Aprendizaje Automatizado y Aplicaciones

Jorge Humberto Moreno Córdoba

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Matemáticas
Medellín, Colombia
2019

Aprendizaje Automatizado y Aplicaciones

Jorge Humberto Moreno Córdoba

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ciencias, Matemáticas aplicadas

Director:
Ph.D. Fernando Alberto Morales Jauregui

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Matemáticas
Medellín, Colombia
2019

A mis padres.

Agradecimientos

Gracias a mis padres por su apoyo permanente. Además, un reconocimiento a mis compañeros y a mi tutor por su constante compañía.

Resumen

Una parte esencial del análisis de datos es la definición de la información o datos suministrados por un fenómeno. Aunque existen diferentes estructuras de datos, en todos los casos es indispensable realizar un preprocesamiento, el cual puede ir desde la categorización de una variable o la reducción de dimensiones, hasta la aproximación de entradas ausentes en la matriz de datos por medio de la compleción de matrices. Este trabajo se centra en el estudio de los métodos básicos para el entendimiento del aprendizaje automatizado con un lenguaje formal, considerando datos estructurados, centrado en el aprendizaje supervisado y no supervisado. Así mismo, se establecen algunos de los métodos que poseen una gran capacidad predictiva. Estas técnicas son reconocidos por sus múltiples aplicaciones en la segmentación de datos y su utilidad en la detección de anomalías.

Palabras clave: Aprendizaje automatizado, estructura de datos, aprendizaje supervisado, aprendizaje no supervisado, segmentación, detección anomalías, compleción de matrices, reducción de dimensiones, predicción.

Abstract

An essential part of data analysis is the defintion of the information or suministered data given by a phenomena. Even though there are different data structures, in every case it is essential to make a pre-processing. This pre-process can go from categorization of a variable or reduction of dimensions to the aproximation of absent entries in the data matrix using matrix completion. This work is centered in the study of the basic methods needed for the understanding of machine learning with a formal language considering structural data, centered in the supervised and unsupervised learning. In this same manner, some of the methods stablished have a great predictive capacity. These methods are recognized by their multiple aplications in data segmentation and their utility in anomaly detection.

Keywords: Machine Learning, data structure, supervised learning, unsupervised learning, clustering, anomaly detection, exact matrix completion, dimensionality reduction, prediction.

Contenido

Agradecimientos VII

Resumen IX

1. Introducción 2

2. Recolección de datos 4
2.1. Estructura de datos . . . 4
2.2. Tipos de datos . . . 6

3. Preprocesamiento de datos 9
3.1. Reducción de dimensiones . . . 9
3.1.1. Análisis de componentes principales (PCA) . . . 10
3.1.2. Porcentajes de variabilidad . . . 13
3.2. Compleción de matrices . . . 15
3.2.1. Matrices reconstruibles . . . 16
3.2.2. Conjunto de muestreo . . . 17
3.2.3. Algoritmo de compleción . . . 18

4. Segmentación 20
4.1. Problema de identificación de clases . . . 20
4.2. K-Medias . . . 22
4.3. K-vecinos más cercanos . . . 25
4.4. Verosimilitud . . . 27
4.5. Mean shift . . . 30
4.6. Otros Métodos . . . 33

5. Predicción 36
5.1. Predicción y error . . . 36
5.1.1. Predicción binaria . . . 37
5.1.2. Predicción discreta . . . 40
5.1.3. Predicción continua . . . 40
5.2. Regresión logística . . . 42
5.3. Árboles de decisión . . . 43

6. Detección de anomalías **47**
6.1. Anomalías por segmentación 48
6.2. Bosque de aislamiento 49
6.3. Otros Métodos 56

A. Anexo: Implementación de compleción de matrices **57**

B. Anexo: Implementación de PCA **61**

C. Anexo: Implementación de K-medias **64**
C.0.1. Aplicación: descomposición de imágenes 66

D. Anexo: Implementación de Mean shift **67**

Bibliografía **70**

1. Introducción

Con la ayuda de la tecnología y los actuales medios de recolección o transmisión de información, se ha convertido en una práctica habitual en la industria el análisis de bases de datos, que en su mayoría tienen un tamaño que resulta imposible un procesamiento manual. No existe una forma estándar o una convención para dar estructura o forma a la información almacenada, esto depende de cada caso: datos estructurados, no estructurados y semiestructurados. Para cada tipo de estructura de datos existen técnicas y procedimientos diferentes para su tratamiento y modelación, sin embargo, sin importar la estructura de los datos se puede decir que a grandes rasgos existen 5 etapas del proceso:

- Diseño: aunque no siempre es posible, es ideal poder tener un diseño o planificación de la estructura de datos, estimar las variables más adecuadas y la cantidad de observaciones necesarias para realizar un modelo determinado. Esto se conoce como diseño de experimentos, un área ampliamente estudiada en la estadística.
- Recolección: por medio de bases de datos, servicios, aplicativos y demás herramientas que recolectan información, es común encontrar cúmulos de datos con volúmenes crecientes y de gran tamaño. Los mecanismos y técnicas de recolección son determinados por las áreas de tecnología de la información.
- Preprocesamiento: es todo el tratamiento o depuración que se realiza sobre la información para lograr que los datos ingresados a un modelo o una visualización sean completos y adecuados. Este paso es relevante en el análisis de datos, porque los errores cometidos en esta parte del proceso pueden afectar la convergencia de los modelos o disminuir la capacidad predictiva. En esta etapa se realizan procesos como normalización de matrices, compleción de matrices, reducción de dimensiones y otras operaciones similares.
- Modelamiento: con base a los datos se crean modelos matemáticos capaces de hacer una predicción para futuras observaciones que tengan las misma variables observadas, detectar anomalías, clasificar observaciones o incluso, tomar decisiones respecto a diferentes escenarios posibles. Para esto se utilizan diferentes herramientas matemáticas como la teoría de grafos, estadística, geometría e incluso la topología, en áreas como la topología computacional o la topología de datos.
- Predicción o implementación: con base a la información inicial y al modelo creado, se usa este resultado para lograr predicciones sobre nueva información y se estructura los resultados

obtenidos para lograr tener una medida de error y si es posible, el modelo aprende de la nueva información para mejorar su capacidad predicativa. En la actualidad se usan modelos robustos que toman decisiones o hacen predicciones con la nueva información en tiempo real, por lo cual se exigen que estas predicciones se hagan de forma eficiente.

El auge del estudio y experimentación en la ciencia de datos está impulsada por la gran necesidad que tienen la industria en diferentes sectores. Estás aplicaciones han evolucionado y se han diversificado en los últimos años, no sólo se usan para segmentación de clientes y estrategias de ventas, sino que también ha migrado al sector financiero con el análisis de fraude en tarjetas de créditos y modelos de fuga de clientes que disminuyen las tasas de deserción en diferentes productos, además de las múltiples aplicaciones en el estudio de los mercados y el trading algorítmico. Las aseguradoras buscan patrones y detectan potenciales fraudes en cobros de seguros. El área de la tecnología y la innovación es un sector amplio, donde se pueden ver aplicaciones en redes sociales o la robótica con la inteligencia artificial, incluso una de las ramas más recientes que ha llamado la atención de miles de científicos, es la conducción autónoma de vehículos por medio del análisis de imágenes y toma de decisiones no supervisadas.

2. Recolección de datos

La ciencia de datos en la actualidad es un foco importante para la aplicación de las matemáticas y la estadística. Aunque la mayoría de las técnicas estén inmersas en procedimientos heurísticos, su fundamento está intrínsecamente relacionado con la interpretación estadística, geométrica o topológica. Por esta razón, en esta sección se exponen los conceptos básicos necesarios para el entendimiento de algunas de las técnicas clásicas del aprendizaje automatizado bajo una formalización.

2.1. Estructura de datos

No existe una convención para dar estructura o forma a la información almacenada, esto depende directamente de las herramientas usadas, el diseño experimental, el conocimiento de la información y el tipo de variables medidas en el fenómeno. En general, la información que se recolecta se suele enmarcar como: datos estructurados, no estructurados o semiestructurados.

Definición 2.1. *Se llaman **datos estructurados** con k observaciones y con n variables $\{V_1, V_2, \cdots, V_n\}$ a una matriz $D_{k\times n}$ con entradas en $\mathbb{R}$*

$$\begin{bmatrix} V_1(1) & V_2(1) & \cdots & V_n(1) \\ V_1(2) & V_2(2) & \cdots & V_n(2) \\ \vdots & \vdots & \cdots & \vdots \\ V_1(k) & V_2(k) & \cdots & V_n(k) \end{bmatrix}.$$

Si a cada observación se le asigna una identificación (número de fila o id), cada fila de la matriz se puede ver como la imagen de una función $V : \{1, 2, \cdots, k\} \to B \subseteq \mathbb{R}^k$ y se denotan como

$$\begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n \\ 1 & V_1(1) & V_2(1) & \cdots & V_n(1) \\ 2 & V_1(2) & V_2(2) & \cdots & V_n(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ k & V_1(k) & V_2(k) & \cdots & V_n(k) \end{bmatrix}.$$

El tipo de datos descritos en la definición 2.1 son predominantes en este estudio, ya que se pueden aprovechar las herramientas de la geometría, el álgebra y la estadística sobre las observaciones, pensando en ellas como un conjunto de k puntos en $\mathbb{R}^n$.

Observación 2.1. *Note que en la definición 2.1, las variables en realidad corresponden a variables aleatorias y por lo tanto, una observación está dada por n mediciones diferentes de un fenómeno o evento. Entonces, desde el punto de vista de la entropía de la información, a más información mayor variabilidad o varianza. En conclusión, incrementar el número de variables o el rango de las variables no simplifica el estudio de un fenómeno.*

Definición 2.2. *Sea $(\Omega, \mathcal{B}, P)$ un espacio de probabilidad, con $Y : \Omega \to D$ y D un conjunto discreto. Cada función inyectiva $f : D \to \mathbb{Z}$ induce una variable aleatoria discreta*

$$Y_f = f \circ Y.$$

Note que f debe ser inyectiva para que Y_f describa la misma información de Y sobre el espacio muestral Ω. Además, es posible que en ciertas aplicaciones algunas de las variables sean textos, pero sin pérdida de generalidad, se puede asumir que siempre se puede definir una función f inyectiva del conjunto de textos de la variable a los números naturales y remplazar esta variable por Y_f. Estas variables se conocen como variables categóricas.

Definición 2.3. *Los **Datos semiestructurados** son aquellos datos que no se limitan a campos determinados, pero que contienen marcadores para separar los diferentes elementos.*

$$\begin{bmatrix} \{v_i := V_i(1)\}_{i \in I_1} \\ \{v_i := V_i(2)\}_{i \in I_2} \\ \vdots \\ \{v_i := V_i(k)\}_{i \in I_k} \end{bmatrix}, \text{ donde } I_i \subseteq \{1, 2, 3, ..., n\} \text{ para todo } i \in \{1, 2, 3, ..., k\}.$$

Los datos semiestructurados se suelen representar como diccionarios o arreglos multidimensionales. Para este tipo de datos se acostumbra a usar métodos de datos estructurados y no estructurados, todo depende si los datos permiten ser estructurados.

Definición 2.4. *Los **datos no estructurados** son aquellos datos que carecen de una estructura específica y no se pueden almacenar en una tabla, ni ser representados por matrices o sus variables no son determinables para todas las observaciones. Es decir, un conjunto de datos que no sea estructurado ni semiestructurado es llamado no estructurado.*

Un ejemplo de datos no estructurados es considerar un registro de correos electrónicos; cada correo tiene un asunto, destinatario y cuerpo diferente, que en este caso son las variables de cada observa-

ción. Note que no es posible categorizar o estructurar la información en un arreglo matricial. Para este tipo de datos se suelen usar algoritmos o métodos de aprendizaje profundo (Deep Learning).

2.2. Tipos de datos

Cuando se hace la recolección de información, adicionalmente a las variables medidas en cada observación, se puede agregar una marcación que puede ser una etiqueta (discreta) o valor (continuo) que representa una cualidad, estado o clasificación de la observación. Como esta marcación se hace de forma manual, no siempre es sencillo recolectar esta información. Por lo tanto, el objetivo principal de muchas aplicaciones es tomar un conjunto de datos marcados y lograr una marcación para otro conjunto de datos no marcados.

Definición 2.5. *Sea D un conjunto de datos estructurados. Se llaman **datos supervisados** si para cada observación se puede asignar una etiqueta por medio de una función f cuyo rango está contenido en $\mathbb{R}$. Si no existe esta función se dice que son **datos no supervisados**.*

$$D = \begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n & f \\ 1 & V_1(1) & V_2(1) & \cdots & V_n(1) & f(1) \\ 2 & V_1(2) & V_2(2) & \cdots & V_n(2) & f(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ k & V_1(k) & V_2(k) & \cdots & V_n(k) & f(k) \end{bmatrix}.$$

Un ejemplo sencillo de datos supervisados se presentan cuando el rango de f es binario; se puede interpretar como cumplir o no determinada propiedad. Por ejemplo, un conjunto de pacientes a los que se le miden algunas variables pueden tener una marcación binaria, positivo o negativo en el diagnostico de una determinada enfermedad.

Definición 2.6. *Dado un conjunto de datos estructurados y no supervisados, si existe un algoritmo (o técnica) capaz de definir una función $f : \mathbb{R}^n \to A \subseteq \mathbb{R}$ de tal forma que $f(V_1(i), V_2(i), \cdots, V_n(i))$ sea una marca para la i-ésima observación, entonces se dice que este algoritmo es un **aprendizaje no supervisado**.*

A la función f se le suele llamar **modelo**. El concepto de aprendizaje se concibe como la capacidad de crear un algoritmo que pueda diferenciar o clasificar observaciones. Los algoritmos más robustos pueden cambiar las marcas cuando se les suministran más observaciones, es decir, redefinen la función f basados en la estructura de las observaciones iniciales y nuevas. De manera informal se suele decir que estos algoritmos robustos "aprenden con la experiencia" y estos se pueden usar para distinguir, predecir, segmentar, tomar una decisión o buscar anomalías en las observaciones.

Este tipo de algoritmos son base esencial de la inteligencia artificial.

Definición 2.7. *Sea $D_{k\times n}$ un conjunto de datos estructurados con variables $\{V_1, V_2, \cdots, V_n\}$ y supervisados con una función de marcación f. Además, sea $B_{s\times n}$ un conjunto de datos estructurados con variables $\{V_1, V_2, \cdots, V_n\}$ y no supervisados, denotados por*

$$D = \begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n & f \\ 1 & V_1(1) & V_2(1) & \cdots & V_n(1) & f(1) \\ 2 & V_1(2) & V_2(2) & \cdots & V_n(2) & f(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ k & V_1(k) & V_2(k) & \cdots & V_n(k) & f(k) \end{bmatrix},$$

$$B = \begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n \\ k+1 & V_1(k+1) & V_2(k+1) & \cdots & V_n(k+1) \\ k+2 & V_1(k+2) & V_2(k+2) & \cdots & V_n(k+2) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ k+s & V_1(k+s) & V_2(k+s) & \cdots & V_n(k+s) \end{bmatrix}.$$

*Se llama **aprendizaje supervisado** a un algoritmo que sea capaz de definir una función de marcado $g : \mathbb{R}^n \to A \subseteq \mathbb{R}$ tal que si $g_i = g(V_1(i), V_2(i), \cdots, V_n(i))$ entonces $g_i = f(i)$ para algunos valores de $1 \leq i \leq k$ y además g es un modelo para B.*

$$\begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n & g \\ k+1 & V_1(k+1) & V_2(k+1) & \cdots & V_n(k+1) & g_{k+1} \\ k+2 & V_1(k+2) & V_2(k+2) & \cdots & V_n(k+2) & g_{k+2} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ k+s & V_1(k+s) & V_2(k+s) & \cdots & V_n(k+s) & g_{k+s} \end{bmatrix}.$$

Una forma de asimilar el aprendizaje supervisado, es notar que un algoritmo de este tipo genera una marcación para unos datos no supervisado basados en unos datos supervisados, donde ambos datos tienen las mismas variables. Al conjunto de datos D, se le suele llamar **datos de entrenamiento**, al proceso de construcción de la función f se le llama **entrenamiento** y, la función g se conoce como **modelo** o **predicción** de D sobre B. Además, si g coincide con f en todas las observaciones de D, $i.e.$ si g es una extensión de f, se dice que **el modelo se ajusta a los datos**. Si el modelo no se ajusta a los datos de entrenamiento, se suelen definir estadísticos o indicadores que permiten estimar de forma cuantitativa que tan ajustado está el modelo con los datos, es decir, una medida del error de predicción, lo cual depende de los datos y del modelo, como se verá en la sección 5.1.

Observación 2.2. *En la practica, cuando se usa un algoritmo de aprendizaje supervisado no se entrena usando todas las observaciones de D. Lo usual es dividir las observaciones de D en dos subconjuntos disjuntos de observaciones D_{train} y D_{test}. Se entrena el modelo g con los **datos de entrenamiento** D_{train} y se verifica luego que tantas coincidencias tiene g y f en las observaciones de D_{test}, que suelen ser llamados **datos de testeo**. Esta practica se usa en analítica de datos para evitar el "sobreajuste" (overfitting), pues muchos algoritmos de aprendizaje supervisados pueden hacer que el modelo g se ajuste a los datos D, pero al incorporar nuevas observaciones en D el modelo g no logra hacer una correcta marcación, es decir, pierde "poder de predicción" porque el modelo depende sólo de las observaciones de D y no obedece a la naturaleza de los eventos que generan los datos D.*

Algunos procedimientos que se verán en este trabajo son sensibles a grandes diferencias entre los rangos en las variables de los datos. Por ejemplo, si una variable representa la edad de un individuo y otra variable representa sus ingresos netos en un año, provocará que la matriz de datos tenga dos variables con escalas diferentes y al llevar las observaciones a $\mathbb{R}^n$ estén demasiado dispersas. Una forma de solucionar esto es estandarizando las variables

$$V_i^{'}(j) = \frac{V_i(j) - \mu_i}{\sigma_i},$$

donde μ_i, σ_i son respectivamente la media y la desviación estándar de V_i. De esta manera se tendría una nueva matriz estandarizada $D^{'}$ donde cada variable tiene un rango comparable

$$D^{'} = \begin{bmatrix} Id & V_1^{'} & V_2^{'} & \cdots & V_n^{'} \\ 1 & V_1^{'}(1) & V_2^{'}(1) & \cdots & V_n^{'}(1) \\ 2 & V_1^{'}(2) & V_2^{'}(2) & \cdots & V_n^{'}(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ k & V_1^{'}(k) & V_2^{'}(k) & \cdots & V_n^{'}(k) \end{bmatrix}.$$

En otras palabras, las variables aleatorias $V_1^{'}, V_2^{'}, \cdots, V_n^{'}$ tienen como rango un intervalo al rededor del cero, ya que una forma simple de entender la estandarización de las variables es pensar que $V_i^{'}$ mide en unidades de desviación estándar, la distancia de los valores de V_i a la media μ_i.

