Algoritmos

Jordi Gironés Roig

PID_00197284

Universitat Oberta de Catalunya

Índice

Introducción

Entender esquemáticamente y también en detalle cómo funcionan algunos de los algoritmos más habituales en BA ayudará al estudiante a utilizar herramientas BA, sabiendo qué pueden dar de sí y a la vez, sabiendo cuáles son los fundamentos científicos que hay detrás de ellas.

Veréis notación y formulación matemática. Este hecho se ha tratado con máxima delicadeza para que el estudiante que no está acostumbrado a la misma no tenga problemas para obviarla y seguir el contenido didáctico sin más problemas, y a la vez, que para el estudiante que sí la domina, pueda usar esta herramienta científica para comprender mejor los conceptos expuestos.

Trabajaremos técnicas de clasificación, segmentación, predicción, jerarquización, asociación y visualización de datos, asimismo, en el documento "Anexo" trabajaremos conceptos estadísticos y algebraicos que constituyen la base matemática de todas estas técnicas.

Es generalmente aceptada la idea de asociar habilidades sintéticas y analíticas a las actividades más o menos científicas como BA; sin embargo, es igualmente importante cultivar habilidades creativas para, a veces buscar y a veces encontrar, patrones nuevos que nos permitan convertir datos en conocimiento.

A través del estudio de los distintos algoritmos, el estudiante observará la aportación creativa de distintos científicos que han contribuido muchas veces, con soluciones sencillas, brillantes y efectivas a problemas realmente complejos *a priori*.

Al finalizar este material didáctico el estudiante será capaz de desarrollar paso a paso los siguientes algoritmos:

- Árbol de decisión
- Análisis de componentes principales
- Asociaciones
- Regresión

Respecto de algoritmos más complejos, el estudiante habrá adquirido un nivel de comprensión esquematizado y general sobre su funcionamiento interno. Estos algoritmos son:

- K-NN
- Redes neuronales
- Support Vector Machines
- K-Means

1. Creación de modelos de datos

Si el scoring consiste en aplicar un algoritmo a un juego de datos con el objetivo de predecir una clase objetivo o encontrar un patrón desconocido, el modelado comprende todas las tareas directamente involucradas en el proceso de construcción y ajuste del mencionado algoritmo.

Empecemos por buscar un origen al nombre y una definición al concepto.

1.1. Familias de algoritmos

1.1.1. Definición de algoritmo

Merece la pena comentar el origen etimológico de la palabra *algoritmo*, por su conexión con los orígenes de las matemáticas modernas. La palabra proviene del nombre *Al-Khwarizmi*, matemático persa que vivió entre los años 750-850. Estrictamente significa 'nativo de Khwarez', actual Uzbekistán.

Su libro más reconocido es *Algebra*, que da nombre a esta disciplina. Sin embargo, el término *algoritmo* trasciende en la historia gracias a otra de sus obras traducida al latín como *Algoritmi de numero Indorum,* que significa 'Algoritmi nos habla sobre los número de la India'. Esos números que despertaron la curiosidad del autor son nuestro actual sistema decimal.

Durante la época medieval, *algoritmo* significaba 'sistema decimal', nada que ver con el concepto moderno.

> "Conjunto de instrucciones bien definidas, ordenadas y finitas, que permiten realizar una actividad mediante pasos sucesivos".
>
> Thomas Cormen (2009). *Introduction to algorithms*. The MIT Press.

Veamos qué tipos de algoritmos podemos encontrar en BA.

1.1.2. Clasificación supervisada

La clasificación supervisada persigue la obtención de un modelo válido para predecir casos futuros a partir del aprendizaje de casos conocidos.

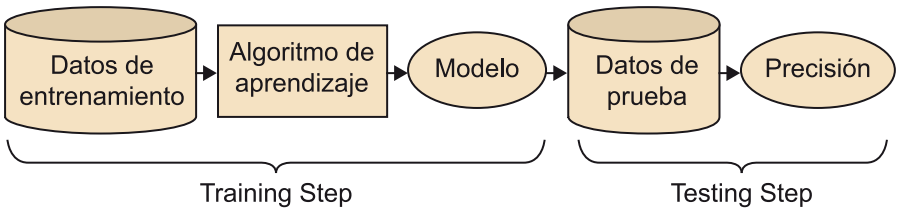
Más formalmente podríamos decirlo del siguiente modo…

A partir de un conjunto de objetos descritos por un vector de características y del que conocemos la clase a la que pertenece cada objeto, se construye un juego de datos llamado de entrenamiento o de aprendizaje, que nos servirá para construir un modelo o regla general que nos va a permitir clasificar objetos nuevos de los que no conocemos la clase a la que pertenecen.

Si la clase a predecir o variable de salida es discreta, diremos que se trata de un problema de clasificación, sin embargo, si la clase a predecir es continua, diremos que se trata de un problema de regresión.

Para los algoritmos de aprendizaje supervisado, una secuencia típica de aprendizaje sería la siguiente:

Figura 1. Aprendizaje supervisado



Fuente: Bing Liu. *Web Data Mining*

Donde vemos que a partir de un juego de datos de entrenamiento se ajusta un modelo de aprendizaje, que acaba estableciendo un modelo, cuyo nivel de precisión es evaluado a partir de un juego de datos de prueba, distinto al de aprendizaje.

Una de las ventajas de este tipo de clasificación es que podemos evaluar la bondad del modelo a partir del cálculo de la tasa de error = $\frac{número\_de\_errores}{número\_total\_de\_casos}$

Otra herramienta para evaluar los modelos de clasificación es la matriz de confusión. Esta nos plasma en una tabla una visión gráfica de los errores cometidos por el modelo.

Figura 2. Matriz de confusión

		Clase verdadera		
		C1	C2	
Clase predicha	C1	a	b	P_1
	C2	c	d	P_2
		π_1	π_2	

Tipos de aciertos:

a = número de clasificaciones correctas en la clase C1 = verdadero positivo.

d = número de clasificaciones correctas en la clase C2 = verdadero negativo.

Tipos de errores:

b = número de clasificaciones incorrectas. Era C2, sin embargo, se clasifica C1 = Error de tipo II o falso negativo.

c = número de clasificaciones incorrectas. Era C1, sin embargo, se clasifica C2 = Error de tipo I o falso positivo.

Proporciones:

p_1 = proporción de casos que el clasificador asigna a la clase C1 = a / (a + b).

p_2 = proporción de casos que el clasificador asigna a la clase C2 = d / (c + d).

π_1 = probabilidad *a priori* de la clase C1.

π_2 = probabilidad *a priori* de la clase C2.

Precisión y *recall*

Supongamos que para un cliente sabemos que hace dos temporadas el 80% de las ventas se concentraron en 10 productos de nuestra cartera.

Con el objetivo de medir la precisión y el *recall* de nuestro algoritmo predictivo, queremos probarlo sobre datos de la temporada pasada.

Nuestro modelo, de los diez productos más vendidos la temporada pasada, clasifica 8 como más vendidos, de los cuales solo 5 realmente estaban entre los 10 más vendidos.

Dicho de otro modo, nuestro modelo ha identificado que 8 productos concentran el 80% de las ventas al cliente, cuando en realidad hay 10 productos que concentran el 80% de las ventas.

Además, de los 8 productos que el modelo ha identificado, 3 eran incorrectos y 5 eran correctos.

Para este modelo diremos que su precisión es 5/8, mientras que su *recall* es 5/10, es decir, tiene una precisión del 62,50% y un *recall* del 50%.

De modo que podemos decir que la precisión es la parte de las instancias clasificadas que eran correctas, mientras que el *recall* es la parte de las instancias correctas que han sido clasificadas.

La capacidad de evaluar la bondad de los modelos de clasificación supervisada hace que sea posible plantear, en el propio algoritmo, tareas de minimización del error, combinando así, bajo un mismo modelo, algoritmos de clasificación y de optimización.

Algoritmos propios de la clasificación supervisada pueden ser K-NN, árboles de decisión, redes bayesianas, redes neuronales, análisis discriminante y máquinas de vectores de soporte o SVM.

1.1.3. Clasificación no supervisada

La clasificación no supervisada persigue la obtención de un modelo válido para clasificar objetos a partir de la similitud de sus características.

Más formalmente podríamos decirlo del siguiente modo:

A partir de un conjunto de objetos descritos por un vector de características y a partir de una métrica que nos defina el concepto de similitud entre objetos, se construye un modelo o regla general que nos va a permitir clasificar todos los objetos.

No se trata de modelos predictivos sino de modelos de descubrimiento de patrones.

Existen dos grandes familias de algoritmos de clasificación no supervisada:

- Los algoritmos **jerárquicos** construyen nodos de forma jerárquica, unos a partir de otros. La representación de los resultados se hace habitualmente mediante dendogramas. Estos se dividen en dos subcategorías:
  - **Aglomerativos o *bottom up***: Esta aproximación parte del supuesto que cada objeto es un nodo o clúster y a medida que evolucionan los pasos del algoritmo, los nodos se van agrupando hasta conseguir un número de nodos aceptable. Un ejemplo es la figura 12.

  - **Divisivos o *top down***: Es la aproximación opuesta, es decir, parte del supuesto que existe un único nodo y a medida que avanza el algoritmo este nodo se va subdividiendo en nuevos nodos, y así sucesivamente.

  Ejemplos de algoritmos jerárquicos pueden ser el método del mínimo o *single linkage* y el método del máximo o *complete linkage*.
- Los algoritmos **particionales**, también llamados algoritmos de optimización, obtienen los nodos a partir de la optimización de una función adecuada para el propósito del estudio. Esta función suele estar relacionada con la métrica seleccionada para establecer el concepto de similitud entre objetos.

Como ejemplos de algoritmos de esta familia podríamos encontrar K-Means.

1.1.4. Algoritmos de aprendizaje reforzado

Los ejemplos de aplicación del aprendizaje por refuerzo más fáciles de identificar son actividades como la conducción automática de vehículos o el pilotaje automático de aviones, pero hay muchas otras, como la interacción de un robot en entornos específicos.

En definitiva, se trata de algoritmos que pretenden capacitar de inteligencia sistemas que básicamente poseen dos características.

1) El aprendizaje de una tarea por parte del sistema se realiza mediante un proceso iterativo de prueba y error en el entorno en el que interactúa.

2) El entorno, de algún modo, debe proporcionar información al sistema sobre si está realizando bien o mal la tarea que está aprendiendo. Esta señal es la que denominamos refuerzo.

Intuitivamente, es el tipo de aprendizaje que pondríamos en práctica con un perro al adiestrarlo para que ejecute alguna actividad. Para ello lo premiamos solo cuando esta se ha ejecutado correctamente. El perro trata iterativamente de repetir una actividad y el dueño lo premia cuando lo hace bien.

Más formalmente diremos que:

> "el aprendizaje por refuerzo consiste en aprender a decidir, ante una situación determinada, qué acción es la más adecuada para lograr un objetivo".
>
> Basilio Sierra Araujo. *Aprendizaje automático: conceptos básicos y avanzados*. Pearson.

De esta definición se desprende que es un tipo de aprendizaje con un fuerte componente selectivo puesto que se debe escoger la mejor acción entre varias posibles.

Por otro lado, también hay un fuerte componente asociativo, puesto que las acciones seleccionadas se asocian a la situación concreta en la que fue tomada. De hecho, como siempre, la realidad es mucho más compleja que la teoría y en el caso de los algoritmos de aprendizaje reforzado lo que se les exige es que esta asociación no se haga de forma individualizada acción a acción, sino que se haga de forma agrupada.

Es decir, el concepto de acción correcta se extiende al concepto de secuencia de acciones correctas. Muchas veces sucede que un coche colisiona, no por una acción incorrecta, sino por una secuencia de acciones que llevan a la colisión, de modo que lo que clasificamos como incorrecto es la secuencia completa de acciones.

El marco matemático en el que se desarrollan este tipo de algoritmos son los procesos de decisión de Markov.

1.2. Clasificación por vecindad. K-Nearest Neighbor

1.2.1. Algoritmo K-NN

K-NN es un algoritmo de aprendizaje supervisado, de modo que a partir de un juego de datos test, su objetivo será el de clasificar correctamente todas las instancias. El juego de datos típico de este tipo de algoritmos está formado por varios atributos descriptivos y un solo atributo objetivo también llamado clase.

En contraste con otros algoritmos de aprendizaje supervisado, K-NN no genera un modelo fruto del aprendizaje con datos de entrenamiento, sino que el aprendizaje sucede en el mismo momento en el que se prueban los datos de test. A este tipo de algoritmos se les llama **lazy learning methods**.

El funcionamiento es el siguiente:

Sea D el juego de datos de entrenamiento, sobre el que no vamos a realizar ningún proceso en específico.

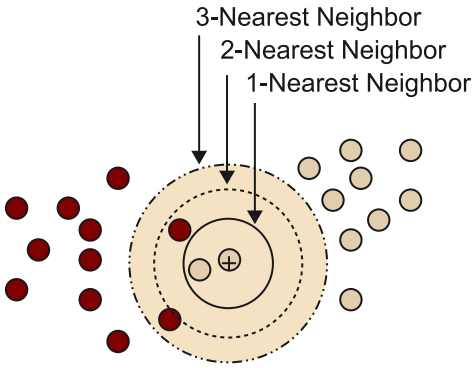
Hacemos que el algoritmo compute una instancia *d* de prueba. Fruto de este proceso, el algoritmo selecciona las *k* instancias más cercanas (de acuerdo con la métrica de similitud utilizada) y se asigna la instancia *d* a la clase más frecuente de entre las instancias seleccionadas como más cercanas.

Sin duda se trata de un algoritmo tremendamente simple y a la par, con un nivel de efectividad similar a otros algoritmos más complejos y elaborados como SVM (Support Vector Machines).

Destacar que k-NN es muy sensible a la variable *k*, de modo que valores distintos nos pueden arrojar resultados también muy distintos. Este valor suele fijarse tras un proceso de pruebas con varias instancias test.

Veámoslo en el siguiente dibujo.

Figura 3. Proceso de clasificación K-NN



Ejemplo extraído del libro *Web Data Mining* de Bing Liu

Para k = 1 el algoritmo clasificará la bola con signo + como blanca.

Para k = 2 el algoritmo no tiene criterio para clasificar la bola con signo +.

Para k>=3 el algoritmo clasificará la bola con signo + como negra.

Su mayor debilidad es la lentitud en el proceso de clasificación puesto que su objetivo no es obtener un modelo optimizado, sino que cada instancia de prueba es comparada contra todo el juego de datos de entrenamiento y será la bondad de los resultados lo que determinará el ajuste de aspectos del algoritmo como el propio valor *k*, el criterio de selección de instancias para formar parte del juego de datos *D* de entrenamiento o la propia métrica de medida de similitud.

1.3. Ganancia de información

Empezaremos por introducir el concepto de **entropía** como medida de cómo de predecible es un resultado en un juego de datos. También puede ser pensada como el grado de desorden o de incertidumbre presente en un juego de datos.

Veamos su expresión matemática:

$$H(Ex) = \sum_{v} \frac{|\{x \in Ex | valor(x) = v\}|}{|Ex|} \cdot \log_2\left(\frac{|Ex|}{|\{x \in Ex | valor(x) = v\}|}\right)$$

Donde |Ex| es el número de experimentos que contiene nuestro juego de datos de entrenamiento Ex.

Donde $|\{x \in Ex | valor(x) = v\}|$ es el número de veces que se da el valor *v* del atributo objetivo *x* o clase, en todo el juego de datos de entrenamiento E_x .

Valores de entropía altos indican que el resultado es muy aleatorio y en consecuencia, poco predecible.

Tomemos como ejemplo los siguientes experimentos:

- Tirar un dado con 6 caras al aire puede darnos 6 posibles resultados {1, 2, 3, 4, 5, 6}. La entropía de este experimento es: $H(Ex) = -6 \cdot \left(\frac{1}{6}\log_2\left(\frac{1}{6}\right)\right) \cong 2{,}58$

- Tirar una moneda al aire puede darnos 2 posibles resultados {cara, cruz}. La entropía de este experimento es: $H(Ex) = -2 \cdot \left(\frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1$

Observamos como la entropía de lanzar un dado con 6 caras es mucho más alta y por ende mucho más aleatoria que la de lanzar una moneda.

La **ganancia de información** o *information gain* nos da una medida de cómo de relevante es un atributo dentro de un juego de datos, de modo que un atributo con mucha ganancia será muy relevante en el juego de datos, es decir, muy determinante para predecir el atributo objetivo o clase.

La ganancia de información refleja el cambio en la entropía del juego de datos cuando tomamos parte de la información como dada.

$$IG(Ex, a) = H(Ex) - H(Ex, a)$$

donde $H(Ex, a) = \sum_{v \in valores(a)} \frac{|\{x \in Ex | valor(x, a) = v\}|}{|Ex|} \cdot H(x \in Ex | valor(x, a) = v)$ es la entropía del juego de datos cuando lo particionamos en base al atributo *a*, o visto de otro modo, es la entropía del atributo *a* en el juego de datos.

En la expresión anterior, el valor $H(x \in Ex | valor(x, a) = v)$ es la entropía del juego de datos cuando fijamos el atributo *a* en su valor *v*, o visto de otro modo, es la entropía del atributo *a* cuando toma el valor *v* en el juego de datos Ex.

Para hacerlo más comprensible, utilizaremos el ejemplo que Bing Liu en su libro Web Data Mining nos propone.

Tabla 1. Juego de datos de solicitudes de préstamo bancario

ID	Edad	Tiene Trabajo	Tiene Casa	Calificación	Clase
1	joven	Falso	Falso	Normal	**no**
2	joven	Falso	Falso	Bueno	**no**
3	joven	Verdadero	Falso	Bueno	**sí**
4	joven	Verdadero	Verdadero	Normal	**sí**
5	joven	Falso	Falso	Normal	**no**
6	medio	Falso	Falso	Normal	**no**

ID	Edad	Tiene Trabajo	Tiene Casa	Calificación	Clase
7	medio	Falso	Falso	Bueno	**no**
8	medio	Verdadero	Verdadero	Bueno	**sí**
9	medio	Falso	Verdadero	Excelente	**sí**
10	medio	Falso	Verdadero	Excelente	**sí**
11	mayor	Falso	Verdadero	Excelente	**sí**
12	mayor	Falso	Verdadero	Bueno	**sí**
13	mayor	Verdadero	Falso	Bueno	**sí**
14	mayor	Verdadero	Falso	Excelente	**sí**
15	mayor	Falso	Falso	Normal	**no**

De acuerdo con nuestra notación anterior, estableceremos que la tabla anterior es a partir de ahora nuestro juego de datos *Ex* donde *x* representa el atributo objetivo o clase.

