Facultad de Ciencias

Estudio del rendimiento en ciclistas mediante técnicas del Aprendizaje Automático

A study of cyclists's perfomance through Machine Learning techniques

Trabajo de Fin de Grado
para acceder al

GRADO EN MATEMÁTICAS

Autora: Claudia Carpintero Borrajo

Directores: Cristina Tirnauca y Adrian Odriozola

Septiembre-2021

Índice

Resumen

En el presente trabajo se analizarán los datos obtenidos de varios ciclistas que participaron en un experimento. En él se recogieron distintas variables fisiológicas (la saturación de oxígeno en la sangre, la concentración de lactato, el nivel de glucosa, etc.), medidas antes y después de someter a los ciclistas a tres pruebas de esfuerzo.

Como primer objetivo, se busca comprender si hay diferencias en el valor de las variables seleccionadas entre atletas de distintas categorías (en el estudio había tanto ciclistas profesionales como amateurs), y agrupar los participantes en dos subconjuntos lo más homogéneos posibles para identificar los rasgos comunes de cada uno de los grupos hallados. En segundo lugar, se pretende encontrar un modelo capaz de predecir el rendimiento de los individuos (en este estudio nos centramos en el valor de potencia media registrado por el potenciómetro de la bicicleta en un trayecto de montaña con pendiente pronunciada durante los 20 minutos de la prueba). Esto permitiría definir entrenamientos personalizados para los deportistas y lograr así mejoras en su rendimiento o evitar algunos problemas clásicos en los deportistas como es el sobreentrenamiento.

Para dar respuesta a las cuestiones planteadas, se emplearán distintas técnicas de aprendizaje supervisado y no supervisado: agrupamiento (*clustering*), clasificación y regresión. En particular, se estudiarán algunos de los algoritmos más conocidos como son: k-medias (k-*means*), *Expectation-Maximization*, k vecinos más cercanos (k-*Nearest Neighbor*), máquinas de soporte vectorial, entre otros. De cada algoritmo se estudiarán sus fundamentos teóricos, así como sus propiedades más relevantes.

Aunque el *dataset* es bastante pequeño como para que los resultados del estudio sean concluyentes, las técnicas empleadas y el software desarrollado son perfectamente escalables en caso de disponer de una cantidad mayor de datos.

Palabras clave: Aprendizaje Automático, *clustering*, clasificación, regresión y visualización de datos.

Abstract

This paper will analyse data obtained from several cyclists who participated in an experiment. This experiment collected different physiological variables (blood oxygen saturation, lactate concentration, glucose level, etc.), that were measured before and after subjecting the cyclists to three performance tests.

As a first objective, we seek to understand whether there are differences in the values of selected variables between athletes of different categories (in the study there were both professional and amateur cyclists), and to group the participants into two subsets as homogeneous as possible in order to identify the common features of each of these groups. Secondly, we would like to find a model capable of predicting the performance of individuals (in this study we focused on the average power value recorded by the bicycle's potentiometer on a steep mountain road during the 20 minutes of the test). This would make it possible to define personalized training for athletes (and thus achieve improvements in their performance) or avoid some classic problems in athletes such as overtraining.

In order to answer the questions posed, different supervised and unsupervised learning techniques will be used: clustering, classification and regression. In particular, some of the best known algorithms will be studied, such as: k-means, expectation-maximization, k-Nearest Neighbours, support vector machines, among others. The theoretical foundations of each algorithm will be studied, as well as their most relevant properties.

Although the dataset is not big enough for the results of the study to be conclusive, the techniques used and the software developed are perfectly scalable in case of having a larger amount of data.

Keywords: Machine Learning, clustering, classification, regression and data visualization.