3. Preprocesamiento de datos

La recolección de información es susceptible a cambios o errores causados por factores humanos en el proceso de generación de información o por errores de los sistemas de recolección. Estos inconvenientes pueden provocar que los datos suministrados no sean confiables, sean incompletos o posean ruido.

Definición 3.1. *Dado un conjunto de datos, señales, imágenes o, en general, cualquier conjunto de información, se dice que la información contiene ruido si un subconjunto de ésta cumple al menos una de las siguientes condiciones:*

1. *No es de interés. Se decide que "no es de interés" o es irrelevante para el fenómeno de estudio.*
2. *Degrada o distorsiona la información de interés o la contamina.*
3. *Impide o limita el estudio o uso de tal información (estorba).*

Dicho subconjunto se denomina ruido.

En las aplicaciones, se suele encontrar ruido en aquellas variables que no influyen directamente en un fenómeno, en el exceso de variables redundantes o constantes, en valores ausentes o en aquellas variables que son combinaciones lineales de las demás. Todo este ruido puede afectar la convergencia de los algoritmos de aprendizaje o disminuir la capacidad predictiva. Por lo tanto, el preprocesamiento es el acto de garantizar que los datos que se usarán en un entrenamiento estén completos y libres de ruido, tanto como sea posible. En esta sección se abordarán algunos de los procedimientos clásicos para el tratamiento de datos que permiten disminuir el ruido.

3.1. Reducción de dimensiones

Uno de los principales problemas que se presenta en la recolección de información es el exceso de variables medidas dentro de una misma observación. Cuando se tienen demasiadas variables puede presentarse uno de los siguientes problemas:

- Pueden existir variables que sean múltiplos escalares de otras variables o variables altamente correlacionadas. Es decir, se tiene información redundante y en la practica es información

innecesaria. Esto genera ruido en el proceso de aprendizaje, en otras palabras, puede crear un modelo desajustado.

- En ocasiones el número elevado de variables puede implicar que el algoritmo del aprendizaje automatizado consuma más recursos y tiempo para la creación del modelo. Esto es un problema realmente importante en la práctica, ya que muchos modelos se deben calcular rápidamente para usar su poder de predicción o de clasificación en tiempo real.

Para reducir dimensiones en un conjunto de datos estructurados no supervisados se usa comúnmente un método clásico que se conoce como Análisis de Componentes Principales (PCA por sus siglas en inglés). Este método fue abordado por Pearson en el siglo XIX y luego por Hotelling a principios del siglo XX, sin embargo, sólo comenzó a ser reconocido y utilizado con la aparición de los computadores.

Observación 3.1. *Si las variables iniciales no están correlacionadas este método carece de utilidad. En la practica no es una exigencia difícil de cumplir, porque comúnmente cuando se recolecta un cumulo grande de información de un conjunto de observaciones en un entorno, es altamente probable que existan similitudes entre algunas observaciones y por lo tanto, esto implicará correlación entre las variables.*

3.1.1. Análisis de componentes principales (PCA)

El objetivo del Análisis de Componentes Principales es construir un conjunto de variables nuevas $V_1', V_2', \ldots, V_p'$ que sean combinaciones lineales de las variables iniciales $V_1, V_2, \ldots, V_m$ de unos datos D.

$$V_i' = \sum_{j=1}^{m} a_{ij} V_j = w_i \cdot V, \qquad V := \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_m \end{bmatrix}.$$

Este nuevo conjunto de variables $\{V_i'\}_1^p$ cumplen que $1 \leq p < m$. De esta forma el conjunto de datos D' resultante tiene $m - p$ variables menos que los datos iniciales D. El valor de p es exactamente el número de componentes principales que se calculen. De forma simple, se puede decir que el análisis de Componentes Princípiales es una aplicación lineal sobre la matriz de datos iniciales $D_{n\times m}$ a una matriz $D'_{n\times p}$ conservando la mayor cantidad de información del fenómeno almacenada en la matriz inicial D. Esto también implica que su implementación y utilización es simple, sin implicar altos gastos computacionales.

Observación 3.2. *Para la aplicación de este método se asume que las variables de los datos están estandarizadas. El análisis de componentes principales es una técnica matemática que no requiere de la suposición de normalidad de los datos, $i.e\ D^*D = DD^*$, aunque si esta condición se presenta, se puede dar una interpretación más profunda de dichos componentes. En estadística, se habla de normalidad multivariada, es decir, que cada variable es una variable aleatoria con distribución normal [13].*

Suponga que se tiene un conjunto de datos estructurados no supervisados X.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ donde } x_i = (V_1(i), V_1(i), \cdots, V_m(i)) \in \mathbb{R}^m.$$

Consideremos la matriz de covarianza $\Sigma = X^tX$. Note que Σ contiene la varianza entre todos variables aleatorias de X. Estimaremos la transformación lineal con base a la matriz de covarianza Σ.

Definición 3.2. *Se define una primera componente principal de X como $w_1 \in \mathbb{R}^m$, el vector unitario que maximiza la función $f : \mathbb{R}^m \to \mathbb{R}$ definida por*

$$f(w) = w^t \Sigma w.$$

Es decir

$$w_1 = \max_{w \in \mathbb{S}^{m-1}} f(w).$$

Note que w_1 está bien definido, porque f es una función continua sobre un compacto de $\mathbb{R}^m$, que es $\mathbb{S}^{m-1} = \{w \in \mathbb{R}^m : \|w\| = 1\}$, y usando el teorema de Weierstrass, se garantiza que siempre se puede escoger una primera componente principal.

Teorema 3.1. *La primera componente principal de X coincide con el autovector v_1 asociado al mayor autovalor λ_1 de Σ. Además*

$$f(w_1) = \lambda_1$$

Demostración. Referirse a [11]. ■

Observación 3.3. *Note que este tratamiento es independiente de los conceptos estadísticos de las variables de las observaciones en la matriz. Si se hace este mismo desarrollo usando el concepto de*

variable aleatoria, se puede decir que w_1 es un vector que maximiza la varianza de las variables, ya que la matriz Σ es precisamente la matriz de covarianza.

La segunda componente principal se construye pensando en una nueva variable como combinación lineal de las iniciales, siempre que la primera y la segunda componentes principales sean ortogonales. Esto desde un punto de vista estadístico se interpreta como dos variables no correlacionadas.

Definición 3.3. *Se define la segunda componente principal de X como el vector unitario $w_2 \in \mathbb{R}^m$ que satisface que*

$$w_2 = max\left\{f(w) : w_1^t \cdot w = 0 \text{ y } |w| = 1\right\}.$$

Y de forma general, se define la k-ésima componente principal de X como el vector unitario $w_k \in \mathbb{R}^m$ que satisface que

$$w_k = max\left\{f(w) : w_i^t \cdot w = 0 \text{ y } |w| = 1, \quad 1 \leq i \leq k-1\right\}.$$

Desde un punto de vista estadístico, cada componente principal es el vector que maximiza la varianza de las variables iniciales, siempre que la componente sea ortogonal a las anteriores. Además, razonando de manera similar a la prueba Variacional del Teorema Espectral se puede generalizar el teorema 3.1.

Teorema 3.2. *La k-ésima componente principal de X coincide con el autovector normalizado v_k asociado al k-ésimo mayor autovalor λ_k de Σ. Además*

$$f(w_k) = \lambda_k$$

cuando $1 \leq k \leq m$.

Dado unos datos estructurados X de dimensión $n \times m$, el algoritmo de la reducción de dimensiones se puede ver como: estandarización de los datos X, encontrar los primeros p autovectores $w_1, w_2, \cdots, w_p$ de la matriz de covarianza Σ de los datos estandarizados, los cuales coinciden con las p componente principales de los datos, siempre que $p < m$; con esto, luego se define para cada observación $x \in \mathbb{R}^m$ una nueva variable V_i' dada por

$$V_i' = w_i \cdot x,$$

para todo $1 \leq i \leq p$, siendo p el número de componentes príncipes calculadas y la dimensión a la cual se quiere reducir cada observación de los datos. Por lo tanto, el nuevo conjunto de datos X' donde cada observación se redujo a p variables se puede expresar como

$$X' = XW^t,$$

donde

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix},$$

con w_i tomado como vector fila para $1 \leq i \leq p$. Note que W^t es de dimensión $m \times p$, por lo tanto X' es de dimisión $n \times p$. Así, se tiene un nuevo conjunto de datos con n observaciones, pero con p variables. En otras palabras, la matriz W^t es quién define la aplicación lineal que se buscaba inicialmente. Más aún, en el contexto de la estadística podemos calcular la matriz de covarianza Λ de W que está dada por

$$\Lambda = W^t W = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p \end{bmatrix}.$$

Note que Λ es una matriz diagonal, porque las componentes principales no sólo son vectores unitarios, sino que fueron tomados de tal forma que $w_i \cdot w_j = 0$ siempre que $i \neq j$. En en otras palabras, los vectores filas de W son ortonormales. Además, por la construcción se tiene la descomposición en valores singulares de Σ

$$\Sigma = W\Lambda W^t.$$

En la práctica se calculan tantas componentes principales como variables nuevas se quieran tener, el objetivo es que el número de variables se vea reducido sustancialmente. *Por lo tanto, es importante saber cuántas componentes principales se deben tomar y cuánta información del fenómeno se pierde al reducir el número de variables de los datos*. Para esto se usa el porcentaje variabilidad, que se expone a continuación.

3.1.2. Porcentajes de variabilidad

Con la construcción de la aplicación lineal anterior, se puede afirmar desde un punto de vista estadístico que cada autovalor de Σ corresponde a la varianza de la componente w_i , es decir,

$var(w_i) = \lambda_i$, ya que consideramos una matriz normalizada inicialmente. Por lo tanto, si se calculan p componentes principales, tendremos la varianza total de estas componentes expresada como

$$\sum_{i=1}^{p} var(w_i) = \sum_{i=1}^{p} \lambda_i = tr(\Lambda).$$

Por las propiedades del operador tr se tiene que

$$tr(\Lambda) = tr(W^t \Sigma W) = tr(\Sigma W^t W) = tr(\Sigma).$$

Esta última igualdad se debe a la ortogonalidad que se impuso en la definición de las componentes principales, es decir, se tiene que $W^t W = 1$, donde 1 es la identidad. Con lo cual

$$tr(\Lambda) = tr(\Sigma) = \sum_{i=1}^{p} var(V_i).$$

Se concluye entonces que la suma de las varianzas de las variables originales y la suma de las varianzas de las componentes son iguales. Esto nos permite definir un **porcentaje de variabilidad** [13], si solamente se toman p componentes principales

$$PV_{pca} = \frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{m} var(V_i)}.$$

Este porcentaje se puede interpretar como la cantidad de información que se conserva en los datos si se usa la aplicación lineal PCA para reducir la cantidad de variables que tienen los datos iniciales, pasando de m a p variables. Es decir, después de la reducción a p variables se pierde un porcentaje de información que está dado por $(1 - PV_{pca})100$. El porcentaje de perdida de información aceptable lo dará la aplicación y el modelo a usar, pero en general se admite un valor de p si la perdida de información está al rededor del $5\,\%$.

Observación 3.4. *También se suele tomar $p = 2$ o $p = 3$ para visualizar datos de altas dimensiones considerando que sólo se pierde una porción de la información en la proyección.*

Una de las aplicaciones más comunes de este método se encuentra en el preprocesamiento de información de entidades financieras, donde las observaciones corresponden a usuarios o clientes. Dado que para cada individuo existen demasiadas variables medibles, desde la información sociodemográfica hasta su historial transaccional en la entidad, se suele tener demasiadas variables que pueden generar ruido en los datos. Al aplicar reducciones de dimensiones, también se obtiene mayor eficiencia en los algoritmos de modelamiento.

Observación 3.5. *Una forma alternativa para hacer reducción de dimensiones es por medio del Análisis de Componentes Independientes (ICA por sus siglas en inglés). Este es un método usado para la separación de señales con un abanico enorme de aplicaciones, entre las cuales está la reducción de dimensiones. [10].*

3.2. Compleción de matrices

En muchas ocasiones cuando se quiere construir una matriz para lograr un conjunto de datos (supervisados o no), es común encontrarse con la imposibilidad de recuperar algunas de las entradas de la matriz debido a errores humanos, problemas con el proceso del levantamiento de información, condiciones del problema o incluso, porque se está intentando transformar datos semiestructurados a datos estructurados. Este problema se conoce como el problema de **compleción de matrices**, que tomó fuerza a principios de siglo, el cual plantea la dificultad de completar un número de entradas que fueron removidas de una matriz. Este problema no es trivial y existen diversas formas de atacarlo desde un punto de vista matemático, siempre que se establezcan algunas condiciones sobre el número de entradas ausentes, dimensiones de la matriz, rango de la matriz u otra condición sobre las características algebraicas de la matriz.

Los matemáticos Terence Tao, Emmanuel J. Candès y Benjamin Recht son referentes en el estudio de este problema y son estos dos últimos, quienes en 2008 establecieron una solución al problema para matrices rectangulares usando optimización convexa, siempre que la matriz sea una matriz genérica de bajo rango. La solución presentada estaba motivada por dos aplicaciones fundamentales que son referencia en la industria:

- Problema de Netflix: en el área de los sistemas de recomendación, los usuarios presentan calificaciones en un subconjunto de entradas en una base de datos, y el vendedor proporciona recomendaciones basadas en las preferencias del usuario. Debido a que los usuarios sólo califican algunos artículos, es deseable inferir su preferencia para elementos sin clasificar. Una instancia especial de este problema es el famoso problema de Netflix. Usuarios (filas de la matriz de datos) tienen la oportunidad de calificar películas (columnas de la matriz de datos) pero los usuarios por lo general, califica muy pocas películas, de modo que hay muy pocas entradas observadas dispersas en esta matriz de datos. Sin embargo, es deseable completar esta matriz para que Netflix pueda recomendar títulos que realmente los usuario estén dispuestos a ver. En este caso, la matriz de datos de todas las clasificaciones de usuarios es de bajo rango, porque se trabaja bajo la hipótesis que sólo unos pocos factores contribuyen a los gustos o preferencias de un individuo, en otras palabras, los gustos de los usuarios se pueden agrupar en unos cuantos segmentos de preferencias.
- Triangulación a partir de datos incompletos: suponga se tiene información parcial sobre las distancias entre ciertos objetos y se quiere reconstruir la geometría de baja dimensión que

describen sus ubicaciones. Por ejemplo, se puede tener una red de sensores de red inalámbrica de baja potencia dispersos aleatoriamente en una región. Suponga que cada sensor sólo tiene la capacidad de construir estimaciones de distancia basadas en lecturas de intensidad de señal de sus compañeros sensores más cercanos. De estas estimaciones de distancia incompletas de los sensores se puede formar una matriz de distancia parcialmente observada. Luego se puede estimar la matriz de distancia verdadera cuyo rango será igual a dos si los sensores son ubicadas en un plano o tres si están ubicadas en un espacio tridimensional. Si la cantidad de sensores es considerablemente grande demandará mucho esfuerzo construir la matriz completa. En este caso, sólo es necesario observar algunas distancias por nodo para tener suficiente información para reconstruir la matriz de distancias de los objetos usando la compleción de matrices.

Más allá de las aplicaciones que motivaron este trabajo, existe una infinidad de aplicaciones en la ciencia y la industria. A continuación se expondrá una metodología para compleción de matrices basada en [4].

Observación 3.6. *La exigencia que se impone sobre el rango de la matriz para que el algoritmo de reconstrucción funcione está asociada a la convergencia y eficiencia computacional del algoritmo.*

3.2.1. Matrices reconstruibles

No todas las matrices de bajo rango se pueden reconstruir. Considere, por ejemplo

$$M = e_1 e_n^t = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

donde e_i es el i-ésimo vector de la base canónica en el espacio euclidiano. Esta matriz tiene un 1 en la esquina superior derecha y todas las otras entradas son 0. Claramente esta matriz no puede ser recuperada a partir de un muestreo de sus entradas a menos que se conozca de antemano la totalidad de sus entradas. La razón es que para la mayoría de los conjuntos de muestreo, sólo se tendrían ceros y no hay forma de adivinar que la matriz no es cero. Por lo tanto, no es posible recuperar todas las matrices de bajo rango a partir de un conjunto de entradas muestreadas. Para saber que matrices de bajo rango se pueden recuperar usando optimización convexa se puede usar un resultado fundamental del álgebra lineal.

Teorema 3.3. *Toda matriz $M \in \mathbb{R}^{n\times m}$ admite una descomposición en valores singulares (DVS)*

$$M = U\Sigma V^t$$

Usando esta descomposición, la matriz M puede escribirse como combinación lineal de matrices de rango 1:

$$M = \sum_{i=1}^{r} \sigma_i u_i v_i^t$$

donde u_i y v_i son los vectores singulares izquierdos y derechos respectivamente, y σ_i es el i-ésimo valor singular de $M^t M$.

Entonces se puede pensar en una matriz genérica de bajo rango de la siguiente manera: M es una matriz **genérica de bajo rango** r, si $r \ll max(m, n)$ y se puede afirmar que la familia de vectores singulares izquierdos de la DVS $\{u_i\}_{i=1}^r$ está distribuida uniformemente al azar en la familia de todos los vectores ortonormales, y de manera similar para la familia de vectores singulares derechos $\{v_i\}_{i=1}^r$.

El trabajo de Candès y Recht se concentra en reconstruir matrices genéricas de bajo rango. En la practica no es difícil cumplir esta condición ya que el gran tamaño de las matrices junto con la asociación de las observaciones con comportamientos o fenómenos que dependan de pocas opciones de variabilidad (individuos, gustos, patrones, etc.) hacen cumplir este requerimiento.

3.2.2. Conjunto de muestreo

Claramente, no se puede esperar reconstruir cualquier matriz M de bajo rango, incluso de rango 1, si el muestreo evita alguna columna o fila de M. Por ejemplo, si definimos la matriz M de rango 1 como

$$M = xy^t$$

con $x, y \in \mathbb{R}^n$, entonces

$$M_{ij} = x_i y_j.$$

Si se hace un muestreo de la matriz M evitando la primera fila, no existe forma de adivinar las entradas de la matriz, ya que no existe ninguna particularidad o preferencia de la primera fila con las demás, es decir, que ningún método podría estimar la estrada x_1 del vector x ya que existen infinitas posibilidades sin restricciones para dicho valor.

Formalmente, suponga que se tiene una matriz $M \in \mathbb{R}^{n \times m}$ genérica de bajo rango, entonces diremos que

$$M^* = \{(m, i^*, j^*) : (i^*, j^*) \in \{1, 2, \cdots, n\} \times \{1, 2, \cdots, m\}, m = M_{i^* j^*}\}$$

es el conjunto de entradas observadas de la matriz M (o muestreo de M) y cumple que $|M^*| < mn$. Llamamos $\Omega = \{(i, j) : (m, i, j) \in M^*, m \in \mathbb{R}\}$ al conjunto de ubicaciones de las entradas observadas. Se establece que la probabilidad de completar la matriz M es directamente proporcional al tamaño del conjunto $|M^*| = |\Omega|$, siempre que Ω sea un conjunto uniformemente aleatorio, es decir, que las entradas del muestreo no tienen predilección o sesgo hacia una fila o columna.