Para empezar, calculemos la entropía de todo el juego de datos *Ex*. Como hay 6 instancias con clase no y 9 instancias con clase sí, tendremos que:

$$H(Ex) = -\frac{6}{15}\log_2\frac{6}{15} - \frac{9}{15}\log_2\frac{9}{15} = 0{,}971$$

Calcularemos ahora la entropía del juego de datos fijando el atributo *edad*:

$$H(Ex, edad) = \frac{5}{15}H(Ex, edad = joven) + \frac{5}{15}H(Ex, edad = medio) + \frac{5}{15}H(Ex, edad = mayor)$$

Determinemos el valor de las entropías parciales, para cada valor del atributo *edad*:

$$H(Ex, edad = joven) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0{,}971$$

Puesto que de entre las instancias con atributo *edad* = *joven* tenemos 3 clases no y 2 clases sí:

$$H(Ex, edad = medio) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0{,}971$$

Puesto que una vez fijado el valor edad = medio, tenemos 2 clases no y 3 clases sí:

$$H(Ex, edad = mayor) = -\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0{,}722$$

Puesto que una vez fijado el valor edad = mayor, tenemos 4 clases no y 1 clase sí.

Finalmente, sustituyendo los valores de las entropías por valor, tenemos la entropía del atributo *edad*:

$$H(Ex, edad) = \frac{5}{15} \cdot 0{,}971 + \frac{5}{15} \cdot 0{,}971 + \frac{5}{15} \cdot 0{,}722 = 0{,}888$$

Llegados a este punto, estamos en disposición de calcular la ganancia de información que nos ofrece el atributo edad:

$$IG(Ex, edad) = H(Ex) - H(Ex, edad) = 0{,}971 - 0{,}888 = 0{,}083$$

Deberíamos comparar esta ganancia con la ganancia del resto de atributos. Veámoslo:

$$IG(Ex, tiene\_casa) = 0{,}971 - 0{,}551 = 0{,}420$$

$$IG(Ex, tiene\_trabajo) = 0{,}971 - 0{,}647 = 0{,}324$$

$$IG(Ex, tiene\_calificacion) = 0{,}971 - 0{,}608 = 0{,}363$$

Claramente el atributo *tiene casa* ofrece mayor ganancia de la información que el resto de atributos. Esto significa que en realidad es el atributo más determinante para predecir el atributo *objetivo* o *clase*, que en nuestro juego de datos es el que determina si el préstamo se aconseja conceder o no.

El concepto de ganancia de la información es clave para la construcción de algoritmos como los árboles de decisión.

1.4. Árboles de decisión

1.4.1. Generalidades

Se trata de algoritmos de clasificación supervisada que dan como resultado un modelo en forma de árbol, llamado árbol de decisión. Es una técnica muy utilizada porque la representación gráfica del árbol facilita mucho la comprensión del modelo.

Estudiaremos el algoritmo C4.5 propuesto por Quinlan, 1993.

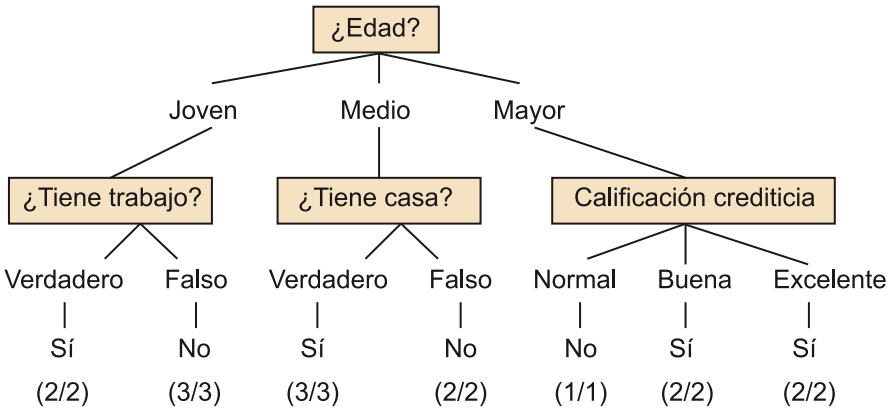
1.4.2. Algoritmo C4.5

Para presentar el funcionamiento de C4.5 tomemos el mismo juego de datos que nos ha servido para estudiar la ganancia de la información (tabla 1).

Observamos cómo un árbol de decisión consiste en un nodo principal construido a partir de uno de los atributos del juego de datos. La pregunta ¿cuál es la edad del solicitante? Nos servirá para segmentar el juego de datos en tres secciones: joven, medio y mayor.

Los nodos internos plantean las preguntas respectivas asociadas al resto de atributos, para al final llegar al nodo final u hoja que contiene la expresión (x/y) con $x \leq y$, significando, por ejemplo, que 2 de cada 2 instancias que han alcanzado la hoja "joven con trabajo" tienen clase "sí", que es la clase que el modelo prevé para esta hoja. O que 3 de cada 3 instancias que han alcanzado la hoja "joven sin trabajo" tienen clase "no", que es la clase que el modelo prevé para esta hoja.

Figura 4. Árbol de decisión



Fuente: *Web Data Mining*, Bing Liu

Probar el modelo de la figura sería tan simple como partir de una instancia nueva e ir respondiendo las preguntas desde el nodo principal, hasta llegar a una hoja donde quedaría determinada la clase a la que nuestro modelo asigna la instancia prueba.

Una vez introducida la representación gráfica de un árbol de decisión, se nos plantea la siguiente pregunta:

Dado un juego de datos, ¿existe un único árbol de decisión que lo representa?

La respuesta es no, de hecho, si tomáramos como nodo principal un atributo distinto de *edad*, obtendríamos un árbol de decisión distinto al de la figura anterior. En cierto modo, el propio árbol de decisión es una descripción intuitiva del juego de datos de entrenamiento.

Siguiendo con la pregunta anterior, uno desearía que el modelo resultante tuviera dos propiedades, la de ser reducido, puesto que facilita su comprensión, y la de ser preciso, puesto que perseguimos la predicción.

De modo que se plantea el reto de, dado un juego de datos, construir el mejor modelo posible en términos de tamaño y precisión.

El algoritmo C4.5 es una de las mejores aproximaciones que existen al problema planteado. Para ello toma el concepto de ganancia de la información.

Para determinar el nodo principal C4.5, calculará la ganancia de la información para cada uno de los atributos del juego de datos de entrenamiento y tomará el atributo con mayor ganancia.

$$IG(Ex, edad) = 0{,}083$$

$$IG(Ex, tiene\_casa) = 0{,}420$$

$$IG(Ex, tiene\_trabajo) = 0{,}324$$

$$IG(Ex, tiene\_calificacion) = 0{,}363$$

En este caso, C4.5 tomará como nodo principal el atributo *Tiene casa* que generará dos particiones, la de Tiene Casa = Verdadero con 6 instancias en clase sí y la de Tiene Casa = Falso que constituirá el nuevo juego de datos (parte sombreada en la siguiente figura) para la siguiente iteración.

Tabla 2. Juego de datos para el caso Tiene Casa = Falso

ID	Edad	Tiene Trabajo	Tiene Casa	Calificación	Clase
1	joven	Falso	Falso	Normal	**no**
2	joven	Falso	Falso	Bueno	**no**
3	joven	Verdadero	Falso	Bueno	**sí**
4	joven	Verdadero	Verdadero	Normal	**sí**
5	joven	Falso	Falso	Normal	**no**
6	medio	Falso	Falso	Normal	**no**
7	medio	Falso	Falso	Bueno	**no**
8	medio	Verdadero	Verdadero	Bueno	**sí**
9	medio	Falso	Verdadero	Excelente	**sí**
10	medio	Falso	Verdadero	Excelente	**sí**
11	mayor	Falso	Verdadero	Excelente	**sí**
12	mayor	Falso	Verdadero	Bueno	**sí**

ID	Edad	Tiene Trabajo	Tiene Casa	Calificación	Clase
13	mayor	Verdadero	Falso	Bueno	**sí**
14	mayor	Verdadero	Falso	Excelente	**sí**
15	mayor	Falso	Falso	Normal	**no**

Inconvenientes

El criterio de la ganancia de la información tiende a favorecer a atributos con más posibles valores que el resto de atributos. Para entenderlo mejor, pensemos en el caso extremo en que todas las instancias de un juego de datos de entrenamiento tuvieran valores distintos en un atributo.

Ese atributo respecto de la clase objetivo tendría entropía 0 y en consecuencia, tendría una ganancia de la información máxima. Sin embargo, generaría un nodo para cada instancia del juego de datos, hecho que vaciaría de sentido la interpretación del resultado.

Como forma de superar este inconveniente, C4.5 también puede utilizar como criterio de selección de atributos, la ratio ganancia. Esta se define como:

$$GainRatio(Ex, a) = \frac{IG(E_x, a)}{-\sum_{j=1}^{s}\left(\frac{|(x \in E_x | valor(x, a) = j)|}{|E_x|} \cdot \log_2 \frac{|(x \in E_x | valor(x, a) = j)|}{|E_x|}\right)}$$

Donde "s" es el número de posibles valores del atributo *a*.

Observemos que en realidad estamos normalizando la ganancia de la información, para eliminar el efecto del número de posibles valores del atributo.

Discretización de atributos continuos

Para atributos con valores continuos, C4.5 lo que hace es discretizarlos en dos intervalos, ya que se considera suficiente. Para ello, analiza la ganancia del atributo para cada una de las posibles selecciones de intervalo y aquel intervalo que ofrece mayor ganancia es el seleccionado.

El sobreentrenamiento

Los algoritmos de árboles de decisión particionan los datos recursivamente hasta que se cumple alguna condición, como la minimización de la entropía o la clasificación de todas las instancias.

Este hecho hace que haya una tendencia a generar árboles con muchos nodos y nodos con muchas hojas. Este tipo de árboles "grandes" suelen tener:

- Mucha precisión si los utilizamos para predecir la clase de un juego de datos de prueba.

- Poca precisión si los utilizamos para clasificar las instancias de un juego de datos de prueba.

Este fenómeno recibe el nombre de sobreentrenamiento y se resuelve añadiendo procedimientos de poda *a posteriori* de la construcción del árbol. La idea para un procedimiento de postpoda es medir el error estimado de cada nodo, de modo que si el error estimado para un nodo es menor que el error estimado para sus subnodos, entonces los subnodos se eliminan.

1.5. Redes neuronales

Las redes neuronales han demostrado ser una buena aproximación a problemas donde el conocimiento de estos es impreciso o variante en el tiempo. Su capacidad de aprender convierte a las redes neuronales en algoritmos adaptativos y elaborados a la vez.

Inspiradas en el funcionamiento biológico de las neuronas, este tipo de algoritmos explotan el concepto de disponer de elementos simples de cómputo, las neuronas, interconectadas (sinapsis) de forma más o menos compleja.

La idea central se encuentra en el concepto de neurona. Veamos cuáles son sus características:

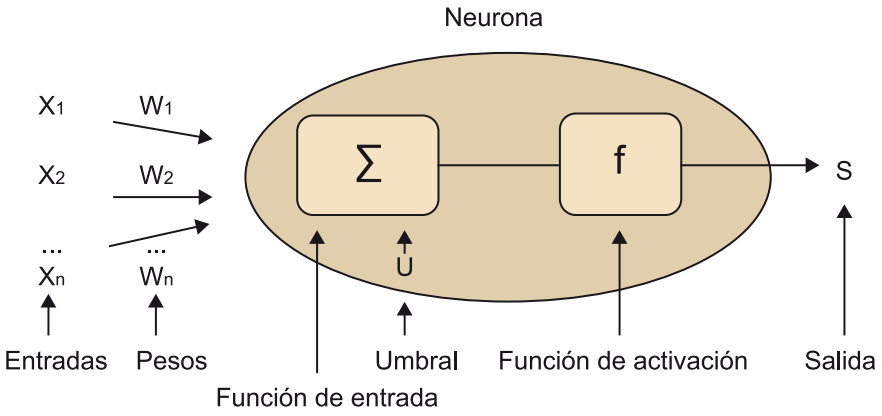
- Recibe un conjunto de señales de entrada procedentes de otras neuronas o de un sistema exterior.

- Estas señales de entrada se transmiten a través de unas conexiones que llevan asociados unos pesos.

- La combinación de señales de entrada y pesos es procesada para dar lugar a unas señales de salida.

En la figura 5 vemos, de forma esquemática, la estructura de una neurona.

Figura 5. Estructura de una neurona



La expresión matemática que describe una neurona es la siguiente:

$$S = f\left(x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + U\right) = f\left(\sum_{i=1}^{n} x_i w_i + U\right)$$

Donde $\vec{x}$ es el vector de entrada, procedente de otra neurona o del exterior.

$\vec{w}$ es el vector de pesos que iremos ajustando, en función del criterio del algoritmo.

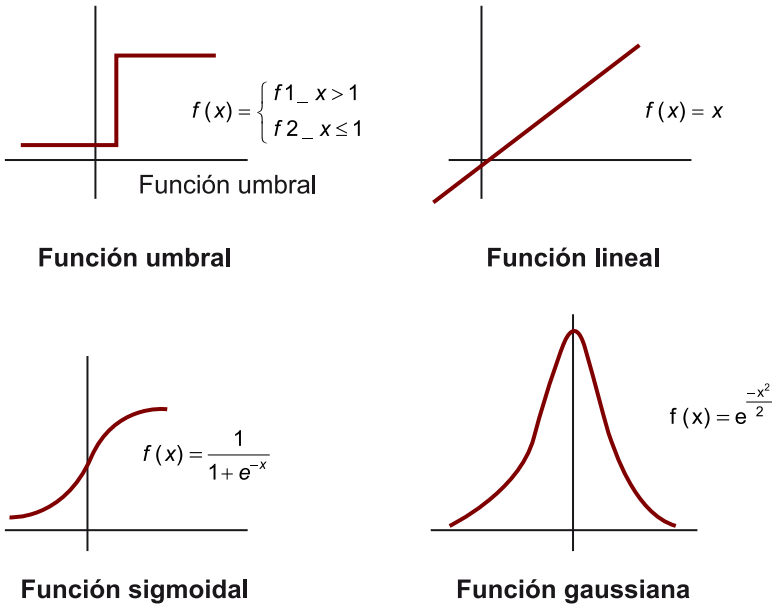
$\vec{x} \cdot \vec{w}$ es el producto escalar entre los vectores $\vec{x}$ y $\vec{w}$

U es el umbral, *bias* en inglés, un valor constante que no depende de ningún valor de entrada.

Tipos de funciones de activación

Las funciones de activación más comunes son las que se muestran en la figura 6.

Figura 6. Funciones de activación



El perceptrón simple

Se trata de una red neuronal especialmente sencilla y en consecuencia especialmente apropiada para entender el funcionamiento de un algoritmo de red neuronal.

El perceptrón simple es un clasificador binario y supervisado, puesto que requiere de datos test, que a partir de un vector de entrada $\vec{x}$, un vector de pesos $\vec{w}$ y un umbral U nos devuelve una salida en forma de resultado binario, 0 o 1.

$$f(x) = \begin{cases} 1\_si\_w \cdot x + b > 0 \\ 0\_en\_caso\_contrario \end{cases}$$

Esta red tan solo consta de una capa de neuronas, en contraposición al perceptrón multicapa.

El algoritmo consiste en modificar el vector de pesos w hasta conseguir un resultado f(x) próximo al esperado de acuerdo con los datos de test.

Redes en función del tipo de propagación

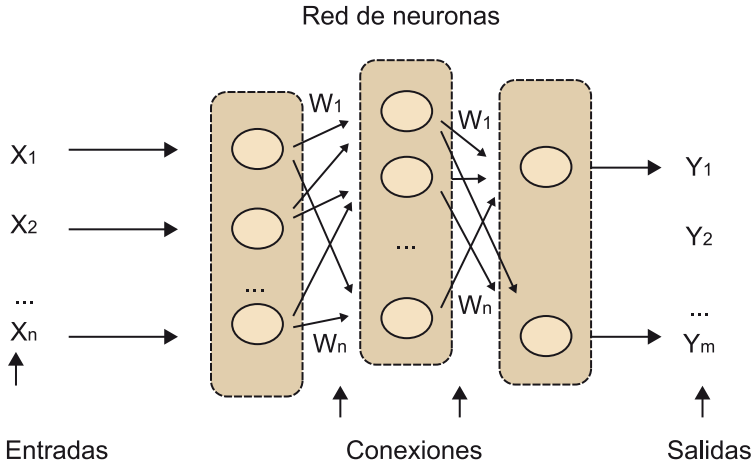
Una red de neuronas artificial puede pensarse como un grafo formado por neuronas organizadas por capas y relacionadas por conexiones que determinan la dirección del grafo. En función de esta dirección, podemos clasificar las redes:

- Redes de **propagación hacia delante** (*feedforward*). No tienen bucles. Algunos ejemplos pueden ser el perceptrón simple y el perceptrón multicapa.

- **Redes recurrentes** (*feedback*). Se producen bucles de retroalimentación. Ejemplos de este tipo de red son las redes competitivas, las redes Kohonen, las redes Hopfield y los modelos ART.

En la figura 7 podemos apreciar un esquema de red neuronal multicapa de propagación hacia delante.

Figura 7. Red de neuronas



La expresión matemática que describe una red de neuronas podría ser la siguiente:

Tomamos x como el vector de datos de entrada, entonces b será el vector de datos de salida de las capas intermedias, w el vector de pesos de la capa inicial y u será el vector umbral también de la capa inicial.

$$b_j = f\left(\sum_{i=1}^{n} x_i w_i + u_j\right)$$

Finalmente, y será el vector de salida de la red neuronal, formado este por el vector b como *input* de la capa interna, w' el vector de pesos de la capa interna y v el vector umbral también de la capa interna.

$$y_j = f\left(\sum_{i=1}^{m} b_i w'_i + v_j\right)$$

El proceso de aprendizaje

Para una red neuronal disponemos de pares $< x, d(x) >$ donde x es el vector de entrada y $d(x)$ es el vector de salida esperado.

El proceso de aprendizaje consiste en ajustar las matrices de pesos $W = (w_{ij})$ iniciales, e intermedios $W' = (w'_{ij})$ y de vectores de umbrales $U = (u_i)$ iniciales, e intermedios $V = (v_i)$, de tal forma que se minimice el error entre la salida de la red $y(\vec{x})$ y la salida deseada $d(\vec{x})$

$$E = \sum ||d(x) - y(x)||$$

Redes en función del paradigma de aprendizaje

Mediante el **aprendizaje supervisado**, a la red se le proporciona una respuesta correcta para cada patrón de entrada. Esto permite ajustar los pesos con el objetivo de aproximar la respuesta de la red a la respuesta correcta o esperada.

En el aprendizaje no supervisado se exploran patrones o correlaciones en los datos de entrada de la red, con el objetivo de organizarlos o clasificarlos.

Existe un tercer paradigma, el híbrido, en el que parte de los pesos se determinan mediante un proceso supervisado, mientras que el resto se determinan mediante un proceso no supervisado.

