraph*.
- *drawRegressionLine*: representa las curvas de regresión (lineal, exponencial, cuadrática y recíproca) de un par de variables del conjunto de entrada.
- *drawNormalPlot*: representa las distribuciones normales de cada clase.

Las características principales de la aplicación son:

- Clasificación usando redes parenclíticas aplicando modelos lineal, exponencial, cuadrático y recíproco.
- Clasificación usando métodos clásicos de machine learning como redes neuronales, Support Vector Machine y árboles de decisión.
- Visualización de estadísticas descriptivas de las variables cuantitativas, tanto de forma global como por grupo.
- Generación de informes.

4.3 Un paseo por la aplicación

La página de inicio se muestra en la Figura 4.5. En la parte izquierda se muestran todas las pestañas disponibles de la aplicación, y en la parte derecha se muestra el contenido de cada pestaña. En este caso, se muestra un formulario para subir el fichero de datos. Este fichero deberá estar en formato CSV y los datos deberán estar separados por coma.

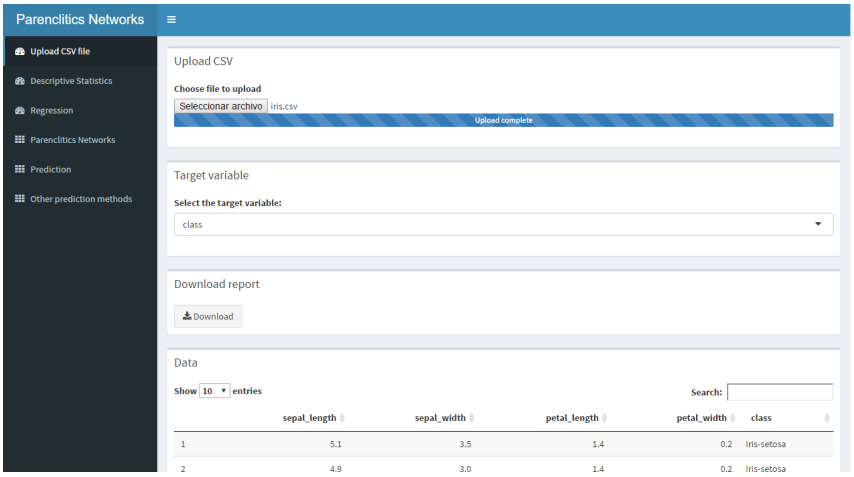


Figura 4.5: Página de inicio de la aplicación

Una vez cargados los datos, se muestran las variables que pueden ser usadas para la clasificación, es decir, aquellas que sean del tipo factor.

En la parte inferior de la pantalla se muestran los datos cargados en forma de tabla. Es posible ordenar las columnas por orden creciente o decreciente y buscar algún datos en el conjunto de datos.

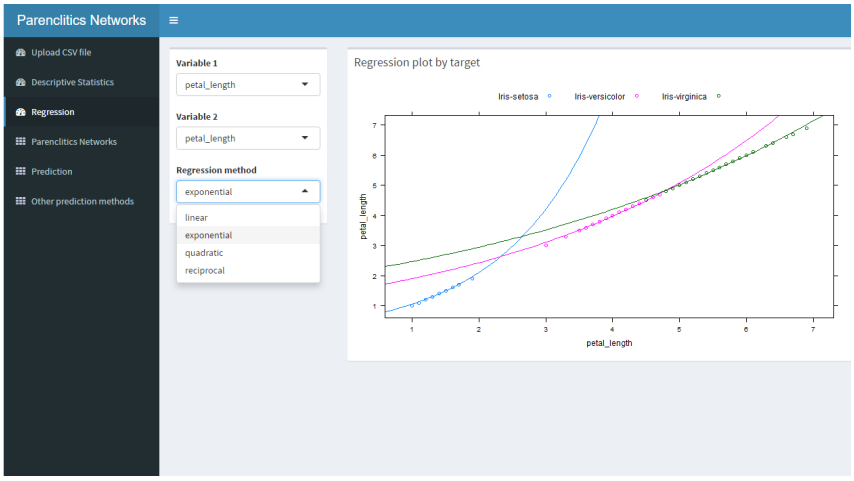


Figura 4.6: Pestaña Regression

En la pestaña *Regression* (Figura 4.6) se representan para cada par de variables sus curvas de regresión (lineal, exponencial, cuadrática y recíproco).
En la pestaña *Parenclitic Network* (Figura 4.7), se pueden ver las distribuciones de cada clase, calculadas a partir del modelo de regresión deseado y la red parenclítica de cada observación (Figura 4.8).

Figura 4.7: Pestaña Parenclitic Network

En la pestaña *Prediction* se muestran las gráficas de las medidas obtenidas de las redes parenclíticas (Figura 4.9) así como la clasificación realizada a partir de estas medidas (Figura 4.10).

Por último en la pestaña *Other prediction methods*, se obtiene la clasificación mediante redes neuronales, Support Vector Machine y árboles de decisión (Figura 4.11) de los datos iniciales.