3.2.3. Algoritmo de compleción

Supongamos que se tiene una matriz genérica $M \in \mathbb{R}^{n\times m}$ de bajo rango, de dimensiones particularmente grandes. Consideremos un muestreo M^* de la matriz M, con un conjunto Ω de índices de las entradas observadas, entonces definimos $\bar{\Omega}$ como el conjunto de índices de las entradas no observadas de la matriz M, por lo tanto $p = \left|\bar{\Omega}\right| = nm - |\Omega|$ es el número de entradas no observadas en M.

Denotaremos por $M^*(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_p)$ a la matriz de valores reales $n \times m$ que se obtiene al completar con orden lexicográfico los p valores no observados en el muestreo M^* con la $p-$tupla $(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_p)$. Note que si se establece que los valores ausentes de la matriz M están ordenados en la $p-$tupla $(\mu_1, \mu_2, \mu_3, \cdots, \mu_p)$, entonces $M = M^*(\mu_1, \mu_2, \mu_3, \cdots, \mu_p)$.

Definición 3.4. *Sea $X \in \mathbb{R}^{n\times m}$ una matriz de rango r. Definimos la norma nuclear de X como*

$$\|X\|^* = \sum_{k=1}^{r} \sigma_k(X),$$

donde $\sigma_k(X)$ denota el k-ésimo valor singular de X.

Note que la norma nuclear de X está bien definida porque X tiene exactamente r valores singulares. Así, cuanto menor sea el rango de la matriz, menor será el gasto computacional necesario para calcular la norma nuclear.

Definición 3.5. *Sea $M \in \mathbb{R}^{n\times m}$ con un muestreo de entradas M^*, entonces este muestreo tiene asociada una función $\Upsilon : \mathbb{R}^p \to \mathbb{R}$ definida por*

$$\Upsilon(x) = \|M^*(x)\|^*.$$

Cabe resaltar que el rango de la matriz $M^*(x)$ dependerá del vector $x \in \mathbb{R}^p$. Sin embargo, se espera que si la matriz es de bajo rango y el muestreo M^* es suficientemente grande o representativa, entonces $M^*(x)$ no supere el rango de la matriz M. Por lo tanto, Candès y Recht concluyen en su trabajo que la mejor matriz que aproxima a la matriz genérica de bajo rango M a partir de un muestreo es $M^*(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_p)$ donde

$$(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_p) = \min_{x\in\mathbb{R}^p} \Upsilon(x).$$

Observación 3.7. *El trabajo de Candès y Recht no sólo demuestra que la matriz que se obtiene al minimizar la norma nuclear es una buena **aproximación heurística**, sino que también muestran que este problema es una optimización convexa, ya que uno de los principales resultados es la convexidad de la función Υ y se puede optimizar de manera eficiente a través de programación*

semidefinida. Por simplicidad, en este trabajo se presentan los resultados de forma práctica, definiendo la función Υ como único objeto de estudio, ya que cualquier aproximación a un mínimo local de la función Υ dará la compleción heurística de la matriz M a partir de M^*.

En la práctica, es indispensable saber que tan acertada es la reconstrucción de la matriz incompleta. Una forma de realizar esta medición es tomar el muestreo M^* de una matriz (o submatriz) M de orden $n \times m$, de la que se conozcan todas sus entradas. Se hace la compleción $M^*(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_p)$ de la matriz M desde el muestreo M^*. Luego se define una tolerancia al error ϵ en las entradas de la matriz. A continuación se hace la diferencia de matrices

$$A = M - M^*(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_p).$$

Se calculan $\epsilon(A)$ como el número de entradas de A que en valor absoluto son mayores a la tolerancia ϵ. Por lo tanto una medida del error de compleción de la matriz M será

$$\frac{\epsilon(A)}{nm}.$$

Cuanto más se acerque este cociente a 0, mejor será la compleción de la matriz. Si el cociente se acerca a 1, se debe imponer una tolerancia mayor o aceptar que la matriz M no es propensa a la reconstrucción.

4. Segmentación

4.1. Problema de identificación de clases

Considere un conjunto de datos D estructurados no supervisados

$$D = \begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n \\ 1 & V_1(1) & V_2(1) & \cdots & V_n(1) \\ 2 & V_1(2) & V_2(2) & \cdots & V_n(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ k & V_1(k) & V_2(k) & \cdots & V_n(k) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_k \end{bmatrix}.$$

Además, diremos que m clases $C = \{C_1, C_2, \cdots, C_m\}$ para los datos D, son m conjuntos disjuntos tales que

$$\bigcup_{i=1}^{m} C_i = \{x_1, x_2, \cdots, x_k\}.$$

Definición 4.1. *Sea D un conjunto datos estructurados no supervisados con m clases C. Una función $\varphi : \{1, 2, \cdots, k\} \to C$ es llamada una clasificación para los datos D sobre el conjunto de clases C. Cada clasificación φ para D induce un conjunto de datos supervisado*

$$D_\varphi = \begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n & \varphi \\ 1 & V_1(1) & V_2(1) & \cdots & V_n(1) & \varphi(1) \\ 2 & V_1(2) & V_2(2) & \cdots & V_n(2) & \varphi(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ k & V_1(k) & V_2(k) & \cdots & V_n(k) & \varphi(k) \end{bmatrix}.$$

Si $m = 2$, se dice que φ es una clasificación binaria.

Note que para cada una de las k observaciones existe m maneras posibles de asignarle una clase en C, entonces en total existen m^k funciones de clasificación. En principio se espera que las aplicaciones presenten matrices de datos con un número de observaciones considerables, provocando que la familia de clasificaciones de un conjunto de datos sea imposible de estudiar por inspección

o por tanteo. Las clasificaciones se construyen a partir de propiedades geométricas, estadísticas o topológicas, para que se ajuste a determinado fenómeno o evento.

Encontrar una función de clasificación acorde a un evento es un problema común en las aplicaciones, por ejemplo, cuando una empresa quiere segmentar sus clientes en diferentes categorías para ofrecer productos diferenciados por segmento, usando el historial de consumo del cliente o su condición socioeconómica. En principio, el problema matemático se puede reducir a definir una clasificación matemáticamente coherente a un conjunto de observaciones que se pueden ver como un conjunto de puntos en $\mathbb{R}^n$. Los métodos o algoritmos que definen funciones de clasificación se llama **algoritmos de agrupamiento** y a los datos supervisados D_φ resultantes de la clasificación se conocen como **segmentación de** D.

Existe muchos algoritmos de agrupamiento, donde cada algoritmo es desarrollado de forma heurística y cada algoritmo tiene sus ventajas y sus debilidades. En principio se puede decir que predominan tres grandes familias de algoritmos de agrupamiento:

- Agrupamientos geométricos: son algoritmos que realizan la clasificación de un conjunto de puntos en $\mathbb{R}^n$ usando propiedades geométricas para encontrar centros de acumulación de puntos, basados en la premisa de que los puntos con la misma clasificación están cercanos, siempre que esta cercanía se mida bajo una métrica preestablecida. Estos son los algoritmos que poseen poco gasto computacional en la mayoría de los casos, pero no funcionan bien cuando se intentan clasificar observaciones con alta variabilidad o con demasiada dispersión.

- Agrupamientos topológicos: estos algoritmos hacen parte del análisis de datos topológicos (TDA por sus siglas en inglés), usualmente usado cuando las observaciones de los datos tienen ruido, están incompletas o con alta variabilidad. La herramienta principal de estos algoritmos es la homología persistente, una adaptación de la homología para los datos, vistos como una nube de puntos. El problema con estos algoritmos está en su complejidad, y en algunos casos, implican un gasto computacional fuerte. También, tradicionalmente se ha considerado que todos los algoritmos basados en grafos hacen parte de esta familia, pero los algoritmos basados en grafos son tan diversos que se podrían considerar como una familia de algoritmos independientes.

- Agrupamientos estadísticos: estos algoritmos establecen una clasificación asumiendo que las observaciones a clasificar en una misma categoría provienen de un mismo fenómeno, por lo tanto, las observaciones con la misma clasificación comparten los mismos estadísticos, por ejemplo, es común asumir en algunos de estos algoritmos que existen m distribuciones que son generadoras de las observaciones, por lo cual, describir estas distribuciones generará una clasificación para las observaciones. Esta familia de algoritmos es muy grande y posee algoritmos basados en estadística Bayesiana y Frecuentista. Estos algoritmos pueden fallar si los datos tienen puntos de acumulación muy cercanos o si existe alguna simetría geométrica en la nube de puntos.

En este trabajo se abordan algunos de los métodos clásicos de agrupamiento que sirven como base para las aplicaciones y el estudio de los demás algoritmos de agrupamiento.

4.2. K-Medias

El algoritmo K-Medias (K-means en inglés) fue creado por MacQueen en 1967 y es el algoritmo de agrupamiento comúnmente conocido por su simplicidad y eficiencia. Este algoritmo está diseñado para clasificar un conjunto de observaciones de una matriz de datos no supervisados dentro de k grupos, donde dicho valor k se establece a priori. Este algoritmo busca exactamente k centros de acumulación (centroides) y a cada observación, se le asocia el centroide más cercano, es decir, que los centroides son los representantes de las clases de clasificación. Dado que cada centroide se calcula como la media de los puntos de la clase que representa, el algoritmo recibe el nombre de K-Medias. Una de las restricciones más relevantes de este método está en las observaciones extremas, es decir, que el cálculo de los centroides es sensible a datos atípicos o distantes de la nube de puntos.

Supongamos que se tiene un conjunto de datos estructurados no supervisados D con n observaciones y m variables denotado por

$$D = \begin{bmatrix} \vec{x_1} \\ \vec{x_2} \\ \vdots \\ \vec{x_n} \end{bmatrix},$$

donde cada observación $\vec{x_i} \in \mathbb{R}^m$. Tomemos inicialmente un conjunto de k centroides de forma aleatoria en la cercanía de las observaciones, que pueden coincidir o no con las observaciones de D. Los centros iniciales los denotamos por la k-tupla

$$M(0) = (\vec{m_1}^{(0)}, \vec{m_2}^{(0)}, ..., \vec{m_k}^{(0)})$$

donde $\vec{m_i}^{(0)} \in \mathbb{R}^m$ para cada $i \in \{1, 2, \cdots, k\}$.

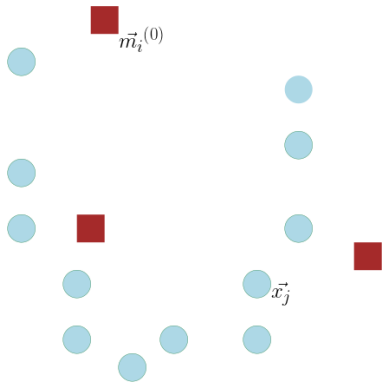


Figura 4-1.: Centroides iniciales.

Observación 4.1. *Se conoce como inicialización de Forgy al método que calcula los k centroides iniciales del algoritmo como k vectores aleatorios tomados de los datos iniciales $\{\vec{x_1}, \vec{x_2}, \cdots, \vec{x_n}\}$.*

Con estos centroides iniciales se puede hacer una partición de las observaciones dadas conocida como **diagrama de Voronoi**. La partición está dada por los conjuntos $S_1^{(0)}, S_2^{(0)}, ..., S_k^{(0)}$ donde

$$S_i^{(0)} = \left\{ \vec{x_p} \in D : \left\| \vec{x_p} - \vec{m}_i^{(0)} \right\| \leq \left\| \vec{x_p} - \vec{m}_j^{(0)} \right\|, 1 \leq j \leq k \right\}.$$

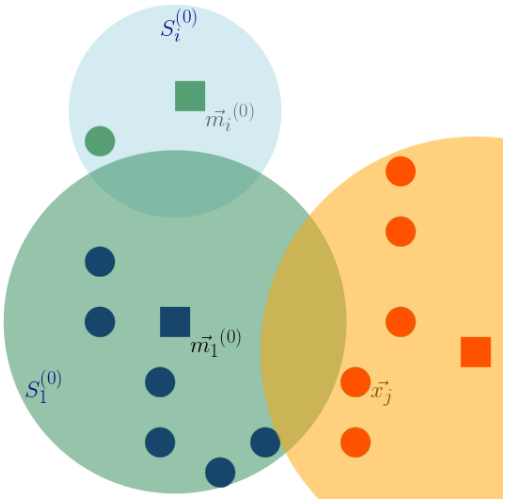


Figura 4-2.: Diagrama de Voronoi inicial.

A priori, los conjuntos $S_1^{(0)}, S_2^{(0)}, ..., S_k^{(0)}$ no son disjuntos. Además, note que cada partición contiene un centroide que la representa. A continuación se recalculan los centroides, tomando como nuevo centroide la media de los vectores de cada partición

$$\vec{m_i}^{(1)} = \frac{1}{\left|S_i^{(0)}\right|} \sum_{\vec{x_j} \in S_i^{(0)}} \vec{x_j},$$

con $1 \leq i \leq k$. Así se obtiene una nueva k-tupla de centroides $M(1) = (\vec{m_1}^{(1)}, \vec{m_2}^{(1)}, ..., \vec{m_k}^{(1)})$.

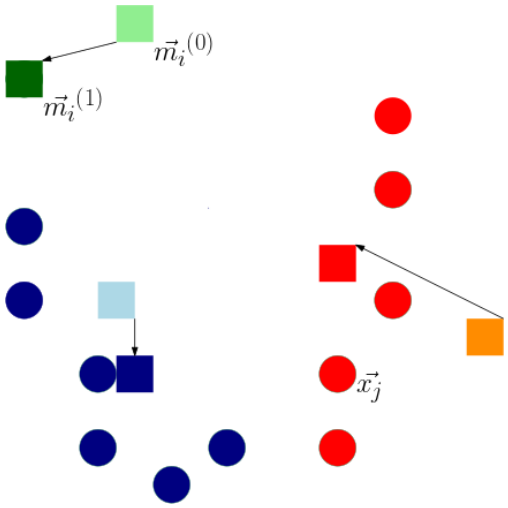


Figura 4-3.: Recalculo de centroides.

Ahora se puede repetir el proceso para el conjunto de centroides $M(1)$ y su diagrama de Voronoi generará un nuevo conjunto de centroides $M(2)$. Se continua repitiendo el proceso hasta que se logre obtener una ϵ-convergencia en el conjunto de centroides $M(w)$, esto quiere decir que para el conjunto $M(w)$ se cumple que

$$\left\| m_i^w - m_i^{w-1} \right\| < \epsilon.$$

Esto para cada centroide, $i \in \{1, 2, \cdots, k\}$. Esta convergencia representa un mínimo local que dependerá de $M(0)$, por lo tanto si no se logra una ϵ-convergencia después de un número de iteraciones considerablemente grande, se debe reiniciar el algoritmo cambiando $M(0)$ o aumentando la tolerancia ϵ de la convergencia.

Observación 4.2. *No se establecen condiciones suficientes para garantizar una convergencia. En la practica, si falla la convergencia se suelen tomar centroides iniciales diferentes y repetir el proceso. Si esto no funciona, se suele estandarizar las variables de las observaciones, hacer reducción de dimensiones, cambiar de métrica o eliminar los valores extrémales (de normal mucho mayor a las demás observaciones) que puedan estar afectando la convergencia.*

Si se logra una ϵ-convergencia, no hay garantías para que los conjuntos finales $S_1^{(w)}, S_2^{(w)}, ..., S_k^{(w)}$ sean disjuntos, porque puede ocurrir que una observación sea equidistante a dos o más centroides.

Existen diferentes formas de lograr una partición disjunta de los datos, una de ella es dejar a las observaciones que están en la intersección de los conjuntos sólo en el conjunto de la partición con más elementos o en el que posea mayor varianza, incluso puede escogerse de forma aleatoria, esto dependerá del tipo de aplicación o del origen de las observaciones. Por lo tanto, podemos decir sin perdida de generalidad, que el algoritmo arroja una partición disjunta $S_1^{(w)}, S_2^{(w)}, ..., S_k^{(w)}$ para el conjunto

$$\left\{\vec{x_1}, \vec{x_2}, \cdots, \vec{x_n}\right\}.$$

Es decir,

$$S_i^{(w)} \cap S_j^{(w)} = \varnothing; \quad i \neq j; \quad i, j \in \{1, 2, \cdots, k\}.$$

Así, podemos definir una clasificación $\varphi : \{1, 2, \cdots, n\} \to \{1, 2, \cdots, k\}$ para los datos D dada por

$$\varphi(i) = j \Leftrightarrow x_i \in S_j^{(w)}.$$

Observación 4.3. *Este algoritmo para calcular los centroides de un conjunto de datos se conoce como* ***algoritmo de Lloyd****. Además, la convergencia del algoritmo no depende de la métrica Euclídea, por lo tanto, la ϵ-convergencia se puede lograr con diferentes métricas. En algunas aplicaciones, será conveniente definir otra métrica para medir la distancia de las observaciones a los centroides y la distancia entre los mismos centroides. Un ejemplo en el que es conveniente cambiar de métrica, es en el agrupamiento de series de tiempo.*

4.3. K-vecinos más cercanos

Este es un método clásico del aprendizaje supervisado que da solución al problema de clasificación de una observación nueva que se introduce en un conjunto de observaciones previamente clasificadas. Es un algoritmo paramétrico, es decir, que es necesario definir una parámetro k antes de efectuar el método. El parámetro k determinará cuantas observaciones cercanas a la nueva observación se van a considerar para la nueva clasificación. Además, no se hace ninguna suposición o desarrollo estadístico, geométrico o topológico para ver la afinidad de la nueva observación con respecto a las demás observaciones de forma global, es por esto, que en ocasiones el algoritmo posee una desventajas en las aplicaciones, ya que este algoritmo supone que la clasificación se hace de forma únicamente local [15].