Redes en función de las reglas de aprendizaje

Aprendizaje supervisado

Reglas de corrección de errores. Establecen un procedimiento iterativo de modificación de pesos con el objetivo de la minimización de la diferencia entre los datos de salida obtenidos y los esperados. Algoritmos que utilizan estas reglas son el perceptrón, Adaline y Madaline.

Regla Boltzmann, basada en probabilidades y utilizada en el algoritmo de aprendizaje de Boltzmann.

La regla Hebbian se basa en la observación biológica de que si las neuronas de ambos lados de la sinapsis se activan de forma sincrónica y repetitiva, entonces la fuerza de la sinapsis se incrementa selectivamente. Se utilizada en el análisis lineal discriminante.

Reglas de aprendizaje competitivo. Utilizan el principio de que las unidades de salida compitan entre sí para su activación, de modo que solo aquella salida que produce menos error es la que finalmente se activará. Estas se utilizan en el algoritmo de cuantización del vector de aprendizaje y en el algoritmo ARTMap.

Aprendizaje no supervisado

Reglas de corrección de errores utilizada en el algoritmo de proyección de Sammon.

Regla Hebbian utilizada en el algoritmo de análisis de componente principal y en el de aprendizaje de memoria asociativa.

Reglas de aprendizaje competitivo, utilizadas en los algoritmos de cuantización de vector, SOM de Kohonen, ART1 y ART2.

Modelos híbridos

Reglas de corrección de errores y de aprendizaje competitivo utilizadas en el algoritmo de aprendizaje RBF.

Backpropagation. Redes de propagación hacia atrás.

Trataremos este algoritmo por separado, por ser posiblemente uno de los clasificadores más potentes que hayan existido hasta el momento. De hecho, en los últimos años se ha posicionado de forma permanente en las mejores posiciones en *benchmarking* de inteligencia artificial.

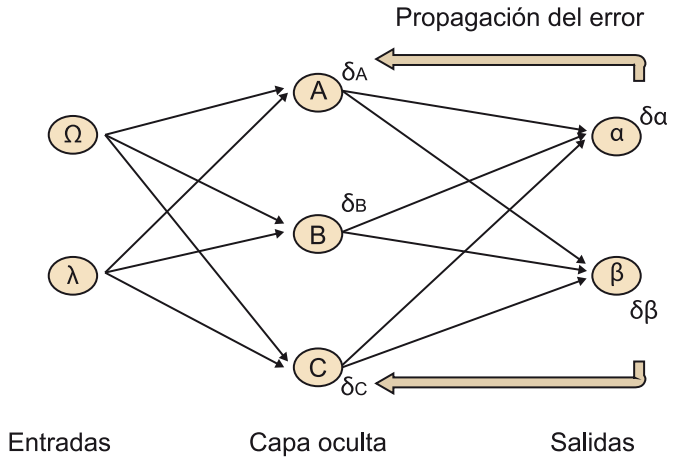
Retomando las clasificaciones de redes neuronales, Backpropagation sería un algoritmo de aprendizaje supervisado del tipo *feedforward*.

Este tipo de red aprende a partir de ejemplos o datos de entrenamiento, es decir, le damos a la red el resultado esperado, y esta recalcula los pesos de la neuronas, de tal modo que una vez ha finalizado el proceso de entrenamiento esta devolverá el resultado esperado para cada entrada concreta.

El nombre de propagación hacia atrás es debido al método de cálculo de los pesos de las neuronas interiores. Los pesos se calculan con el objetivo de minimizar el error producido por la red en su salida real respecto de su salida esperada.

Tal y como indica la figura 8, al no conocer las salidas esperadas para las neuronas internas, lo que se hace es propagar el error hacia atrás, desde la salida de la red hasta el origen de la misma.

Figura 8. Red de neuronas Backpropagation



En la figura 8 los datos de entrada se propagan de izquierda a derecha, mientras que el error, entendido como la diferencia entre el valor de salida y el valor esperado, se calculará de derecha a izquierda.

De hecho, delta A será función de delta alfa y delta beta.

1.6. SVM Support Vector Machines

> "No existe nada más práctico que una buena teoría."
>
> Vladimir Vapnik

Con esta frase Vladimir Vapnik daba a entender el porqué de los reconocidos resultados del método de las máquinas de soporte desarrollado en los años noventa fruto de sus trabajos sobre aprendizaje estadístico.

Las SVM son capaces de producir buenos modelos para resolver problemas de clasificación binaria, pero también para tareas de regresión, de multiclasificación y de agrupamiento. Estas propiedades han llevado a las SVM a ser considerados los mejores algoritmos para aplicaciones de clasificación de texto.

La gran aportación de Vapnik radica en que construye un método que tiene por objetivo producir predicciones en las que se puede tener mucha confianza, en lugar de lo que se ha hecho tradicionalmente, que consiste en construir hipótesis que cometan pocos errores.

La hipótesis tradicional se basa en lo que se conoce como minimización del riesgo empírico ERM, mientras que el enfoque de las SVM se basa en la minimización del riesgo estructural SRM, de modo que lo que se busca es construir modelos que estructuralmente tengan poco riesgo de cometer errores ante clasificaciones futuras.

En su origen, las SVM resuelven un problema de clasificación binaria en la que a partir de unos datos de entrenamiento de la forma (vector, clasificación binaria) se construye un hiperplano (recta en el plano) capaz de dividir los puntos en dos grupos.

Si las SVM solo son capaces de resolver problemas lineales de clasificación binaria, parece lógico preguntarse dónde radica su importancia en el mundo del *business analytics*. Para poder dar respuesta a esta cuestión, necesitamos antes introducir algunos conceptos.

Funciones kernel

Una función kernel es una función k: $X \times X \to \Re$, que asigna a cada objeto del espacio de entrada X, un valor real correspondiente al producto escalar de las imágenes de dichos objetos en un espacio F, al que llamamos espacio de características.

De modo que $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$, donde $\varphi : X \to F$

Técnicamente, para garantizar que la función kernel es realmente un producto escalar en el espacio F, exigiremos que sea simétrica y semi-definida positiva.

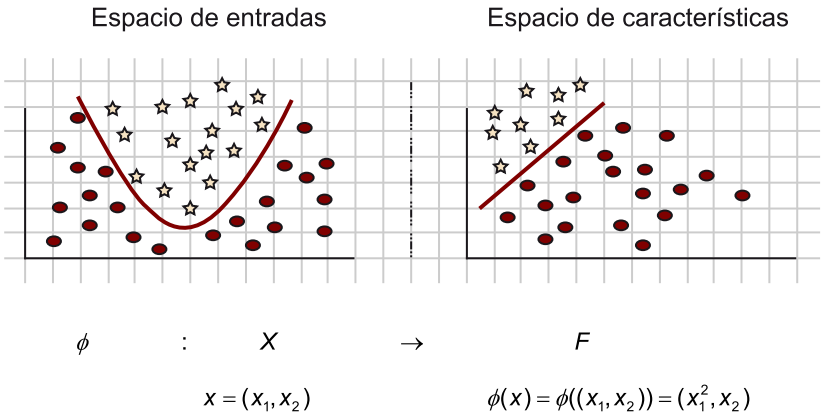
Función kernel

Diremos que nuestra función kernel es simétrica si cumple que k(x,y)=k(y,x) para todo x,y del espacio X.

Diremos que nuestra función kernel es semidefinida positiva si cumple que k(x,y)>=0 para todo x,y del espacio X.

En la siguiente figura vemos cómo en el espacio de entradas, la función que es capaz de separar puntos de estrellas es una función no lineal, sin embargo, en el espacio de características, la función que separa puntos de estrellas es lineal.

Figura 9. SVM Función kernel



La gran utilidad de las funciones kernel es que nos permite utilizar algoritmos lineales como SVM para abordar problemas no lineales. Además, esto es posible hacerlo sin necesidad de conocer explícitamente la función kernel.

Margen del hiperplano

Para entender bien el concepto de margen, procederemos a plantear formalmente el problema de clasificación binaria, y estudiaremos cómo lo afrontan estos dos algoritmos.

Partimos de una muestra de entrenamiento S formada por n elementos, de modo que $S = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$, donde cada x_i pertenece al espacio de entrada X y la clase $y_i \in \{+1, -1\}$ es la clasificación binaria que asociamos a cada objeto x_i. Si consideramos que $X \subseteq \Re$, la clasificación binaria en este dominio se puede realizar mediante una función lineal $f : X \to \Re$ que asigne valores positivos a los ejemplos de la clase +1 y valores negativos a los ejemplos de la clase -1.

La función signo sería $h(x) = sign(f(x)) = sign(\langle w, x\rangle + b)$ donde $\langle w, x\rangle$ es el producto escalar de los dos vectores.

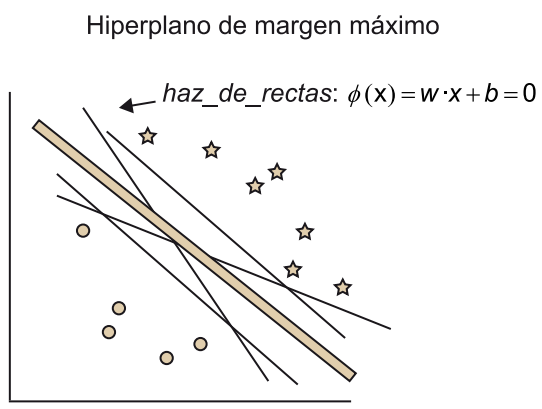
De modo que a nivel de notación podemos decir que $\langle w, x\rangle = w \cdot x$

Llegados a este punto, el problema se reduce a encontrar valores adecuados para el vector ***w*** y valores adecuados para el término independiente ***b*** de tal manera que consigan diferenciar los objetos de ambas clases.

Mediante la siguiente figura podemos observar que si realmente ambas clases son linealmente separables, entonces este problema tiene infinitas soluciones, es decir, existen infinitas rectas $\varphi(x) = w \cdot x + b = 0$ capaces de separar los objetos en dos grupos.

Variando el valor *w* obtenemos diferentes inclinaciones de la recta de separación, y variando el parámetro *b*, obtenemos diferentes desplazamientos de la recta se separación.

Figura 10. SVM margen



Parece lógico preguntarse con cuál de estas rectas obtenemos una mejor separación. Otros algoritmos no se preocupan en absoluto de esta cuestión y seleccionan la primera recta que clasifica correctamente todos los casos.

Todas las rectas de la figura clasifican correctamente todos los casos, sin embargo, gráficamente parece obvio que la recta con mayor grosor es la más óptima de todas. Recordemos que la recta de separación se construye en base a un juego de datos de entrenamiento y que se le exige que para futuros datos sea también capaz de clasificar correctamente.

Bajo estas premisas parece razonable pensar que la mejor recta será aquella que esté más alejada de los dos grupos de puntos, dicho de otra forma, será aquella recta que defina una frontera más ancha entre los dos grupos, ya que de este modo tendrá más "margen" para clasificar los futuros puntos.

Para una recta dada, definiremos su margen como la menor de las distancias entre:

La recta y el grupo de clases positivas, y la recta y el grupo de clases negativas.

Es posible demostrar que este problema no presenta extremos locales. Este hecho facilita el procedimiento de búsqueda de la recta que maximice el margen.

Extremos de una función

Entendemos por extremos de una función como los valores más grandes, máximos, o los valores más pequeños, mínimos, que toma la función en todo el dominio de la función o en una región concreta del dominio.

Si los extremos se toman con referencia a todo el dominio de la función, diremos que se trata de extremos globales o absolutos.

Si los extremos se toman con referencia a una región concreta del dominio de la función, diremos que se trata de extremos locales.

Clasificador lineal basado en producto escalar

Veremos a continuación el desarrollo teórico y un ejemplo práctico de un caso de clasificación lineal basado en un concepto equivalente a la distancia euclidiana en la notación vectorial. Se trata del producto escalar.

Esto nos ayudará mucho a comprender mejor cómo funciona la función signo en SVM.

Partiremos de un juego de datos de entrenamiento en el que disponemos de los consumos de 4 productos por parte de nuestros clientes. Nuestro objetivo será el de clasificar a nuestros clientes en dos grupos en función del consumo que han hecho a lo largo de un periodo determinado de los productos A, B, C y D.

Tabla 3. Datos de entrenamiento

Clase	Prod. A	Prod. B	Prod. C	Prod. D
+1	5,10	3,50	1,40	0,20
+1	4,90	3,00	1,40	0,20
-1	6,10	2,90	4,70	1,40
-1	5,60	2,90	3,60	1,30

Procederemos a calcular los centroides de cada grupo, centroide p para las instancias de clase +1 y centroide n para las instancias de clase -1.

Tabla 4. Centroides

Clase	Prod A	Prod B	Prod C	Prod D	
+1	5,00	3,25	1,40	0,20	Centroide p
-1	5,85	2,90	4,15	1,35	Centroide n

Centroide

Por centroide entendemos un punto medio, es decir, un punto que simétricamente estaría en el centro.

Con los centroides definidos, ya podemos construir una función signo del modo siguiente:

$$h(x)=\begin{cases}+1, si\_distancia(x,\ p)<distancia(x,\ n)\\ -1, en\_caso\_contrario\end{cases}$$

Si tomamos como medida de distancia la euclidiana, tenemos que:

$$h(x)=signo(||x-n||^2-||x-p||^2)$$

Si tomamos la notación vectorial de la distancia euclidiana, tenemos que:

$$||x-n||^2-||x-p||^2=\langle x,\ x\rangle+\langle n,\ n\rangle-2\langle x,\ n\rangle-\langle x,\ x\rangle-\langle p,\ p\rangle+2\langle x,\ p\rangle=$$

$$=2\langle x,\ p\rangle-2\langle x,\ n\rangle-\langle p,\ p\rangle+\langle n,\ n\rangle=2\langle x,\ p-n\rangle-\langle p,\ p\rangle+\langle n,\ n\rangle$$

Obtención del hiperplano de separación

Recordemos que, tal y como habíamos establecido anteriormente, el problema se reduce a encontrar los valores adecuados para el vector ***w*** y valores adecuados para el término independiente ***b*** de tal manera que consigan diferenciar los objetos de ambas clases.

Nuestro objetivo es, entonces, encontrar la relación entre el vector *w* y el parámetro *b*:

$$h(x)=signo(2\langle x,\ p-n\rangle-\langle p,\ p\rangle+\langle n,\ n\rangle)=$$

$$=signo\left(\langle x,\ p-n\rangle-\frac{1}{2}\cdot(\langle p,\ p\rangle+\langle n,\ n\rangle)\right)=signo(\langle w,\ x\rangle-b)$$

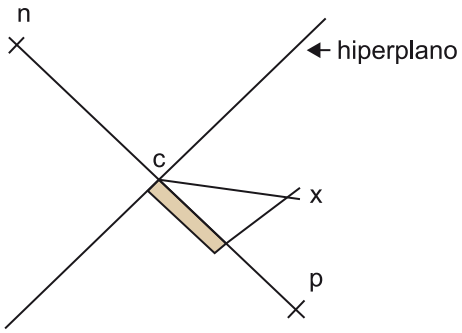
De modo que $w=p-n;\ b=\frac{1}{2}\cdot(\langle p,\ p\rangle+\langle n,\ n\rangle)$

Para ver una aplicación de la función signo, procedamos a calcular el signo de una instancia nueva, que llamaremos *x*.

Recordemos que disponemos de un hiperplano, dos centroides y un punto nuevo. Tomaremos el punto *c* como el punto equidistante entre los centroides *p* y *n*. Entonces la predicción para este nuevo punto *x* vendrá determinada por el signo del producto escalar de los vectores: $\langle cp, cx\rangle$.

En la siguiente figura podemos apreciar de una forma gráfica cómo la medida del producto escalar es una medida de la distancia del punto x al hiperplano (espacio sombreado):

Figura 11. SVM producto escalar



De modo que $h(x) = signo(\langle p - c, x - c\rangle) =$
$= signo(\langle p - c, x\rangle + \langle c, c - p\rangle) = signo(\langle w, x\rangle + b)$

Si tomamos los valores de *w* y de *b*, tenemos que:

$$w = p - c = p - \frac{p+n}{2} = \frac{p-n}{2}$$

$$b = \langle c, c - p\rangle = \left\langle \frac{p+n}{2}, \frac{p+n}{2} - p \right\rangle = \left\langle \frac{p+n}{2}, \frac{n-p}{2} \right\rangle =$$

$$= \frac{\langle p, n\rangle - \langle p, p\rangle + \langle n, n\rangle - \langle p, n\rangle}{4} = \frac{\langle n, n\rangle + \langle p, p\rangle}{4}$$

Volviendo al ejemplo práctico, vamos a clasificar la siguiente instancia *x*:

Tabla 5. Instancia a clasificar

Clase	Prod A	Prod B	Prod C	Prod D
?	4,90	3,10	1,50	0,10

$$w = \frac{p-n}{2} = (-0{,}425; 0{,}175; -1{,}375; -0{,}575)$$

$$b = \frac{\langle n, n\rangle - \langle p, p\rangle}{4} = 6{,}02875$$

$$signo(\langle w, x\rangle + b) =$$

$$signo(\langle(-0{,}425; 0{,}175; -1{,}375; -0{,}575), (4{,}90; 3{,}10; 1{,}59; 0{,}10)\rangle + 6{,}02875) =$$

$$signo(2{,}36875) = +1 = clase$$

Conclusiones

SVM es un sistema de aprendizaje lineal que encuentra el hiperplano de margen máximo, capaz de separar ejemplos asociados a clases positivas de ejemplos asociados a clases negativas.

El aprendizaje se formula como un problema de optimización cuadrática.

Para aquellos casos en los que la frontera entre clases positivas y clases negativas no es una función lineal (recta o hiperplano) deberemos recurrir a una función kernel que nos transforme un problema no lineal, de separación en el espacio X de datos de entrada, en un problema lineal, de separación en un espacio de características.

Gracias a la separación entre el algoritmo de aprendizaje y las funciones kernel, ambas se pueden tratar de forma separada, de modo que es recomendable experimentar entre diferentes funciones kernel sin necesidad de modificar el algoritmo de aprendizaje.

Limitaciones de las SVM

Solo funcionan para espacios numéricos, de forma que para atributos categóricos será necesario un proceso previo de conversión de valores categóricos a numéricos.

Por ejemplo, una forma de hacerlo sería crear un nuevo atributo para cada valor categórico, asignándole un valor 1 si el valor aparece y un 0 si el valor no aparece.

Solo está pensado para separar entre dos clases. Para casos de clasificación entre múltiples clases se pueden usar varias estrategias, como la comparación de uno contra el resto.

El hiperplano producido por una SVM puede ser complicado de interpretar por parte de los usuarios. Pensemos en espacios con muchas dimensiones.

Además, la comprensión empeora si se han utilizado funciones kernel. Por este motivo, las SVM suelen usarse en entornos en los que la comprensión humana no es un requisito.

