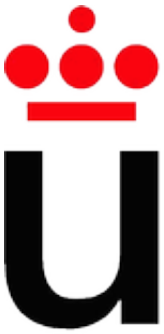


Universidad
Rey Juan Carlos

Título

Trabajo Fin de Máster

Escuela Técnica Superior de Ingeniería Informática

Ingeniería de Sistemas de Decisión

Curso 2015–2016

José Ignacio Escribano Pablos

Tutores:
Tutores

Índice general

Índice de figuras iii

Índice de tablas v

1 Introducción 1

2 Introducción al Machine Learning 3
2.1. Aprendizaje supervisado 4
2.1.1. El proceso de Machine Learning 6
2.1.2. Redes neuronales 7
2.1.3. Support Vector Machine 11
2.1.4. Árboles de decisión 14
2.1.4.1. Algoritmo ID3 14
2.2. Aprendizaje no supervisado 18
2.2.1. Algoritmo de las k-Medias 19
2.3. Aprendizaje por refuerzo 20

3 Redes parenclíticas 21
3.1. Introducción a la teoría de grafos 22
3.1.1. Algunas medidas 28
3.2. Método de redes parenclíticas 28

4 Diseño de la aplicación 29
4.1. Tecnología utilizada 29

5 Aplicaciones 31

6 Conclusiones 33
6.1. Mejoras y futuro trabajo 33

Bibliografía 35

Índice de figuras

2.1. Estructura de una neurona 8
2.2. Neurona de McCulloch y Pitts 8
2.3. Perceptrón 9
2.4. Conjuntos linealmente separables y no separables 10
2.5. Función XOR 10
2.6. Perceptrón multicapa 11
2.7. Distintos clasificadores lineales 12
2.8. Margen en SVM 12
2.9. Kernel 13
2.10. Árbol de decisión 14
2.11. Árbol de decisión tras la ejecución del algoritmo ID3 18
2.12. Esquema del aprendizaje por refuerzo 20

3.1. Esquema de redes parenclíticas 22
3.2. Ejemplo de grafo 23
3.3. Ejemplo de grafo completo 23
3.4. Ejemplo de unión de unión e intersección de grafos 25
3.5. Ejemplo de subgrafos de un grafo 26
3.6. Ejemplo de grafo dirigido con bucle 27
3.7. Ejemplo de grafo ponderado 27

Índice de tablas

2.1. Datos para regresión lineal 6
2.2. Linealización de distintos modelos 7
2.3. Definición de la función XOR 10
2.4. Datos para el ejemplo del algoritmo ID3 16

1

Introducción

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. 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.

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.

  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.

  Por ejemplo, el valor de las acciones de una determinada empresa a lo largo del tiempo puede ser 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).

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.

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.

  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.

  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.

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 $(\mathbf{x}, \mathbf{y})$, donde x_i son las entradas, y y_i son las salidas.

Una característica importante de los algoritmos de machine learning es la capacidad de generalización: el algoritmo debería de 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 en 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.

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 $[0, \infty)$.

Regresión lineal

La regresión lineal viene dada por

$$y_i = \sum_{i=1}^{n} \alpha_i x_i + \alpha_0 \tag{2.1}$$

donde y_i es la variable salida y x_i es la variable de entrada.

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

$$\alpha = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.2}$$

En el caso bidimensional el modelo viene dado por

$$y = \alpha_1 x + \alpha_0 \tag{2.3}$$

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

$$\begin{cases} \left(\sum_{i=1}^{n} x_i^2\right)\alpha_1 & + & \left(\sum_{i=1}^{n} x_i\right)\alpha_0 & = & \sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} x_i\right)\alpha_1 & + & n\alpha_0 & = & \sum_{i=1}^{n} y_i \end{cases} \tag{2.4}$$

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,

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

Resolviendo el sistema lineal, se tiene que,

$$\begin{aligned} \alpha_1 &\approx -1.607 \\ \alpha_0 &\approx 8.642 \end{aligned}$$

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

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 método permite ajustar modelos que, en principio, no son lineales como, por ejemplo,

$$y = ax^m$$

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

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 distintos modelos.