Suponga que se tiene un conjunto de datos supervisados D por medio una función de clasificación $\varphi : \{1, 2, \cdots, n\} \to C$. Podemos denotar de forma simple

$$D = \begin{bmatrix} \vec{x}_1 & \varphi(1) \\ \vec{x}_2 & \varphi(2) \\ \vdots & \vdots \\ \vec{x}_n & \varphi(n) \end{bmatrix},$$

donde $\vec{x}_i \in \mathbb{R}^m$ representa la i-ésima observación con m variables. Sea $\vec{x} \in \mathbb{R}^m$ una observación nueva no clasificada, es decir, una observación que no está en D. Se quiere construir una extensión $\bar{\varphi} : \{1, 2, \cdots, n+1\} \to C$ de φ que sea una clasificación para el nevo conjunto de datos $\bar{D}$, es decir,

$$\bar{D} = \begin{bmatrix} \vec{x}_1 & \bar{\varphi}(1) \\ \vec{x}_2 & \bar{\varphi}(2) \\ \vdots & \vdots \\ \vec{x}_n & \bar{\varphi}(n) \\ \vec{x} & \bar{\varphi}(n+1) \end{bmatrix} = \begin{bmatrix} \vec{x}_1 & \varphi(1) \\ \vec{x}_2 & \varphi(2) \\ \vdots & \vdots \\ \vec{x}_n & \varphi(n) \\ \vec{x} & \bar{\varphi}(n+1) \end{bmatrix}.$$

Sea k un entero positivo tal que $1 < k < n$ y consideremos $d : \mathbb{R}^m \times \mathbb{R}^m \to [0, +\infty)$ una métrica para $\mathbb{R}^m$. Consideremos el conjunto τ_k formado por k observaciones de D

$$\tau_k = \{\vec{x}_{i_1}, \vec{x}_{i_2}, \cdots, \vec{x}_{i_k}\}$$

tal que

$$\begin{gathered} d(\vec{x}_{i_1}, \vec{x}) = min(\{d(\vec{x}_i, \vec{x}) : 1 \leq i \leq n)\}), \\ d(\vec{x}_{i_2}, \vec{x}) = min(\{d(\vec{x}_i, \vec{x}) : \vec{x}_i \notin \{\vec{x}_{i_1}\}, 1 \leq i \leq n\}), \\ d(\vec{x}_{i_3}, \vec{x}) = min(\{d(\vec{x}_i, \vec{x}) : \vec{x}_i \notin \{\vec{x}_{i_1}, \vec{x}_{i_2}\}, 1 \leq i \leq n\}), \\ \vdots \\ d(\vec{x}_{i_k}, \vec{x}) = min(\left\{d(\vec{x}_i, \vec{x}) : \vec{x}_i \notin \left\{\vec{x}_{i_1}, \vec{x}_{i_2}, \cdots, \vec{x}_{i_{k-1}}\right\}, 1 \leq i \leq n\right\}). \end{gathered}$$

En otras palabras, τ_k contiene las k observaciones de D más cercanas a x bajo la métrica d. Se suele decir que τ_k son los k-vecinos más cercanos a x bajo la métrica d. Si k es un entero que lo permite, se define $c_{\tau_k} \in C$ como la clase de mayor ocurrencia en la k-tupla

$$(\varphi(i_1), \varphi(i_2), \cdots, \varphi(i_k)).$$

Es decir, c_{τ_k} es la clasificación con mayor ocurrencia en los k-vecinos más cercanos a x. Por lo tanto se obtiene la clasificación $\bar{\varphi}$ para $\bar{D}$ dada por

$$\bar{D} = \begin{bmatrix} \vec{x}_1 & \bar{\varphi}(1) \\ \vec{x}_2 & \bar{\varphi}(2) \\ \vdots & \vdots \\ \vec{x}_n & \bar{\varphi}(n) \\ \vec{x} & \bar{\varphi}(n+1) \end{bmatrix} = \begin{bmatrix} \vec{x}_1 & \varphi(1) \\ \vec{x}_2 & \varphi(2) \\ \vdots & \vdots \\ \vec{x}_n & \varphi(n) \\ \vec{x} & c_{\tau_k} \end{bmatrix}.$$

Note que esta clasificación de $\bar{D}$ depende de la métrica d y del parámetro k que define la vecindad de x. Como se puede ver en ejemplo de la figura **4-4**, $c_{\tau_3} \neq c_{\tau_8}$. Además, no todo valor de k permitirá una extensión de φ, ya que no se puede garantizar que siempre haya una clase de mayor ocurrencia para todas las vecindades de x. En la practica, se suele usar diferentes valores de k y comparar la clasificación que arroja el algoritmo con cada parámetro, antes de elegir un valor final para la constante.

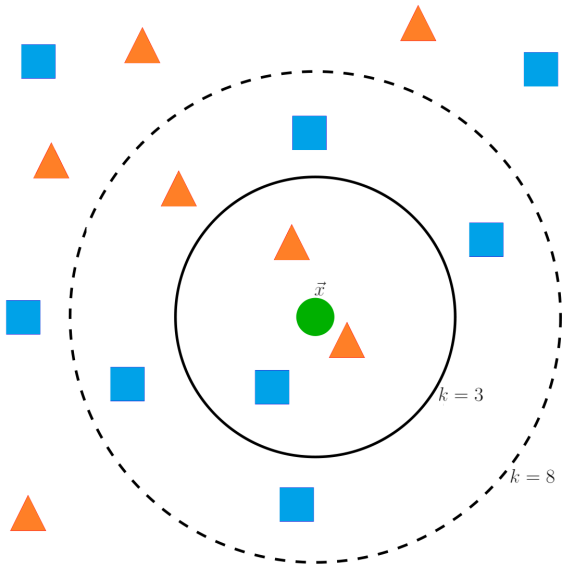


Figura 4-4.: Ejemplo de $\bar{D}$ para puntos en el plano con clasificación binaria. con $k = 3$ y $k = 8$.

4.4. Verosimilitud

La forma estadística clásica para definir una función de clasificación es suponer que las observaciones provienen de un conjunto de funciones de distribución y asignarle a cada observación una distribución de origen. Aunque este problema se puede resolver de forma general, con el objetivo de ilustrar la solución estadística clásica, consideremos unos datos no supervisados D de una sola variable con rango en los reales positivos. Además se considera un conjunto de clases $C = \{0, 1\}$. Se pretende establecer una clasificación binaria para

$$D = \begin{bmatrix} V_1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Consideremos la función de distribución de probabilidad exponencial $exp(\lambda_1)$ cuya función de densidad de probabilidad está dada por

$$d_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & si \quad 0 \leq x \\ \\ 0 & si \quad x < 0 \end{cases}.$$

La hipótesis inicial de la cual parte esta construcción, es suponer que cada una de las observaciones provienen de alguna de las siguientes distribuciones exponenciales:

$$exp(\lambda_1),$$
$$exp(\lambda_2),$$

tal que, una observación pertenecerá a la clase 0 si proviene de la distribución con parámetro λ_1 o pertenecerá a la clase 1 si proviene de la distribución con parámetro λ_2. Supongamos por simplicidad que las observaciones de D son no negativas. Así, podemos definir las funciones de densidad

$$d_{\lambda_1}(x) = \lambda_1 e^{-\lambda_1 x},$$
$$d_{\lambda_2}(x) = \lambda_2 e^{-\lambda_2 x}.$$

Llamaremos Ω al conjunto de todas las funciones de clasificación que se pueden definir para los datos D sobre el conjunto de clases C, el cual contiene 2^n elementos. Por lo tanto, si $\varphi \in \Omega$ es una función de clasificación, ésta induce un vector de clasificación

$$z = (\varphi(1), \varphi(2), \cdots, \varphi(n)).$$

Así, se puede definir la función de verosimilitud como

$$L(D, \varphi, \lambda_1, \lambda_2) = \prod_{i=1}^{n} \left[(1 - \varphi(i)) d_{\lambda_1}(x_i) + \varphi(i) d_{\lambda_2}(x_i)\right].$$

Note que para cada factor del producto se tiene una suma y uno de los dos términos de la suma se hace cero, es decir, $(1-\varphi(i))d_{\lambda_1}(x_i)=0$ si la i-ésima observación pertenece a la clase 1 o $\varphi(i)d_{\lambda_2}(x_i)=0$ si la i-ésima observación pertenece a la clase 0. Además, la función L, bajo la hipótesis inicial, estima la probabilidad de que las observaciones de los datos D provengan de las distribuciones exponenciales con parámetros λ_1,λ_2 si están clasificadas mediante φ.

Teorema 4.1. *Si se tiene unos datos no supervisados D con observaciones de una sola variable real positiva y se fija una función de clasificación $\varphi\in\Omega$, entonces la función de verosimilitud L, como función de λ_1,λ_2, alcanza el máximo en*

$$\bar{\lambda}_1^{(\varphi)}=\frac{\sum\limits_{\varphi(i)\neq 0}1}{\sum\limits_{\varphi(i)\neq 0}x_i},$$

$$\bar{\lambda}_2^{(\varphi)}=\frac{\sum\limits_{\varphi(i)\neq 1}1}{\sum\limits_{\varphi(i)\neq 1}x_i}.$$

Demostración. Ver [3]. ■

Por el teorema anterior, podemos decir que si la clasificación de D es φ, entonces lo más probable es que las distribuciones de las cuales provienen los datos sean

$$exp(\bar{\lambda}_1),$$

$$exp(\bar{\lambda}_2).$$

Por lo tanto, se puede definir la función $L^{'}:\Omega\to\mathbb{R}$ por

$$L^{'}(\varphi)=L(D,\varphi,\bar{\lambda_1}^{(\varphi)},\bar{\lambda_2}^{(\varphi)}).$$

Desde un punto estadístico, $L^{'}(\varphi)$ estima la probabilidad de que los datos D tengan la marcación φ, bajo la hipótesis inicial. Así, se puede asegurar que la función de clasificación más probable para los datos D, es aquella que maximiza la función $L^{'}$. En otras palabras, se puede afirmar que la mejor clasificación posible para los datos bajo la hipótesis inicial es φ, donde alcanza el máximo $L^{'}$.

Observación 4.4. *La debilidad de este razonamiento radica en el crecimiento exponencial de las evaluaciones necesarias para encontrar el máximo de la verosimilitud, 2^n. Hay métodos más elaborados, con mayor simplicidad en el cálculo si se tiene un número elevado de observaciones y*

además, no requieren que las observaciones de los datos sean unidimensionales, ni que la clasificación sea binaria. Uno de estos algoritmos es el Modelo Mezcla Gaussiana (GMM por sus siglas en inglés), el cual es un método heurístico comúnmente usado para realizar agrupamientos de forma estadística, suponiendo que las observaciones provienen de m distribuciones multivariadas, con la ventaja de poder definir un estadístico bayesiano (BIC, bayesian information criterion) que permite estimar el ajuste del modelo.

El método de encontrar una clasificación optimizando la función de verosimilitud, se conoce como **Máxima Verosimilitud**.

4.5. Mean shift

Este es un método estadístico y geométrico de agrupamiento en el cual se busca clasificar los puntos a partir de la distancia y la densidad. Así, se relaciona cada observación con el centro de densidad más cercano, lo que genera una clasificación. Este método fue presentado en 1975 por Fukunaga y Hostetler [8].

Definición 4.2. *Decimos que la función $K : \mathbb{R}^n \to \mathbb{R}$ es un kernel en $\mathbb{R}^n$ si existe una función real k no negativa, par e integrable que satisface*

$$\int_{-\infty}^{+\infty} k(x)dx = 1,$$

y tal que para todo $X \in \mathbb{R}^n$

$$K(X) = k(\|X\|).$$

La función k se llama soporte del kernel.

Está definición está basada en [5]. Note que si K es un kernel para $\mathbb{R}^n$ entonces

$$\sup_{X\in\mathbb{R}^n} K(X) < \infty,$$

y

$$\lim_{\|X\|\to\infty} K(X) = 0.$$

Estas propiedades ayudan a que algunos algoritmos iterativos que usan estimación de densidad basada en un kernel puedan encontrar un punto de convergencia. Además, note que existe una familia infinita de kerneles, ya que si $k(x)$ es el soporte de un kernel, entonces $\lambda k(\lambda x)$ también es un soporte de otro kernel si $\lambda > 0$.

Nombre del kernel	**Soporte**
Rectangular	$k(x) = \frac{1}{2}$ si $\|x\| < 1$, 0 en otro caso.
Triangular	$k(x) = (1 - \|x\|)$ si $\|x\| < 1$, 0 en otro caso.
Cosenoidal	$k(x) = \frac{\pi}{4} cos(\frac{\pi}{2}x)$
Gaussiano	$k(x) = (\sqrt{2\pi}e^{\frac{1}{2}x^2})^{-1}$
Logística	$k(x) = (e^x + 2 + e^{-x})^{-1}$

Tabla 4-1.: Soportes de kerneles comúnmente usados

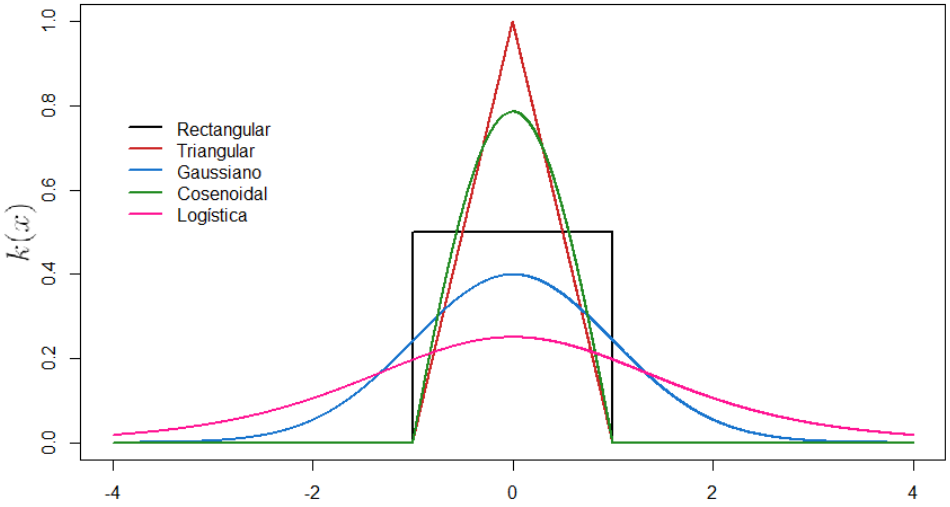
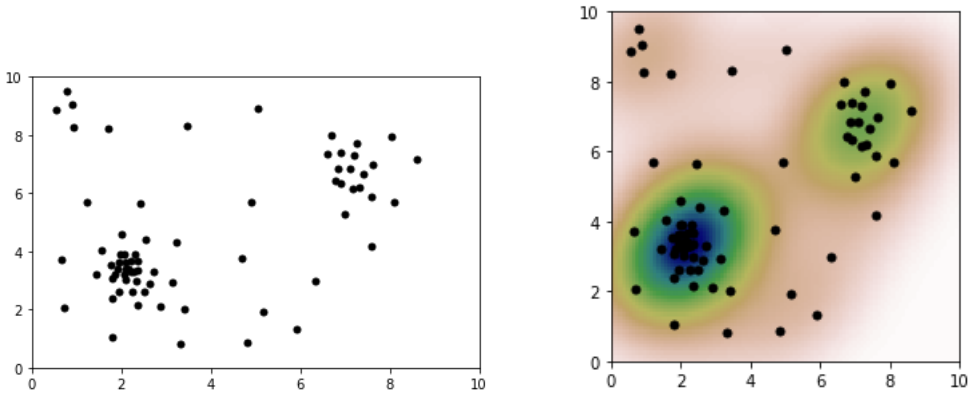


Figura 4-5.: Gráficas de soportes comúnmente usados.

Dado que el soporte de un kernel es simétrico y está normalizado, se suele usar en estadística Bayesiana como estimador de densidad de probabilidad. En otras palabras, un kernel puede ser usado para estimar la densidad en una δ-vecindad de un conjunto finito de vectores $\{x_1, x_2, \cdots k_s\} \subset \mathbb{R}^n$ definiendo la función

$$P(x) = \frac{1}{|B(x,\delta)|} \sum_{x_i \in B(x,\delta)} K(x - x_i).$$



(a) Datos bidimensionales. (b) Densidad estimada.

Figura 4-6.: Ejemplo de densidad estimada con kernel Gausssiano.

En su trabajo, Fukunaga y Hostetle definieron una función que sirve para estimar el centro de densidad en una δ-vecindad, a saber

$$m(x) = \frac{\sum\limits_{x_i \in B(x,\delta)} K(x - x_i)x}{\sum\limits_{x_i \in B(x,\delta)} K(x - x_i)}.$$

La diferencia $m(x) - x$ se llama mean shift. Ahora consideremos un conjunto de datos no supervisados

$$D = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

donde cada observación $x_i \in \mathbb{R}^m$ es un vector fila. Tomemos un Kernel K para $\mathbb{R}^m$ y fijemos los parámetros $\delta, \epsilon > 0$. Así, podemos tomar el arreglo de vectores fila

$$X^{(0)} = (x_1, x_2, ..., x_n).$$

Si para cada observación consideramos una δ-vecindad, entonces podemos definir el arreglo de vectores fila

$$X^{(1)} = (m(x_1), m(x_2), ..., m(x_n)) = (x_1^{(1)}, x_2^{(1)}, ..., x_n^{(1)}).$$

En este paso consideramos la relación de equivalencia conocida como igualdad con tolerancia ϵ, es decir,

$$x_i^{(1)} \equiv x_j^{(1)} \quad \Longleftrightarrow \quad \left\| x_i^{(1)} - x_j^{(1)} \right\| < \epsilon.$$

Luego se repite el procedimiento para calcular $X^{(2)}$ con igualdad basada en la tolerancia

$$X^{(2)} = (m(x_1^{(1)}), m(x_2^{(1)}), ..., m(x_n^{(1)})) = (x_1^{(2)}, x_2^{(2)}, ..., x_n^{(2)}).$$

El proceso se repite l veces, de tal forma que $X^{(l)} \equiv X^{(l-1)}$ bajo la tolerancia ϵ, ya que $x \to m(x)$ en $\mathbb{R}^m$ bajo este proceso iterativo. Por lo tanto, en las entradas de $X^{(l)}$ aparecen solo p vectores distintos, con $p < n$. Así, podemos afirmar que

$$X^{(l)} = (x_1^{(l)}, x_2^{(l)}, ..., x_n^{(l)}),$$

donde $x_i^{(l)} \in \{y_i : 1 \leq i \leq p\} \subseteq \mathbb{R}^m$.

Observación 4.5. *Este algoritmo es ampliamente usado en diferentes aplicaciones por su rápida convergencia y su simplicidad en la implementación. Sin embargo, aún no se conoce una prueba rigurosa para la convergencia del algoritmo. Aliyari Ghassabeh mostró la convergencia del algoritmo cuando las observaciones de los datos están en $\mathbb{R}$, pero el caso unidimensional tiene aplicaciones limitadas.*

Existen dos formas de asignar una clasificación a partir de $X^{(l)}$:

I Se define $\varphi(i) = j$ cuando se tiene que $x_i^{(l)} = y_j$.

II Se define $\varphi(i) = j$ cuando se cumple que $d(y_j, x_i) = min\{d(x_i, y_k) : 1 < k < p\}$

Así, este algoritmo de agrupamiento depende de tres parámetros: el Kernel K, la tolerancia ϵ y lo que se conoce como el ancho de banda δ. La selección de estos parámetros dependerá de la aplicación y de los datos.

4.6. Otros Métodos

Existen muchos métodos diferentes para lograr un agrupamiento de un conjunto de datos. Por practicidad, este trabajo expone los métodos básicos para el entendimiento del problema de agrupamiento. Sin embargo, existen otras técnicas ampliamente usadas:

- **Agrupamiento espectral:** este método construye un grafo en la nube de puntos, asociando a éste una matriz de similitud de rango menor a la dimensión de los datos. Así, se logra una reducción de dimensiones de los datos al proyectarlos en el espacio generado por el espectro de la matriz de similitud. Una vez se logra la proyección de los datos en el espacio de menor dimensión, se usa comúnmente K-medias o Mean shift para hacer el agrupamiento. Este algoritmo es popular por su capacidad de segregación de objetos dentro una imagen [21][1].