1. Introducción

Hoy en día, debido a los grandes avances tecnológicos, se dispone de cantidades ingentes de datos de los que se puede extraer mucho conocimiento si se cuenta con las herramientas adecuadas, como es el caso de las técnicas y métodos de Aprendizaje Automático (en inglés, *Machine Learning*). Se trata de un campo que pertenece a la Inteligencia Artificial y se encuentra detrás de numerosas aplicaciones cotidianas como las recomendaciones de películas en Netflix, reconocimiento facial o de voz, la creación de vehículos autónomos, etc. En base a lo escrito por Arthur Samuel en [25], puede definirse como un "campo de estudio que da a las computadoras la capacidad de aprender sin ser programadas de forma explícita" (véase [20]). Por lo tanto, la investigación en este área busca crear sistemas capaces de aprender por ellos mismos, con la finalidad de predecir hechos futuros, realizar recomendaciones, clasificaciones de datos, eventos o etiquetas. Para ello se utilizan distintas técnicas y algoritmos que crean modelos predictivos, patrones de comportamiento, etc.

Un sistema de Aprendizaje Automático se nutre de experiencias y evidencias en forma de datos (*dataset*), con los que interpretar patrones y/o comportamientos. Dentro de las distintas formas de caracterizar el Aprendizaje Automático se encuentran [22, pgs. 1-11]:

- Aprendizaje supervisado: tiene como objetivo principal crear un modelo que aprenda de un conjunto de datos etiquetados para poder realizar predicciones sobre datos que no pertenecen al conjunto de entrenamiento. Una de las más típicas aplicaciones de este tipo de aprendizaje es la clasificación de correos en deseados o no deseados (en inglés, *ham* o *spam*). Dentro del aprendizaje supervisado, dependiendo de si el valor a predecir es discreto o continuo, se encuentran dos subcategorías: clasificación y regresión, respectivamente.

- Aprendizaje no supervisado: en este caso, se parte de datos con una estructura desconocida. Mediante técnicas específicas, se explora la disposición de los mismos con el objetivo de encontrar información valiosa.

- Aprendizaje reforzado: pretende desarrollar un sistema (agente) que mejore su actuación a través de interactuar con el ambiente. Un ejemplo típico de este aprendizaje es el jugador virtual de ajedrez.

En este trabajo nos centraremos en los primeros dos paradigmas enumerados. En particular, se va a estudiar el problema del agrupamiento (*clustering*) dentro del aprendizaje no supervisado y los problemas de regresión y clasificación en el seno del aprendizaje supervisado, a los que se intentará dar respuesta mediante distintos algoritmos de aprendizaje.

El objetivo de este trabajo es estudiar, mediante técnicas y algoritmos de Aprendizaje Automático, la fatiga en los ciclistas, así como su rendimiento. Para ello contaremos con los datos del experimento "The limits of the performance during prolonged endurance exercise" [33] (comunicación personal, aún sin publicar). Se considera fatiga deportiva, según [1], el estado en el que el deportista no puede mantener el nivel de rendimiento de la práctica deportiva para llegar al alto rendimiento. Debido a la cantidad de factores que pueden influir en la fatiga deportiva, aún es escaso el conocimiento que se tiene sobre el tema. Un problema que poseen muchos deportistas es el síndrome de sobreentrenamiento o síndrome de fatiga crónica. Suele ser el resultado de un largo e intenso proceso de entrenamiento que ocasiona un estado permanente de fatiga que lleva al sobreentrenamiento. Este es un tipo de fatiga global. En función de dónde se encuentre la fatiga, puede tratarse de fatiga central o fatiga periférica; la que se entiende por fatiga central afecta a nuestro sistema nervioso central, mientras que la segunda se conoce como fatiga física y afecta a nuestros músculos.