1.7. *Clustering* aglomerativo y dendrogramas

Para estudiar con más detalle cómo funcionan estos algoritmos, recomendamos leer antes el anexo que trata el tema de distancia o similitud. Este nos permitirá comparar dos objetos o puntos, para así, posteriormente, poder agrupar objetos cercanos o similares.

1.7.1. *Clustering* y segmentación

Clustering y *segmentation* traducidos como 'agrupamiento' y 'segmentación' constituyen el ámbito de conocimiento correspondiente a las técnicas no supervisadas, ya que los datos que se proporcionan al sistema no tienen asociada ninguna etiqueta o información añadida por un revisor humano. En otras palabras, su objetivo es el de encontrar grupos similares en los juegos de datos.

Los algoritmos de agrupación jerárquica son de tipo aglomerativo cuando partiendo de una fragmentación completa de los datos, estos se van fusionando hasta conseguir una situación contraria, es decir, todos los datos se unen en un solo grupo. En este caso hablaremos de *clustering* o agrupamiento.

Asimismo, diremos que son de tipo divisivo cuando partiendo de un grupo que contiene todos los datos, se procede a una división progresiva hasta conseguir tener un grupo para cada dato. En este caso hablaremos de segmentación.

Conceptualmente ambos tipos, los aglomerativos y los divisivos, son equivalentes, sin embargo, los algoritmos aglomerativos son de más fácil construcción simplemente por el hecho de que mientras solo hay un modo de unir dos conjuntos, hay varios modos de separar un conjunto de más de dos elementos.

A través de los algoritmos aglomerativos, es posible construir recomendadores basados en modelo, de modo que para producir una recomendación, solo es necesario asignar una instancia nueva a uno de los grupos generados por el modelo.

Merece la pena observar que para llevar a cabo la operación de recomendación, tan solo será necesario almacenar la descripción de los grupos generados en el modelo, y no todo el juego de datos entero, suponiendo así un ahorro notable en recursos.

El principal inconveniente de este tipo de algoritmos es que no son capaces de determinar por sí mismos el número idóneo de grupos a generar, sino que es necesario fijarlos de antemano, o bien definir algún criterio de parada en el proceso de construcción jerárquica.

1.7.2. Dendrogramas

El dendrograma es un diagrama que muestra las agrupaciones sucesivas que genera un algoritmo jerárquico aglomerativo.

Sin duda, es una forma muy intuitiva de representar el proceso de construcción de los grupos, sin embargo, se les critica que no ofrecen información sobre la distancia entre los distintos objetos. Para dotar al dendrograma de la idea de distancia, se puede utilizar el tamaño de las flechas o se puede trabajar con tonalidades de un mismo color.

En la figura 12 vemos un ejemplo de dendrograma aglomerativo en el que inicialmente tenemos una colección de 5 algoritmos individuales.

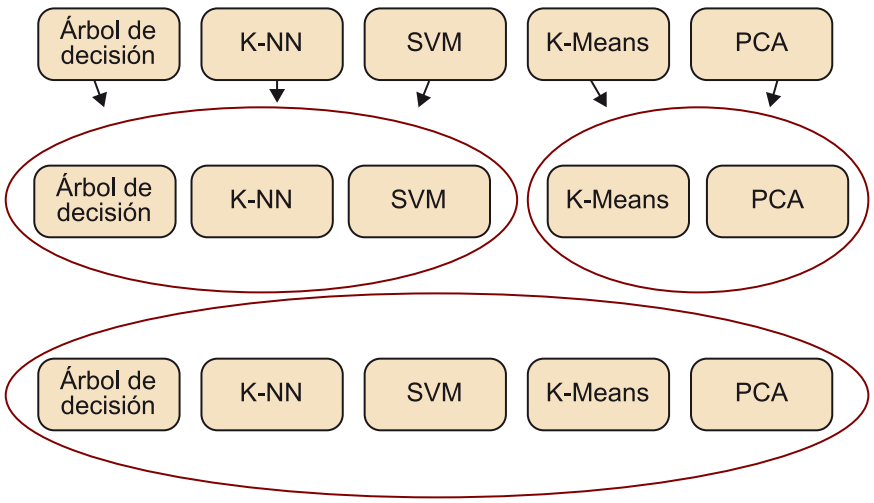
En un segundo paso hemos agrupado los algoritmos en dos categorías:

- Algoritmos supervisados: Árbol de decisión, K-Nearest Neighbor, Support Vector Machine.

- Algoritmos no supervisados: K-Means, Principal component analysis.

Finalmente, en un tercer paso se agrupan todos los algoritmos en una sola categoría que podríamos llamar “algoritmos habituales en Business Analytics”.

Figura 12. Dendrograma aglomerativo



Criterios de enlace

Los algoritmos jerárquicos aglomerativos, para construir grupos, necesitan de un concepto de distancia entre objetos y de un criterio de enlace para establecer la pertenencia a un grupo u otro. Algunos de los criterios más utilizados para medir la distancia entre dos grupos A y B son los siguientes:

1) Enlace simple o *single linkage*

Tomaremos como criterio la distancia mínima entre elementos de los grupos:

$$\min\{d(x, y) \mid x \in A, y \in B\}$$

Puede ser apropiado para encontrar grupos de forma no elíptica, sin embargo, es muy sensible al ruido en los datos y puede llegar a provocar el efecto cadena.

Este consiste en el hecho de que puede llegar a forzar la unión de dos grupos, que *a priori* deberían permanecer bien diferenciados, por el hecho de que estos compartan algún elemento muy próximo.

2) Enlace completo o *complete linkage*

Tomaremos como criterio la distancia máxima entre elementos de los grupos:

$$\max\{d(x, y) \mid x \in A, y \in B\}$$

No produce el efecto cadena, pero es sensible a los valores *outliers*, sin embargo, suele dar mejores resultados que el criterio simple.

3) Enlace medio o *average linkage*

Tomaremos como criterio la distancia media entre elementos de los grupos:

$$\frac{1}{|A|\,|B|}\sum_{x\in A}\sum_{y\in B} d(x, y)$$

Se trata de un criterio que trata de mitigar los inconvenientes de los dos anteriores sin acabar de resolverlos por completo.

4) Enlace centroide

La distancia entre dos grupos será la distancia entre sus dos centroides. Presenta la ventaja de que su coste computacional es muy inferior al de los criterios anteriores, de modo que está indicado para juegos de datos de gran volumen.

1.8. *Clustering* o clasificador

En este apartado estudiaremos un algoritmo de clasificación aglomerativo y una generalización del mismo que permite a su vez mejorar otros algoritmos de clasificación.

1.8.1. Algoritmo k-means

El algoritmo k-means o k-medias está considerado como un algoritmo de clasificación no supervisado. Este requiere que de antemano se fijen los *k* grupos que se quieren obtener.

Supongamos que disponemos de un juego de datos compuesto por *n* casos o instancias, por ejemplo, cada caso podría ser un cliente del que hemos seleccionado *m* atributos que lo caracterizan.

Llamaremos X a este juego de datos $X=\{x_1, x_2,...x_i,...x_n\}$ donde cada x_i podría ser un cliente con atributos $x_i=(x_{i1}, x_{i2},...x_{im})$, como pueden ser por ejemplo ventas, promociones, distancia al centro de distribución logística, etc.

Para clasificar nuestro juego de datos X mediante el algoritmo k-means seguiremos los siguientes 5 pasos:

1) De entre los n casos seleccionaremos k, que llamaremos semillas y denotaremos por c_j, $j=1,...,k$. Cada semilla c_j identificará su clúster C_j

2) Asignaremos el caso x_i al clúster C_j cuando la distancia entre el caso x_i y la semilla c_j sea la menor entre todas las semillas. Es decir $d(x_i, c_j)=\min\{d(x_i, c_t)\}, t=1,...,k$

3) Calcular la mejora que se produciría si asignáramos un caso a un clúster al que no pertenece actualmente. Se debería seleccionar un criterio apropiado, para medir esta mejora. Un ejemplo podría ser el de minimizar la distancia de las distintas instancias o casos a sus respectivos centros.

4) Hacer el cambio que proporciona una mayor mejora.

5) Repetir los pasos 3 y 4 hasta que ningún cambio sea capaz de proporcionar alguna mejora.

Como se puede intuir, el algoritmo k-means puede presentar numerosas versiones en función de la métrica de distancia que se utilice y en función del criterio que se seleccione para medir la mejora producida por la asignación de un caso a un clúster distinto del actual. Normalmente el criterio tratará de minimizar alguna función.

También es fácil intuir que el algoritmo no siempre alcanzará un óptimo global, puede darse el caso de que antes encuentre un óptimo local. Además, también es posible que el algoritmo no sea capaz de encontrar ningún óptimo,

bien porque simplemente el juego de datos no presente ninguna estructura de clústeres, bien porque no se haya escogido correctamente el número *k* de clúster a construir.

El algoritmo k-means

El algoritmo k-means contiene un paso de optimización. Se trata de la optimización de la mejora obtenida al encontrar nuevos centros. Como todo proceso de optimización, cuando encontramos un extremo debemos preguntarnos si se trata de un extremo local o absoluto.

Ejemplo k-means

Veamos un ejemplo sencillo en el que tomaremos para simplificar, como medida de distancia, el cuadrado de la distancia euclidiana, y como criterio de mejora, la minimización de la suma de distancias de cada caso a su semilla correspondiente.

Tomemos como nuestro juego de datos $X=\begin{pmatrix}1 & 1\\ 2 & 2\\ 3 & 0\\ 7 & 3\\ 8 & 4\end{pmatrix}\begin{matrix}\_caso\_1\\ \_caso\_2\\ \_caso\_3\\ \_caso\_4\\ \_caso\_5\end{matrix}$, aplicaremos el algoritmo k-means tomando k=2 y tomando como semillas iniciales los casos 1 y 3.

Siguiendo con nuestra notación, tenemos que $c_1=(1,1)$ y $c_2=(3,0)$ serán respectivamente los centros de los clústeres C_1 y C_2.

Partición 0

Para los casos 2, 4 y 5 decidiremos a cuál de los clústeres pertenecen:

$$\left.\begin{aligned} d^2(x_2, c_1)&=(2-1)^2+(2-1)^2=2\\ d^2(x_2, c_2)&=(2-3)^2+(2-0)^2=5\end{aligned}\right\}\text{ De modo que } x_2\in C_1$$

$$\left.\begin{aligned} d^2(x_4, c_1)&=(7-1)^2+(3-1)^2=40\\ d^2(x_4, c_2)&=(7-3)^2+(3-0)^2=25\end{aligned}\right\}\text{ De modo que } x_4\in C_2$$

$$\left.\begin{aligned} d^2(x_5, c_1)&=(8-1)^2+(4-1)^2=58\\ d^2(x_5, c_2)&=(8-3)^2+(4-0)^2=41\end{aligned}\right\}\text{ De modo que } x_5\in C_2$$

Resumiendo, tenemos que la partición P_0 está formada por 2 clústeres que contienen los siguientes casos : $C_1=\{x_1, x_2\}$, $C_2=\{x_3, x_4, x_5\}$

Partición 1

En primer lugar calcularemos las medias o centros de los 2 clústeres: $\bar{x}_1=(1{,}5; 1{,}5)$, $\bar{x}_2=(6; 2{,}3)$

Nuestro criterio para valorar si las siguientes particiones son mejores o no será el de minimizar la distancia de los casos a sus respectivos centros.

Calculemos entonces el valor del criterio S para la Partición P_0:

$$S(P_0) = (1-1{,}5\,;\,1-1{,}5)^2 + (2-1{,}5\,;\,2-1{,}5)^2 +$$

$$+(3-6\,;\,0-2{,}3)^2 + (7-6\,;\,3-2{,}3)^2 + (8-6\,;\,4-2{,}3)^2 = 23{,}7$$

Ahora deberemos cambiar cada caso de clúster siempre que el cambio suponga una mejora en el valor $S(P_0)$

Por ejemplo, si asignamos el caso 3 al clúster C_1 podemos ver cómo se produce una mejora significativa en el valor del criterio $S(P_1)$, donde P_1 sería la nueva partición formada por el clúster $C_1 = \{x_1, x_2, x_3\}$ y $C_2 = \{x_4, x_5\}$ y donde $\bar{x}_1 = (2,1)$, $\bar{x}_2 = (7{,}5{,}3{,}5)$ serían las nuevas medias o centros:

$$S(P_1) = (1-2\,;\,1-1)^2 + (2-2\,;\,2-1)^2 + (3-2\,;\,0-1)^2$$

$$+(7-7{,}5\,;\,3-3{,}5)^2 + (8-7{,}5\,;\,4-3{,}5)^2 = 5$$

Como este cambio mejora el valor del criterio *S*, lo daríamos por bueno.

Después de haber desarrollado numéricamente este ejemplo tan simple, es fácil entender que uno de los problemas que presenta k-means es el gran número de cálculos que requiere.

Para mitigar este inconveniente se han desarrollado alternativas. Una de las más ingeniosas y eficientes es el algoritmo Canopy Clustering, válido para mejorar en general los algoritmos de clasificación.

1.8.2. *Canopy clustering* y *map reducing*

La idea brillante que subyace a esta técnica es que podemos reducir drásticamente el número de cálculos que requieren los algoritmos aglomerativos como k-means, introduciendo un proceso previo de generación de grupos superpuestos (*canopies*) a partir de una métrica más sencilla de calcular (*cheapest metric*).

De forma que solo calcularemos distancias con la métrica inicial, más estricta y pesada en cálculos, para los puntos que pertenecen al mismo *canopy*.

Podríamos resumirlo diciendo que previamente, mediante una métrica simple, decidimos qué puntos están definitivamente lejos y en consecuencia, para estos puntos alejados ya no valdrá la pena malgastar más cálculos con una métrica más exigente.

En realidad el método *canopy clustering* divide el proceso de *clustering* en dos etapas:

- En una primera etapa, usaremos una métrica sencilla en cálculos, con el objetivo de generar los *canopies* o subgrupos superpuestos de puntos.

- Además, lo haremos de modo que cada punto pueda pertenecer a más de un *canopy* y a su vez todos los puntos tengan que pertenecer al menos a un *canopy*.

- En una segunda etapa, utilizaremos un método de segmentación tradicional, como por ejemplo el método k-means, pero lo haremos con la siguiente restricción:
  "No calcularemos la distancia entre puntos que no pertenecen al mismo *canopy*".

Para facilitar la comprensión de lo que estamos haciendo en realidad, vamos a situarnos en los dos casos extremos:

1) Supongamos, como consecuencia de la primera etapa, que nuestro universo de puntos cae por completo en un solo *canopy*. Entonces el método de segmentación por *canopies* sería exactamente igual al del método de segmentación tradicional seleccionado, es decir, k-means en nuestro ejemplo.

2) Supongamos que como resultado de la primera etapa, generamos *canopies* relativamente pequeños y con muy poca superposición. En este caso, al aplicar la técnica tradicional solo dentro de cada *canopy*, habremos ahorrado un gran número de cálculos.

Veamos de una forma gráfica cómo funciona el algoritmo:

Figura 13. *Canopy clustering* (1)

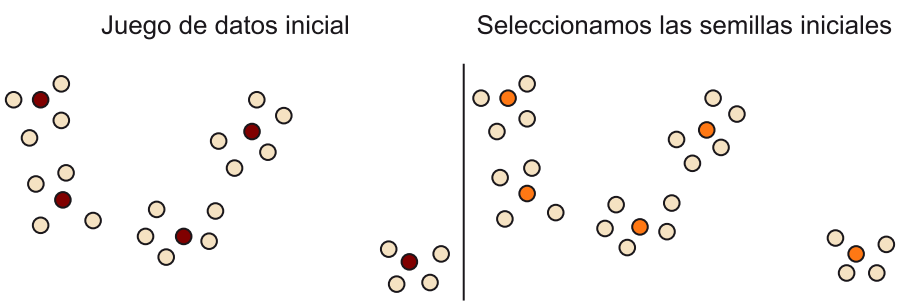


Figura 14. *Canopy clustering* (2)

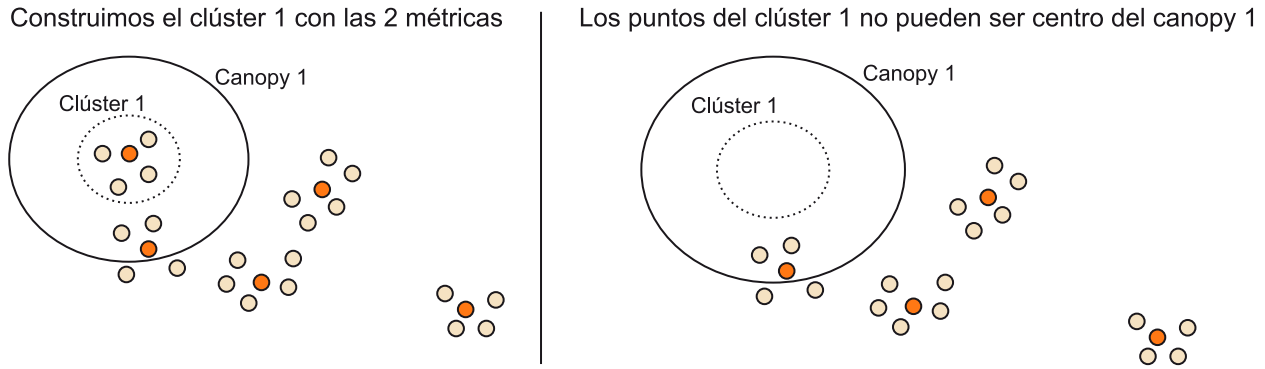


Figura 15. *Canopy clustering* (3)

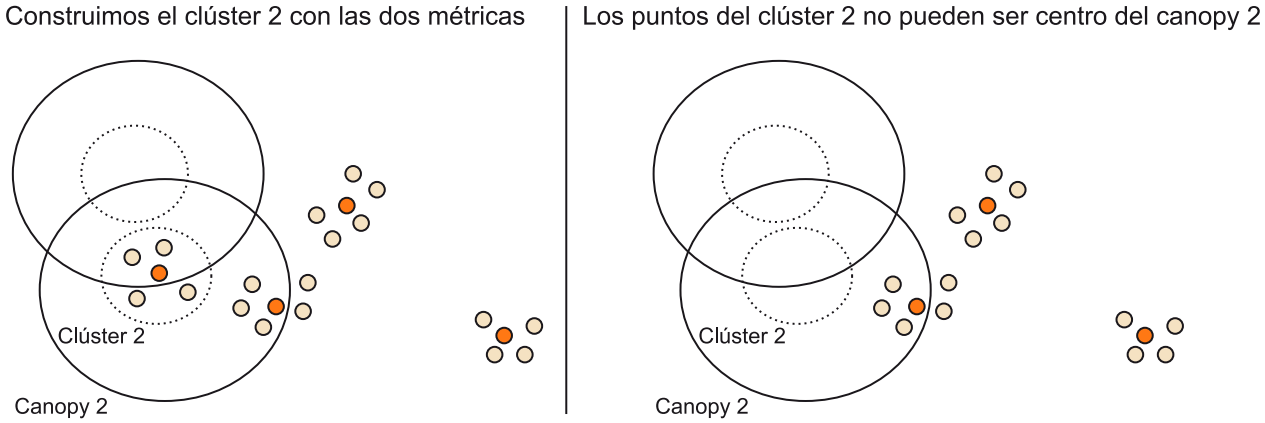


Figura 16. *Canopy clustering* (4)