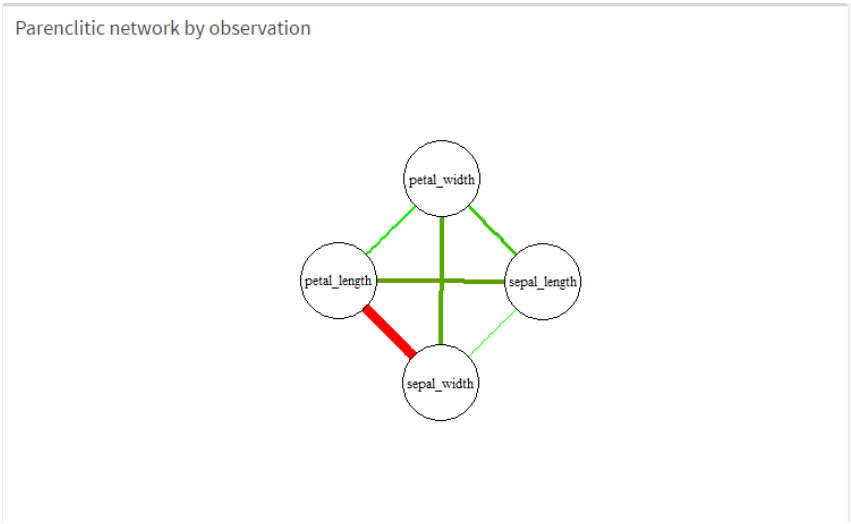


Figura 4.8: Red parenclítica

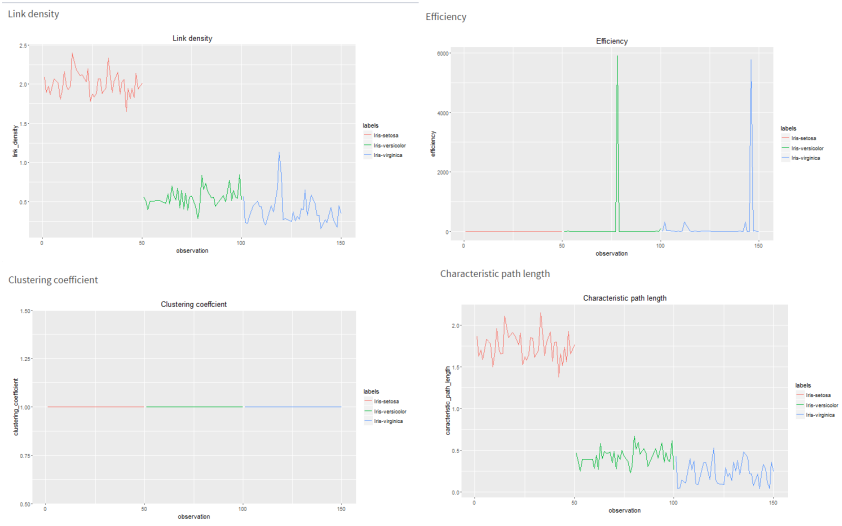


Figura 4.9: Medidas

Classification

	predicted	labels	classification
5	Iris-versicolor	Iris-setosa	Bad
7	Iris-versicolor	Iris-setosa	Bad
16	Iris-versicolor	Iris-setosa	Bad
18	Iris-versicolor	Iris-setosa	Bad
21	Iris-versicolor	Iris-setosa	Bad
31	Iris-versicolor	Iris-setosa	Bad
34	Iris-versicolor	Iris-setosa	Bad
37	Iris-versicolor	Iris-setosa	Bad
49	Iris-versicolor	Iris-setosa	Bad
55	Iris-versicolor	Iris-versicolor	Good

Figura 4.10: Clasificación con redes parenclíticas

Classification with decision tree

	predicted	labels	classification
7	Iris-setosa	Iris-setosa	Good
12	Iris-setosa	Iris-setosa	Good
19	Iris-setosa	Iris-setosa	Good
21	Iris-setosa	Iris-setosa	Good
22	Iris-setosa	Iris-setosa	Good
23	Iris-setosa	Iris-setosa	Good
30	Iris-setosa	Iris-setosa	Good
36	Iris-setosa	Iris-setosa	Good
39	Iris-setosa	Iris-setosa	Good
44	Iris-setosa	Iris-setosa	Good

Figura 4.11: Predicción usando árboles de decisión

En el próximo capítulo aplicaremos esta aplicación a una serie de conjuntos de datos.

5

Aplicación al análisis de datos de cáncer de mama

En este capítulo utilizaremos la aplicación descrita en el Capítulo 4 para aplicarla a la detección de tumores en cáncer de mama.

5.1 Datos utilizados

El conjunto de datos ha sido obtenido de [2]. Consta de 699 observaciones y 10 variables (más la variable de clasificación). Éstas son:

1. Código de la muestra
2. Espesor
3. Uniformidad del tamaño celular
4. Uniformidad de la forma celular
5. Adhesión marginal
6. Tamaño de las células epiteliales
7. Bare nuclei
8. Cromatina
9. Normal Nucleoli
10. Mitosis
11. Clase (B = beningno; M = maligno)

Todas las variables toman valores numéricos entre 1 y 10, excepto el número de muestra.

Puesto que esta variable no aporta nada, la eliminamos, dejando 10 variables (9 + variable de clasificación).

Un subconjunto de los datos se puede ver en la Figura 5.1.

Data

Show 10 entries Search:

	Clump.Thickness	Uniformity.of.Cell.Size	Uniformity.of.Cell.Shape	Marginal.Adhesion	Single.Epithelial.Cell.Size	Bare.Nuclei
1	5	1	1	1	2	1
2	5	4	4	5	7	10
3	3	1	1	1	2	2
4	6	8	8	1	3	4
5	4	1	1	3	2	1
6	8	10	10	8	7	10
7	1	1	1	1	2	10
8	2	1	2	1	2	1
9	2	1	1	1	2	1
10	4	2	1	1	2	1

Showing 1 to 10 of 683 entries Previous 1 2 3 4 5 … 69 Next

Figura 5.1: Subconjunto de los datos de cáncer de mama

Con la aplicación podemos ver cuál es la distribución de cada una de las variables. Por ejemplo, en el caso del espesor la distribución por clase de tumor se puede ver en la Figura 5.2.