Los individuos que participaron en el experimento en el que se basa este trabajo son ciclistas; en esta modalidad, el sobreentrenamiento más frecuente es el parasimpático (es difícil de identificar y generalmente se diagnostica tarde). En este caso, el atleta aparentemente presenta buena salud (buena alimentación, no hay pérdida de peso o insomnio, puede dormir de hecho más de lo normal). Los principales síntomas del sobreentrenamiento en ciclistas son los siguientes:

- Disminución de la frecuencia cardíaca en reposo

- Rápida recuperación de la frecuencia cardíaca post-esfuerzo

- Hipoglucemia durante esfuerzos

- Síntomas depresivos

- Disminución de niveles de ácido láctico máximos

Para diagnosticar dicho síndrome, la capacidad de realizar tareas rutinarias debe verse reducida en un 50 %. Con el fin de evitar este y otros problemas en los atletas, los entrenadores y deportistas necesitan herramientas prácticas, económicas y útiles. Con ellas se podría determinar el grado de respuesta fisiológica y mecánica a la fatiga, y así adaptar los entrenamientos de forma individual a cada deportista. Y para ello hace falta tener datos y saber analizarlos.

El trabajo comienza con la explicación de los fundamentos teóricos de las técnicas a aplicar (Sección 2). En la Sección 2.1 se introduce el Aprendizaje no supervisado y dentro de ella, en las Secciones 2.1.1 y 2.1.2, se detallan los algoritmos *k-means* y *Expectation-Maximization*, respectivamente. En la Sección 2.2 se trata el Aprendizaje supervisado y en la Sección 2.2.1 se introducen las máquinas de soporte vectorial, el algoritmo *k-Nearest Neighbor* y el método de regresión logística. Finalmente, en la Sección 2.3, se aborda la técnica de reducción de dimensiones t-SNE. En la Sección 3 se discute todo lo referente al análisis del *dataset* obtenido del experimento [33], que consiste en un conjunto de variables de un grupo de ciclistas al realizar una serie de pruebas. Se comienza con la descripción del experimento, sus variables y una visualización del *dataset* mediante la técnica t-SNE anteriormente mencionada (Secciones 3.1.1 y 3.1.2). Con el objetivo de tener una visión mas clara y global del *dataset* con el que contamos, se realiza un análisis descriptivo en la Sección 3.2. A continuación, en la Sección 3.3, se aplican los distintos algoritmos explicados, se comienza con los algoritmos de Aprendizaje supervisado (máquinas de soporte vectorial, *k-Nearest Neighbor* y regresión logística) en la Sección 3.3.1 y en la Sección 3.3.2 se exponen algunas variables interesantes a raíz de la aplicación de los anteriores algoritmos así como las posibles relaciones que guardan con el rendimiento y la fatiga. Se continua intentando dar respuesta al problema de *clustering* mediante los algoritmos de *k-means* y *Expectation-Maximization* en la Sección 3.3.3. Para concluir, en la Sección 3.4 se exponen unas conclusiones sobre los resultados obtenidos (Sección 3.4.1) y sobre el trabajo a nivel personal (3.4.2).

Para terminar la introducción, se exponen los objetivos del trabajo.

A nivel del experimento:

- Entender la influencia de las variables en la fatiga y en el rendimiento de los ciclistas participantes en el experimento [33].
- Conseguir buenos modelos que permitan predecir el rendimiento de los ciclistas (en particular, la variable de potencia media).

En el ámbito teórico:

- Comprender el funcionamiento de los distintos modelos de Aprendizaje Automático en función de algunos de sus parámetros.

2. Marco teórico

2.1. Aprendizaje no supervisado

Un problema clásico dentro del aprendizaje no supervisado es el problema del agrupamiento. Según [22, pg. 353], este problema (*clustering*) consiste en lo siguiente: partiendo de una serie de datos, se busca en ellos estructuras en función de sus posibles similitudes. Por tanto, los elementos que pertenezcan al mismo "grupo" presentarán cierta similitud en algún sentido y los elementos de distintos "grupos" no la guardarán (o será notablemente menor). A cada grupo identificado se le llama *cluster* y se define como el *problema de agrupamiento* la tarea de dividir el *dataset* en *clusters*.