- **Agrupamiento jerárquico:** Este algoritmo agrupa las observaciones de forma jerárquica, basadas en una métrica. La jerarquía se puede visualizar con un árbol, donde la altura en la que se encuentre el nodo asociado a la observación determina la jerarquía. Hay dos formas de realizar este algoritmo: descendente, donde todas las observaciones inician en el mismo grupo y ascendente, donde cada observación inicia en su propio grupo. Este algoritmo es ideal para estimar el número de clases que admiten los datos. El inconveniente principal de este algoritmo es que crece de la forma n^2, ya que se debe computar siempre la matriz de distancia de los datos [16].

- **Modelo de mezclas gaussianas:** Este es un método de clasificación estadístico que parte de suponer K distribuciones generadoras de los datos. Así, asociar cada observación a la distribución genera una clasificación. Para este método es necesario hacer suposiciones de las distribuciones. Este método de clasificación es ideal para agrupar observaciones que describan comportamientos, personas o fenómenos naturales [14].

- **Propagación de afinidad:** Este método se basa en una matriz de afinidad M, donde cada entrada $0 \leq m_{ij} \leq 1$ representa la afinidad entre la i-ésima observación y la j-ésima observación. Por lo tanto la matriz M es una matriz simétrica donde todas las entradas de la diagonal son iguales a 1. La matriz de afinidad describe el comportamiento topológico de los datos, ya que dos observaciones afines no son necesariamente cercanas con la métrica Euclídea. El algoritmo no necesita que se le defina el número de clases. El algoritmo estima los centros de afinidad de los datos y con esto logra la clasificación. Este algoritmo es ideal para clasificar datos con topologías no Euclídeas. Un ejemplo de esto es la clasificación de rostros o fotografías de personas, ya que si a una persona le tomas dos fotografías, una fotografía frontal sin desplazamiento facial y la otra con una mueca o un peinado diferente, desde un punto de vista Euclídeo las fotos son distantes, pero desde un punto de vista topológico, las personas de las fotografías son altamente afines [6].

- **DBSCAN:** El agrupamiento espacial basado en densidad de aplicaciones con ruido (Density-Based Spatial Clustering of Applications with Noise), es un método basado en densidad y con la capacidad de formar categorías que no guarden alguna proporción o forma geométrica entre ellas. Este es un algoritmo frecuentemente citado en la literatura. Este método es muy eficiente para buscar anomalías por categoría, pero no es completamente determinista, es decir, puede que algunos de las observaciones no queden clasificadas por el método. Para estas observaciones sin clasificar en la practica se suele usar K-vecinos más cercanos, para

completar la clasificación [20].

Observación 4.6. *Los métodos descritos son heurísticos, no existe una forma de compararlos entre sí de forma general. Determinar cuál usar, dependerá de la aplicación y los datos. Es recomendable usar varios métodos y hacer algún tipo de validación o comparación del resultado de cada modelo seleccionado. Además, se debe tener en cuenta que algunos de estos algoritmos admiten variaciones o generalizaciones que pueden ser apropiadas para algunas aplicaciones particulares.*

5. Predicción

5.1. Predicción y error

Las predicciones son todos aquellos modelos creados bajo aprendizaje supervisado bajo la definición 2.7. Este es uno de los conceptos más usados en el aprendizaje automatizado y existe una gran variedad de algoritmos geométricos y estadísticos; como ejemplo simple de estos está la clasificación de K-vecinos más cercanos. Las aplicaciones de estos conceptos fue un impulso importante en el estudio del aprendizaje automatizado, porque ayudó a la industria a anticipar eventos perjudiciales o saber de antemano las futuras elecciones de los usuarios o clientes para lograr optimizar sus ganancias o mejorar servicios, entre otras aplicaciones. Pero al extenderse el uso de los modelos de predicción, se hace necesario hablar del concepto de error, que es una forma de estimar la capacidad predictiva de un modelo [19].

Definición 5.1. *Sea D un conjunto de datos supervisados*

$$D = \begin{bmatrix} Id & V_1 & V_2 & \cdots & V_n & f \\ 1 & V_1(1) & V_2(1) & \cdots & V_n(1) & f(1) \\ 2 & V_1(2) & V_2(2) & \cdots & V_n(2) & f(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ k & V_1(k) & V_2(k) & \cdots & V_n(k) & f(k) \end{bmatrix}.$$

Considere J, I una partición aleatoria de $\{1, 2, \cdots, k\}$, es decir, J, I son definidos de forma aleatoria y cumplen que

$$I \cup J = \{1, 2, \cdots, k\},$$

$$I \cap J = \varnothing.$$

Llamamos datos de entrenamiento D_{train} a los datos supervisados que se obtienen de tomar de D sólo las filas que aparezcan en J. Así mismo, llamados D_{test} a los datos supervisados que se obtienen de tomar de D sólo las filas que aparezcan en I. Se dice que la pareja (D_{train}, D_{test}) son una partición de entrenamiento para D.

Observación 5.1. *Uno de los métodos clásicos de predicción es la regresión lineal, aunque existen pocas aplicaciones en las cuales este método tenga un buen ajuste en los datos de entrenamiento. Para el caso de predicción binaria, el método clásico de predicción es separar las observaciones como puntos de $\mathbf{R}^n$ con un hiperplano si los puntos son separables; cuando el conjunto de puntos no es separable, es decir, que no existe un hiperplano $\mathbf{R}^n$ que pueda dividir la nube de puntos en dos grupos, se usa un Kernel para embeber la nube de puntos en $\mathbf{R}^{n+1}$ y lograr separabilidad. Este método se conoce como Maquinas de Soporte Vectorial (Support Vector Machine for One Class), donde los puntos cercanos al hiperplano (o sobre el hiperplano) son los que caracterizan la separación y suelen llamarse vectores de soporte. En este trabajo se tratarán otros métodos adicionales de predicción que consideren diferentes planteamientos geométricos o estadísticos.*

Supongamos D, un conjunto de datos supervisado bajo una función de etiqueta f con una partición de entrenamiento (D_{train}, D_{test}). Sea B un conjunto de datos no supervisados con las mismas variables de D. Si bajo un aprendizaje supervisado se define una predicción g sobre B a partir de D_{train}, entonces g también se puede usar como modelo para D_{test}, esto debido a que D y B tienen las mismas variables para cada observación, en particular, las observaciones de D_{test} están en el dominio de g. Por lo tanto, se consigue una doble marcación en las observaciones de D_{test}:

$$D_{test} = \left[\begin{array}{cccccc|c} Id & V_1 & V_2 & \cdots & V_n & f & g \\ i_1 & V_1(i_1) & V_2(i_1) & \cdots & V_n(i_1) & f(i_1) & g_{i_1} \\ i_2 & V_1(i_2) & V_2(i_2) & \cdots & V_n(i_2) & f(i_2) & g_{i_2} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ i_p & V_1(i_p) & V_2(i_p) & \cdots & V_n(i_p) & f(i_p) & g_{i_p} \end{array}\right],$$

donde $g_{i_s} = g(V_1(i_s), V_2(i_s), \cdots, V_n(i_s))$. Dado que la partición de entrenamiento fue hecha de forma aleatoria, la matriz de doble etiqueta o matriz de error E se puede usar para estimar un error de predicción

$$E = \begin{bmatrix} f(i_1) & g_{i_1} \\ f(i_2) & g_{i_2} \\ \vdots & \vdots \\ f(i_p) & g_{i_p} \end{bmatrix}.$$

La forma de estimar el error dependerá de la función f, ya que el objetivo del problema de predicción es lograr que el modelo g sea lo más cercano a una extensión de la función f.

5.1.1. Predicción binaria

Consideremos el caso inicial: las funciones f, g son binarias con rango en $\{0, 1\}$. Tomaremos como consideración que si una observación está marcada con 0, entonces la observación es favorable,

pero si tiene la etiqueta 1 representa un caso desfavorable. Esto se usa porque en las aplicaciones las etiquetas binarias representan en general dos estados posibles, donde uno de estos estados es un caso desfavorable que se quiere predecir, por ejemplo un fraude, la fuga de un cliente o la materialización de un riesgo. Por esto, las observaciones marcadas con 0 por la función g que supervisa los datos se les llaman negativos reales y los marcados con 1, se les llaman positivos reales. Así, cuando la función de marcado inicial y la predicción marcan una observación con 1 se le llama un verdadero positivo, es decir, un acierto en la predicción sobre el grupo de observaciones que representan un riesgo, y cuando se predice un 1 a una observación marcada inicialmente favorable 0, se suele llamar falso positivo. Análogamente, se tienen los verdaderos negativos y los falsos negativos. [7]

Definición 5.2. *Sea E una matriz de error con p observaciones de una predicción binaria.*

- *Definimos la tasa de **positivos reales** P como el número de observaciones marcadas con 1 por la función g en la matriz E, $i.e$,*

$$P = \sum_{j=1}^{p} g(i_j).$$

- *Definimos la tasa de **negativos reales** N como el número de observaciones marcadas con 0 por la función g en la matriz E, $i.e$,*

$$N = \sum_{j=1}^{p} (1 - g(i_j)).$$

- *Llamamos TP a la tasa de **verdaderos positivos** que se define como el número de observaciones de E etiquetadas con 1 por ambas funciones, $i.e$,*

$$TP = \sum_{j=1}^{p} f(i_j) g_{i_j}.$$

- *Llamamos TN a la tasa de **verdaderos negativos** que se define como el número de observaciones de E etiquetadas con 0 por ambas funciones, $i.e$,*

$$TN = \sum_{j=1}^{p} (1 - f(i_j))(1 - g_{i_j}).$$

- *Llamamos FP a la tasa de **falsos positivos** que se define como el número de observaciones de E etiquetadas con 0 por f y etiquetadas con 1 por g, $i.e$,*

$$FP = \sum_{j=1}^{p} (1 - f(i_j))(g_{i_j}).$$

- *Llamamos FN a la tasa de **falsos negativos** que se define como el número de observaciones de E etiquetadas con 1 por f y etiquetadas con 0 por g, $i.e$,*

$$FN = \sum_{j=1}^{p} (f(i_j))(1 - g_{i_j}).$$

La matriz de confusión asociada a la matriz de error E se define como

$$E_c = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}.$$

En algunos casos para facilitar la recordación del contenido de la matriz de confusión se escribe de manera informal como

$$\begin{bmatrix} & 1 & 0 \\ 1 & TP & FN \\ 0 & FP & TN \end{bmatrix}.$$

Note que la $tr(E_c)$ es exactamente el número de observaciones donde f y g coinciden. Por lo tanto, en el caso de predicciones binarias se usa la matriz de confusión para estimar el error: el error cometido por el modelo g en la predicción es

$$p - tr(E_c).$$

Sin embargo, esta medida de error no siempre contiene la información suficiente para determinar la exactitud del modelo. Dependiendo de la aplicación, puede ocurrir que se quieran modelos que minimicen FN o minimicen FP, también puede ocurrir que se quiera maximizar $tr(E_c)$ sin importar que el error se concentre en FN o FP. Algunas de las medidas de error o estadísticos de predicción que se pueden usar se muestran en la tabla **5-1** [19].

Nombre del estadístico	**Estadístico**
Sensibilidad o tasa verdadera positiva	$TPR = \frac{TP}{P}$
Especificidad o tasa negativa verdadera	$TNR = \frac{TN}{N}$
Precisión o valor predictivo positivo	$PPV = \frac{TP}{TP+FP}$
Valor predictivo negativo	$NPV = \frac{TN}{TN+FN}$
Tasa de fallos o tasa de falsos negativos	$FNR = \frac{FN}{P}$
Tasa de caída o de falsos positivos	$FPR = \frac{FP}{N}$
Tasa de descubrimiento falso	$FDR = \frac{FP}{FP+TP}$
Tasa de omisión falsa	$FOR = \frac{FN}{FN+TN}$
Exactitud	$ACC = \frac{TP+TN}{P+N}$
Media armónico de precisión o puntaje $F1$	$F1 = 2\frac{PPV \cdot TPR}{PPV+TPR}$

Tabla 5-1.: Estadísticos comunes para una predicción binaria.

Para elegir el estadístico adecuado en una aplicación, se debe detectar cuál es el error que tiene mayor impacto en la aplicación. Por ejemplo, si la aplicación se usa para predecir la fuga de clientes o determinar qué usuarios van a adquirir un nuevo producto, se querrá minimizar el número de falsos positivos. Si se desea hacer una segmentación binaria de personas para determinar quienes reciben o no determinado beneficio, será relevante que $tr(E_c)$ sea lo más cercana a p, entonces es recomendable medir la exactitud o la media armónica de precisión.

5.1.2. Predicción discreta

Supongamos ahora el caso en que f es una función discreta, es decir, f y g son dos funciones de clasificación para los datos sobre un mismo conjunto C de m clases. Entonces para este caso se puede extender el concepto de matriz de confusión.

Definición 5.3. *Sean f y g dos funciones de clasificación para unos datos D sobre un conjunto de clases $C = \{C_1, C_2, \cdots, C_m\}$. Entonces se define la matriz de confusión $E_c \in C^m \times C^m$ asociada a f y g, como aquella matriz donde la entrada E_{ij} está dada por el numero de observaciones etiquetadas por f como C_i y etiquetadas por g como C_j.*

Al igual que en el caso binario, $tr(E_c)$ es el número total de observaciones en el que f y g coinciden. Por lo tanto la tasa de error en la predicción discreta para p observaciones está dada por

$$\frac{p - tr(E_c)}{p}.$$

Una forma práctica de visualizar la predicción discreta es asociar a cada observación un par ordenado en $\mathbf{R}^2$, donde a la observación x se le asocia el par (i, j) si $f(x) = C_i$ y $g(x) = C_j$. Todos los puntos en el plano fuera de la recta identidad están asociados a errores de predicción, permitiendo apreciar de forma visual cómo se distribuye la tasa de error de predicción.

5.1.3. Predicción continua

Si la función de marcado y el modelo son funciones con rango en $\mathbf{R}$ no tiene sentido definir la matriz de confusión. Una forma simple de visualizar el ajuste del modelo g con la función de marcado f, es asociar a cada punto un par ordenado definido por la correspondiente fila de la matriz de error E, así para el j-ésimo punto, se gráfica el par ordenado $(f(i_j), g_{i_j})$ en $\mathbf{R}^2$, correspondiente a la j-ésima fila de E. Note que el error disminuye si los puntos se concentran cerca de la recta identidad [23].

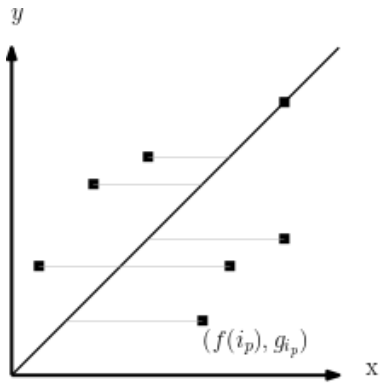


Figura 5-1.: Filas de la matriz E en el plano y su distancia horizontal a la recta identidad.

Se puede definir un error con base en una tolerancia: se calcula F_δ como el número de puntos que tengan una distancia mayor a δ de la recta identidad, donde esta distancia se puede medir de forma horizontal, vertical o perpendicular. Se define el **error de predicción con una tolerancia** δ como

$$\frac{F_\delta}{p}.$$

Este indicador cuantitativo habla de forma global sobre los errores de predicción, pero no da información acerca de las predicciones más allá de δ. Por ejemplo, puede ocurrir que los valores más allá de delta tiendan a concentrarse. Así, la medida del error se puede acompañar con algún estadístico que ayude a describir el error de predicción más allá de la tolerancia:

- El **error acumulado de predicción** permite evaluar el sesgo o el error total que se obtiene con el modelo g:

$$\sum_{j=1}^{p}(\left|f(i_j) - g_{i_j}\right|.$$

- La **desviación media absoluta del error** estima la desviación o error absoluto que se puede esperar en promedio de la predicción:

$$\sum_{j=1}^{p}\frac{\left|f(i_j) - g_{i_j}\right|}{p}.$$

- El **error cuadrático medio** es una forma de medir error promedio en la predicción, pero castigando el estadístico cuando la diferencia de los valores entre las etiquetas sea mayor a uno:

$$\sum_{j=1}^{p}\frac{(f(i_j) - g_{i_j})^2}{p}.$$

5.2. Regresión logística

Este es un método de aprendizaje supervisado para predicción binaria de forma probabilística. Es un método clásico que realiza la marcación al estimar la probabilidad de que una observación no supervisada tenga la marcación 1, que se conoce usualmente como estado de default. Este tipo de técnicas también se usan para definir puntajes que representen cuantitativamente, qué tan cerca está una observación del estado de default. Una aplicación común de esta medición se puede ver en el puntaje de riesgo crediticio que usan las entidades financieras, una escala de medición de la probabilidad de impago de cliente [18].

Supongamos un conjunto de datos supervisados D con n variables y k observaciones.

$$D = \begin{bmatrix} x_1 & f(1) \\ x_2 & f(2) \\ \vdots & \vdots \\ x_k & f(k) \end{bmatrix} = \begin{bmatrix} V_1(1) & V_2(1) & \cdots & V_n(1) & f(1) \\ V_1(2) & V_2(2) & \cdots & V_n(2) & f(2) \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ V_1(k) & V_2(k) & \cdots & V_n(k) & f(k) \end{bmatrix}.$$

La regresión logística supone que el estado de la observación o marcación binaria de la función f está relacionado con las variables de las observaciones, por lo tanto se espera que las variables de los datos sean independientes, es decir, el método asume que las columnas de la matriz D son linealmente independientes. Así, para la i-ésima observación x_i, el modelo logístico establece que la probabilidad condicional está determinada de forma paramétrica como

$$Pr(f(i) = 1|x_i) = \frac{1}{1 + exp(-\alpha - \langle \beta, x_i \rangle)},$$

donde $\beta = (\beta_1, \beta_2, \cdots, \beta_n) \in \mathbf{R}^n$ es un vector de parámetros que junto con el parámetro α determinan la función de probabilidad. De forma equivalente, la función de probabilidad se puede presentar usando la función $logit$, a saber

$$logit(Pr(f(i) = 1|x_i)) = log\left(\frac{Pr(f(i) = 1|x_i)}{1 - Pr(f(i) = 1|x_i)}\right) = \alpha + \langle \beta, x_i \rangle.$$

Para estimar los $n+1$ parámetros de la función de probabilidad se usa máxima verosimilitud, como se realizó en la sección 4.4. Definimos la función verosimilitud como

$$L(\alpha, \beta) = \prod_{i=1}^{k} \left[(1 - f(i))Pr(f(i) = 0|x_i) + f(i)Pr(f(i) = 1|x_i)\right].$$

Para simplificar los cálculos computacionales, se suele maximizar $log(L(\alpha, \beta))$ que se presenta como

$$log(L(\alpha,\beta)) = \sum_{i=1}^{k} log([(1-f(i))Pr(f(i)=0|x_i) + f(i)Pr(f(i)=1|x_i)]).$$

Una vez estimados los parámetros α, β con máxima verosimilitud, entonces se puede definir un modelo g para un conjunto de datos no marcados B que tiene las mismas variables de D. Si y_i es la i-ésima observación de B, se define la clasificación binaria g como

$$g(y_i) = \begin{cases} 1 & (1+exp(-\alpha - \langle \beta, y_i \rangle))^{-1} > 0{,}5 \\ 0 & (1+exp(-\alpha - \langle \beta, y_i \rangle))^{-1} \leq 0{,}5 \end{cases}.$$

5.3. Árboles de decisión

Este es un método clásico para realizar predicciones de una función de marcado discreto. Aunque existen muchos algoritmos alternativos diferentes para este método, en este trabajo se expone el modelo clásico que se genera de forma simple y aunque no carece de poder predictivo, sí puede acarrear dificultades computacionales para predicciones sobre un gran volumen de observaciones o con un número elevado de variables [17].