2.1.1 El proceso de Machine Learning

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

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.

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. 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.

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.

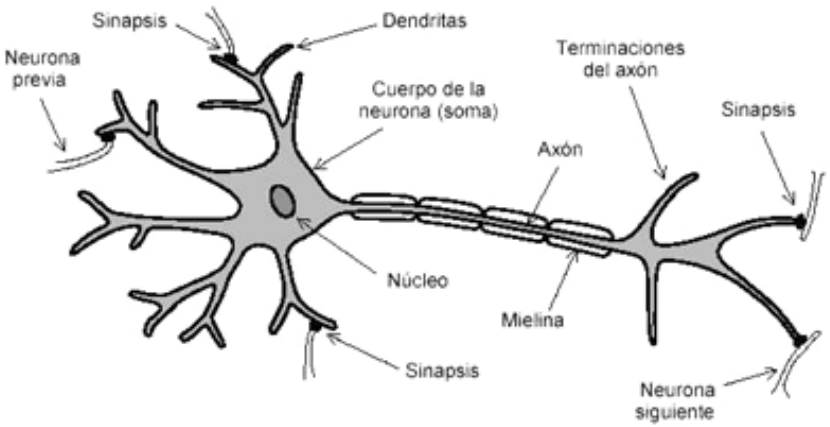


Figura 2.1: Estructura de una neurona

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

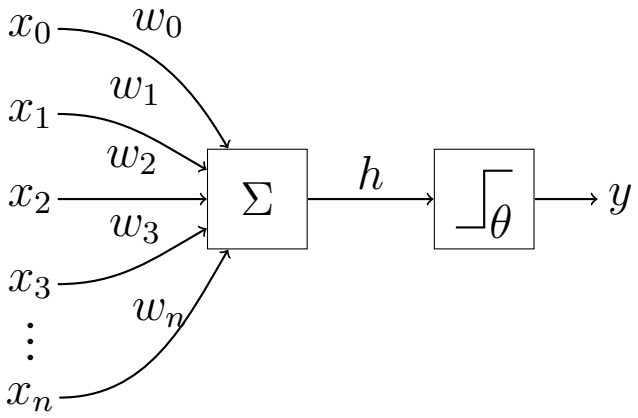


Figura 2.2: Neurona de McCulloch y Pitts

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

Llamaremos h a

$$h = \sum_{i=1}^{n} w_i x_i \tag{2.5}$$

a la suma de las entradas multiplicadas por los pesos.

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

$$y = g(h) = \begin{cases} 1 & \text{si } h > \theta \\ 0 & \text{si } h \leq \theta \end{cases} \tag{2.6}$$

Un problema obvio de la neurona de MacCulloch 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.