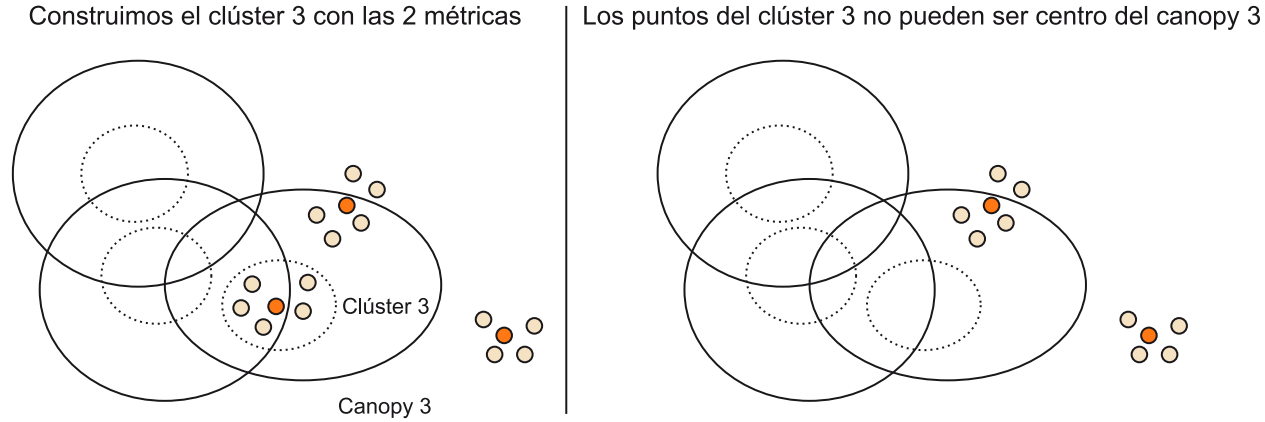


Figura 17. *Canopy clustering* (5)

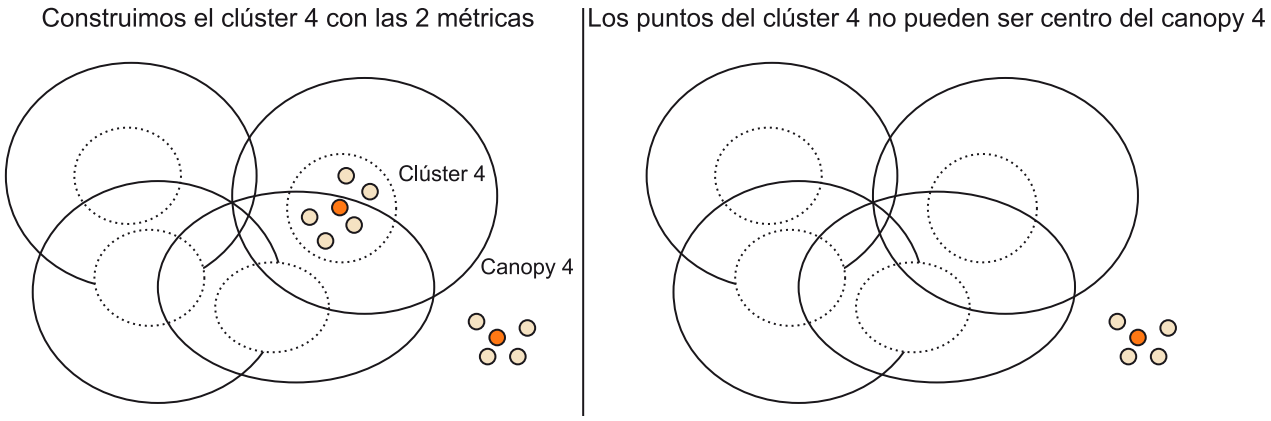
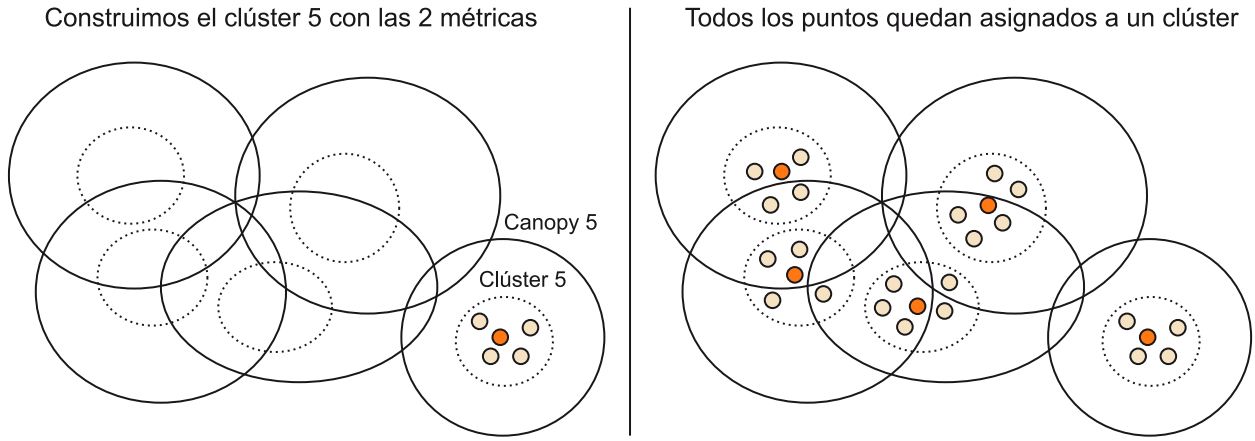


Figura 18. *Canopy clustering* (6)



Una vez visto de forma gráfica cómo evoluciona el algoritmo, quizá se entenderá con más facilidad su enunciado.

Canopy clustering

Mientras haya puntos no marcados {

- Seleccionar un punto no fuertemente marcado y nombrarlo centro de *canopy*.
- Marcar todos los puntos contenidos en un cierto umbral, como puntos del mismo *canopy*
- Marcar fuertemente todos los puntos contenidos en un umbral más fuerte.

}

Ventajas e inconvenientes

El *clustering* jerárquico presenta varias ventajas respecto de otros algoritmos de *clustering* particionales como k-means. Es capaz de trabajar con cualquier definición de distancia o similitud. Además, así como k-means solo produce los clústeres al final del proceso, la jerarquización de los clústeres permite al analista explorar los clústeres generados en cada nivel.

Dependiendo de la tipología del problema a resolver, se ha probado que los algoritmos jerárquicos pueden dar mejores resultados que k-means.

El principal problema de los algoritmos jerárquicos es su coste computacional y el espacio en memoria que se requiere para llevarlos a cabo. Comparado con k-means, suelen ser muy ineficientes y desaconsejables para grandes volúmenes de datos.

Existen varias técnicas para superar estos problemas. Un ejemplo podría ser el de utilizar un método de *clustering* más eficiente, con el objetivo de conseguir muchos pequeños clústeres, y posteriormente utilizar solo los centroides de estos pequeños clústeres para aplicar un algoritmo jerárquico.

¿Qué algoritmo utilizar?

Es una constante a lo largo del estudio de los distintos algoritmos del ámbito *clustering* plantearse la siguiente pregunta.

Dado un juego de datos. ¿Cuál es el algoritmo de *clustering* más adecuado?

Por supuesto no hay una respuesta ni mágica ni contundente a esta cuestión, pero sí que hay una relación de buenas prácticas que deberán ayudarnos a tomar la opción más acertada.

Utilizar varios algoritmos, varias definiciones de distancia y varios parámetros como el *k* del k-means. Todo ello para posteriormente analizar cuidadosamente los resultados obtenidos, compararlos y sacar conclusiones basadas en el conocimiento del problema, de los datos y del funcionamiento de los propios algoritmos.

Es importantísimo que el analista sea consciente de las limitaciones de los algoritmos que está utilizando y a la vez comprenda perfectamente el dominio del problema al que se está enfrentando.

1.9. PCA Análisis de componentes principales

En inglés *principal component analysis*. Se trata de una técnica estadística cuyo objetivo es la reducción de la dimensión o número de atributos. Se basa en el supuesto de que la mayor parte de la información de un juego de datos puede ser explicada por un número menor de variables o atributos.

Un análisis de componentes principales tiene sentido si existen altas correlaciones entre las variables, ya que esto es indicativo de que existe información redundante y, por lo tanto, un número menor de variables serán capaces de explicar gran parte de la variabilidad total.

La selección de factores o variables se realiza de manera que el primero recoja la mayor proporción posible de la variabilidad original, el segundo factor debe recoger la máxima variabilidad posible no recogida por el primero, y así sucesivamente.

Un aspecto importante en PCA es la interpretación de los resultados, ya que esta no es directa. Dependerá de la relación entre los atributos iniciales y los factores principales obtenidos, así como del signo y magnitud de sus correlaciones, que deberán ser interpretadas con respecto al significado del juego de datos. Lo veremos mediante un ejemplo.

Fundamentos matemáticos

Para poder entender el funcionamiento del PCA se requieren algunas herramientas básicas que nos proporcionan la estadística y el álgebra. Se recomienda leer los anexos de estadística y álgebra donde se introducen definiciones de conceptos básicos y no complejos, pero fundamentales para entender PCA.

1.9.1. Método del análisis de componentes principales

Desarrollaremos el método sobre la base de un ejemplo para facilitar su comprensión.

En la tabla 6 tenemos para los productos X e Y las ventas anuales en 10 zonas geográficas de nuestro mercado.

Tabla 6. Zonas de venta por producto

	Prod X	Prod Y	X-media	Y-media	cov(X,Y)
Zonas de venta	2,5	2,4	0,69	0,49	0,3381
	0,5	0,7	-1,31	-1,21	1,5851
	2,2	2,9	0,39	0,99	0,3861
	1,9	2,2	0,09	0,29	0,0261
	3,1	3	1,29	1,09	1,4061
	2,3	2,7	0,49	0,79	0,3871
	2	1,6	0,19	-0,31	-0,0589
	1	1,1	-0,81	-0,81	0,6561
	1,5	1,6	-0,31	-0,31	0,0961
	1,1	0,9	-0,71	-1,01	0,7171
Media	1,81	1,91		cov(X,Y) =	0,5539

Primer paso: Sustraer la media

Tal y como apreciamos en la tabla 6, nuestro primer paso será sustraer la media de cada atributo y calcular la covarianza entre ambos atributos.

Una covarianza positiva ya nos indica que en la mayoría de las zonas de venta ambos productos se encuentran conjuntamente por encima o por debajo de sus respectivas medias.

Segundo paso: Obtener la matriz de covarianza

De la tabla 4 es fácil obtener la matriz de covarianza de nuestro juego de datos.

$$\mathrm{cov}(X, Y) = \begin{pmatrix} 0{,}5549 & 0{,}5539 \\ 0{,}5539 & 0{,}6449 \end{pmatrix}$$

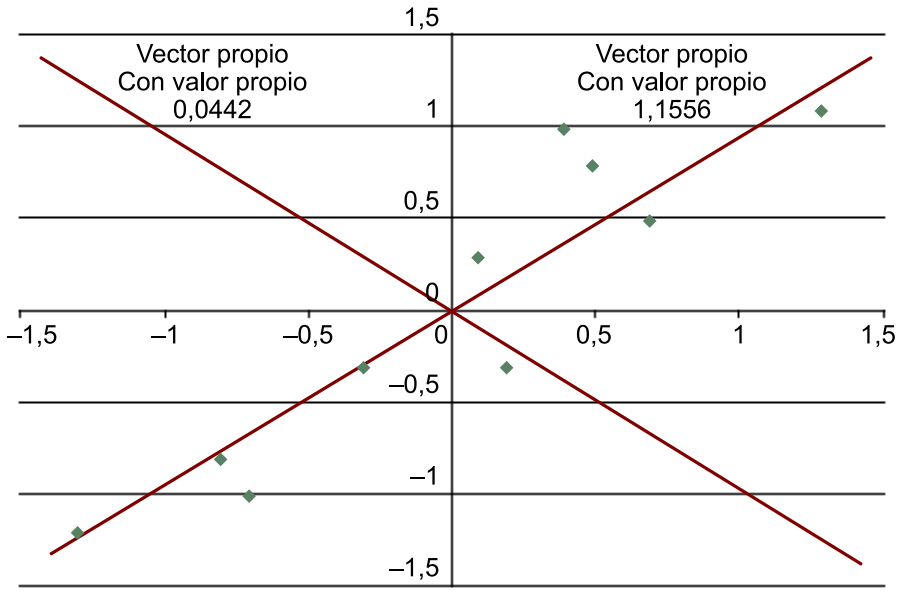
Tercer paso: Vectores y valores propios

Los vectores y los valores propios de la matriz de covarianza son los siguientes:

$$\lambda_1 \cdot v_1 = 1{,}1556 \cdot \begin{pmatrix} -0{,}6779 \\ -0{,}7352 \end{pmatrix};\ \lambda_2 \cdot v_2 = 0{,}0442 \cdot \begin{pmatrix} -0{,}7352 \\ 0{,}6779 \end{pmatrix}$$

Observemos que hemos seleccionado los dos vectores propios de longitud 1.

Figura 19. Análisis de componentes principales



Cuarto paso: Proyección de los datos sobre los vectores principales

El siguiente paso será proyectar los datos originales corregidos con la media, sobre nuestro nuevo eje de coordenadas, formado por nuestros vectores propios o componentes principales.

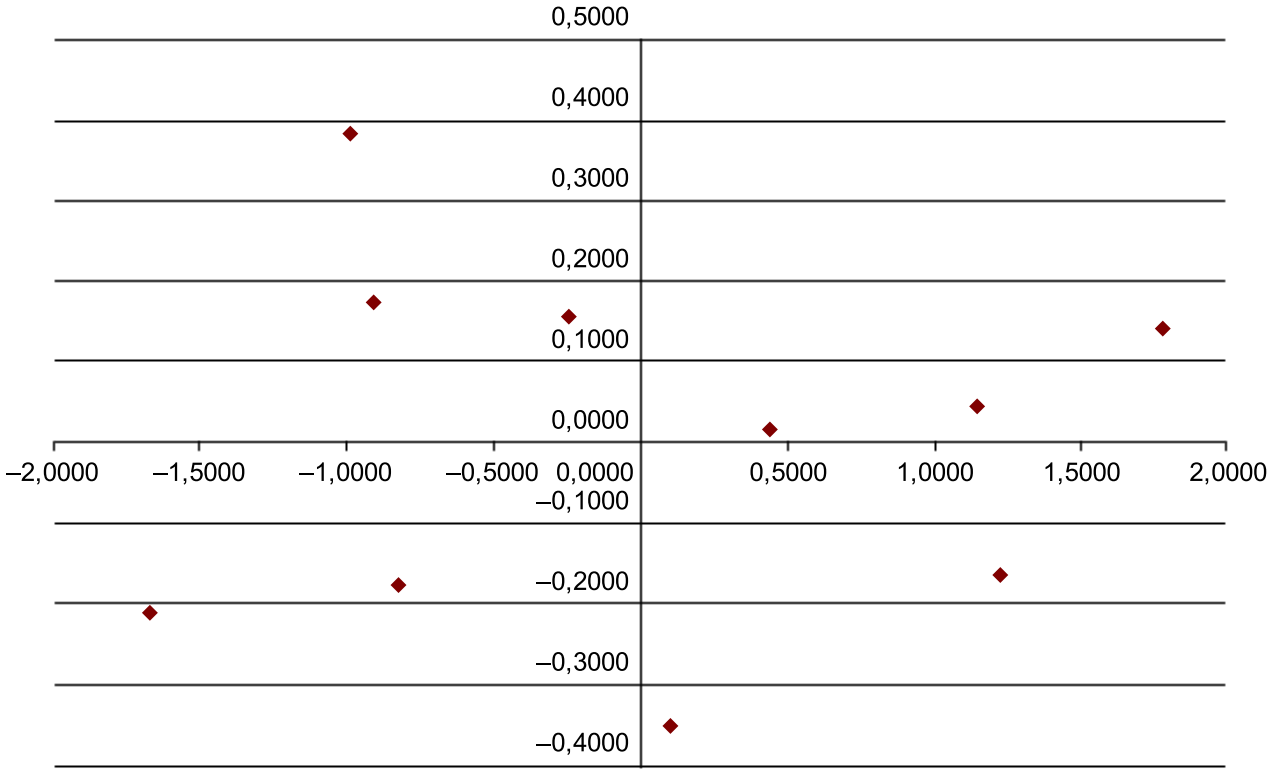
Tabla 7. Proyección de las ventas de los productos X e Y por zonas de ventas

Proyección sobre v1	Proyección sobre v2
-0,8280	-0,1751
1,7776	0,1429
-0,9922	0,3844
-0,2742	0,1304

Proyección sobre v1	Proyección sobre v2
-1,6758	-0,2095
-0,9129	0,1753
0,0991	-0,3498
1,1446	0,0464
0,4380	0,0178
1,2238	-0,1327

La figura 20 muestra la proyección de los datos originales corregidos por la media, sobre los componentes principales v1 y v2.

Figura 20. Proyección sobre los componentes principales



Quinto paso: Interpretación de los resultados

El vector propio v_1 debido al peso de su valor propio λ_1 acumula el 96,32% de la variabilidad del juego de datos, mientras que el vector propio v_2 solo explica el 3,68% de la variabilidad total.

Nuestro ejemplo solo tiene 2 dimensiones y no tendría demasiado sentido descartar una de las dos, pero en juegos de datos de más de 3 atributos, descartar atributos hasta quedarnos solo con 2 o 3 podría facilitar la tarea de representar gráficamente el juego de datos, respetando al máximo la variabilidad de los datos originales.

Con la proyección de los datos sobre los nuevos ejes, en realidad lo que estamos haciendo es representar los datos en función de los patrones que existen entre los atributos o variables originales.

Con la proyección tenemos representados los puntos (las ventas por zonas de cada uno de los dos productos) en función de la contribución de las ventas de cada producto al total de ventas. Hemos pasado de ver datos absolutos a ver datos relativos.

1.10. Asociaciones

La búsqueda de asociaciones constituye una actividad fundamental en la minería de datos, y consiste en encontrar relaciones entre tuplas o combinaciones de valores en un juego de datos. Los algoritmos que resuelven este problema suelen ser muy eficientes en términos de coste computacional. Además, la lógica que utilizan es relativamente simple y la información que proporcionan suele ser de gran utilidad. Esto convierte las asociaciones en un recurso muy frecuente.