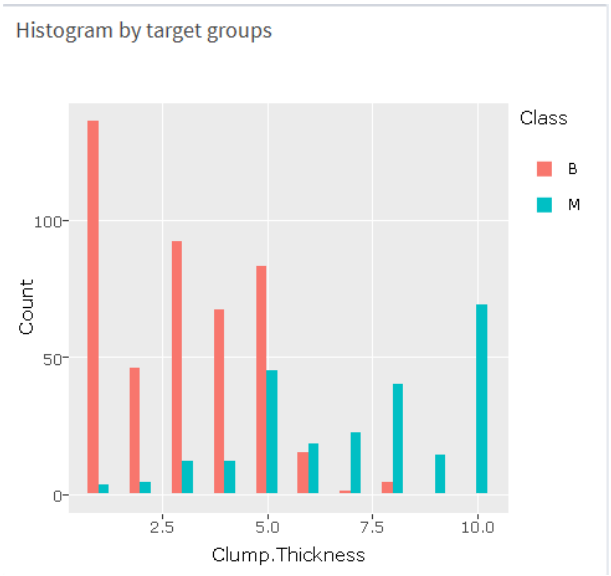
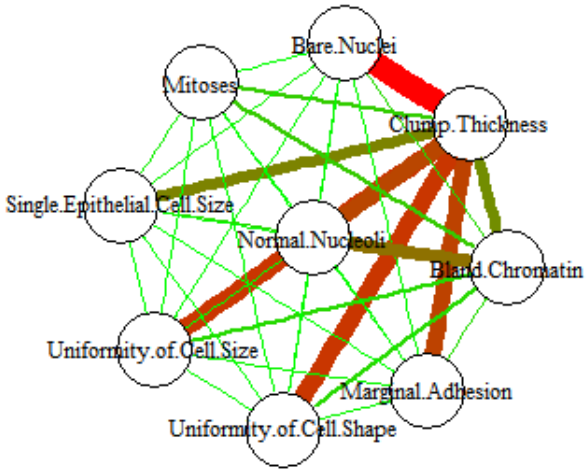
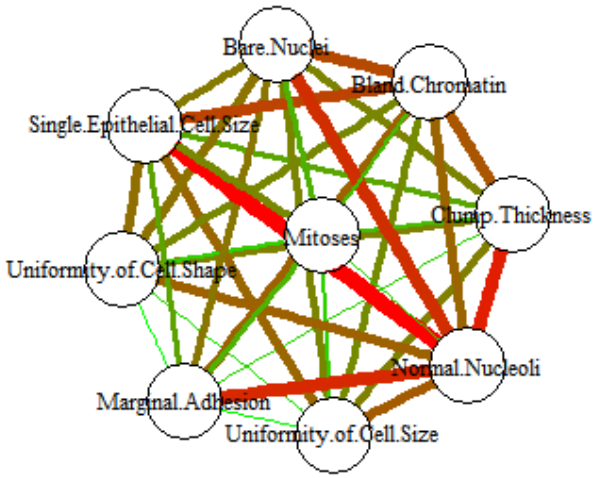


Figura 5.2: Distribución del espesor por clase de tumor

Se puede observar que la presencia de un espesor menor está presente en tumores benignos, y a medida que el espesor es mayor se producen más tumores malignos.



(a) Red parenclítica de la observación 1

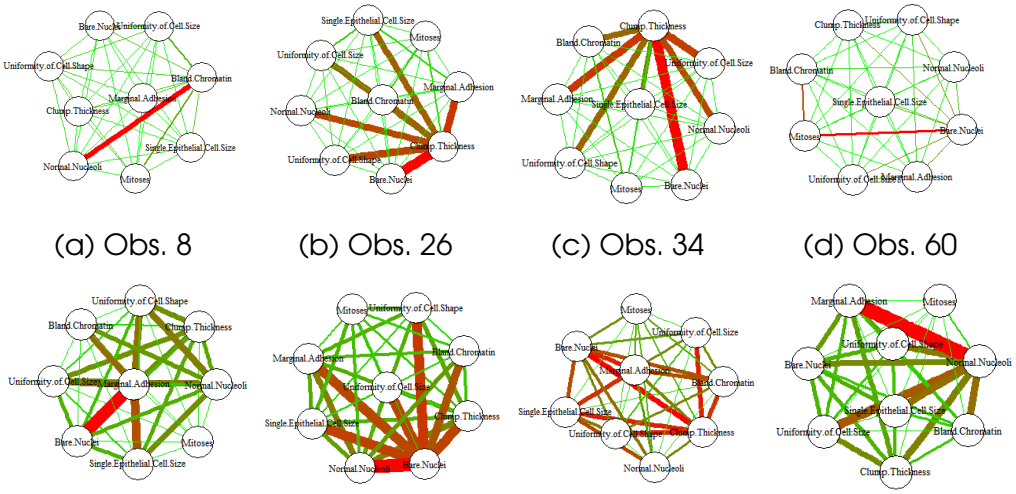


(b) Red parenclítica de la observación 2

Figura 5.3: Redes parenclíticas con modelo de regresión lineal

En la Figura 5.3 se pueden observar las redes parenclíticas de las dos primeras observaciones obtenidas con el modelo lineal.

Podemos ver si existen diferencias entre las redes parenclíticas de las observaciones con tumores benignos y malignos. La Figura 5.4 muestra cuatro observaciones de cada clase de tumor.