En [2, pg.153] se describen muchas situaciones de la vida real en las que partir un conjunto de datos en subconjuntos afines puede traer grandes beneficios, pues ayuda a resumir los datos y entenderlos mejor. Algunas de las aplicaciones son: la segmentación de clientes para la adecuación de productos, en campañas de marketing y el análisis de redes sociales (en este caso, los *clusters* pueden estar formados por personas que pertenecen a la misma comunidad, al mismo núcleo familiar o mismo grupo de amigos y podría recopilar información sobre los amigos o relaciones de cada persona con el fin de sugerir perfiles afines a ellos). Otra gran utilidad del *clustering* es como pre-procesado de datos para luego aplicar modelos de clasificación.

Se ha diseñado una gran variedad de algoritmos para resolver el problema del clustering; las diferencias entre los principales modelos radica en las formas en la que se determina la similitud entre los datos para construir los diferentes *clusters*. En [24] se hace una clasificación de los principales métodos de *clustering* y se explican las ideas que hay detrás de cada método:

- Métodos basados en representantes (*representative-based methods*): Definen la similitud mediante distancias.
- Métodos jerárquicos (*hierachical methods*): Se particiona el *dataset* de forma consecutiva a distintos niveles de agrupación (jerarquías).
- Métodos probabilísticos (*probabilistic-methods*): La pertenencia de un dato a un *cluster* no es excluyente (se determina la probabilidad de pertenencia a cada uno de los clusters existentes).
- Métodos basados en densidad (*density-based methods*): La idea es detectar las zonas en las que hay mayor concentración de puntos y dónde se encuentran más dispersos; estos métodos son beneficiosos porque al no tener la noción de distancia pueden determinar *clusters* con formas arbitrarias.
- Métodos basados en cuadrículas (*graph-based methods*): Discretizan el espacio del objeto en un número de cuadrículas finito que se derivan de una estructura cuadriculada; su principal ventaja es la rapidez.

2.1.1. Algoritmo *k-means*

El algoritmo *k-means*, también conocido como heurístico de *Lloyd*, pertenece a la familia de los métodos basados en representantes. Son los más sencillos ya que se apoyan en la noción intuitiva de distancia para definir la similitud entre puntos. El objetivo del algoritmo es minimizar el error cuadrático de aproximar cada punto del *dataset* por su representante más cercano.

k-means comienza con la inicialización de los k representantes $\{c_1, \ldots, c_k\}$ que llamaremos centroides (k será el número de *clusters* determinado por el usuario), que forman el conjunto S, mediante un heurístico claro (sin ambigüedad). Puede ser, por ejemplo, una muestra k-dimensional aleatoria del *dataset* $D = \{(x_i)_{i=1}^n\} \subseteq \mathbb{R}^d$. Se continúa asignando datos a representantes iterativamente de la siguiente forma (tal como está descrito en [2, pgs. 207-210]):

- **Paso de asignación:** El algoritmo comienza asignando cada dato a su centroide más cercano del conjunto S, utilizando la función de distancia euclídea. Se forman así los *clusters* $C_1, \ldots, C_k$. El conjunto C_i será el formado por los puntos del *dataset* que se encuentren más próximos al punto c_i que al resto de elementos de S.
- **Paso de optimización:** Se determinan los centroides óptimos c_i de cada *cluster* C_i como aquellos que minimizan la función $\sum_{x_j \in C_i} Dist(x_j, c_i)^2$, donde $\text{Dist}(x, y) = \|x - y\|_2$ es la distancia euclídea. Al derivar la función a minimizar respecto de cada centroide e igualando a 0, se obtiene que el centroide de cada *cluster* se recalcula como la media aritmética de los puntos del *cluster*. Esto es,

$$c_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \tag{1}$$

  con $|C_i|$ el número de elementos del *cluster* C_i.