Consideremos un conjunto de datos marcados D con sus variables categorizadas. Consideremos además unos datos B no marcados con las mismas variables. Con el fin de lograr una ilustración simple del método, asumiremos que las observaciones tienen dos variables binarias V_1, V_2. Podemos decir que V_1 toma valores en el conjunto de marcas $C(V_1) = \{A, B\}$, así mismo V_2 toma valores en $C(V_2) = \{X, Y\}$, adicionalmente supondremos que las marcaciones de D están en las categorías $C(f) = \{0, 1, 2\}$. Más precisamente

$$D = \begin{bmatrix} x_1 & f(1) \\ x_2 & f(2) \\ \vdots & \vdots \\ x_k & f(k) \end{bmatrix}, \quad \text{con } x_i \in C(V_1) \times C(V_2) = \{A, B\} \times \{X, Y\}.$$

A continuación, se construye el universo de posibilidades U, que es el conjunto con todas las combinaciones posibles de las variables y la función de marcado, $i.e.$,

$$U = C(f) \times C(V_1) \times C(V_2).$$

Creamos un grafo simple no dirigido, que resulta ser un árbol con un nodo de origen que llamaremos nodo de decisión en el nivel cero. En el nivel 1 del grafo habrá un nodo por cada marcación

en $C(f)$. Para $i > 1$ se tendrá en el i-ésimo nivel del grafo la variable $i - 1$, de tal forma que el árbol tendrá una rama por cada elemento de U y por lo tanto, la altura del grafo será igual a la cardinalidad de U.

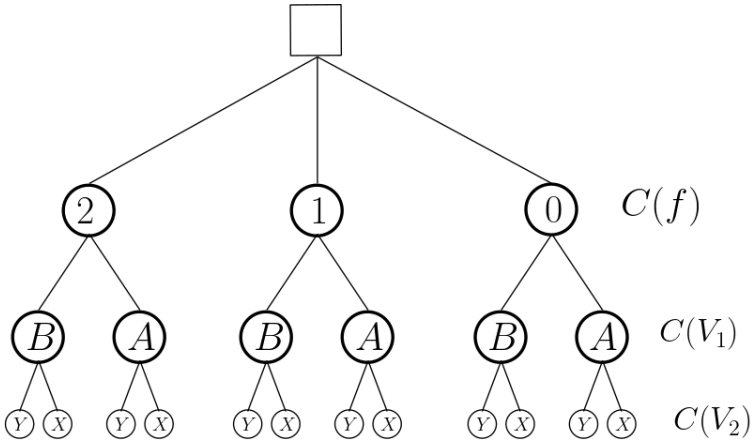


Figura 5-2.: Árbol de decisión con 2 variables binarias y 3 categorías de marcado.

Ahora, para cada arista que no se conecte con el nodo de decisión se le dará un peso, de acuerdo a la probabilidad de ocurrencia que tiene una observación del nivel i condicionada al nivel $i + 1$. Por ejemplo, para la conexión del nodo A con el nodo X se calcula el peso de la arista dado por $P(V_1 = A|V_2 = X)$. Estas probabilidades se calculan con base en las observaciones de los datos de entrenamiento D, tomando en cuenta que las probabilidad condicional está dada por

$$P(V_1 = A|V_2 = X) = P(A|X) = \frac{P(A \cap X)}{P(X)},$$

donde $P(X)$ es el número de observaciones marcadas con X en la variable V_2, dividido por el número total de observaciones de D. De igual manera, $P(A \cap X)$ es el número de observaciones marcadas con A en la variable V_1 y marcadas con X en la variable V_2, dividido por el número total de observaciones de D.

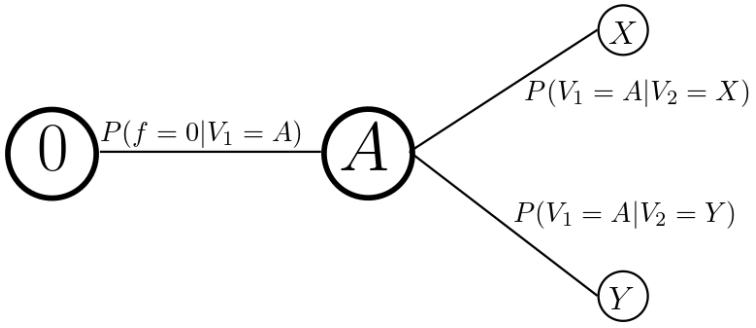


Figura 5-3.: Ejemplo de cálculo de los pesos de las ramas del árbol.

Una vez calculado el peso de las aristas, el árbol recibe el nombre de árbol de decisión y es usado para realizar las predicciones sobre las observaciones de los datos B. Para cada observación $z \in B$ se podan las ramas del árbol que no correspondan a las ocurrencias de las variables en z, y las aristas que tocan el nodo de decisión se marcan con el producto de los pesos de la rama. Finalmente, a z se le dará la marcación dada por el nodo del primer nivel que tenga un mayor peso en el nodo de decisión. Por ejemplo, supongamos que $z = (A, X)$, entonces al podar el árbol de decisión se tiene el grafo mostrado en la gráfica **5-4**, donde

$$P_0 = P(0|A)P(A|X),$$
$$P_1 = P(1|A)P(A|X),$$
$$P_2 = P(2|A)P(A|X).$$

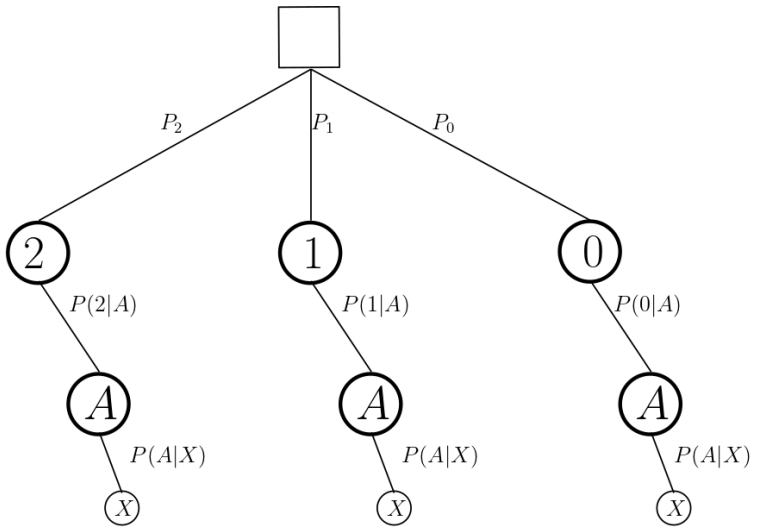


Figura 5-4.: Árbol de decisión podado para $z = (A, X)$.

Por lo tanto, el modelo del aprendizaje supervisado evaluado en la observación z está dado por

$$g(z) = \begin{cases} 0 & \text{si } P_0 = max(\{P_0, P_1, P_1\}), \\ 1 & \text{si } P_1 = max(\{P_0, P_1, P_1\}), \\ 2 & \text{si } P_2 = max(\{P_0, P_1, P_1\}). \end{cases}$$

Observación 5.2. *Note que el árbol crece de forma proporcional al rango de la función de marcado y al rango de las variables. Una forma de disminuir los cálculos es podar el árbol en el entrenamiento por medio de una función que permitirá eliminar variables que tengan poca correlación con la función de marcado, por ejemplo la función χ^2. Aunque este paso se puede evitar si se hace una reducción de dimensiones en el preprocesamiento de los datos. Otro impedimento*

puede ser si existen demasiadas observaciones en los datos de entrenamiento, lo cual puede hacer un sesgo sobre el cálculo de las probabilidades o aportar complejidad computacional. Una solución a este problema es tomar m muestras aleatorias de las observaciones de entrenamiento y generar m arboles de decisión y así, la predicción se hará con el valor más ocurrente en los m modelos. Esta sería una versión simple del algoritmo conocido como arboles aleatorios (Random Forest).

Este método tiene diferentes formas de usarse en las aplicaciones. Por ejemplo, suponga que un conjunto de afiliados o clientes de determinada empresa son segmentados, esto con el objetivo de ofrecer productos más específicos a cada grupo, para estrategias comerciales o para ofrecer beneficios diferenciados. Esto se puede hacer mediante cualquier aprendizaje no supervisado visto en el capitulo 4. Ahora, si llega un cliente nuevo, el cual no fue incluido en el entrenamiento del modelo, no es necesario recalcular la clasificación inicial. Para el cliente nuevo se realiza un árbol de decisión, y así se le asignará la categoría en la cual tiene mayor probabilidad de pertenecer. Esto será posible, si el nuevo individuo posee las mismas variables que fueron usadas con los individuos de la clasificación inicial. Note que en este caso también puede usar el método de k-vecinos más cercanos, pero este método depende de un parámetro externo k, mientras que el árbol de decisión es puramente probabilístico y por lo tanto, tendrá menos sesgo en la clasificación del nuevo individuo.

Observación 5.3. *Otro método de predicción discreta y continua, son las redes neuronales artificiales. Este método y sus variaciones tienen gran poder predictivo y es frecuente encontrarlos en diversas aplicaciones y en la literatura.*

6. Detección de anomalías

Desde el punto de vista de la estadística descriptiva, una anomalía o valor atípico dentro de un conjunto finito de números reales, es aquel valor que es diferente a la mayoría de los elementos del conjunto, siempre que la diferencia de la anomalía se establezca de forma geométrica o estadística. Por ejemplo, si se tiene un conjunto de valores reales tomado de un muestreo o la medición de alguna variable aleatoria, se puede considerar que un valor p es una anomalía si

$$p > (Q_3 + 1{,}5(Q_3 - Q_1)),$$

o bien

$$p < (Q_1 - 1{,}5(Q_3 - Q_1)),$$

donde Q_1, Q_3 representan el primer y tercer cuartil respectivamente. Hallar los valores que cumplen estas desigualdades se conoce como la búsqueda de valores atípicos con distancias intercuartiles. Más aún, si en este ejemplo se aumenta la constante $1{,}5$ por un valor real mayor, se dice que las anomalías tendrían mayor severidad. Así mismo, si la aplicación o la recolección de información lo permite, se puede establecer que un valor es atípico si su norma sobrepasa determinado limite o si su distancia a la media excede lo esperado.

Siguiendo la misma idea de la estadística descriptiva, se dice que una observación de un conjunto de datos es una anomalía si se puede diferenciar de las demás bajo un concepto estadístico o geométrico. Así, la detección de anomalías se puede ver como una clasificación binaria, donde las observaciones etiquetadas con la clase 1 difieren de las observaciones marcadas con la clase 0, por medio de un estadístico o una apreciación geométrica.

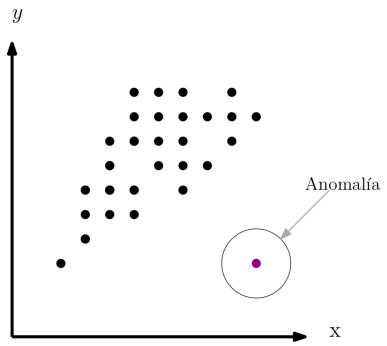


Figura 6-1.: Ejemplo de observaciones anómalas en datos bidimensionales.

La detección se anomalías se puede hacer por medio de aprendizajes supervisado y no supervisado, generando un abanico de posibilidades en las aplicaciones. En la industria existen aplicaciones muy importantes de este concepto, una de las más destacadas es la detección de anomalías en las transacciones de tarjetas de crédito: usando datos sociodemográficos y transaccionales se puede establecer que una transacción es una anomalía. En muchas ocasiones esto resulta en un intento de fraude.

6.1. Anomalías por segmentación

Si pensamos en las observaciones de unos datos como una nube de puntos en $\mathbf{R}^n$ y queremos encontrar en ésta algunas observaciones anómalas, será preferible que la nube de puntos esté concentrada, ya que de manera intuitiva: cuánto más dispersa esté la nube, más difícil será encontrar anomalías. Basados en esta idea, si los datos admiten una segmentación, entonces la búsqueda de anomalías será más efectiva si se busca las observaciones anómalas dentro de cada segmento [19].

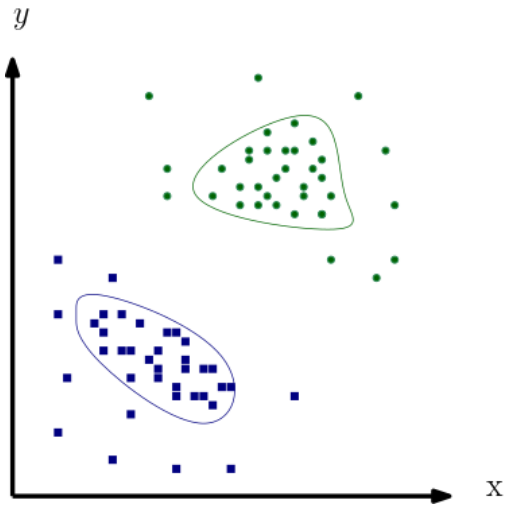


Figura 6-2.: Anomalías en agrupamiento de datos bidimensionales.

Existen muchas maneras de determinar cuándo en una nube de puntos una observación se puede considerar anómala. Se puede tomar el conjunto de distancias de los puntos al centro o media del grupo y aplicar la búsqueda de valores atípicos con distancias intercuartiles. También se puede asumir que una observación es anómala si su distancia al centro del grupo es mayor a determinado percentil. Sin embargo, en muchas aplicaciones, es preferible no asumir que la distancia es un factor completamente influyente para la detección de la anomalía, esto puede deberse a la distribución de la nube de puntos o tal vez, porque una nube de puntos tenga más de un punto de acumulación. Si esto ocurre, lo recomendable es crear un puntaje que determine qué observaciones realmente están aisladas del grupo. Aunque existen diversas formas de crear un puntaje de aislamiento o un

puntaje de anomalía, expondremos una manera simple. Considere el conjunto M que contiene las m observaciones de un mismo grupo. Tomamos

$$n = \max_{x_i \in M} |B(x_i, \delta)|, \text{ donde } \delta > 0.$$

Para un parámetro fijo δ, el valor n representa el número máximo de vecinos que puede tener una observación del conjunto M en una δ-vecindad. Así, la función P definida por

$$P(x_i) = \frac{|B(x_i, \delta)|}{n} 100,$$

determinará el puntaje de aislamiento para cada observación de forma local. En otras palabras, este puntaje de aislamiento es el porcentaje de densidad de puntos en una vecindad con respecto a la máxima densidad hallada en el grupo. Por lo tanto, puede considerarse que para un puntaje cercano a cero se tiene una anomalía, teniendo en cuenta que este puntaje se debe establecer para cada grupo estimado en la segmentación inicial. Note que este puntaje se puede generalizar usando un kernel para estimar la densidad de puntos en la δ-vecindad.

Observación 6.1. *El umbral a partir del cual una observación se considera una anomalía dependerá de la forma como se defina el puntaje y de la aplicación, por lo que es recomendable hacer una análisis descriptivo antes de asumir un límite, ya que tomar un límite inferior considerablemente grande puede implicar un número elevado de falsos positivos.*

6.2. Bosque de aislamiento

Este es un método propuesto por Fei Tony Liu, Kai Ming Ting y Zhi-Hua Zhou en el 2008. Aunque el método es reciente, está ampliamente difundido en diferentes lenguajes de programación. El método en ocasiones puede tener un gasto computacional considerable en el entrenamiento del modelo. Sin embargo, su aplicación sorprende por sus altas tasas de aciertos en múltiples aplicaciones, entre las que se encuentra la detección de anomalías en seguros y tarjetas de crédito. Además, aporta a las matemáticas y la ciencia de datos un complejo problema teórico, ya que el modelo se construye sobre una hipótesis que hasta el momento es un problema abierto. Este método introduce el uso del aislamiento como un medio más eficaz para detectar anomalías que las medidas clásicas comúnmente utilizadas, distancia y densidad. A continuación expondremos el trabajo presentado en [12].

Considere una nube de puntos M, que por simplicidad tomaremos $M \subset \mathbf{R}^2$. Si $x \in M$, diremos que una p-secuencia de x en $\mathbf{R}^2$ es una secuencia de p particiones, donde la primera partición $A^{(1)}$ y $B^{(1)}$, divide a $\mathbf{R}^2$ en dos partes por medio de una recta horizontal o vertical. En otras palabras.

se tiene que $A^{(1)} \cup B^{(1)} = \mathbf{R}^2$ y la intersección $A^{(1)} \cap B^{(1)}$ es una recta horizontal o vertical. Sin pérdida de generalidad, podemos asumir que $x \in A^{(1)}$.

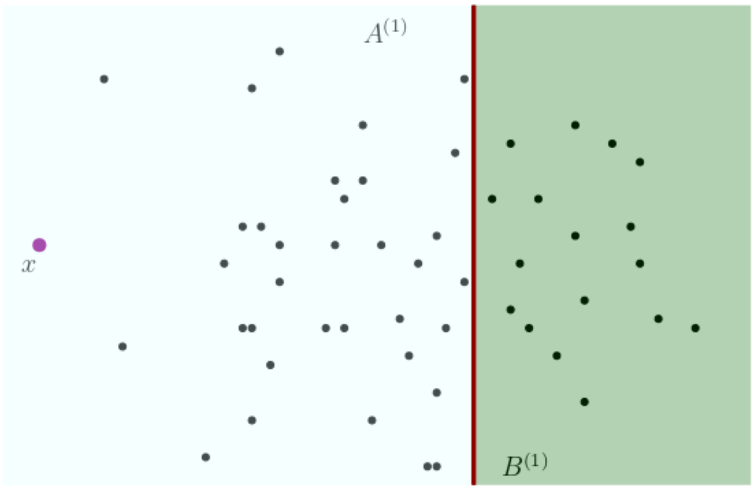


Figura 6-3.: Primera partición.

La segunda partición divide a $A^{(1)}$ en $A^{(2)}$ y $B^{(2)}$ por medio de una recta horizontal o vertical, donde $x \in A^{(2)}$.