(a) Obs. 8 (b) Obs. 26 (c) Obs. 34 (d) Obs. 60

(e) Obs. 41 (f) Obs. 43 (g) Obs. 50 (h) Obs. 51

Figura 5.4: Redes parenclíticas por clase de tumor con modelo de regresión lineal. En la primera fila se muestran las observaciones con tumor benigno y, en la segunda, con tumor maligno

Se puede observar que en el caso de las observaciones con tumor benigno existe un menor peso en las aristas (color verde de las aristas), y en las observaciones con tumor maligno existe un mayor entre las aristas (color rojo de las aristas).

Puesto que disponemos de varios modelos de regresión, puede que el modelo lineal no sea el más indicado, por lo que calculamos la regresión para cada uno de los modelos de regresión (lineal, exponencial, cuadrática y recíproca), junto con la de los modelos de machine learning clásicos (árboles de decisión, Support vector Machine y redes neuronales), y calculamos la curva ROC de cada clasificador (Figura 5.5).

Se puede observar que el mejor clasificador por redes parenclíticas es el obtenido con el modelo de regresión recíproco, seguido del exponencial y el cuadrático. El peor clasificador es el lineal. Respecto a los algoritmos clásicos de machine learning, el mejor clasificador son los árboles de decisión seguido de los Support Vector Machine y de las redes neuronales.

Sobre todos los clasificadores, el mejor son los árboles de decisión, seguido de los Support Vector Machine y las redes parenclíticas con modelo recíproco.