Este proceso iterativo se repetirá hasta que se alcancen el número máximo de iteraciones permitidas por el usuario o hasta que la diferencia entre los centroides de cada *cluster* de dos iteraciones consecutivas sea menor que cierto umbral ϵ. El segundo criterio de parada puede expresarse en función de la iteración t en la que se encuentre el algoritmo como

$$\sum_{i=1}^{k} (\|c_i^t - c_i^{t-1}\|_2)^2 \leq \epsilon \tag{2}$$

El pseudocódigo del algoritmo *k-means* encontrado en [31] se muestra en el Algoritmo 1. A continuación, se detalla la **complejidad computacional** del algoritmo [31, pgs. 369-374] en términos del número de clusters (k), tamaño del *dataset* (n) y la dimensión de los puntos del *dataset* (d). El paso de asignación requiere una complejidad temporal $\mathcal{O}(nkd)$, debido a que para cada elemento del *dataset* se ha de calcular la distancia a cada centroide, lo que requiere d operaciones pues los datos se encuentran en un espacio d-dimensional. El paso de actualización del centroide requiere una complejidad temporal $\mathcal{O}(nd)$, ya que se deben añadir n puntos d-dimensionales. La complejidad temporal total del algoritmo *k-means* es, por tanto, $\mathcal{O}(nkd)$.

Algoritmo 1 Algoritmo *k-means*

Entrada: *dataset* D, número de *clusters* k, tolerancia ϵ

$t = 0$
Inicialización aleatoria de los centroides $c_1^0, \ldots, c_k^0 \in D$
repetir
  //Paso de asignación del *cluster*
  para todo $x_j \in D$ **hacer**
    $j* \leftarrow \text{argmin}_i\{\|x_j - c_i^t\|^2\}$
    //Se asigna x_j al centroide más cercano
    $C_{j*} \leftarrow C_{j*} \cup x_j$
  fin para
  //Paso de actualización de los centroides
  para i=1 hasta k **hacer**
    $c_i^{t+1} \leftarrow \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$
  fin para
  $t \leftarrow t + 1$
hasta que $\sum_{i=1}^{k} \|c_i^t - c_i^{t-1}\|^2 \leq \epsilon$
devolver $(C_1, \ldots, C_k)$

Por otro lado, como se muestra en [28], el problema de encontrar el número de *clusters* óptimos para cualquier *dataset* (es decir, que minimicen globalmente la suma de las distancias euclídeas al cuadrado de cada elemento del *dataset* al centroide más cercano), es un problema NP-duro[1] (en inglés, *NP-hard*, donde las siglas vienen de *nondeterministic polynomial*). Incluso para $k = 2$ o $d = 2$ (fijos), [8] y [17] muestran que sigue siendo un problema NP-duro. Uno de los mejores algoritmos conocidos hasta la fecha logra realizar esta tarea en un tiempo $\mathcal{O}(n^{dk+1})$ [13]; para números concretos de n y d, esta complejidad puede verse reducida (más detalles en [28]).

Algunas de las situaciones en las que el algoritmo *k-means* no funciona correctamente son las explicadas en [3]:

- El caso en el que los *clusters* son de distinto tamaño (Figura 1).
- El caso en el que los *clusters* tienen distinta densidad (Figura 2).
- El caso en el que los *clusters* no son convexos (Figura 3).

Otro tema a destacar en *k-means* es su inicialización. Las dos formas más populares se conocen como *k-means++* y la inicialización de forma aleatoria. La segunda consiste simplemente en seleccionar los centroides aleatoriamente de entre los elementos del *dataset* (se escogen tantos como el número de *clusters* en que se quiere dividir el conjunto de datos). En cambio, *k-means++* es más sofisticada y se resume en los siguientes pasos, como muestra [22, pgs. 358-359]:

- **Paso (1):** Se crea un conjunto vacío S en el que se almacenarán los k centroides.
- **Paso (2):** A continuación, se selecciona aleatoriamente del conjunto D el primer centroide c_1 y se añade a S.
- **Paso (3):** Para cada $x_i \in D \setminus S$ se calcula la distancia euclídea mínima (representada a continuación por $\text{Dist}(x_i, S)$) a cualquiera de los centroides que contiene S.
- **Paso (4):** Para seleccionar aleatoriamente el centroide c_p, se utiliza la probabilidad con distribución ponderada $\frac{\text{Dist}(x_p,S)}{\sum_i \text{Dist}(x_i,S)}$ para todo $x_p \in D \setminus S$ (de esta forma, los puntos x_p más cercanos a los centroides ya elegidos tendrán menos probabilidad de salir elegidos como siguiente c_p).
- **Paso (5):** Se continúan repitiendo los pasos (3) y (4) hasta que se seleccionan los k centroides.
- **Paso (6):** Se procede con el algoritmo *k-means* usual.