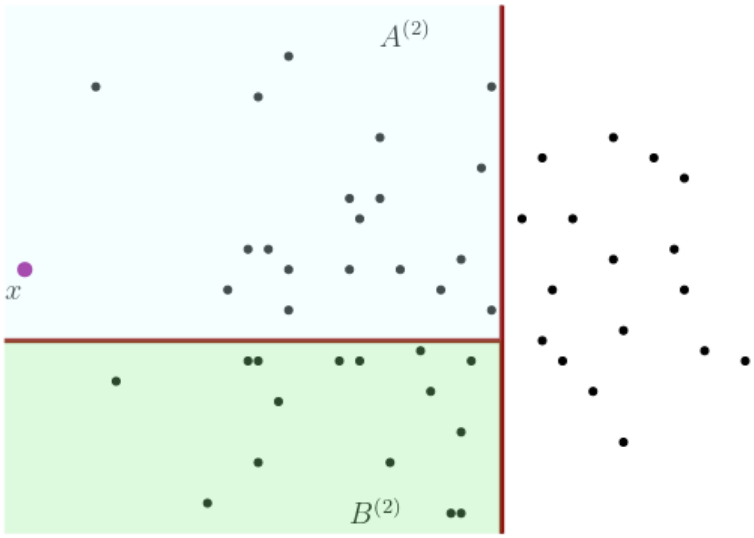


Figura 6-4.: Segunda partición.

La tercera partición divide a $A^{(2)}$ en $A^{(3)}$ y $B^{(3)}$ por medio de una recta horizontal o vertical, donde $x \in A^{(3)}$. Así, se continúa el proceso hasta lograr $A^{(p)} \cap M = \{x\}$.

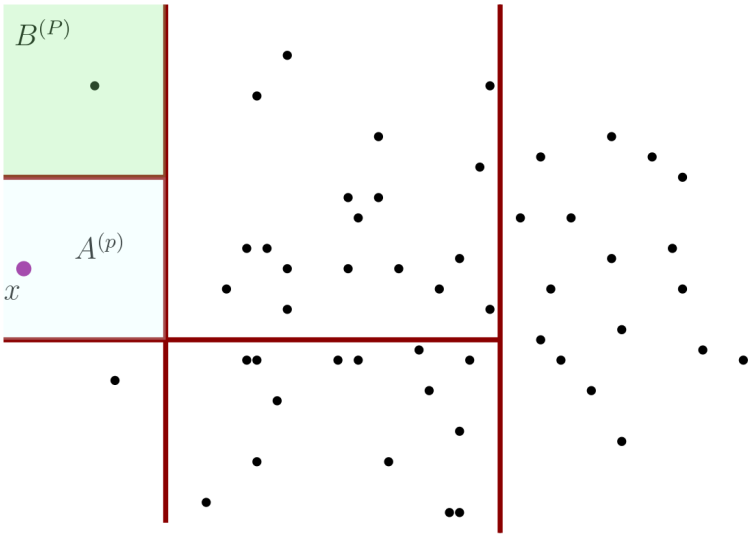


Figura 6-5.: p-ésima partición.

Ahora, si las rectas de las particiones de la p-secuencia fueron tomadas de forma aleatoria, se dice que es un aislamiento para x. Además, note que todo aislamiento genera un grafo. Este grafo es un árbol binario donde la rama de mayor longitud tiene exactamente p aristas. Por lo tanto, si S es un árbol de aislamiento para x, diremos que la altura del árbol es $h(S) = p$, la función que mide la mayor rama del árbol de aislamiento inducido por S. Diremos que el bosque de aislamiento de x, denotado por $U(x)$, es el conjunto de todos los arboles de aislamiento de x.

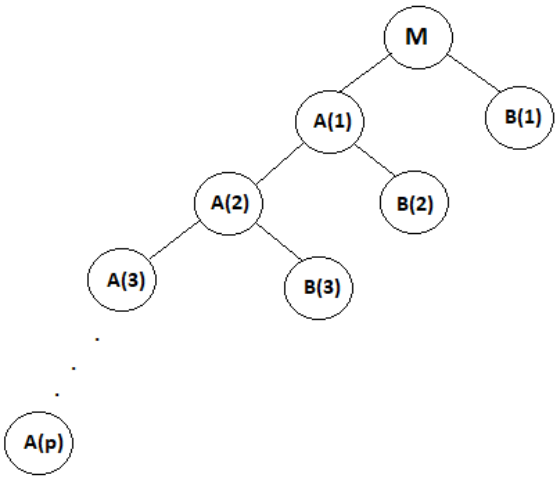
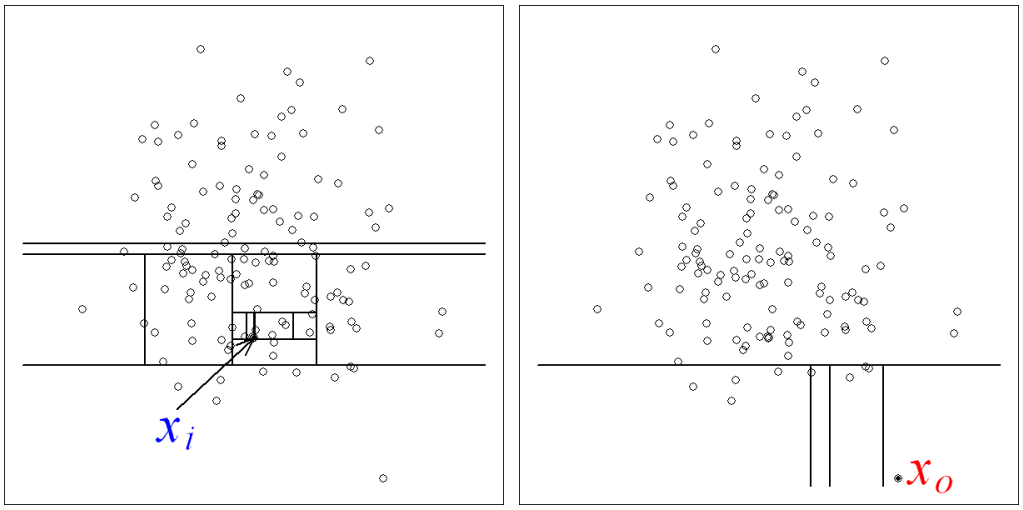


Figura 6-6.: Árbol de aislamiento de longitud p.

El método de bosques de aislamiento se basa en dos hipótesis:

1. Las observaciones anómalas son pocas y son diferentes, por lo tanto estas observaciones atípicas son más susceptibles al aislamiento. Esto quiere decir que para las observaciones que están en los sectores menos densos de la nube, se obtendrán arboles de aislamiento con menor altura.



(a) Aislamiento para un punto no distante x_i. (b) Aislamiento para un punto distante x_o.

Figura 6-7.: Comparación de aislamientos. Imagen tomada de [12].

2. En el bosque de aislamiento de un punto x predomina una altura promedio. Es decir, si tomamos contables muestras aleatorias y finitas del bosque $U(x)$, digamos $M_1, M_2, M_3 \cdots$, entonces el promedio la i-ésima muestra está dado por

$$\overline{M}(i) = \sum_{S \in M_i} \frac{h(S)}{|M_i|},$$

con $|M_i|$ el número de arboles de la i-ésima muestra. Entonces se tiene que

$$\lim_{i \to \infty} \overline{M}(i) = p.$$

El valor p se conoce como altura promedio del bosque de aislamiento de x. Adicionalmente, la existencia de este valor p no depende de la función h, entonces se suele normalizar a h, para que las mediciones de las alturas de los arboles estén acotadas. Esta hipótesis es un conjetura. En la actualidad sólo existen simulaciones con gran número de muestras que permiten sospechar que la hipótesis es valida, sin embargo no existe una prueba rigurosa.

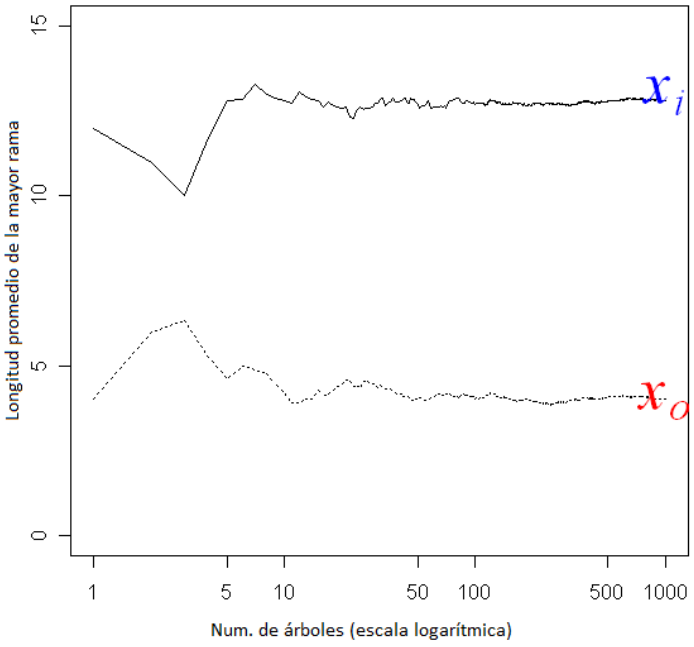


Figura 6-8.: Valor promedio de h en simulaciones. Imagen tomada de [12].

Así, si se tiene un conjunto de datos con n observaciones se define la constante armónica

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n},$$

donde $H(i)$ es el número armónico, que se puede estimar por el i-ésimo logaritmo natural más la constante de Euler, es decir, $H(i) = ln(i) + 0{,}5772156649$. Con esto podemos definir una función de score o puntaje para cada observación x de los datos

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}},$$

donde $E(h(x))$ es el promedio de h sobre una muestra de árboles de aislamiento de x, es decir, se toma una muestra $V \subset U(x)$ con $|V| = p$, y entonces

$$E(h(x)) = \sum_{S \in V} \frac{h(S)}{p}.$$

Otra forma de verlo, es que $E(h(x))$ es un estimador de altura promedio del bosque de aislamiento de x. Al definir el puntaje de esta manera se obtienen algunas relaciones:

- si $E(h(x)) \to c(n)$ entonces $s \to 0.5$,
- si $E(h(x)) \to 0$ entonces $s \to 1$,
- si $E(h(x)) \to n - 1$ entonces $s \to 0$.

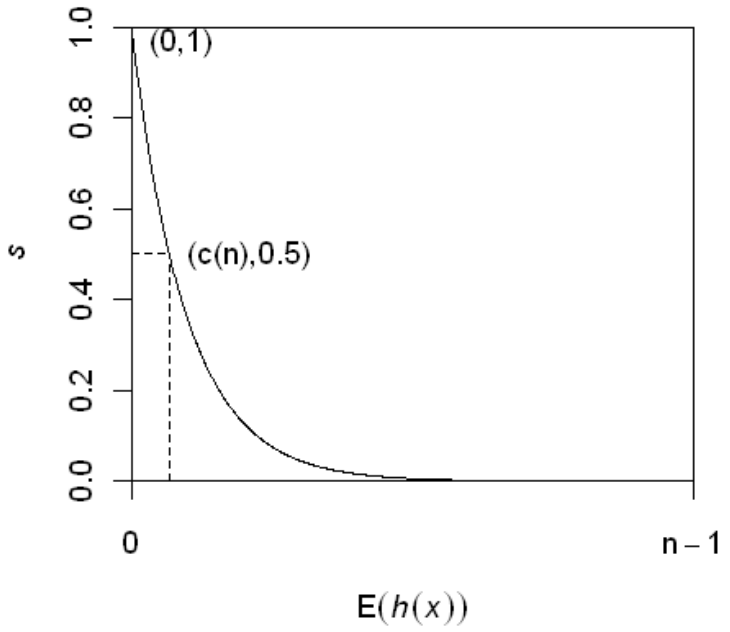


Figura 6-9.: Relación entre $E(h(x))$ y s. Imagen tomada de [12].

En conclusión, si s arroja un valor cercano a 1, entonces la observación es potencialmente una anomalía. Así mismo, si s arroja un valor inferior a 0.5, entonces la observación es común y no aislada. Para determinar a partir de qué valor de s se puede considerar anomalía, es común usar un mapa de contorno (o mapa de calor) para los puntos, como se muestra en la figura **6-10**. En este ejemplo se establece que si $0.6 \leq s$, entonces es una anomalía.

Observación 6.2. *Este proceso se puede generalizar para observaciones en* $\mathbf{R}^n$ *si se define el aislamiento de una observación de forma análoga, haciendo una secuencia de separaciones con hiperplanos paralelos a los ejes coordenados, y generando igualmente un árbol de aislamiento. Lo demás, será exactamente igual. Por otro lado, note que para la construcción del modelo puede hacerse necesario un gasto computacional considerablemente alto al computar* E *para cada observación. Lo que se suele hacer en la practica es aplicar una reducción de dimensiones y entrenar el modelo sólo con una muestra representativa de los datos.*

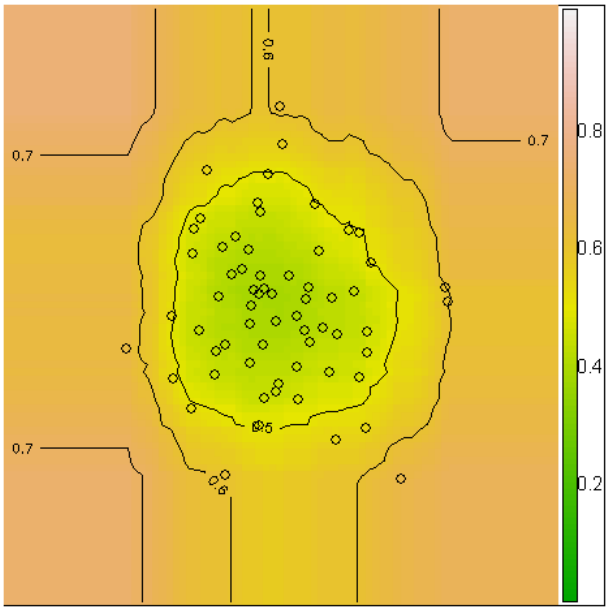


Figura 6-10.: Ejemplo de mapa de contorno o calor. Imagen tomada de [12].

Una aplicación interesante de este método es la detección de transacciones anómalas en entidades financieras. Se plantea una idea para detección de anomalías de forma preventiva o post mortem. Considere la serie de tiempo dada por el histórico del saldo del afiliado en una ventana de tiempo, por producto o saldo global, lo cual depende del objetivo del modelo. Si la serie de tiempo del saldo tiene m entradas y se tienen n afiliados, se obtiene una matriz de datos D con n observaciones en $\mathbb{R}^m$. En primer lugar será recomendable realizar segmentación de los afiliados, para esto se pueden definir diferentes distancias apropiadas para las series de tiempo:

- Le métrica de L_p o la métrica Euclídea. Permiten tener una medida de distancia común o fácil de comparar.
- La métrica de Canberra es una versión ponderada de L_1, que mide los cambios proporcionales de la serie de tiempo:
$$d(x,y) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i| + |y_i|}.$$
- Usando el índice de correlación de Pearson, se puede definir una distancia acotada en $[0,2]$, útil para definir una afinidad basada en la correlación:
$$d(x,y) = 1 - \rho(x,y) = 1 - \frac{\sigma_{xy}}{\sigma_x \sigma_y} = 1 - \frac{E[(x-\mu_x)(y-\mu_y)]}{\sigma_x \sigma_y}.$$

Con alguna de estas métricas se puede usar cualquier algoritmo de agrupamiento visto en la sección 4 que admita cambio de métrica. Puede ser K-medias o agrupamiento jerárquico.

Observación 6.3. *En ocasiones, hacer comparación punto a punto de las series de tiempo no es realmente apropiado. Si para la aplicación se quiere que una serie de tiempo comparada con una translación horizontal de la misma serie tengan una distancia cercana a cero, será necesario definir una métrica dinámica. Para esto se suele usar una comparación de series de tiempo conocida como* ***Dynamic Time Warping****, que da origen a una métrica donde no se hace comparación punto a punto [22]. Lo cual provoca una clasificación donde los afiliados son segmentados en una misma categoría si tienen saldos parecidos y hacen operaciones (retiros y pagos) similares en la misma ventana tiempo.*

Supongamos que la detección de anomalías se quiere efectuar sobre tarjetas de crédito. Entonces para hacer una detección post mortem de los clientes con transacciones atípicas, bastará solo usar el método de bosques de aislamiento para cada uno de los segmentos. Incluso pueden agregarse variables intrínsecas a la serie de tiempo, como la media, el número de picos, la transformada de Fourier discreta, etc. Ahora, si se quiere una alerta en tiempo real, se puede adicionar otras variables exógenas a la serie de tiempo y sus variables intrínsecas: el monto de la transacción, hora del día, mercado en el que efectúa la transacción, distancia física si aplica, etc. Acá el bosque de aislamiento podrá dar una medida de riesgo de fraude de la transacción en tiempo real para cada segmento.

6.3. Otros Métodos

Existen otros algoritmos que son referentes en la detección de anomalías. Con el desarrollo de este trabajo, estos métodos se pueden considerar complementarios:

- **Maquinas de Soporte Vectorial:** es un método que proyecta los datos en un espacio de dimensión mayor usando un kernel, para luego hacer clasificación binaria de los datos por medio de una separación por un hiperplano. Este método es de bajo costo de ejecución, pero en comparación con otros métodos, en general, tiene alta tasa de falsos positivos [19].
- **Elliptical Envelope:** es un método estadístico de aprendizaje no supervisado que se basa en la matriz de covarianza de los datos. La ventaja de este método es que se puede definir un parámetro porcentual llamado contaminación, que representa la cantidad de anomalías que se espera encontrar en los datos. Este método es apropiado si a priori se sabe cual es la tasa de anomalías presente en los datos [9].
- **LOF:** recibe su nombre por las siglas de Local Outlier Factor. Es un método de aprendizaje no supervisado que detecta anomalías en las observaciones de los datos al medir la desviación local de una observación determinada con respecto a sus vecinos. Es un método que se basa en la detección de anomalías en una vecindad, lo cual puede fallar si existen anomalías que se agrupan o están cercanas [2].

A. Anexo: Implementación de compleción de matrices

Suponga que se tiene una matriz 20×40, con un total de 800 entradas, y cada entrada es un número real en $[0, 2.7]$. A esta matriz se le remueven 200 entradas de forma aleatoria para formar un muestreo de la matriz inicial. Se hace compleción de la matriz haciendo minimización de la función Υ con una tolerancia al error de $\epsilon = 0.1$.