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

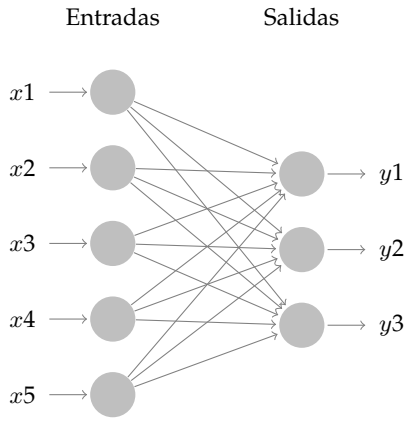


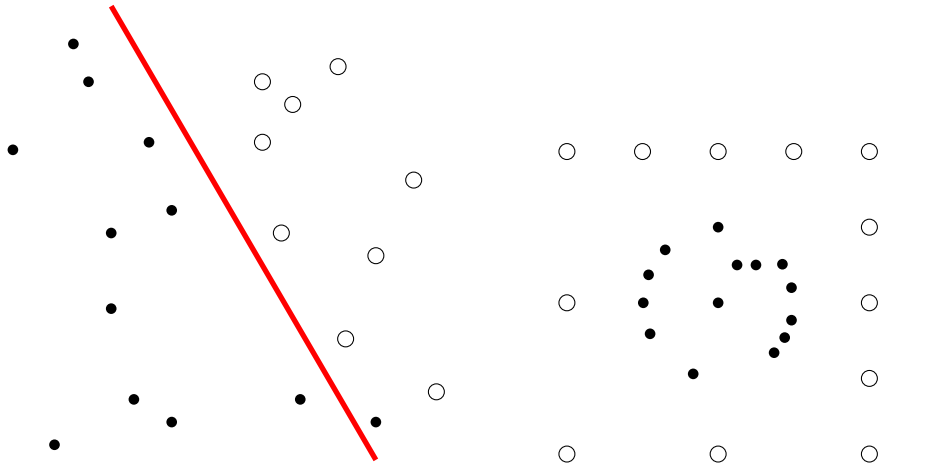
Figura 2.3: Perceptrón

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. Un conjunto es linealmente separable si existe un hiperplano que separe dos clases.
Un ejemplo clásico que no puede aprender un perceptrón es la función XOR: $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$, que se define de la como se ve en la Tabla 2.3.

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

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



(a) Linealmente separable (b) No linealmente separable

Figura 2.4: Conjuntos linealmente separables y no separables

Tabla 2.3: Definición de la función XOR

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

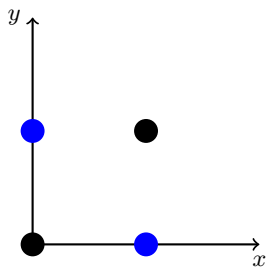


Figura 2.5: Función XOR

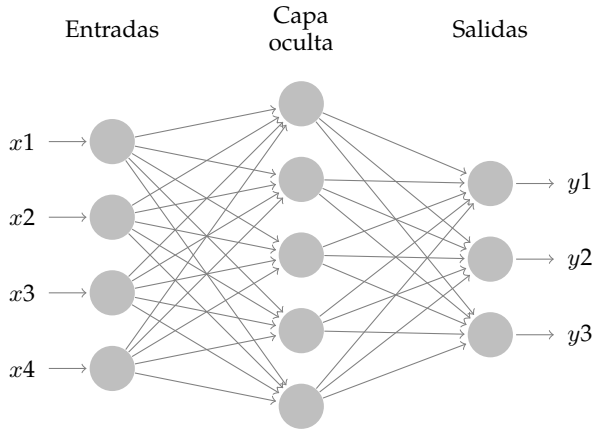


Figura 2.6: Perceptrón multicapa

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

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 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. 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.

La Figura **??** 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.

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 que el problema anterior se puede plantear como un problema de programación cuadrática con la siguiente estructura:

$$\min \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{2.7}$$

$$\text{s.a.} \quad y_i(\mathbf{w}^T x_i + b) \geq 1 \; \forall i = 1, \ldots, n \tag{2.8}$$