[1] Un problema NP-duro es aquel cuyo lenguaje asociado no se ha podido probar que pertenece a la clase *NP*, pero cualquier lenguaje que se encuentre en la clase *NP* puede reducirse polinomialmente a él.

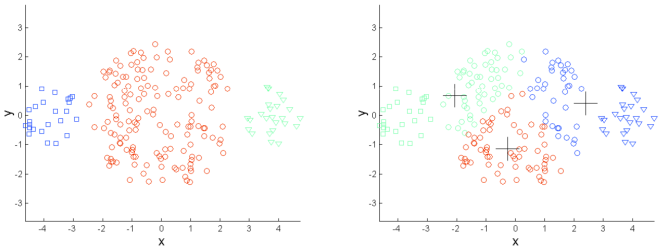


Figura 1: Resultado de *k-means* con un *dataset* en el que los grupos no son de tamaño similar [3]. En la primera gráfica se muestran los datos originales divididos en las tres clases y en la segunda los datos agrupados según el algoritmo.

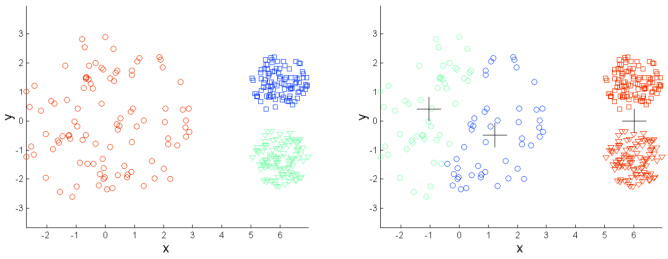


Figura 2: Resultado de *k-means* con un *dataset* en el que los grupos no tienen densidades similares [3]. En la primera gráfica se muestran los datos originales divididos en las tres clases y en la segunda los datos agrupados según el algoritmo.

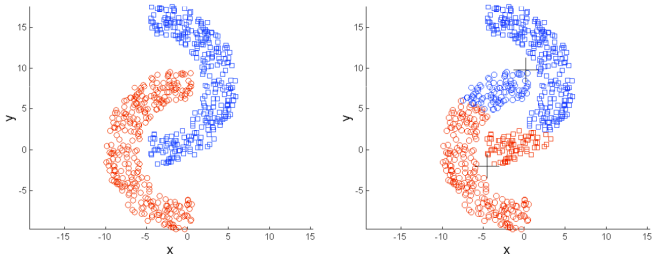


Figura 3: Resultado de *k-means* con un *dataset* en el que los grupos no son convexos [3]. En la primera gráfica se muestran los datos originales divididos en las dos clases y en la segunda los datos agrupados según el algoritmo.

Existen muchas variantes de *k-means* dependiendo de la forma de recalcular los representantes en cada iteración. *k-means* utiliza la media, pero podrían utilizarse:

- **La moda:** Entonces se trata del algoritmo *k-medoids*, y se recalculan en el paso de actualización de los medoides (en este caso se obliga a que los representantes de cada *cluster* pertenezcan al *dataset*, en el caso de *k-means* podría no ocurrir); como ventaja, este método limita la influencia de los *outliers*. Como contrapartida, tiene una complejidad temporal cuadrática en n, el cardinal del *dataset*.
- **La mediana:** En este caso el algoritmo se llama *k-medians*. Funciona bien con cantidades muy grandes de datos, pero supone que los puntos se encuentran normalmente distribuidos.