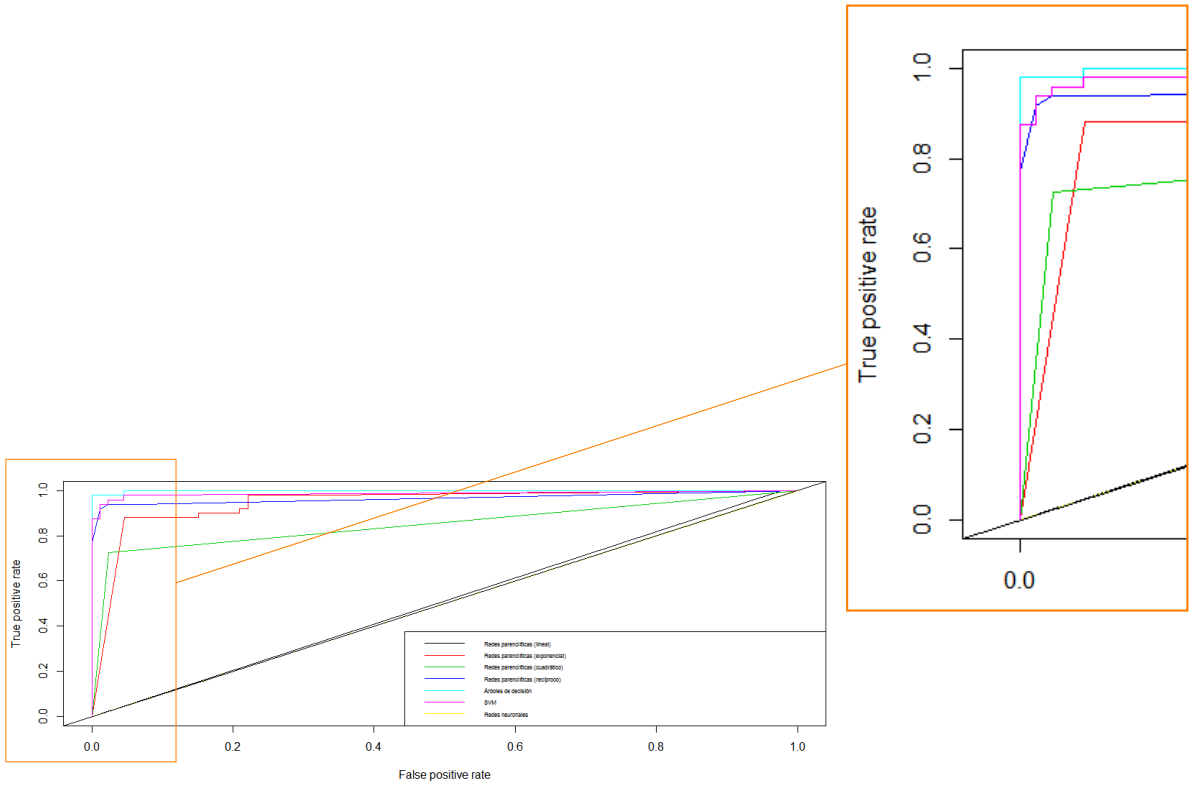


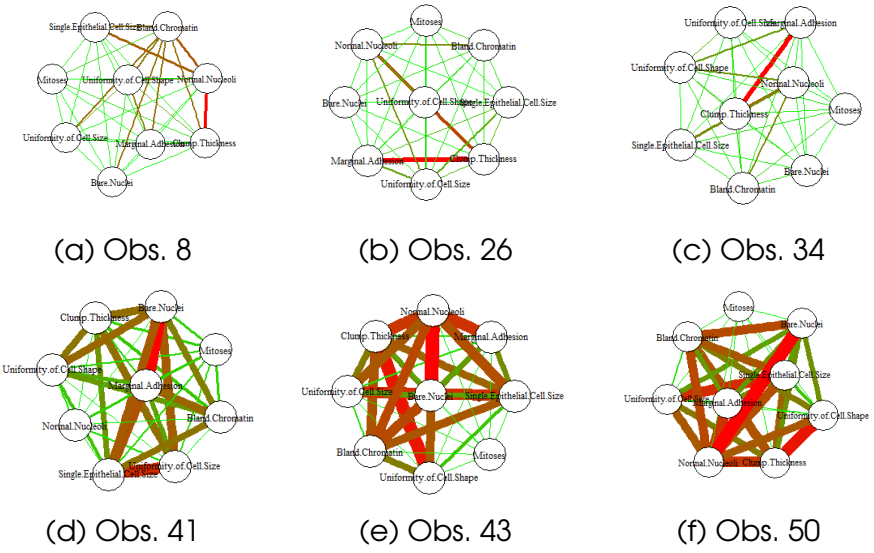
Figura 5.5: Curvas ROC

Puesto que el mejor modelo de redes parenclíticas es el modelo recíproco, veamos si existen diferencias entre las redes parenclíticas según la clase de tumor (Figura 5.6).

Se puede observar que los pacientes con tumores malignos tienen una cantidad de aristas con mayor peso (representadas con el color rojo), mientras que en el caso de los pacientes con tumores benignos se da el caso contrario, existe una cantidad mayor de aristas con peso pequeño (representado en color verde). Así pues, parece que existen diferencias según el tipo de tumor (benigno y maligno) atendiendo solamente a la estructura de la red parenclítica.

5.2 Umbralización de las redes parenclíticas

Hasta ahora, las redes parenclíticas usadas eran grafos completos, es decir, que existe una arista entre cada par de nodos (Ver Definición 4). Esto implica que algunas medidas no son adecuadas sobre grafos completos, ya que siempre se obtiene el mismo valor de la medida. Para solventar este problema, normalizaremos (en el intervalo $[0, 1]$) los pesos de las aristas de cada una de las redes parenclíticas, y eliminaremos las aristas cuyos pesos



(a) Obs. 8 (b) Obs. 26 (c) Obs. 34

(d) Obs. 41 (e) Obs. 43 (f) Obs. 50

Figura 5.6: Redes parenclíticas por clase de tumor con modelo de regresión recíproco. En la primera fila se muestran las observaciones con tumor benigno y, en la segunda, con tumor maligno

sean menores o iguales que un valor $x \in [0, 1]$, y realizaremos la clasificación de igual manera que en los casos anteriores (Figura 5.7).
Si aplicamos este procedimiento a los datos de cáncer de mama para cada uno de los cuatro tipos de regresión (lineal, exponencial, cuadrático y recíproco), con paso de umbralización 0.01, atendiendo a la tasa de aciertos, los resultados se muestran en la Figura 5.8.

Se puede observar que el método de umbralización mejora el porcentaje de clasificación de cada uno de los métodos de regresión. En especial, mejora el método de regresión lineal, que con las redes parenclíticas con grafos completos apenas superaba el 50 % de acierto, con el método de umbralización supera el 90 %. En los otros métodos de regresión, se produce el mismo efecto, pero la mejora es inferior que en el caso del método de regresión lineal. Notar que se producen fuertes variaciones en la tasa de acierto para valores muy cercanos de la umbralización. Esto puede ser debido a que al realizar el entrenamiento con redes neuronales, ésta se queda atrapada en un mínimo local, lo que produce una baja tasa de aciertos. El valor óptimo x para los cuatro métodos están el intervalo $[0.55, 0.65]$, ya que es estos valores maximizan la tasa de acierto.