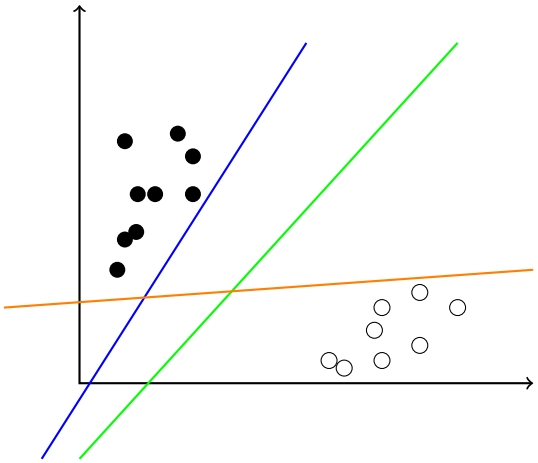


Figura 2.7: Distintos clasificadores lineales

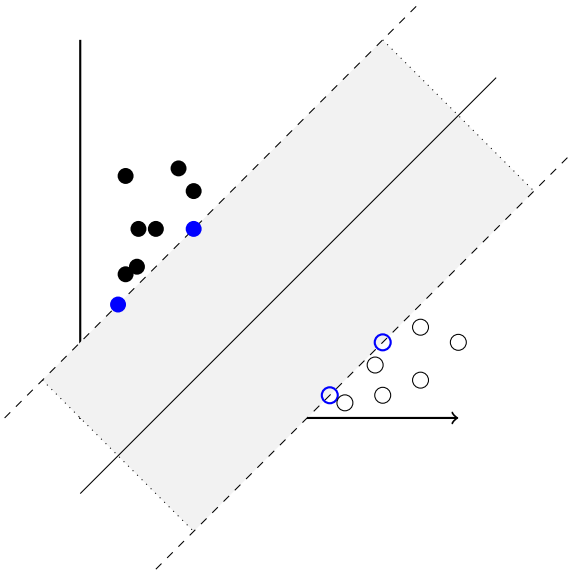


Figura 2.8: Margen en SVM

donde **x** es el vector de entradas, **w** es un vector de 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).

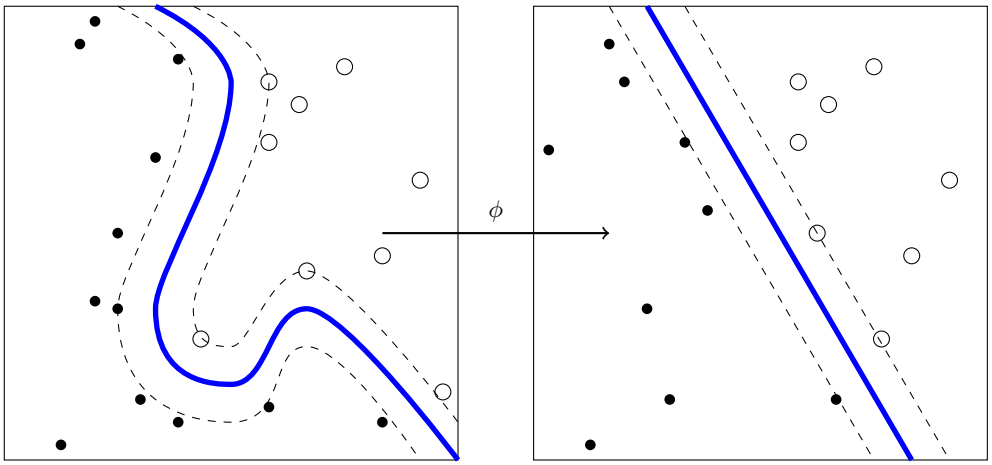


Figura 2.9: Kernel

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

Los kernels más habituales son:

- Polinomiales de grado s

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^s \tag{2.9}$$

- Sigmoidales con parámetros κ y δ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} - \delta) \tag{2.10}$$

- Funciones de base redial con parámetro σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\left(\frac{\mathbf{x} - \mathbf{y}}{2\sigma^2}\right)^2) \tag{2.11}$$

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 **??**.

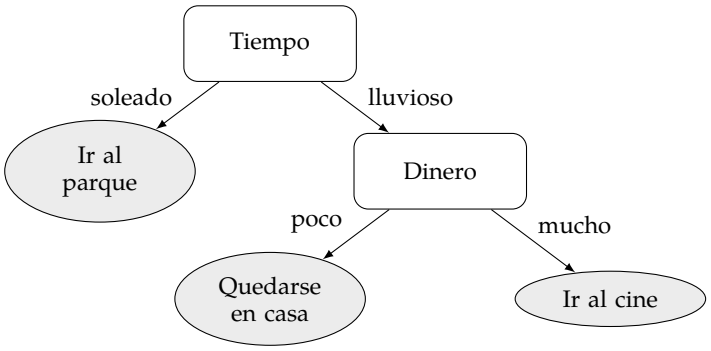


Figura 2.10: Árbol de decisión

Una de las ventajas de los árboles de decisión es que pueden convertirse en una unión de conjunción programarse de la forma "Si ... Entonces ...".

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 se define como

$$E(p) = -\sum_i p_i \log_2 p_i \tag{2.12}$$

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

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

Entonces,

$$E(p) = -(0.6 \log_2 0.6 + 0.4 \log_2 0.4) \tag{2.13}$$
$$= -(-0.44 - 0.52) \tag{2.14}$$
$$= 0.96 \tag{2.15}$$

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

$$G(S, F) = E(S) - \sum_{f \in \text{valores}(F)} \frac{|S_f|}{|S|} E(S_f) \tag{2.16}$$

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 etiqueta,
  - 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.