Un ejemplo clásico de aplicación de asociaciones sería en una cesta de la compra donde tratamos de establecer relaciones como…

…"Si compra el producto A, probablemente también comprará el producto B"

Pero, ¿qué significa este "probablemente"?...

1.10.1. Especificación de transacciones, esperanza y soporte

Estudiemos el ejemplo que nos propone Bing Liu en su libro *Web Data Mining*, en el que se nos presenta un grupo de transacciones T, que se corresponden con siete cestas de la compra, realizadas en una tienda.

Tabla 8. Cesta de la compra de 7 clientes

t1	Bistec, pollo, leche
t2	Bistec, queso
t3	Queso, botas
t4	Bistec, pollo, queso
t5	Bistec, pollo, ropa, queso, leche
t6	Pollo, ropa, leche
t7	Pollo, leche, ropa

Utilizaremos este ejemplo a lo largo del desarrollo de todo el capítulo de asociaciones. Nos servirá para acompañar todas las definiciones y planteamientos.

El problema que resuelven las reglas de asociaciones podríamos formularlo del siguiente modo:

Sea $I = \{i_1, i_2, \ldots, i_k\}$ un juego de k datos, en nuestro caso

I = (bistec, queso, pollo, leche, ropa, botas).

Sea $T = \{t_1, t_2, \ldots, t_n\}$ un juego de n transacciones, donde por ejemplo

t_1 = (bistec, pollo, leche)

Una regla de asociación es una implicación de la siguiente forma

$X \rightarrow Y$, donde $X \subset I$, $Y \subset I$, y $X \cap Y = 0$

Por ejemplo, la regla (bistec, pollo) → leche cumple con la definición.

Para medir el grado de precisión de una regla disponemos de dos métricas, el soporte y la esperanza.

Soporte (*support*)

El soporte de una regla $t_i : X \rightarrow Y$, es el porcentaje de transacciones en T que contienen $X \cup Y$ y puede ser visto como una medida de la probabilidad de que se dé $X \cup Y$ en nuestro espacio de transacciones T.

$$\text{Soporte} = \frac{|X \cup Y|}{n}$$

El soporte nos da una idea del grado de cobertura de una regla. En definitiva, es el porcentaje de transacciones que aglutinan o bien X o bien Y.

Para valores demasiado bajos, nos indica que la regla no va a ser de gran utilidad por su escasa incidencia.

Esperanza (*confidence*)

La esperanza de una regla $t_i : X \rightarrow Y$, es el porcentaje de las transacciones en T que contienen X y que además también contienen Y. Puede ser visto como una medida de la probabilidad de que se dé $X \cap Y$ en nuestro espacio de transacciones T.

$$\text{Esperanza} = \frac{|X \cap Y|}{|X|}$$

Nos indica el grado de previsibilidad de la regla, es decir, de entre las transacciones que contienen *X*, es el porcentaje de las que además contienen *Y*.

Valores demasiado bajos nos indicarán que la regla será poco fiable.

Una vez introducido el paradigma de los algoritmos de asociaciones, y establecidas las definiciones básicas, pasamos a estudiar con detalle uno de los algoritmos de asociaciones más utilizado. El algoritmo Apriori.

1.10.2. Algoritmo Apriori

El algoritmo Apriori establece valores mínimos para las medidas de soporte y esperanza de una regla. Pongamos en nuestro ejemplo un soporte mínimo del 30% y una esperanza mínima del 80%.

En base a estos dos mínimos, se ejecutan dos pasos:

- La generación de frecuencias: en la que seleccionaremos aquellas combinaciones de productos de la compra que tengan un soporte por encima del mínimo establecido.

- Generación de reglas: en el que sobre las combinaciones obtenidas en el paso anterior, se generarán reglas y se seleccionarán solo aquellas que tengan esperanza por encima de la mínima establecida.

Generación de frecuencias de tuplas

La generación de frecuencias de tuplas consiste en un proceso secuencial en el que para cada tupla, es decir, para cada valor *k*, ejecutaremos tres funciones, una función de generación de candidatos, una función de poda y una función de selección de candidatos.

Función generación de candidatos

Nuestro juego de datos *I* tiene *k* productos, generaremos combinaciones de productos con *k* elementos para j = 1,2,… *k* llamémosles k-tuplas, y calcularemos la frecuencia de aparición en el juego de transacciones *T* para cada una de ellas.

Función de poda

La función anterior nos habrá generado una lista de candidatos. Sobre esta lista se aplicará un proceso de "poda" o eliminación, consistente en exigir que cualquier subconjunto de (k-1) elementos del candidato de *k* elementos, deba haber superado el soporte mínimo establecido.

Función selección de candidatos

Finalmente, de entre aquellos candidatos que hayan superado la poda anterior, consideraremos válidos solo aquellos con un soporte superior al mínimo establecido.

Veámoslo en nuestro ejemplo. Recordemos que nuestro juego de datos de productos *I* contiene k = 6 productos distintos:

1) Aplicaremos la función de generación de candidatos para k=1 y obtenemos la tabla de frecuencias de la 1-tupla F_1

Tabla 9. Frecuencias para 1-tupla

1-tupla	Frecuencia	
Queso	5	
Bistec	4	
Pollo	4	
Leche	4	
Ropa	3	
Botas	1	descartada

En este caso k = 1, no tiene sentido aplicar la función de poda para k-1 = 0.

La función de selección de candidatos nos descarta el producto “Botas” por tener un soporte de 1/7 < 30%

2) Aplicaremos la función de generación de candidatos para k = 2 y obtenemos la tabla de frecuencias de la 2-tupla F_2

Tabla 10. Frecuencias para 2-tupla

2-tupla	Frecuencia	
Bistec, queso	3	
Bistec, pollo	3	
Bistec, ropa	1	descartada
Bistec, leche	2	descartada
Queso, pollo	2	descartada
Queso, ropa	1	descartada
Queso, leche	1	descartada
Pollo, ropa	3	
Pollo, leche	4	

2-tupla	Frecuencia	
Ropa, leche	3	

En este caso la función de poda se aplica en subconjuntos de k-1 = 2-1 = 1 productos, es decir, a los productos individuales. Observamos que todos los productos individuales presentes en la 2-tupla tienen un soporte superior al 30%, de modo que todas las 2-tupla superan la poda.

La función de selección de candidatos nos descarta todos los pares con soporte 1/7 o 2/7 < 30%.

3) Aplicaremos la función de generación de candidatos para k = 3 y obtenemos la tabla de frecuencias de la 3-tupla F_3

Tabla 11. Frecuencias para 3-tupla

3-tupla	Frecuencia
Bistec, queso, pollo	Eliminado en poda
Bistec, queso, ropa	Eliminado en poda
Bistec, queso, leche	Eliminado en poda
Bistec, pollo, ropa	Eliminado en poda
Bistec, pollo, leche	Eliminado en poda
Bistec, ropa, leche	Eliminado en poda
Queso, pollo, ropa	Eliminado en poda
Queso, pollo, leche	Eliminado en poda
Queso, ropa, leche	Eliminado en poda
Pollo, ropa. Leche	3

En este caso la función de poda se aplica en subconjuntos de k-1 = 3-1 = 2 productos, es decir, a los pares de productos. Observamos, por ejemplo, el caso (bistec, queso, pollo) de donde obtenemos los subconjuntos:

- (Bistec, queso) con un soporte 3/7 > 30%.
- (Bistec, pollo) con un soporte 3/7 > 30%.
- (Queso, pollo) con un soporte 2/7 < 30% que provoca la eliminación, por poda, de todo el subconjunto de 3 productos.

Finalmente, la función de selección de candidatos acepta al único candidato que supera la poda, por tener un soporte 3/7 > 30%.

Uno de los problemas de los algoritmos de reglas de asociación es que generan un gran número de reglas y combinaciones de productos, hecho que dificulta su posterior estudio. El problema de identificar las reglas más interesantes supone un ámbito de estudio específico, que ha aportado algoritmos como Frequent-Pattern-Growth, que queda fuera del alcance de este documento.

Generación de reglas

En ocasiones solo el hecho de disponer de las k-tuplas con un nivel de soporte aceptable puede ya ser suficiente para cubrir las necesidades de lo que se busca.

Para dar un paso más, generaremos las reglas de asociaciones del siguiente modo.

Para cada juego de datos de frecuencias "f" consideraremos todos sus subconjuntos no vacíos "s" y construiremos la siguiente función:

$$(f - s) \rightarrow s, \text{ si esperanza } = \frac{|f|}{|f - s|} \geq \text{ esperanza mínima}$$

Apliquémoslo a nuestro ejemplo:

F_1 = { (pollo:5, bistec:4, queso:4, leche:4, ropa:3) }

F_2 = { (bistec, queso):3, (bistec, pollo):3, (pollo, ropa):3,

(pollo, leche):4, (ropa, leche):3 }

F_3 = { (pollo, ropa, leche):3 }

A título de ejemplo desarrollaremos reglas solo para el caso F_3:

1) Reglas con 1 solo elemento consecuencia:

Se generan a partir de todas las combinaciones posibles de F_3 y solo se seleccionan las que tienen esperanza > 80%.

Regla 1: pollo, ropa leche con esperanza 3/3 > 80%

Regla 2: pollo, leche ropa con esperanza 3/4 < 80% se descarta

Regla 3: ropa, leche pollo con esperanza 3/3 > 80%

2) Reglas con 2 elementos consecuencia:

Fruto del paso anterior, disponemos de un conjunto de productos consecuencia, que llamamos H_1 = {leche, pollo} por ser los únicos con esperanza > 80%.

Con H_1 generaremos la regla 4, a partir de la función

$(F_3 - H_1) \rightarrow H_1$

Regla 4: ropa leche, pollo , con esperanza 3/3 = $\frac{|(Pollo, Ropa, Leche)|}{|(Ropa)|} > 80\%$

De modo que nuestro algoritmo Apriori ha generado para nuestro ejemplo 3 reglas con esperanza superior al 80%. Las reglas 1, 3 y 4.

1.10.3. Algoritmo MS-Apriori

Minimum Support Apriori es una generalización del algoritmo Apriori y consiste en posibilitar que cada producto individual tenga su propio nivel de soporte mínimo MIS (*minimum item support*).

De este modo se define soporte de una regla como el mínimo soporte de entre los productos que la componen.

Tomando el ejemplo anterior, tenemos que el soporte para el par (bistec, queso) = 3/7 > 30%, su hubiéramos establecido para el bistec un soporte mínimo del 60% y para el queso del 70%, a esta regla se le exigiría un soporte mínimo del 60%.

Esta extensión del algoritmo Apriori tiene la ventaja de permitir al analista descartar aquellos productos en los que no está interesado, asignándoles soportes mínimos elevados.

1.11. Técnicas estadísticas de regresión

En ocasiones puede resultar interesante poder predecir un atributo que ocurrirá en el futuro o simplemente conocer el valor de un atributo que es difícil de medir. Bajo estas circunstancias, puede ser práctico utilizar otros atributos conocidos para modelar a partir de ellos el comportamiento del atributo objetivo.

Este modelo es lo que conocemos por regresión. Para entender bien los fundamentos de la regresión, empezaremos por desarrollar un ejemplo práctico de regresión lineal simple.

Regresión lineal simple

La regresión lineal simple trabaja solo con dos variables, una explicatoria, habitualmente el eje de las x, y otra variable a predecir, habitualmente el eje y.

También encontraréis en la literatura, variable independiente para el eje x, y variable independiente para el eje y.

Desarrollaremos la formulación del problema a partir de un juego de datos, totalmente ficticio, en el que cada instancia refleja dos ratios. El número de visitas que un mismo comercial ha hecho a 12 clientes distintos y la facturación obtenida por cliente.

Tabla 12. Relación entre visitas y ventas

	Visitas	**Ventas**			
	x	**y**	**xx**	**xy**	**yy**
Cliente 1	1,00	100,00	1,00	100,00	10.000,00
Cliente 2	3,00	109,10	9,00	327,30	11.902,81
Cliente 3	5,00	122,70	25,00	613,50	15.055,29
Cliente 4	6,00	136,30	36,00	817,80	18.577,69
Cliente 5	8,00	149,90	64,00	1.199,20	22.470,01
Cliente 6	10,00	163,60	100,00	1.636,00	26.764,96
Cliente 7	11,00	177,20	121,00	1.949,20	31.399,84
Cliente 8	15,00	190,80	225,00	2.862,00	36.404,64
Cliente 9	16,00	209,00	256,00	3.344,00	43.681,00
Cliente 10	18,00	222,60	324,00	4.006,80	49.550,76
Cliente 11	19,00	240,70	361,00	4.573,30	57.936,49
Cliente 12	20,00	258,90	400,00	5.178,00	67.029,21
Suma	**132,00**	**2.080,80**	**1.922,00**	**26.607,10**	**390.772,70**

Por ejemplo, nuestro vendedor ha realizado 3 visitas comerciales al cliente 2, obteniendo al final del periodo unas ventas de 109,10 euros.

Es importante mencionar que asumimos como un hecho certero que la relación entre las dos variables, número de visitas y ventas obtenidas, es una relación lineal, es decir, que a partir del número de visitas podemos influir de una forma directa sobre las ventas obtenidas.

La regresión lineal simple encuentra una recta que atraviesa los n puntos del juego de datos, de tal forma que consigue hacer mínima la suma de las distancias entre los puntos del juego de datos y los puntos equivalentes en la recta de regresión.

Precisamente, por el hecho de ser una regresión simple, es especialmente apropiada para utilizarla con fines didácticos.

Veamos cómo obtener la recta de regresión correspondiente al juego de datos en la tabla 12.

Nuestra recta deberá ser de la forma $y = \alpha + \beta x$, además le exigiremos que sea la que hace mínima la suma de las distancias entre los puntos del juego de datos y los puntos de la recta, es decir $\underset{\alpha\beta}{Min}\sum_i^n \left(y_i - \alpha - \beta x_i\right)^2$

Es posible demostrar que bajo estas premisas los valores de alfa y beta serán:

$$\beta_{\min} = \frac{\sum_i^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_i^n (x_i - \overline{x})^2} = \frac{\sum_i^n x_i y_i - \frac{1}{n}\sum_i^n x_i \sum_i^n y_i}{\sum_i^n x_i 2 - \frac{1}{n}\left(\sum_i^n x_i\right)^2} = \frac{nS_{xy} - S_x S_y}{nS_{xx} - S_x^2}$$

$$\alpha_{\min} = \frac{S_y - \beta_{\min} S_x}{n}$$

Para facilitar un poco la comprensión de las fórmulas, hemos utilizado la siguiente equivalencia en la notación $\sum = S$ de modo que por ejemplo $\sum_i^n x_i y_i = S_{xy}$

Esta recta, además, posee una propiedad interesante, beta, su pendiente, coincide con el coeficiente de correlación entre ambas variables, corregido por la desviación estándar del eje x, es decir, $\beta_{\min} = \frac{Cov(x, y)}{Var(x)}$

Si aplicamos estas fórmulas a nuestro juego de datos, obtendremos lo siguiente:

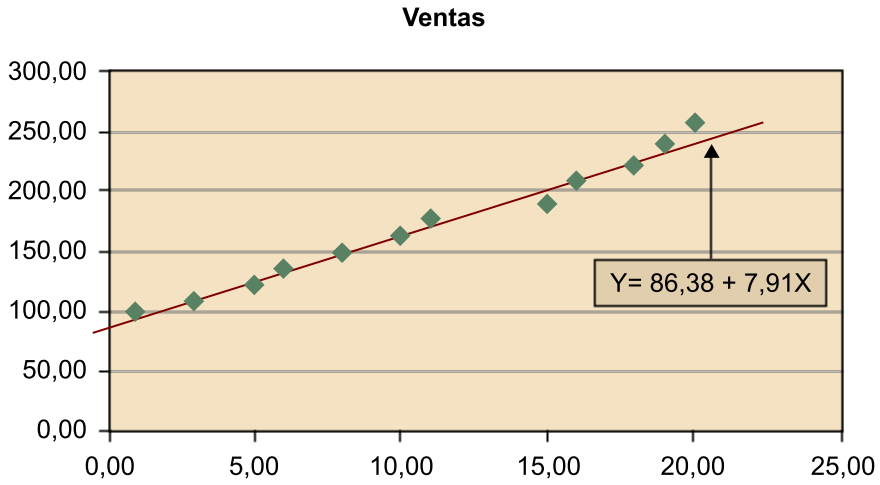
Sx = 132 ; Sy = 2.080,80 ; Sxx = 1.922 ; Sxy = 26.607,10 ; Syy = 390.772,70

$$\beta_{\min} = \frac{nS_{xy} - S_x S_y}{nS_{xx} - S_x^2} = \frac{12 \cdot 26.607,10 - 132 \cdot 2.80,80}{12 \cdot 1.922 - 1.922^2} = 7,91$$

$$\alpha_{\min} = \frac{S_y - \beta_{\min} S_x}{n} = \frac{2.080,80 - 7,91 \cdot 132}{12} = 86,38$$

De modo que nuestra recta de regresión es $y = 86,38 + 7,91 \cdot x$

Figura 21. Recta de regresión simple



A través de la figura 21 es fácil observar cómo la recta de regresión es la recta que mejor modela el atributo, y = ventas, de nuestro juego de datos, sin embargo, no deja de ser una buena aproximación en lugar de ser cien por cien exacto.

Obviamente, cualquier aproximación contiene un cierto grado de desviación. Medir la desviación del modelo es tan importante como obtener el modelo en sí.

Por su complejidad, no mostraremos la formulación correspondiente al intervalo de desviación asociado a una recta de regresión. Además se trata de cálculos fáciles de obtener a partir de un software con capacidades estadísticas.

De todos modos sí que merece la pena mencionar que, siguiendo el ejemplo en la tabla 12, deberíamos poder decir que para x = 6 con un 95% de precisión el intervalo de los posibles valores de y sería $133{,}84€ \pm 2{,}46€$

Regresión lineal múltiple

Para predecir las ventas por cliente, además de utilizar el número de visitas que nuestro comercial realiza al cliente en un período de tiempo, también podríamos haber utilizado variables explicatorias adicionales, como el número de artículos diferentes que el cliente suele comprar de nuestro catálogo.

Bajo estas premisas, la regresión lineal múltiple tratará de modelar la variable y = ventas a partir de las variables explicatorias x = número de visitas, z = número de artículos distintos que suele comprar.

$$y = \alpha + \beta \cdot x + \gamma \cdot z$$

Otros tipos de regresión

Aparte de la regresión lineal en sus versiones simple y múltiple, también disponemos de modelos de regresión para figuras más complejas:

- Regresión exponencial $y = \alpha + \beta^x$.
- Regresión cuadrática $y = \alpha + \beta \cdot x + \gamma \cdot x^2$.
- Regresión logarítmica $y = \alpha + \beta \cdot \log(x)$ se utiliza para predecir variables categóricas dependientes en base a una o más variables independientes. En realidad lo que se modela es la probabilidad de que se dé un valor concreto de la variable categórica dependiente.