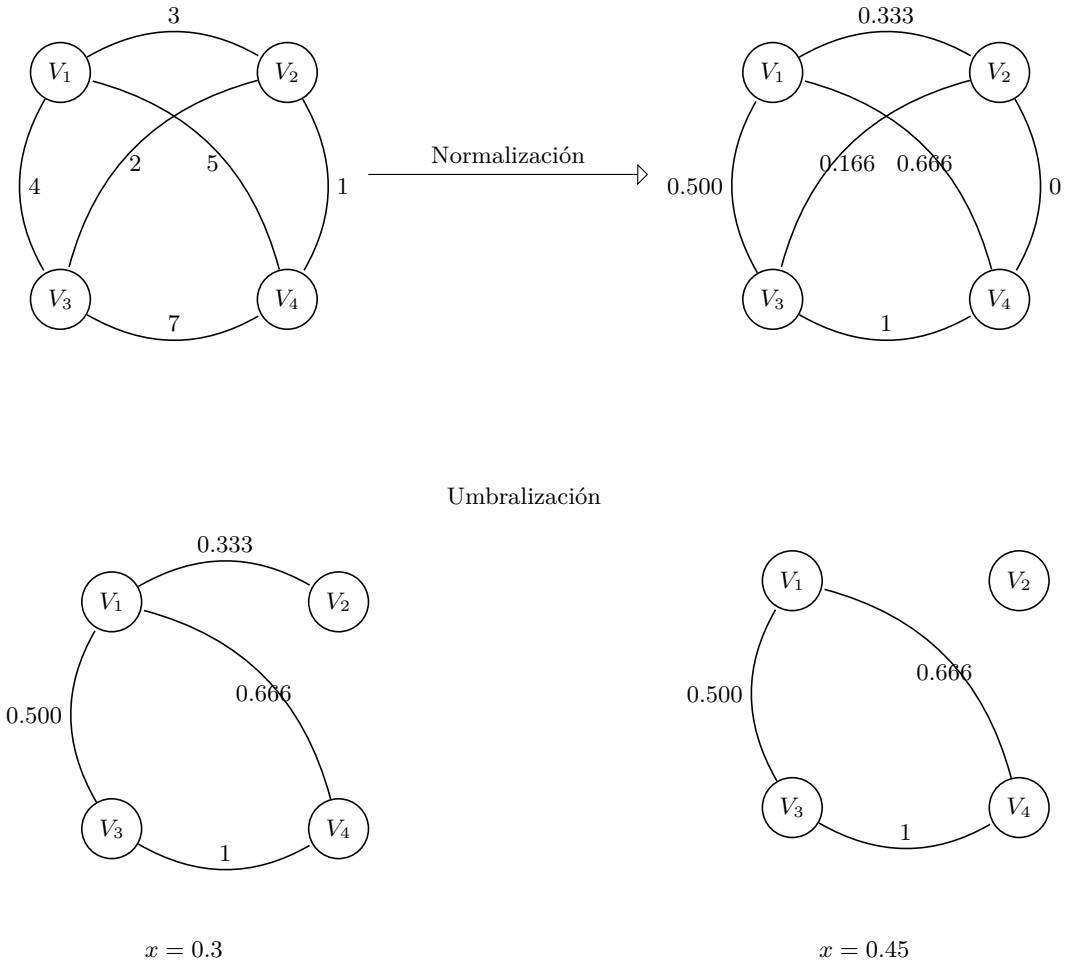


Figura 5.7: Ejemplo de umbralización de una red parenclítica. En primer lugar, se normaliza las aristas de la red en el intervalo $[0, 1]$, y por cada $x \in [0, 1]$ se eliminan las aristas menores o iguales que x

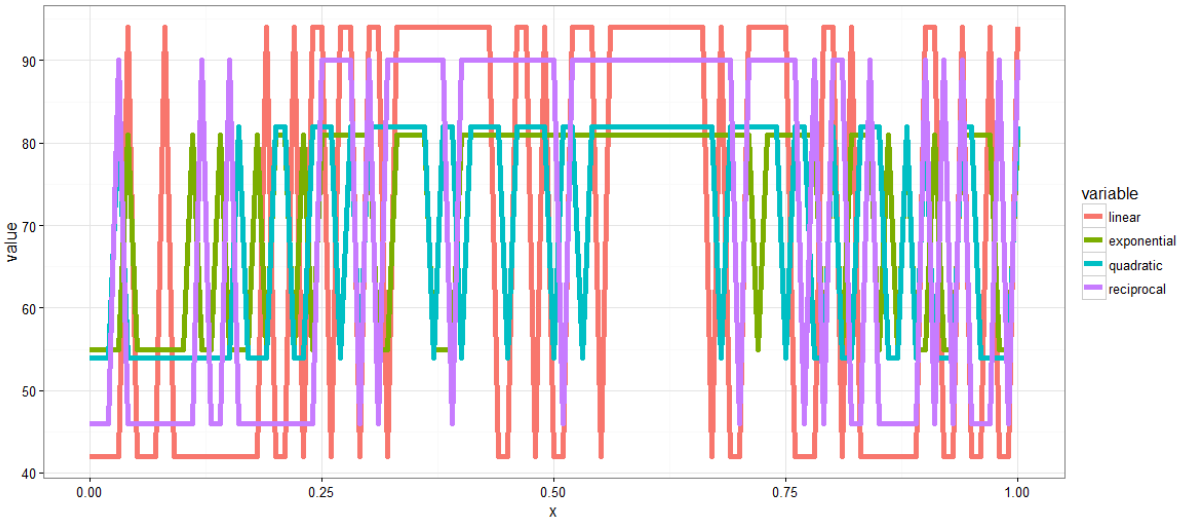


Figura 5.8: Porcentaje de acierto por cada tipo de método de regresión, usando la umbralización de las redes parenclíticas

6

Conclusiones

Durante la memoria hemos visto qué es el Machine Learning, qué tipos existen según el tipo de aprendizaje y qué tareas permiten resolver. Para cada uno de los tipos de aprendizaje hemos visto qué es lo que caracteriza a cada uno, qué tareas se pueden resolver y algunos de los algoritmos que se pueden aplicar.

Hemos visto que es posible usar la teoría de grafos para desarrollar un nuevo algoritmo de Machine Learning para realizar la tarea de clasificación binaria. Hemos visto cómo generalizar el método a una clasificación multiclase, cómo aplicar distintos métodos de regresión para realizar la clasificación.

Hemos desarrollado una aplicación que analiza y compara un conjunto de datos según distintos algoritmos de Machine Learning.

Por último, hemos aplicado la aplicación al diagnóstico de tumores de cáncer de mama, obteniendo grandes resultados, aunque sin llegar al nivel de los algoritmos clásicos. Además, hemos visto que umbralizando las redes parenclíticas es posible obtener mejores resultados que con los grafos completos.

6.1 Mejoras y futuro trabajo

- Implementación de un módulo de validación para comprobar que los resultados no dependen de la selección aleatoria de los conjuntos de entrenamiento y prueba.

- Optimización de los algoritmos de redes parenclíticas: se deben optimizar los algoritmos de aprendizaje de redes parenclíticas para minimizar la cantidad de memoria empleada y el tiempo de procesamiento para conjuntos de datos muy grandes.

- Mejora en visualización de la aplicación: mejora de la interfaz gráfica para conseguir una mejor experiencia de usuario, con la consiguiente migración a HTML 5 y CSS 3 "nativo".
- Nuevos métodos de regresión: adición de nuevos métodos de regresión como la regresión logística o la no lineal, o incluso, algoritmos de Machine Learning que permitan la tarea de regresión, como SVM o redes neuronales.
- Persistencia de las redes parenclíticas en bases de datos de grafos: almacenamiento persistente de las redes parenclíticas en bases de datos de grafos como OrientDB [9], Titan [45] o Neo4j [27] para realizar futuros procesamientos o visualizaciones.