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

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

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

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

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 **??**.

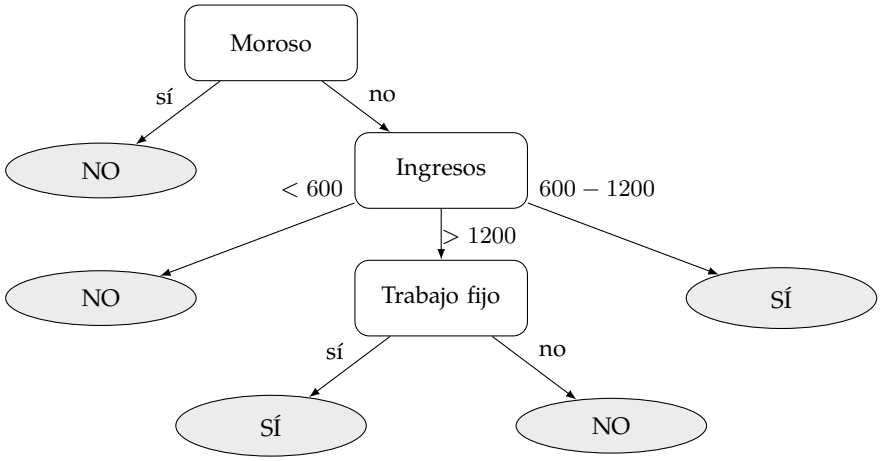


Figura 2.11: Árbol de decisión tras la ejecución del algoritmo ID3

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. El aprendizaje no supervisado 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.

2.2.1 Algoritmo de las k-Medias

Supongamos que queremos dividir nuestros 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 medio 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

$$d_i = \min_j d(x_i, \mathbf{x_i}) \tag{2.17}$$

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

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i \tag{2.18}$$

      donde N_j es el número de puntos del clúster j.
  - Hasta que el centro del clúster para de moverse.

2.3 Aprendizaje por refuerzo

El aprendizaje por refuerzo rellena el hueco entre el aprendizaje supervisado, donde el algoritmo se 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 se 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 busca 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 [7, 9] se pueden encontrar algunos algoritmos sobre aprendizaje por refuerzo y algunas de sus aplicaciones.

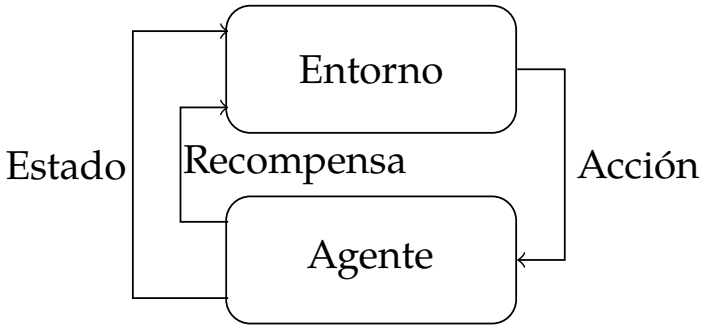


Figura 2.12: Esquema del aprendizaje por refuerzo

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 [11, 12] (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.

En [10] 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 [13] se usan las redes parenclíticas para la optimización de series temporales multivariantes.

En [6] se utilizan redes parenclíticas para la clasificación de los datos de metilzación 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 ver en la Figura **??**. Partiendo de unos datos estructurados, 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 se obtienen distintas medidas, que se pasan a un algoritmo de machine learning clásico, que obtiene la clasificación final.