Linealización

Por un lado podemos pensar la regresión logarítmica como una regresión lineal en la que la variable independiente o explicatoria en lugar de ser x es $\log(x)$.

Por otro lado, utilizando las propiedades de las operaciones con logaritmos, podríamos llegar a convertir una regresión exponencial en lineal.

$$y = \alpha \cdot e^{\beta \cdot x} \ln(y) = \ln(\alpha \cdot e^{\beta \cdot x}) = \beta \cdot x + \ln(a)$$

Donde en lugar de predecir la variable y, ahora predeciremos la variable $\ln(y)$.

Causalidad

Es importante tener en cuenta que el hecho de que dos variables estén relacionadas no significa necesariamente que una sea causa de la otra, ya que perfectamente puede existir una tercera variable explicatoria que influya también directamente sobre la variable a predecir.

En nuestro ejemplo, donde hemos encontrado una relación lineal directa entre el número de visitas que un comercial hace a sus clientes y la facturación obtenida por cliente, no podemos decir que los incrementos de facturación se deben exclusivamente al incremento de visitas, puesto que podrían también deberse a una tercera variable, como puede ser las acciones comerciales de la competencia.

Extrapolación

Otro error común es el de extrapolar resultados más allá del rango de las variables explicatorias. Trasladándonos a nuestro ejemplo, no podemos extrapolar ninguna conclusión para casos más allá de 20 visitas porque perfectamente podríamos provocar la fatiga del cliente y conseguir un descenso de ventas en lugar de un incremento.

2. Visualización de datos

El análisis visual de datos es importante tanto en las fases previas a un estudio numérico más preciso, como en fases finales de presentación de resultados.

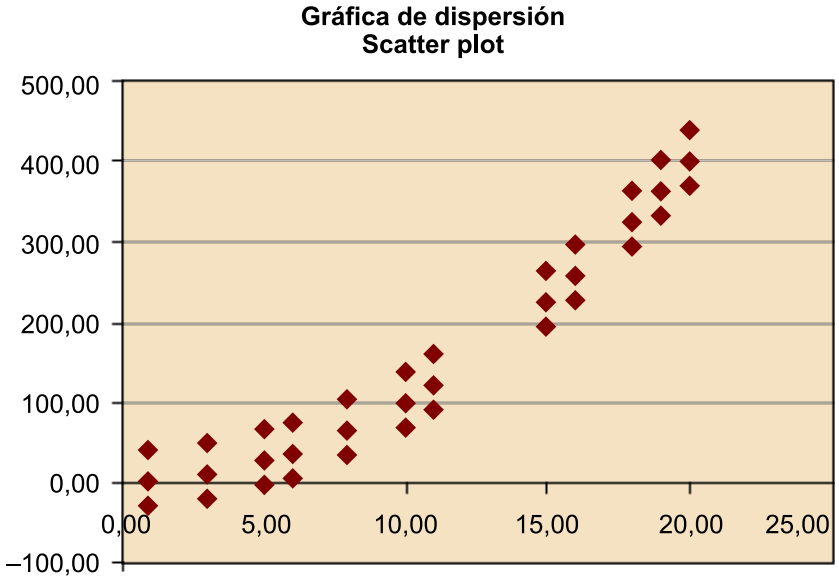
Retomando el caso de la regresión, por ejemplo, antes de intentar inferir la variable dependiente en función de la variable independiente, deberíamos visualizar un diagrama de dispersión para así poder ver de forma gráfica si tiene sentido una aproximación al problema de predicción vía una regresión lineal, cuadrática, polinomial, logarítmica, etc.

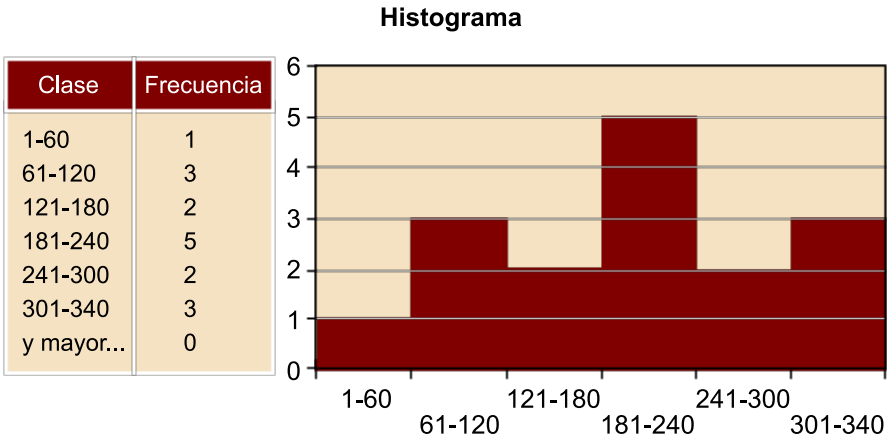
Del mismo modo, independientemente de que el consumidor de la información tenga más o menos habilidades numéricas, una representación gráfica siempre facilita enormemente la comprensión del contenido informativo del resultado.

De hecho, en un estudio, renunciar a la información gráfica sería como renunciar a la inteligencia visual para comprenderlo.

En la siguiente figura mostramos un ejemplo de gráfica de dispersión y otro de histograma o diagrama de frecuencias. Ambas obtenidas con la herramienta MS Excel.

Figura 22. Ejemplos de gráfica





Otros gráficos muy utilizados son los gráficos de superficie (*surface plot*), mapas jerarquizados (*tree map*), gráficos de línea con múltiples variables (*parallel coordinate plot*) y dendrogramas (ver figura 12).

Histograma con MS Excel

MS Excel no ofrece la posibilidad de obtener histogramas de forma directa, sin embargo, con algunos pasos sencillos podemos construirlo.

- Es necesario tener activado el complemento de análisis de datos. Una vez activado o instalado, procederemos del siguiente modo.
- Accedemos a Datos, Análisis de datos y seleccionamos Histograma.
- Informamos de los rangos de entrada y de clases. Esto nos genera una tabla de frecuencias por clase.
- Creamos una gráfica de barras sobre esta tabla. Accedemos a la configuración de la tabla en Formato de serie de datos y fijamos el ancho del intervalo a cero.

Visual data discovery

Comprende un conjunto de funcionalidades orientadas a favorecer una comprensión visual de la información contenida en los datos yendo más allá del simple gráfico.

Este tipo de funcionalidades actualmente se integran en herramientas especialmente orientadas al análisis de datos y suelen ofrecer las siguientes prestaciones:

- Acceso a datos de forma interactiva. Intercambio de filas por columnas, arrastrar columnas nuevas, navegar de datos agregados al detalle de los mismos.

- Simulaciones y estudios tipo *What if*, con la posibilidad de guardarse los diferentes estudios en forma de versiones, de modo que sean comparables entre ellas o incluso con la funcionalidad de poder comparar escenarios con datos reales.

- Facilidad para el diseño de informes con posibilidad de combinar gráficos con resúmenes numéricos de frecuencias u otros estadísticos.

- Disponibilidad de columnas con operadores estadísticos, como covarianzas, desviaciones estándar, intervalos de confianza, etc.

- Gráficos con movimiento de variables dentro de un rango, incorporando conceptos nuevos como la velocidad de cambio. Burbujas móviles, por ejemplo.

- Funciones de optimización de objetivos.

- Funciones de secuenciación de tareas. Por ejemplo, obtención de datos, transformación de datos, generación de gráficos, puesta a disposición de informes a los consumidores de información, y a su vez, programar todos estos pasos para que se ejecuten con una cierta periodicidad.

Multidimensional *scaling*

MDS tiene por objetivo estudiar la similitud entre variables en un juego de datos, o dicho de otro modo, busca patrones de distancia o similitud entre variables.

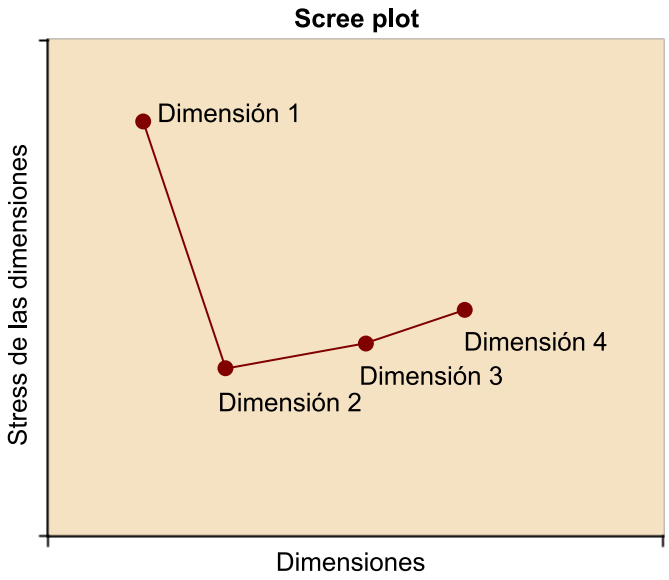
En realidad se trata de un proceso en forma de combinación de técnicas estadísticas, que dan como resultado un gráfico de distancias o similitud entre variables.

El procedimiento es el siguiente, en primer lugar se construye una matriz de distancias o similaridades.

En segundo lugar ejecutaríamos un proceso de análisis de factores (*scree plot or factor analysis*) con el objetivo de establecer cuantas dimensiones son las que contienen la mayoría de la información.

Por ejemplo, si queremos comparar 5 almacenes en función de sus movimientos logísticos (entradas y salidas de mercancías). El análisis de factores nos dirá si para dibujar gráficamente la similitud entre los 5 almacenes es más apropiado hacerlo en un gráfico de 1 dimensión, 2 dimensiones o 3 dimensiones. Donde cada dimensión será la combinación de las variables iniciales (ver el capítulo dedicado al análisis de componentes principales).

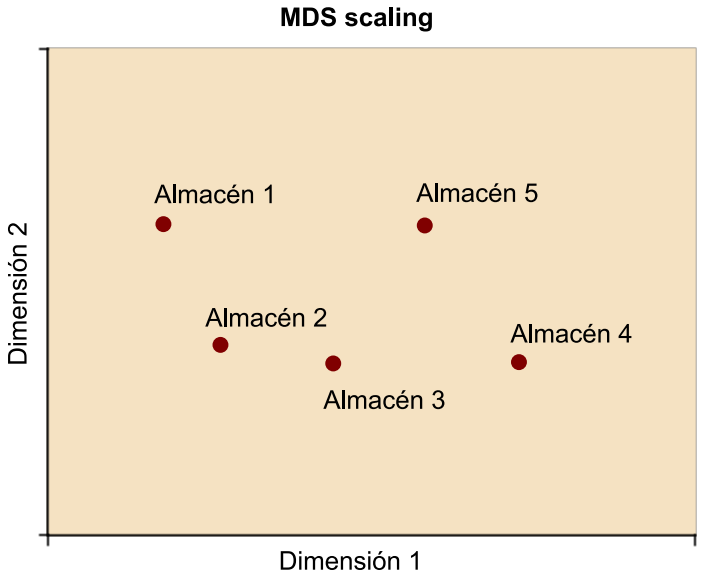
Figura 23. Análisis de factores



En nuestro caso, a través del *scree plot* vemos claramente como 2 dimensiones son suficientes para explicar la mayoría de la variabilidad de la información en el juego de datos.

El tercer y último paso consiste en dibujar un gráfico de dispersión de puntos, donde, volviendo al ejemplo de los almacenes, cada punto sería un almacén. Este gráfico nos permitirá ver qué almacenes tienen una actividad similar y qué almacenes tienen una actividad singular.

Figura 24. Multidimensional *scaling*



A modo de resumen, algunos autores consideran MDS *scaling* como una alternativa al análisis de factores, de hecho podríamos ver MDS como un complemento visual del análisis de factores, contribuyendo de este modo a una mayor comprensión gráfica de la estructura de los datos.

3. Anexo

3.1. Distancia o similitud

Cuando hablamos de distancia, en realidad nos estamos refiriendo a una forma de cuantificar cuán similares son dos objetos, variables o puntos.

Planteado de esta forma, el concepto de distancia es muy abstracto y etéreo, por este motivo los científicos quieren ponerle algunos límites o condiciones:

Para un conjunto de elementos X se considera distancia cualquier función (X x X) → R que cumpla las siguientes tres condiciones:

- La no negatividad: La distancia entre dos puntos siempre debe ser positiva.
  $d(a, b) \geq 0 \forall\ a, b \in X$

- La simetría: La distancia entre un punto a y un punto b debe ser igual a la distancia entre el punto b y el punto a.
  $d(a, b) = d(b, a) \forall\ a, b \in X$

- La desigualdad triangular: La distancia debe coincidir con la idea intuitiva que tenemos de ella en cuanto a que es el camino más corto entre dos puntos.
  $d(a, b) \leq d(a, c) + d(b, c) \forall\ a, b, c \in X$

Bajo estas tres condiciones, vamos a centrarnos en tres definiciones de distancia muy peculiares: La euclidiana o clásica, la estadística o de Gauss y la que propuso Mahalanobis.

1) Distancia euclidiana

La distancia euclidiana coincide plenamente con lo que nuestra intuición entiende por distancia. Podríamos llamarle la distancia "ordinaria".

Su expresión matemática para un espacio, por ejemplo de dos dimensiones, sería:

$$d(X, Y) = d((x_1, x_2), (y_1, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Inconveniente

Utilizar esta distancia en un proceso de segmentación presenta el problema de que no tiene en cuenta las diferentes unidades de medida en las que pueden estar expresadas la variables *X* e *Y*.

Por ejemplo:

Si la variable *X* representa la edad de los ratones, entonces tomará valores de entre 0 y 3.

Si la variable *Y* representa la longitud de la cola del ratón en milímetros, puede tomar valores entre 90 y 120.

Parece claro que cuando calculemos la "distancia o similitud" entre dos individuos, pesará injustamente mucho más la longitud de la cola que la edad.

2) Distancia estadística o de Gauss

Distancia estadística

Debemos este avance al brillante matemático alemán, Carl Friedrich Gauss (1777 – 1855).

Para superar la distorsión provocada por las diferentes unidades de medida usadas en las distintas variables estudiadas, tenemos la distancia estadística, que simplemente "normaliza" la variables para situarlas todas bajo la misma escala.

Su expresión analítica para un espacio, por ejemplo de dos dimensiones, en la que nuestra distribución de puntos en estas dimensiones tuviera una desviación estándar σ:

$$d(X, Y) = d((x_1, x_2), (y_1, y_2)) = \sqrt{\left(\frac{x_2 - x_1}{\sigma(X)}\right)^2 + \left(\frac{y_2 - y_1}{\sigma(Y)}\right)^2}$$

Inconveniente

Nuevamente, este concepto de distancia sigue teniendo problemas.

No tiene en cuenta la correlación entre las variables, es decir, si nuestras variables de trabajo fueran totalmente independientes, no habría ningún problema, pero si tienen algún tipo de correlación, entonces "una influye sobre la otra" y esta "influencia" no queda bien reflejada si usamos la distancia estadística.

Por ejemplo, las variables *entrenar* y *rendimiento* están correlacionadas, de modo que más entrenamiento siempre implica más rendimiento, pero esta regla no se cumple infinitamente, ya que entrenamiento infinito no implica rendimiento infinito (lo vemos continuamente en el deporte).

Si no tenemos en cuenta esta correlación y queremos comparar a dos deportistas (medir su distancia) podemos llegar a conclusiones erróneas.

3) Distancia de Mahalanobis

Prasanta Chandra Mahalanobis (India) en 1936 se dio cuenta de esta carencia y propuso corregir la distorsión provocada por la correlación de las variables mediante la siguiente expresión.

La versión simplificada para un espacio, por ejemplo de dos dimensiones, con un conjunto de puntos de varianza $\sigma^2(X)$, $\sigma^2(Y)$ y covariancia cov(X,Y) sería:

$$d(X, Y) = d((x_1, x_2), (y_1, y_2)) =$$

$$= \sqrt{(x_1 - y_1, x_2 - y_2) \begin{bmatrix} \sigma^2(X) & \mathrm{cov}ar(X, Y) \\ \mathrm{cov}ar(Y, X) & \sigma^2(Y) \end{bmatrix}^{-1} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix}}$$

Relación de las tres distancias

Que Mahalanobis generalice a Gauss y Gauss a Euclides forma parte de lo que se ha dado en llamar "la belleza de las matemáticas". Veámoslo.

Si en nuestra distribución de puntos, las dos dimensiones son totalmente independientes, tendremos que su covarianza será cero, de manera que la distancia de Mahalanobis coincide con la distancia Estadística.

$$d(X, Y) = \sqrt{(x_1 - y_1, x_2 - y_2) \begin{bmatrix} \sigma^2(X) & 0 \\ 0 & \sigma^2(Y) \end{bmatrix}^{-1} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix}} =$$

$$= \sqrt{\left(\frac{x_2 - x_1}{\sigma(x)}\right)^2 + \left(\frac{y_2 - y_1}{\sigma(y)}\right)^2}$$

Si en la misma distribución con variables independientes tenemos que su distribución está "normalizada", es decir, que las dos variables tienen la misma escala, entonces su varianza será 1, de manera que la distancia estadística coincide con la distancia de Euclides.

$$d(X, Y) = \sqrt{(x_1 - y_1, x_2 - y_2) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Desarrollemos un ejemplo

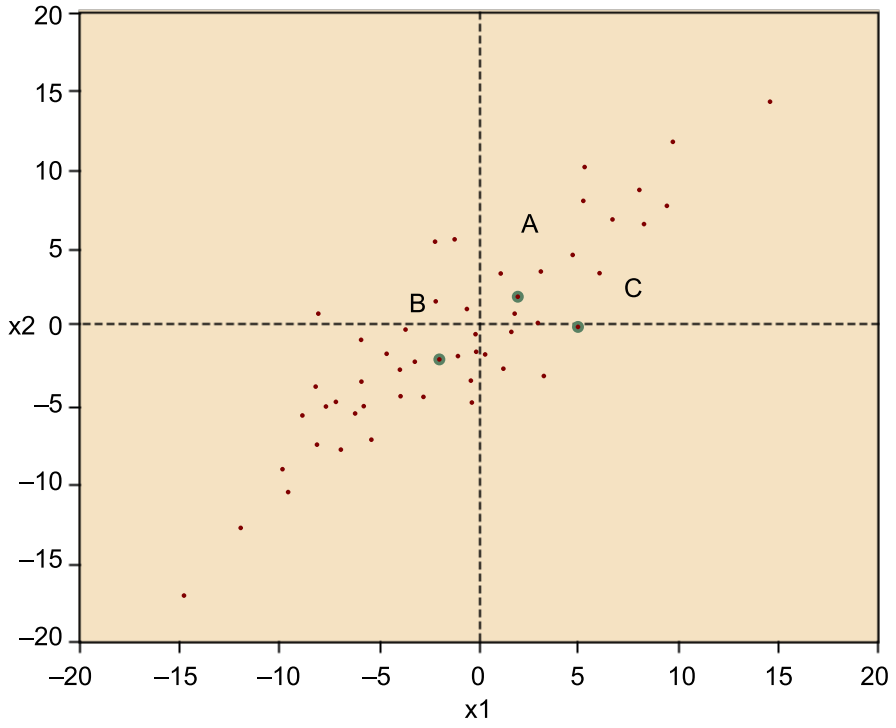
Supongamos que disponemos de un juego de datos formado por pares de números de variables x1 y x2. Con media (0,0), varianza 5 para las dos variables y covarianza 3 entre las dos variables.