Observación A.1. *Se usa la convención de -1 para representar las entradas de la matriz que fueron removidas, es decir, las entradas ausentes en el muestreo. Además, se usa un paquete auxiliar de Python que ayuda a calcular el mínimo de la función Υ.*

En primer lugar creamos una matriz de bajo rango de forma aleatoria.

```
# -*- coding: utf-8 -*-
"""
Este algoritmo genera de forma aleatoria
una matriz de bajo rango completa y ademas
almacena la misma matriz con algunos valores
ausentes.
"""

import numpy as np
import pandas as pd
import random

# Se definene las dimensiones de la matrix
rango = 5
non_ranked_entries = 200

shape = (20, rango)
a = np.random.random_sample(100).reshape(shape)

shape = (rango, 40)
b = np.random.random_sample(200).reshape(shape)
```

```
22
23 c = np.dot(a, b)
24
25 # Se seleccionan datos aleatorios para remover.
26 # Por convencion, los valores ausentes se guardan
27 # como -1
28 vector_other = np.concatenate(c)
29 choose = random.sample(range(800), non_ranked_entries)
30 vector_other[choose] = -1
31 matrix_incomplete = vector_other.reshape((20, 40))
32
33 # Se almacenana las dos matrices
34 # Matriz completa para medir el error
35 pd.DataFrame(c).to_csv("complete_matrix.csv",
36                        index=False)
37 # La matriz a reconstruir
38 pd.DataFrame(matrix_incomplete).to_csv("incomplete_matrix.csv",
39                                        index=False)
```

Ahora, aprovechando las funciones de optimización del paquete **Scipy**, se realiza la compleción de la matriz y la estimación del error de compleción.

```
1 # -*- coding: utf-8 -*-
2 import numpy as np
3 from scipy.optimize import minimize
4 import pandas as pd
5
6 # Se carga la matrices
7 ranking_matrix = pd.read_csv("incomplete_matrix.csv").values
8 complete_matrix = pd.read_csv("complete_matrix.csv").values
9
10 # Se almacenan  indices de
11 # las entradas incompletas de la matriz
12 entries_incomplets_positions = ranking_matrix < 0
13
14
15 def objective_function(m):
16     """
17     Se define la funcion Upsilon. Es la funcion que es objeto
18     de la optimizacion
19     :param m: Matriz
20     :return: Funcion upsilon evaluada en la matriz m
```

```
    """
    # Se completa la matriz incial del muestreo,
    # con los valores de m
    new_rank_matrix = ranking_matrix.copy()
    new_rank_matrix[entries_incomplets_positions] = m

    # Se hace la descomposicion en valores singulares
    # usando el paquete numpy
    # A = U*diag(s)*V, donde A es la matriz
    # m[user_entries_positions]
    u, s, v = np.linalg.svd(new_rank_matrix)

    # Retornamos entonces la suma de los valores singulares
    # que estan en la diagonal de la matriz s.
    return sum(s)

# Se cran las restricciones de la optimizacion:
# La matriz tiene entradas observadas entre 0 y 2.7.
# Por lo tanto se usa esta restriccion optimizar
# el proceso de complecion
cons = ({'type': 'ineq',
         'fun': lambda x: np.array(x)},
        {'type': 'ineq',
         'fun': lambda x: np.array(2.7 - x)}
        )

# Se hace las minimizacion con restricciones
res = minimize(objective_function, np.zeros(sum(sum(
    entries_incomplets_positions))),
               method="SLSQP", constraints=cons)


# Se construye la matriz a partir del vector
# que optimisa la funcion Upsilo
min_matrix = ranking_matrix.copy()
min_matrix[entries_incomplets_positions] = res.x

# Se guarda la matriz
pd.DataFrame(min_matrix).to_csv("computed_matrix.csv",
                                index=False)

"""
```

```
Se considera un dato mal reconstruido si su diferencia
con la entrada real de la matriz supera la tolerancia
"""

# Se calcula el error de la recuperacion de la matriz
resta = np.abs(complete_matrix - min_matrix)
number_of_wrong_classifications = sum(sum(resta >= 0.1))
print "Reconstruye mal  el {0:.2f}% de los datos removidos. " \
      "".format(number_of_wrong_classifications * 100.0 / 200)

# Se calcula el error de la recuperacion de la matriz
resta = np.abs(complete_matrix - min_matrix)
total_unclasified = complete_matrix[ranking_matrix == -1].sum()
total_errors = resta[ranking_matrix == -1].sum()
print "Los datos mal caclulados difieren " \
      "en promedio {0:.2f}% de los reales" \
      "".format(total_errors*100.0/total_unclasified)
```

Las entradas bien reconstruidas con este algoritmo, basados en la tolerancia ϵ, serán mayores al $99\,\%$. En otras palabras, el algoritmo sólo se desviará en menos del $1\,\%$ de las entradas ausentes cuando realiza la reconstrucción. Además, la desviación porcentual de las entradas reconstruidas más allá de la tolerancia no superan el $0.8\,\%$.

B. Anexo: Implementación de PCA

Se tiene una matriz de 64 columnas que representa unos datos. Se le aplica el algoritmo de PCA para reducir y obtener datos de sólo 3 variables por observación.

```
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
# Los siguientes son paquetes auxiliares.
# Paquete para estandarizar la matriz
from sklearn.preprocessing import StandardScaler
# Paquetes para la optimizacion y el tratamiento
# algebraico de la matriz
from scipy.optimize import minimize
from scipy.linalg import eigvals, norm

# Se toman datos de 1797x63
data = pd.read_csv('numbers.csv')
# Se convierte el DataFrame a matriz
X = data.values

# Dimensiones de la matriz X
rows, cols = X.shape
print 'La matriz X tiene {col} columnas y {row} filas'.format(col=cols,
    row=rows)

# Se estandariza la matriz
scaler = StandardScaler(copy=True, with_mean=True, with_std=True).fit(X
    )
X = scaler.transform(X)
X = np.matrix(X)

# Se define la funcion a optimizar
A = X.transpose() * X
def fun(x):
    xx = np.matrix(x)
```

```
30     return - (xx * A * xx.transpose())[0, 0] / (xx * xx.transpose())[0,
           0]
31
32 # Note que se define la funcion en negativo, porque el paquete de
       sklearn
33 # solo trae minimizacion, por lo tanto para hallar el maximo de una
34 # funcion se busca de forma equivalente el minimo del negativo de la
       funcion
35
36 # Primera componente principal
37 # Punto inicial
38 x0 = np.array([1.0 / (cols ** 0.5)] * cols)
39 # Se halla el maximo de la funcion -fun
40 res = minimize(fun, x0, method='Powell')
41
42 # Maximo valor que alcanza la funcion -fun
43 sg1 = -res.fun
44 # Se normaliza el w para definir la primera componente principal
45 w1 = np.matrix(res.x / np.linalg.norm(res.x))
46 # Se imprime el maximo de la funcion -fun
47 # y la primer componente principal
48 print np.round(sg1)
49 print w1
50
51 # Note que el maximo coincide con el mayor valor propio de A
52 print 'Mayor valor propio de A: ', max(eigvals(A))
53
54
55 # Segunda componente principal
56 X2 = X - X * w1.transpose() * w1
57 A = X2.transpose() * X2
58 x0 = np.array([1.0 / (cols ** 0.5)] * cols)
59 res = minimize(fun, x0, method='Powell')
60 sg2 = -res.fun
61 w2 = np.matrix(res.x / np.linalg.norm(res.x))
62 # La segunda componente principal
63 print np.round(sg2)
64 print w2
65 # Note que el maximo coincide con el mayor valor propio de A
66 print 'Mayor valor propio de A :', max(eigvals(A))
67
68 # Tercera componente principal
```

```
69 X3 = X2 - (X * w1.transpose()) * w1
70 A = X3.transpose() * X3
71 x0 = np.array([1.0 / (cols ** 0.5)] * cols)
72 res = minimize(fun, x0, method='Powell')
73 sg3 = -res.fun
74 w3 = np.matrix(res.x / np.linalg.norm(res.x))
75 # La tercera componente principal
76 print np.round(sg3)
77 print w3
78 # Note que el maximo coincide con el mayor valor propio de A
79 print 'Mayor valor propio de A :', max(eigvals(A))
```

C. Anexo: Implementación de K-medias

Se generó un conjunto de datos de forma manual para probar k-medias. Se replica el algoritmo de de Lloyd para calcular los centroides con $\epsilon = 1e^{-7}$ y $w \leq 20$.

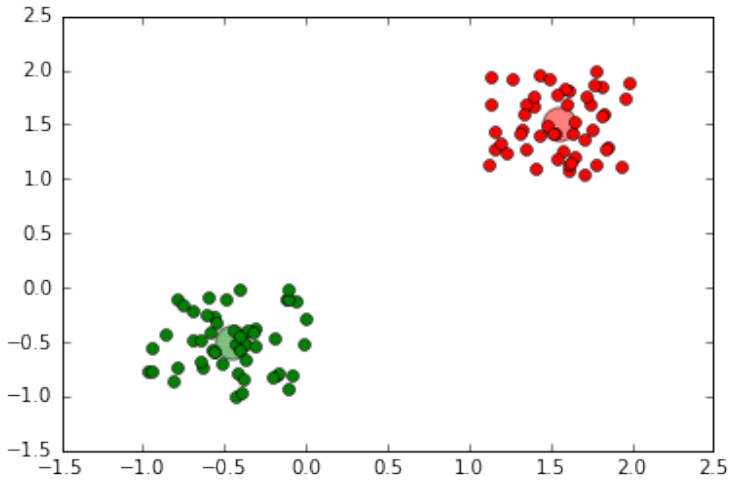


Figura C-1.: Resultado del agrupamiento con k=2.

El algoritmo usado en Python 2.7 se muestra a continuación.

```
# -*- coding: utf-8 -*-
import numpy as np
import random
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("random_data_kmeans.csv").values
n_clusters = 2

def calcule_new_centers(prev_centers):
    """
```

```
12          Esta funcion calcula los nuevos centroides dados los anteriores
13          segun el algoritmo de Lloyd.Lo que hace es encontrar para cada
14          punto el centro mas cercano. Al final, halla la media
15          aritmetica de los puntos que tienen un centro en comun y esa
16          media sera un nuevo centro.
17
18      :param np.ndarray prev_centers: Centroides de la iteracion anterior
19      :return: Numpy array con la lista de centroides actuales
20      """
21      # En este diccionario las keys significan el i-esimo centro y
22      # los valores seran (despues del proximo for)
23      # el conjunto de puntos que estan mas cerca del centro i
24      # que de cualquier otro centro.
25      cluster_points = {i: [] for i in range(n_clusters)}
26      for point in data:
27          distances = [np.linalg.norm(point - center) for center in
                prev_centers]
28          min_cluster_index = min(range(n_clusters), key=lambda i:
                distances[i])
29          cluster_points[min_cluster_index].append(point)
30
31      new_centers_ = []
32      for i_ in range(n_clusters):
33          new_centers_.append(sum(cluster_points[i_])*1.0/len(
                cluster_points[i_]))
34      return np.array(new_centers_)
35
36  # Se definen los centroides iniciales, maximo de iteraciones
37  # y la toleracia epsilon
38  centers = random.sample(data, n_clusters)
39  max_iterations = 20
40  tol = 1e-7
41
42  # Se realiazan las iteraciones
43  for i in range(max_iterations):
44      new_centers = calcule_new_centers(centers)
45      center_distances = [np.linalg.norm(nc - c) for c, nc in zip(
            new_centers, centers)]
46      if sum(center_distances) < tol:
47          break
48      centers = new_centers
49
```

```
# Se asigna la clacificacion
classification = []
for point in data:
    distances = [np.linalg.norm(point - center) for center in centers]
    min_cluster_index = min(range(n_clusters), key=lambda i: distances[
        i])
    classification.append(min_cluster_index)
classification = np.array(classification)

# Se hace una graficaca de los puntos donde cada color
# represente una categoria de clasifiacion distinta
for i, c in zip(range(n_clusters), 'rgbkcmyw'[:n_clusters]):
    group = data[classification == i]
    plt.plot(group[:, 0], group[:, 1], 'o', c=c)
    plt.scatter(centers[i][0], centers[i][1], 300, c=c, alpha=0.5)

plt.show()
```

C.0.1. Aplicación: descomposición de imágenes

Cuando una imágenes tiene poca variabilidad en sus colores, con contrates y se conocen la cantidad de objetos que se quieren extraer de la imagen, se puede usar K-medias para separa la imagen. Lo primero que se debe hacer es descomponer la imagen en colores primarios, lo que genera una matriz conocida como **matriz RGB**. Para el ejeplo siguiente se tomó una imagen y se le efectúo K-medias con $k = 3$ y $\epsilon = 1\text{e}^{-10}$ para lograr una descomposición de los elementos de la imagen[1].

(a) Imagen original. (b) Clase 1 en K-medias. (c) Clase 2 en K-medias. (d) Clase 3 en K-medias.

Figura C-2.: Descomposición de imágenes usando K-medias.

Para imágenes con mayor complejidad se suele usar agrupamiento espectral y se quiere hacer comparaciones en imágenes, es usual utilizar propagación de afinidad.

[1]Imagen tomada de www.freshwallpapers.net

D. Anexo: Implementación de Mean shift

Para un conjunto de datos de muestra se desarrolló el algoritmo de agrupamiento Mean shift en Python. Estos datos fueron creados de forma ilustrativo. El algoritmo funcionará para cualquier conjunto de datos, excepto la gráfica de las observaciones que está diseñada para datos bidimensionales.

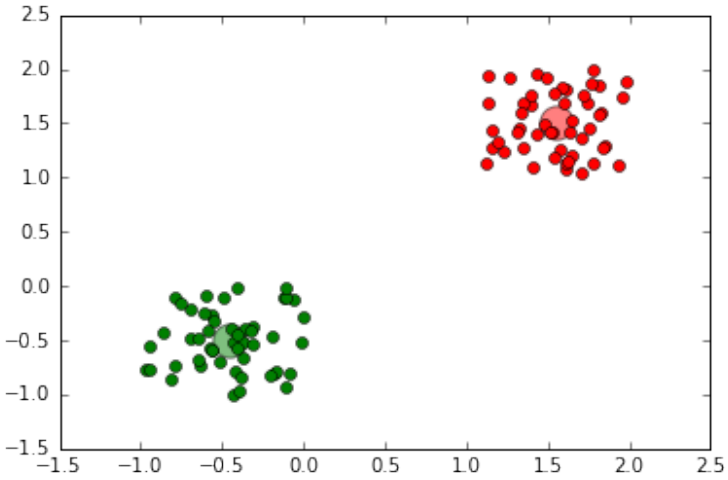


Figura D-1.: Resultado del agrupamiento con Mean shift.

```
# coding: utf-8
import numpy as np
# Paquete auxiliar para las graficas
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("random_data_meanshift.csv").values

current_points = data
max_iterations = 30 # Limite de iteraciones
tol = 1e-6 # Tolerancia de comparacion
```

```
12 delta = 1 # radio de la vecindad
13 epsilon = delta*1e-6
14 points_classification = range(len(current_points))
15
16 # Definimos la funcion de kernel gaussiano.
17 def kernel(x):
18     return 1.0/(2*np.pi)*np.exp(-0.5*np.power(np.linalg.norm(x), 2))
19
20 def remove_similar_clusters(original_points_classification,
       clusters_representantes):
21     """
22         Esta funcion remueve puntos de transformation_points
23         que esten a una distancia menor que epsilon.
24         Crea entonces una nueva lista con los puntos
25         representantes y calcula la nueva clasificacion para
26         los puntos originales en terminos de los nuevos
27         representantes.
28     """
29     # En current_images, la posicion i-esima correspondera con j (
           despues del siguiente bucle),
30     # donde j es el indice del cluster que esta a distancia menor que
           epsilon de i.
31     current_images = range(len(clusters_representantes))
32     for i in range(len(clusters_representantes)):
33         for j in range(i+1, len(clusters_representantes)):
34             p1 = clusters_representantes[i]
35             p2 = clusters_representantes[j]
36             if all(p1 - p2 < epsilon):
37                 current_images[j] = i
38
39     cluster_indexes = list(set(current_images))
40     new_transformation_points = [clusters_representantes[index] for
           index in cluster_indexes]
41     new_points_classification = [cluster_indexes.index(current_images[
           classiff]) for classiff in original_points_classification]
42     return np.array(new_points_classification), np.array(
           new_transformation_points)
43
44
45 def get_new_data_points(points):
46     """
47         Aqui se calculan nuevos representantes de clusters
```

```
48          basado en el delta global ya definido.
49      """
50      new_points = []
51      for point in points:
52          cluster = [a for a in points if np.linalg.norm(a-point) < delta
                ]
53          int1 = sum([kernel(x-point)*x for x in cluster])
54          int2 = sum([kernel(x-point) for x in cluster])
55          new_points.append(int1*1.0/int2)
56      return np.array(new_points)
57
58
59  for i in range(max_iterations):
60      # Se aplican el paso iterativo a los nuevos puntos
61      transformation_points = get_new_data_points(current_points)
62
63      # Se eliminan los representantes que esten a una distancia
64      # muy cercana
65      points_classification, new_calculated_points =
            remove_similar_clusters(points_classification,
            transformation_points)
66      current_points = new_calculated_points
67
68  # Se hace la grafica de los datos
69  for i, c in zip(range(len(current_points)), 'rgbkcmyw'[:len(
        current_points)]):
70      group = data[points_classification == i]
71      plt.plot(group[:, 0], group[:, 1], 'o', c=c)
72      plt.scatter(current_points[i][0], current_points[i][1], 300, c=c,
            alpha=0.5)
73
74  plt.show()
```

Bibliografía

[1] BOUTSIDIS, C. ; GITTENS, A. ; KAMBADUR, P.: Spectral Clustering via the Power Method - Provably. (2015)

[2] BREUNIG, M. ; KRIEGEL, H. ; R.NG ; SANDER, J.: LOF: Identifying Density-Based Local Outliers. (2000)

[3] CAM, L. L.: Maximum Likelihood; An Introduction. (1990)

[4] CANDÈS, E. ; RECHT, B.: Exact Matrix Completion via Convex Optimization. (2018)

[5] D. COMANICIU, P. M.: Mean shift: a robust approach toward feature space analysis. (2002)

[6] DUECK, D. ; FREY, B. J.: Non-metric affinity propagation for unsupervised image categorization. (2007)

[7] FAWCETT, T: An introduction to ROC analysis. (2005)

[8] FUKUNAGA, K. ; HOSTETLER, L.: The Estimation of the Gradient of a Density Function, with Applications in Pattern. (1975)

[9] HOYLE, B. ; RAU, M. ; PAECH, K. ; BONNETT, C. ; SEITZ, S. ; WELLER, J.: Anomaly detection for machine learning redshifts applied to SDSS galaxies. (2010)

[10] ISOMURA, T. ; TOYOIZUMIB, T.: A Local Learning Rule for Independent Component Analysis. (2016)

[11] LAX, PETER D.: *Linear Algebra and Its Applications*. New Jersey : WILEY, 2007

[12] LIU, F. ; TING, K. ; ZHOU, Z.: Isolation Forest. (2008)

[13] MARÍN, J.: Análisis de Componentes Principales, Departamento de Estadística, Universidad Calor III de Madrid.

[14] MCLACHLAN, G. ; PEEL, D.: *Finite Mixture Models*. Wiley : New York, 2000

[15] MOUJAHID, A. ; INZA, I ; LARRA, P: Clasificadores K-NN, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad del País Vasco.

[16] ROKACH, L. ; MAIMON, O.: *Clustering Methods*. US : Springer, 2005

[17] ROKACH, L. ; MAIMON, O.: *Data mining with decision trees: theory and applications*. US : World Scientific Pub Co Inc., 2008

[18] SALAS, M.: La regresión logística . Una aplicación a la demanda de estudios universitarios. (1996)

[19] SAMMUT, C. ; WEBB, G.: *Encyclopedia of Machine Learning*. Boston : Springer, 2010

[20] SANDER, J. ; ESTER, M. ; H.KRIEGEL ; XU, X.: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. (1997)

[21] SHI, J. ; MALIK, J.: Normalized Cuts and Image Segmentation. (2000)

[22] SILVA, D. ; BATISTA, G.: Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation. (2015)

[23] VILLARREAL, F.: Introducción a los Modelos de Pronósticos, Departamento de Matemática, Universidad Nacional del Sur.