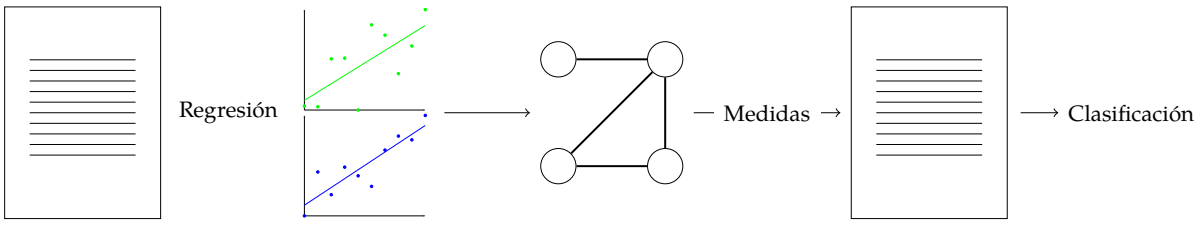


Figura 3.1: Esquema de redes parenclíticas

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.

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 véctices $x, y \in V$ 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 dibujados los vértices y los arcos es irrelevante. sólo importa qué pares de nodos forman una arista y cuáles no.

Ejemplo 7. La Figura 3.2 muestra la representación gráfica de un grafo. Matemáticamente, el grafo es el par (V, E) donde

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

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.

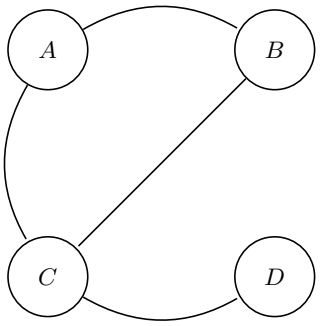


Figura 3.2: Ejemplo de grafo

Ejemplo 8. El grafo de la Figura 3.2 es un grafo finito, puesto que el número de vértices del grafo es 4.

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 3.3 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.

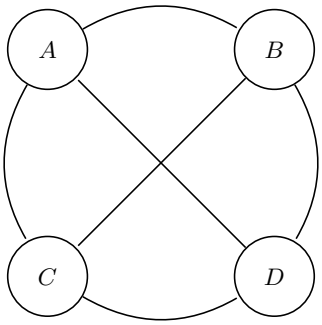


Figura 3.3: Ejemplo de grafo completo

Definición 5. Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos. Decimos que G y G' son isomorfos, y escribimos $G \simeq G'$, si existe una biyección $\phi : V \to V'$ tal que $xy \in E \iff \phi(x)\phi(y) \in E' \ \forall x, y \in V$. La aplicación ϕ recibe el nombre de isomorfismo. Si $G = G'$, ϕ se dice que es un automorfismo.

Podemos definir operaciones sobre grafos, como la unión o la intersección.

Definición 6. Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos, se definen la unión y la intersección de grafos como

$$G \cup G' := (V \cup V', E \cup E')$$
$$G \cap G' := (V \cap V', E \cap E')$$

Si $G \cap G' = \emptyset$, entonces G y G' son disjuntos.

Ejemplo 10. La Figura 3.4 muestra la unión e intersección de grafos. Si G y G' son respectivamente

$$G = (\{A, B, C, D, E\}, \{\{A, B\}, \{B, C\}, \{B, D\}, \{C, E\}, \{D, E\}\})$$
$$G' = (\{C, D, E, F\}, \{\{C, D\}, \{C, E\}, \{D, F\}, \{E, F\}\})$$

Por definición,

$$G \cup G' = (\{A, B, C, D, E, F\}, \{\{A, B\}, \{B, C\}, \{B, D\}, \{C, E\}, \{D, E\}, \{C, D\}, \{D, F\}, \{E, F\}\})$$

$$G \cap G' = (\{C, E\}, \{\{C, E\}\})$$

Definición 7. Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos. Si $V' \subseteq V$ y $E' \subseteq E$, se dice que G' es un subgrafo de G (y G es un supergrafo de G').

Ejemplo 11. La figura 3.5 muestra algunos de los subgrafos de $G = (V, E)$ donde

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

Del mismo modo, $G' = (V', E')$ y $G'' = (V'', E'')$ donde

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

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

Se ve claramente que $V' \subseteq V$ y $E' \subseteq E$, por lo que G' es un subgrafo de G (o G es un supergrafo de G'). Análogo para G''.