A

Despliegue de la aplicación en www.shinyapps.io

Para instalar la aplicación necesitaremos seguir los siguientes pasos[1]:

1. Instalar R.

```
sudo apt-get install r-base
```

2. Instalar el paquete *Shiny* de R.

```
sudo su - -c "R -e "\"
↪ install.packages('shiny',
↪ repos='https://cran.rstudio.com/') \""
```

3. Descargar e instalar RStudio [30]

4. Instalar Shiny Server.

```
sudo apt-get install gdebi-core
wget https://downloads3.rstudio.org/ubuntu-
↪ 12.04/x86_64/shiny-server-1.4.2.786-amd64.deb
sudo gdebi shiny-server-1.4.2.786-amd64.deb
```

5. Ir a la carpeta *r-package* del código.

```
cd r-package
```

[1]De ahora en adelante, supondremos que estamos en un entorno Ubuntu. Para instalar R y RStudio en Windows y Mac OS X seguir las instrucciones de [19].

6. Hacer doble click sobre *r-package.Rproj*

7. Instalar los siguientes paquetes de R dentro de la sesión de RStudio.

```
install.packages(c("shinydashboard", "plotly",
↪  "DT", "igraph", "rmarkdown", "pander",
↪  "gridExtra", "stringi", "e1071", "nnet",
↪  "rpart", "ggplot2", "lattice", "lme4"))
```

8. Hacer click en *Run App* (Figura A.1).

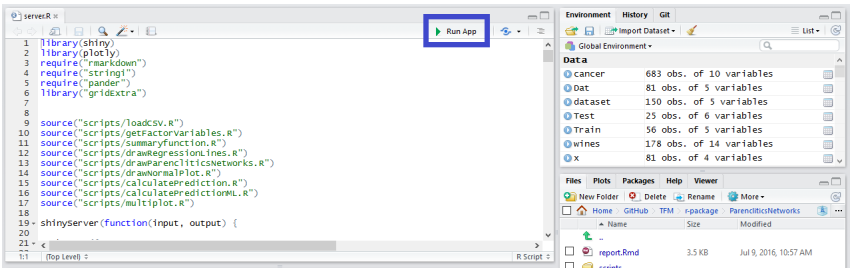


Figura A.1: Hacer click en *Run App*

9. Hacer click en *Publish* (Figura A.2). R pedirá instalar unos paquetes necesarios para desplegar la aplicación. Respondemos afirmativamente.

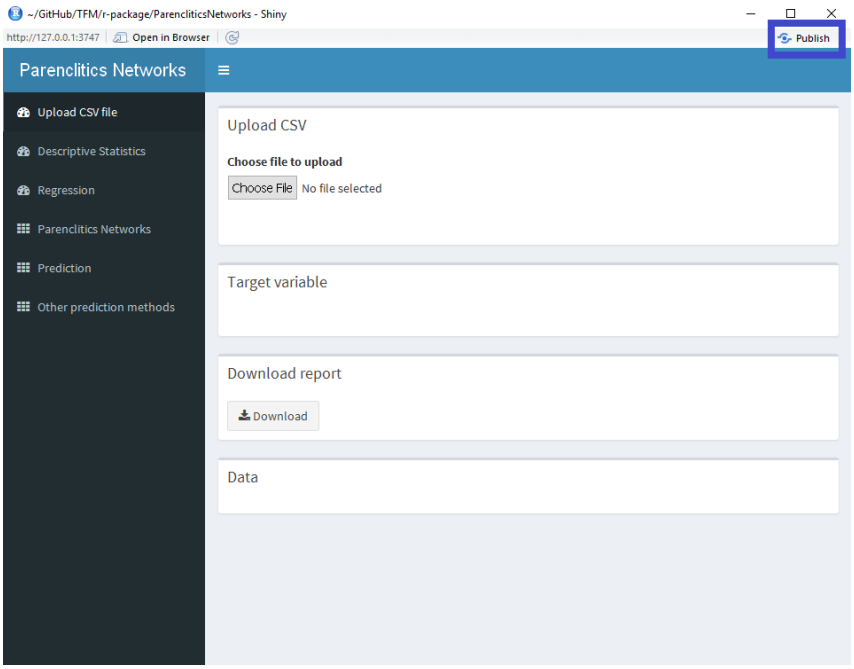


Figura A.2: Hacer click en *Publish*

10. Crear una cuenta en `www.shinyapps.io`.
11. Dar un nombre al dominio.
12. Hacer click en la opción *Tokens* del menú *Account*. Hacer click en *Show* y en *Copy to clipboard*.
13. Pegar en el cuadro de texto de RStudio y hacer click en *Ok*.
14. Tras unos minutos, la aplicación estará desplegada en la nube.

Bibliografía

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] Breast Cancer Wisconsin (Diagnostic) Data Set. `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`. Accedido el 03/07/2016.

[3] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[5] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000.

[6] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[7] Luciano da F. Costa, Francisco A. Rodrigues, Gonzalo Travieso, and P. R. Villas Boas. Characterization of complex networks: A survey of measurements. 2005.

[8] Reinhard Diestel. *Graph theory*. Graduate texts in mathematics. Springer, New York, Berlin, Paris, 1997. Trad. de : Graphentheorie, paru en 1996 chez Springer.

[9] OrientDB: Distributed Multi-Model Graph/Document Database. `orientdb.com`. Accedido el 13/11/2016.