Primero observamos que en nuestro juego de datos las dos variables no están expresadas en la misma escala, porque de lo contrario tendrían varianza 1 y además las dos variables no son totalmente independientes, porque si no tendrían covarianza 0. Con este juego de datos tenemos la matriz de covarianza

$$\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

Nos fijamos en tres puntos de nuestro juego de datos, el punto A = (2,2), el punto B = (-2,-2) y el punto C = (0,5)

Figura 25. Ejemplo distancia Mahalanobis, Gauss, Euclides



Queremos determinar cuál de los puntos B o C se encuentra más cerca del punto A.

Para ello utilizaremos la distancia de Euclides, la estadística y la de Mahalanobis, y veremos que ofrecen resultados distintos.

Euclides nos diría que C está más cerca de A:

$$d(A, B) = d((2,2), (-2,-2)) = \sqrt{(2-(-2))^2 + (2-(-2))^2} = \sqrt{32}$$

$$d(A, C) = d((2,2), (0,5)) = \sqrt{(2-0)^2 + (2-5)^2} = \sqrt{13}$$

La distancia estadística también nos diría que C está más cerca de A:

$$d(A, B) = d((2,2), (-2,-2)) = \sqrt{\frac{(2-(-2))^2}{5} + \frac{(2-(-2))^2}{5}} = \sqrt{6,4}$$

$$d(A, C) = d((2,2), (0,5)) = \sqrt{\frac{(2-0)^2}{5} + \frac{(2-5)^2}{5}} = \sqrt{2,6}$$

Veremos que para Mahalanobis B es el punto más cercano a A y no C.

La inversa para una matriz 2x2 es la siguiente:

$$\begin{bmatrix} a & d \\ c & b \end{bmatrix}^{-1} = \frac{1}{ab-cd}\begin{bmatrix} b & -d \\ -c & a \end{bmatrix}$$

Calculemos la inversa de nuestra matriz de covarianza:

$$\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}^{-1} = \frac{1}{25-9}\begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix} = \begin{bmatrix} \frac{5}{16} & \frac{-3}{16} \\ \frac{-3}{16} & \frac{5}{16} \end{bmatrix}$$

Estamos ya en disposición de calcular la distancia entre A y B según Mahalanobis:

$$d(A, B) = d((2,2), (-2,-2)) = \sqrt{(2-(-2), 2-(-2))\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}^{-1}\begin{pmatrix} 2-(-2) \\ 2-(-2) \end{pmatrix}} =$$

$$= \sqrt{(4,4)\begin{bmatrix} \frac{5}{16} & \frac{-3}{16} \\ \frac{-3}{16} & \frac{5}{16} \end{bmatrix}(4,4)} = \sqrt{\begin{pmatrix} \frac{8}{16} \\ \frac{8}{16} \end{pmatrix}(4,4)} = \sqrt{4}$$

Calculemos la distancia entre A y C según Mahalanobis:

$$d(A, C) = d((2,2), (0,5)) = \sqrt{(2-0, 2-5)\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}^{-1}\begin{pmatrix} 2-0 \\ 2-5 \end{pmatrix}} =$$

$$= \sqrt{(2,-3)\begin{bmatrix} \frac{5}{16} & \frac{-3}{16} \\ \frac{-3}{16} & \frac{5}{16} \end{bmatrix}\begin{pmatrix} 2 \\ -3 \end{pmatrix}} = \sqrt{\begin{pmatrix} \frac{19}{16} \\ \frac{-21}{16} \end{pmatrix}\begin{pmatrix} 2 \\ -3 \end{pmatrix}} = \sqrt{\frac{101}{16}} = \sqrt{6,3125}$$

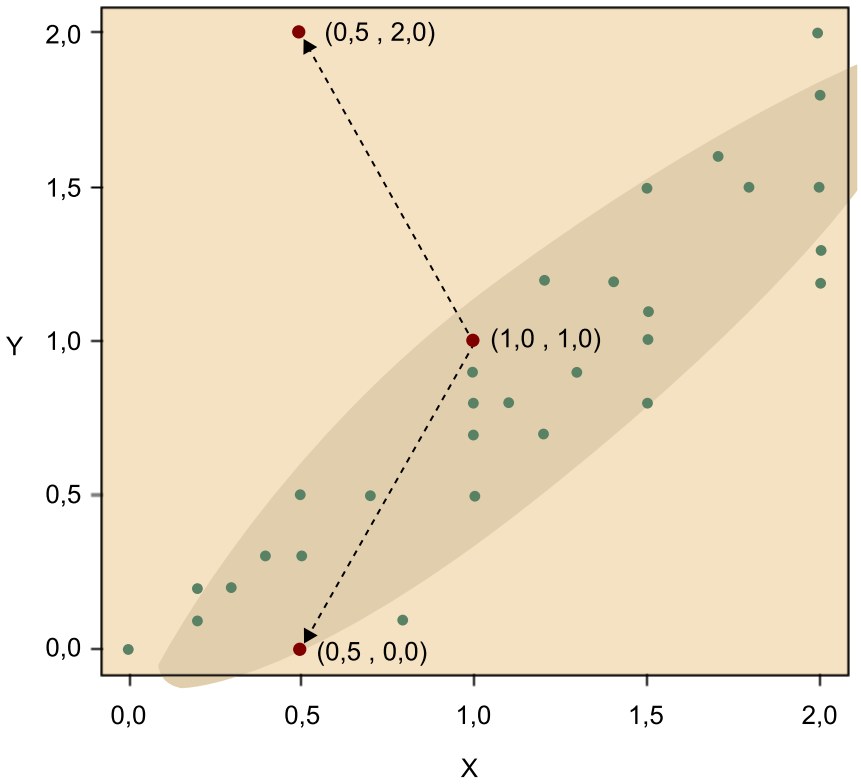
Un valor *outlier* o atípico en un juego de datos es aquel que, aparentemente, se desvía del resto de valores similares. En *business analytics* identificar estos puntos es importante puesto que distorsionan los resultados en la mayoría de algoritmos.

La distancia de Mahalanobis ha demostrado ser un buen detector de estas anomalías. Veámoslo en el siguiente apartado.

Mahalanobis como buen detector de *outliers*

La definición de distancia propuesta por Mahalanobis responde a la idea intuitiva de que los puntos que se encuentran en una zona densamente poblada deberían considerarse más cercanos entre ellos que con respecto a puntos fuera de esta zona de mayor densidad.

Figura 26. Mahalanobis como buen detector de *outliers*



Observemos en la figura 26 que la mayoría de puntos se encuentran en la zona sombreada. Esta es la zona con mayor densidad de puntos en la figura.

Mediante el ejemplo de la figura 26 veremos que el punto (0.5, 2.0) es un *outlier*.

Para ello, fijémonos en los siguientes tres puntos:

- El punto (0.5, 0.0) es un punto perfectamente integrado en la zona densa de puntos.

- El punto (1.0, 1.0) es también un punto perfectamente integrado en la zona de puntos con mayor densidad.

- El punto (0.5, 2.0) es claramente un valor *outlier*, al menos la intuición y la observación visual así lo indican.

Veremos cómo, según la distancia euclidiana, los tres puntos son equidistantes, esto es porque esta distancia no tiene en cuenta la densidad de puntos.

Veremos también cómo, según la distancia de Mahalanobis, el punto (0.5, 2.0) está claramente alejado del resto.

Distancia euclidiana entre (0.5, 0.0) y (1.0, 1.0) = 1.118034.

Distancia euclidiana entre (1.0, 1.0) y (0.5, 2.0) = 1.118034.

Observamos que estos puntos, según Euclides, son equidistantes tal y como nos indica nuestra intuición.

Distancia Mahalanobis entre (0.5, 0.0) y (1.0 , 1.0) = 1.930186.

Distancia Mahalanobis entre (1.0, 1.0) y (0.5 ,2.0) = 3.889715.

Sin embargo Mahalanobis nos hacer ver que si tomamos en consideración la densidad de puntos, el punto (0.5, 0.0) está más cerca de (1.0, 1.0) por estar en una zona densa o más poblada de puntos.

De hecho, tomando como métrica de distancia la de Mahalanobis, la zona sombreada de la figura 26 se corresponde con una circunferencia de radio 1.930186. Podríamos tomar como criterio, por ejemplo, que todos los puntos fuera de la zona sombreada son puntos *outliers*.

3.2. Estadística y álgebra

3.2.1. Estadística

Sea $X = (x_1, x_2, \ldots x_n)$ un atributo o variable y sus valores. Sobre este atributo podemos calcular los siguientes estadísticos.

La media $\overline{X} = \frac{\sum_1^n x_i}{n}$.

La desviación estándar $s = \sqrt{\frac{\sum_1^n (x_i - \overline{X})^2}{n}}$ nos sirve para medir la dispersión de los valores de la variable X.

La varianza $s^2 = \mathrm{var}(X) = \frac{\sum_1^n (x_i - \overline{X})^2}{n}$, también es una medida de la dispersión.

El siguiente estadístico nos será útil para poder comparar pares de variables X, Y:

La covarianza $\mathrm{cov}(X, Y) = \frac{\sum_1^n (x_i - \overline{X}) \cdot (y_i - \overline{Y})}{n}$ nos sirve para medir la dispersión de los valores de una variable respecto de la otra.

No es difícil comprobar las siguientes identidades $\mathrm{cov}(X, Y) = \mathrm{cov}(Y, X)$; $\mathrm{cov}(X, X) = \mathrm{var}(X)$

Lo importante de la covarianza más que el valor es el signo. Valores positivos indican que ambos atributos crecen del mismo modo. Valores negativos indican que cuando un atributo crece, el otro decrece. Covarianza cero indica que los dos atributos son independientes.

Cuando trabajemos con más de dos atributos, será conveniente familiarizarnos con la matriz de covarianza. Se trata de una matriz cuadrada que identificará las relaciones entre los pares de atributos respecto de la variabilidad de sus valores respectivos.

Como ejemplo, mostramos la matriz de covarianza para un juego de datos con 3 atributos x, y, z:

$$\mathrm{cov} = \begin{pmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,X) & \mathrm{cov}(Y,Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{cov}(Z,Z) \end{pmatrix} = \begin{pmatrix} \mathrm{var}(X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,X) & \mathrm{var}(Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{var}(Z) \end{pmatrix}$$

En la tabla 13 vemos un ejemplo de cálculo de covarianza entre dos atributos. Las horas que un estudiante ha dedicado al estudio, y las calificaciones que ha obtenido.

Tabla 13. Juego de datos de horas de estudio y calificaciones.

	Horas de estudio	Calificaciones			
	H	C	$(H_i - \overline{H})$	$(C_i - \overline{C})$	$(H_i - \overline{H}) \cdot (C_i - \overline{C})$
Estudiantes	9	39	–4,92	–23,42	115,13
	15	56	1,08	–6,42	6,95
	25	93	11,08	30,58	338,97
	14	61	0,08	–1,42	0,12
	10	50	–3,92	–12,42	48,63
	18	75	4,08	12,58	51,38
	0	32	–13,92	–30,42	423,30
	16	85	2,08	22,58	47,05
	5	42	–8,92	–20,42	182,05
	19	70	5,08	7,58	38,55
	16	66	2,08	3,58	7,47
	20	80	6,08	17,58	106,97
Media	13,92	62,42	62,42	Cov (H, C) =	112,70

Vistos los conceptos estadísticos que necesitamos, vamos ahora a por los conceptos algebraicos. Dada una matriz M, estudiaremos qué son sus vectores propios y sus valores propios.

3.2.2. Álgebra

Vectores propios (*eigenvectors*)

Dada una matriz cuadrada M nxn, diremos que v es un vector propio de M y que λ es un valor propio de M si se cumple que $M \cdot v = \lambda \cdot v$

Veamos primero un ejemplo sobre la matriz $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$

Si la multiplicamos por el vector $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ tendremos que $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \cdot \begin{pmatrix} 3 \\ 2 \end{pmatrix}$

$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ Es un vector propio de la matriz $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$ y 4 es una valor propio de la matriz $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$

Sin embargo, si la multiplicamos por el vector $\begin{pmatrix} 1 \\ 3 \end{pmatrix}$ tendremos que $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$ y vemos como el vector $\begin{pmatrix} 1 \\ 3 \end{pmatrix}$ no es vector propio de la matriz.

Interpretación de los vectores propios

Podemos pensar el producto de una matriz por un vector $M \cdot v$ como una transformación del vector o como una proyección del vector v sobre una recta o sobre un plano. Pues bien, si se cumple $M \cdot v = \lambda \cdot v$ significa que el vector v ya se encuentra en la recta o plano de proyección de la matriz.

Propiedades de los vectores propios

- Solo podemos encontrar vectores propios en matrices cuadradas nxn. Además, no todas las matrices cuadradas tienen vectores propios. Lo que sí se cumple es que si una matriz nxn tiene vectores propios, entonces el número de vectores propios que tendrá será exactamente *n*.

- Si consideramos un escalar de un vector propio, al multiplicarlo por la matriz, igualmente obtendremos un múltiplo λ del escalar del vector propio. Por este motivo tiene sentido hablar también de valores propios λ. Esto es debido al hecho de que, cuando multiplicamos un escalar por un vector, simplemente estamos alterando la longitud del vector, pero no la dirección del mismo.

- Todos los vectores propios de una matriz son perpendiculares entre sí. Esta propiedad es crucial puesto que significa que podemos representar todas las instancias del juego de datos en función de las nuevas coordenadas formadas por los vectores propios.

Valores propios (*eigenvalues*)

Un valor propio siempre va asociado a su respectivo vector propio, y su magnitud es una medida de la importancia del vector propio con respecto al resto de vectores propios de la matriz.

Un valor propio mayor indica que su vector propio asociado explica la mayor parte de la variabilidad de la matriz o juego de datos. Esta propiedad nos permitirá ordenar los vectores propios por relevancia y poder determinar si descartamos los vectores propios menos representativos con el objetivo de reducir la dimensionalidad del juego de datos, perdiendo así, una mínima cantidad de información por ello.

Calcular vectores y valores propios

Excede del ámbito de este material didáctico profundizar en este tipo de operaciones. Sin embargo, merece la pena comentar que existen herramientas software como R o Matlab que disponen de funciones para realizar estos cálculos.

A título de información y como apoyo a los ejemplos que se presentan a continuación, adjuntamos las fórmulas para obtener los vectores y los valores propios de una matriz 2x2.

Tomemos como matriz de covarianza $\begin{pmatrix} a & c \\ c & b \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \cdot \begin{pmatrix} x \\ y \end{pmatrix}$, entonces:

Los dos valores propios son $\lambda = \dfrac{a+b \pm \sqrt{(a+b)^2 - 4 \cdot (a \cdot b - c^2)}}{2}$

Los dos vectores propios son $x = \pm \dfrac{\sqrt{1+\left(\frac{\lambda-a}{c}\right)^2}}{1+\left(\frac{\lambda-a}{c}\right)^2}$; $y = \dfrac{\lambda - a}{c} \cdot x$

Resumen

Hemos profundizado en el funcionamiento interno de algoritmos más o menos complejos. El estudiante a nivel esquemático comprende la operativa de los algoritmos y a nivel matemático ha encontrado en este material didáctico herramientas suficientes para estudiar con más profundidad la mayoría de ellos.

Inicialmente se han presentado las tipologías de algoritmos, distinguiendo entre supervisados y no supervisados, la mayoría de ellos tienen por objetivo la clasificación y la segmentación descriptiva o predictiva.

Se ha estudiado con detalle el concepto de ganancia de la información como base para poder entender bien el mecanismo de los árboles de decisión.

También nos hemos atrevido con algoritmos realmente complejos de seguir en su nivel más detallado, como son las redes neuronales y las máquinas de vectores de soporte, de donde hemos extraído una herramienta potente como son las funciones kernel presentes en otros tipos de algoritmos.

Igualmente, el concepto de distancia o similitud, explicado en el documento anexo, aparece de forma insistente y forma parte de la base de la mayoría de algoritmos de segmentación.

Se ha trabajado K-Means como exponente de los algoritmos de *clustering* jerárquico y hemos visto la importancia de la selección de una métrica adecuada en este tipo de algoritmos. Asimismo, se ha visto *canopy-clustering* como una técnica brillante que optimiza K-Means a nivel de coste computacional y además permite su computación en paralelo.

Con relación a los algoritmos de asociaciones, se han estudiado los conceptos de esperanza y soporte, sencillos de entender y a la vez cruciales en la construcción del algoritmo Apriori.

En el capítulo de análisis de componentes principales, se ha trabajado en detalle el problema de la reducción de la dimensionalidad y de la correlación entre las variables. Se trata de una herramienta que en muchas ocasiones supondrá un paso previo o un paso exploratorio antes de utilizar otros algoritmos.

Bibliografía

McCallum, A.*Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching*.

Sierra Araujo, B. (2006). *Aprendizaje Automático: conceptos básicos y avanzados*. Pearson Educación.

Nisbert, R.;Elder, J.; Miner, G. (2009). "Handbook of Statistical Analysis and Data Mining Applications". *Academic Press*.

Bink Liu (2011). *Web Data Mining. Exploring*. Springer.

Hastie, T.; Tibshirani, R.; Friedman, J. (2001). *The elements of statistical learning*. Nueva York: Springer.

Hernández Ovallo, J.; Ramírez Quintana, M. J.; Ferri Ramírez, C. (2004). *Introducción a la minería de datos*. Madrid: Pearson Prentice-Hall.

Artículos

Bisciglia, C. (2007). "Distributed Computing Seminar" (Lectura 4). Google.

Smith, L. (2002). *A tutorial on Principal Component Analysis*.

Terrádez Gurrea, M.*Análisis de componentes principales*. UOC.

Fernández Rebollo, F. *Redes de neuronas*. Universidad Carlos III de Madrid.

Muñoz Gutiérrez, C.*Redes neurales*. Departamento de Lógica y Filosofía de la Ciencia.

Burges, C. J. C. (1998). "A tutorial on support vector machines for pattern recognition". *Data mining and knowledge discovery* (núm. 2, pág. 121-167).

Dietterich, T. G. (1997). "Machine-learning research: Four current directions". *AI magazine* (pág. 97-136).