Definición 8. Sea $G = (V, E)$ un grafo (no vacío). El grado de un vértice $v \in V$, denotado por $d_G(v) = d(v)$, se define como el número de vértices adyacentes a v.

Si todos los vértices de G tienen el mismo grado k, el grafo G es regular.

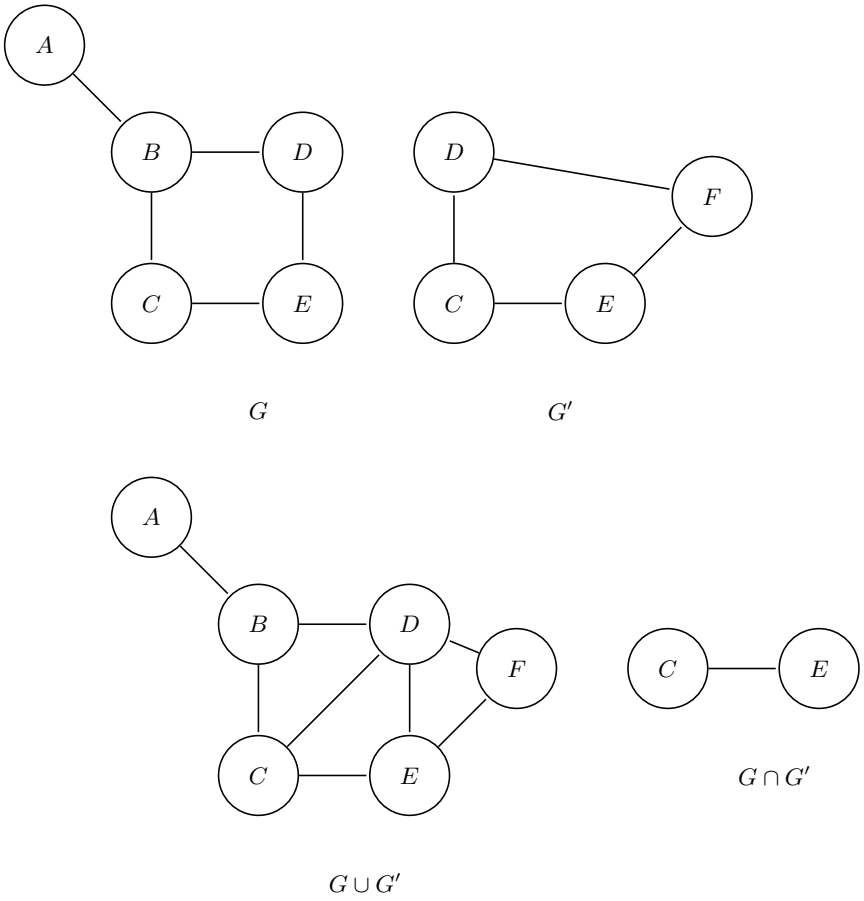


Figura 3.4: Ejemplo de unión de unión e intersección de grafos

Definición 9. Se define el grado medio de un grafo $G = (V, E)$ como el número

$$d(G) = \frac{1}{|V|} \sum_{v \in V} d(v) \tag{3.1}$$

Definición 10. Un camino es un grafo no vacío $P = (V, E)$ de la forma

$$V = \{x_0, x_1, \ldots, x_k\} \qquad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$$

donde $x_i \neq x_j \; \forall i \neq j$.

Los vértices x_0 y x_k se denominan final del camino P. Los vértices $x_1, \ldots, x_k$ se denominan vértices interiores del camino P.

El número de aristas del camino se denomina longitud del camino.

Definición 11. Un grafo no vacío G se dice conexo si cualquier par de vértices están unidos por un camino de G.