[10] e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. `https://cran.r-project.org/web/packages/e1071/index.html`. Accedido el 02/07/2016.

[11] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

[12] Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx. *Regression - Models, Methods and Applications*. Springer, 2013.

[13] Tristan Fletcher. Machine learning for financial market prediction.

[14] ggplot2: An Implementation of the Grammar of Graphics. `https://cran.r-project.org/web/packages/ggplot2/index.html`. Accedido el 02/07/2016.

[15] Glossary of Terms of Machine Learning. `http://robotics.stanford.edu/~ronnyk/glossary.html`. Accedido el 12/11/2016.

[16] G.E. Hinton and T.J. Sejnowski. *Unsupervised Learning: Foundations of Neural Computation*. A Bradford Book. MCGRAW HILL BOOK Company, 1999.

[17] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, March 1991.

[18] igraph: Network Analysis and Visualization. `https://cran.r-project.org/web/packages/igraph/index.html`. Accedido el 02/07/2016.

[19] Install R, RStudio, and R Commander in Windows and OS X. `https://www.andrewheiss.com/blog/2012/04/17/install-r-rstudio-r-commander-windows-osx`. Accedido el 13/11/2016.

[20] Introducción al Big Data sin perderse en un mar de DATOS. `http://www.gmv.com/blog_gmv/introduccion-al-big-data`. Accedido el 13/11/2016.

[21] Alexander Karsakov, Thomas Bartlett, Iosif Meyerov, Alexey Zaikin, and Mikhail Ivanchenko. Parenclitic network analysis of methylation data for cancer identification, 2015.

[22] Machine Learning. `https://global.britannica.com/technology/machine-learning`. Accedido el 12/11/2016.

[23] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[24] J.H. Mathews and K.D. Fink. *Numerical Methods Using MATLAB*. Featured Titles for Numerical Analysis Series. Pearson Prentice Hall, 2004.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[26] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[27] Neo4j: The World's Leading Graph Database. https://neo4j.com. Accedido el 13/11/2016.

[28] nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models. https://cran.r-project.org/web/packages/nnet/index.html, note = Accedido el 02/07/2016.

[29] A. B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, 1962.

[30] Open source and enterprise-ready professional software for R. https://www.rstudio.com/. Accedido el 09/07/2016.

[31] The R Project for Statistical Computing. https://www.r-project.org. Accedido el 02/07/2016.

[32] R Contributed Packages. https://cran.r-project.org/web/packages. Accedido el 02/07/2016.

[33] Lior Rokach and Oded Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2008.

[34] Roundup Of Analytics, Big Data & BI Forecasts And Market Estimates, 2016. http://www.forbes.com/sites/louiscolumbus/2016/08/20/roundup-of-analytics-big-data-bi-forecasts-and-market-estimates-201 Accedido el 13/11/2016.

[35] rpart: Recursive Partitioning and Regression Trees. https://cran.r-project.org/web/packages/rpart/index.html. Accedido el 02/07/2016.

[36] RStudio Server. https://www.rstudio.com/products/rstudio/#Server. Accedido el 02/07/2016.

[37] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[38] Shiny: A web application framework for R. http://shiny.rstudio.com. Accedido el 02/07/2016.

[39] Shiny Dashboard. https://rstudio.github.io/shinydashboard. Accedido el 02/07/2016.

[40] Shiny Server. https://www.rstudio.com/products/shiny/shiny-server. Accedido el 10/07/2016.

[41] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[42] Phil Simon. *Too big to ignore : The business case for big data*. Wiley, New Delhi, 2013 (R).

[43] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[44] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[45] Titan: Distributed Graph Database. `titan.thinkaurelius.com`. Accedido el 13/11/2016.

[46] What is big data? `https://www.oreilly.com/ideas/what-is-big-data`. Accedido el 13/11/2016.

[47] M. Zanin, D. Papo, P.A. Sousa, E. Menasalvas, A. Nicchi, E. Kubik, and S. Boccaletti. Combining complex networks and data mining: Why and how. *Physics Reports*, 635:1 – 44, 2016. Combining complex networks and data mining: Why and how.

[48] Massimiliano Zanin, Joaquín Medina Alcazar, Jesus Vicente Carbajosa, Marcela Gomez Paez, David Papo, Pedro Sousa, Ernestina Menasalvas, and Stefano Boccaletti. Parenclitic networks: uncovering new functions in biological data. *Scientific Reports*, 4:5112 EP –, May 2014. Article.

[49] Massimiliano Zanin, Joaquín Medina Alcazar, Jesus Vicente Carbajosa, David Papo, M. Gomez Paez, Pedro Sousa, Ernestina Menasalvas, and Stefano Boccaletti. Parenclitic networks: a multilayer description of heterogeneous and static data-sets, 2013.

[50] Massimiliano Zanin, David Papo, José Luis González Solís, Juan Carlos Martínez Espinosa, Claudio Frausto-Reyes, Pascual Palomares Anda, Ricardo Sevilla-Escoboza, Rider Jaimes-Reategui, Stefano Boccaletti, Ernestina Menasalvas, and Pedro Sousa. Knowledge discovery in spectral data by means of complex networks. *Metabolites*, 3(1):155, 2013.

[51] Massimiliano Zanin, Pedro Sousa, David Papo, Ricardo Bajo, Juan García-Prieto, Francisco del Pozo, Ernestina Menasalvas, and Stefano Boccaletti. Optimizing functional network representation of multivariate time series. *Sci Rep*, 2:630, Sep 2012. 22953051[pmid].