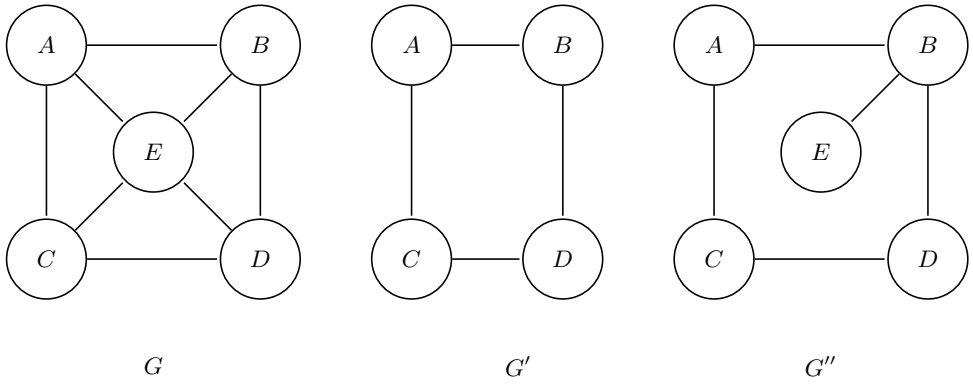


Figura 3.5: Ejemplo de subgrafos de un grafo G

Definición 12. Sea $G = (V, E)$ un grafo. Un subgrafo conexo maximal de G se llama componente conexa de G.

Definición 13. Un clique es un conjunto de nodos mutuamente conectados entre sí.

Ejemplo 12. Un triángulo es un clique formado por tres nodos.

Definición 14. 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 13. La Figura 3.6 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:

init :	E	$\to$	V	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

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

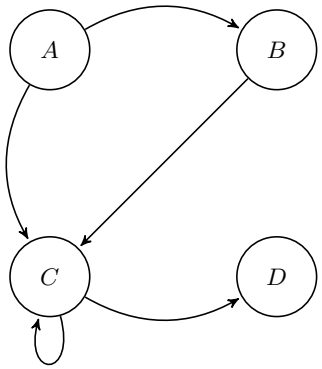


Figura 3.6: Ejemplo de grafo dirigido con bucle

Definición 15. Un grafo ponderado es un grafo con una función 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 14. El grafo $G = (V, E)$ con

$$V = \{A, B, C\} \tag{3.2}$$
$$E = \{\{A, B\}, \{A, C\}, \{B, C\}\} \tag{3.3}$$

es un grafo. Si le añadimos la función w : $E \to \mathbb{R}$ definida de la siguiente forma, G es un grafo ponderado (ver Figura 3.7).

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

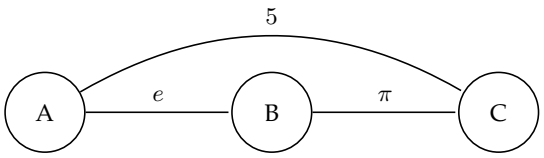


Figura 3.7: Ejemplo de grafo ponderado

Definición 16. Dado un grafo ponderado $G = (V, E)$ y w : $E \to \mathbb{R}$, decimos que una arista $e \in E$ incide en el vértice $v \in V$, si ter$(e) = v$.

Definición 17. La fuerza de un vértice en un grafo ponderado se define como la suma de todos los pesos de sus aristas incidentes.

3.1.1 Algunas medidas

Definición 18. 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.

$$d(G) = \frac{1}{n(n-1)} \sum_{i,j} a_{i,j} \tag{3.4}$$

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 19. Se define coeficiente de clustering C del grafo G como

$$C(G) = \frac{3N_{\triangle}}{N_3} \tag{3.5}$$

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

Definición 20. Se define eficiencia de un grafo G como

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

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 eficiencia es la media armónica de todos los caminos del grafo de competitividad.

Para calcular la distancia entre los nodos del grafo se pueden aplicar los algoritmos de Dijkstra o Floyd (Ver [2]).

Definición 21. Se define longitud del camino característico[2] del grafo G de la siguiente forma:

$$\mathrm{L}(\mathcal{G}) = \frac{1}{\binom{n}{2}} \sum_{i \neq j} d(i,j) \tag{3.7}$$

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.

En [4] se pueden encontrar una extensa lista con distintas medidas sobre grafos.

3.2 Método de redes parenclíticas

[1] Traducción de link density

[2] Traducción de characteristic path length

4

Diseño de la aplicación

4.1 Tecnología utilizada

5

Aplicaciones

6

Conclusiones

6.1 Mejoras y futuro trabajo

Bibliografía

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

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

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

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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].