2.1.2. Algoritmo *Expectation-Maximization* (EM)

El enfoque del *k-means* es un ejemplo de "asignación dura de *clustering*" (en inglés, *hard-assigment clustering*), donde los datos solo pueden pertenecer a un único *cluster*. A continuación, se explicará un enfoque que permite considerar una "asignación blanda" (*soft-assignment*) de los datos a los *clusters* de acuerdo con [31, pgs. 380-396]; así, cada dato pertenecerá con cierta probabilidad a cada uno de los *clusters* (se trata de un método probabilístico). De nuevo, el número de *clusters* es proporcionado por el usuario. Se trata de un modelo basado en la probabilidad.

Sea $D = \{(x_i)_{i=1}^n\} \subseteq \mathbb{R}^d$. Denotemos por X_a una variable aleatoria que corresponde a la variable a-ésima de los datos. Se empleará X_a para denotar el a-ésimo vector columna de la matriz aleatoria $\mathbf{X} = (X_1, X_2, \ldots, X_d)$, con x_j una muestra simple de $\mathbf{X}$.

Modelo compuesto de Gaussianas

Se asume que cada *cluster* C_i se caracteriza por una distribución normal multivariante, es decir

$$f_i(x) = f(x \mid \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}{2}\right\} \tag{3}$$

donde la media del *clusters* $\mu_i \in \mathbb{R}^d$ y el determinante de la matriz de varianzas-covarianzas $|\Sigma_i|$ ($\Sigma_i \in \mathbb{R}^{d\times d}$) son parámetros desconocidos. $f_i(x)$ es la probabilidad (dada por la función de densidad) de que x pertenezca al *cluster* C_i. Se asume que la función de densidad de $\mathbf{X}$ viene dada por el modelo compuesto de gaussianas de los k *clusters* normales, definido como sigue

$$f(x) = \sum_{i=1}^{k} f_i(x)P(C_i) = \sum_{i=1}^{k} f(x \mid \mu_i, \Sigma_i)P(C_i) \tag{4}$$

donde las probabilidades $P(C_i)$ se llaman *mixture parameters*, y deben satisfacer la propiedad

$$\sum_{i=1}^{k} P(C_i) = 1 \tag{5}$$

El modelo de combinación de gaussianas queda entonces caracterizado por la media μ_i, la matriz de varianzas-covarianzas Σ_i y la *mixture probability* $P(C_i)$ para cada una de las k distribuciones normales. Se puede escribir el conjunto de parámetros del modelo de forma compacta de la siguiente manera

$$\theta = \{\mu_1, \Sigma_1, P(C_1), \ldots, \mu_k, \Sigma_k, P(C_k)\}$$

Algunas de las razones por las que se utiliza una suma de campanas de Gauss en lugar de otras distribuciones son: que esta distribución aparece con frecuencia en la realidad, es el caso límite de otras distribuciones comunes y porque pueden generarse puntos conociendo solamente la media y desviación típica.

Estimador de máxima verosimilitud

Dado un *dataset* D, se define la verosimilitud de θ como la probabilidad condicionada de los datos de D dado el parámetro del modelo. En otras palabras, la verosimilitud es la probabilidad de que ocurra una muestra si es cierta la estimación que se ha efectuado sobre el parámetro y se denota por $P(D \mid \theta)$. Cada dato x_j se considera una muestra aleatoria simple de $\mathbf{X}$ (es decir, es una muestra aleatoria independiente e idénticamente distribuida). La función de verosimilitud de θ viene dada por

$$P(D \mid \theta) = \prod_{j=1}^{n} f(x_j)$$

El objetivo de la estimación de máxima verosimilitud (MLE) es escoger el parámetro θ que maximice dicha función de verosimilitud

$$\theta^* = \text{argmax}_\theta \{P(D \mid \theta)\}$$

Normalmente, se maximiza el logaritmo de la función de verosimilitud debido a que, tomando el logaritmo, el productorio pasa a ser un sumatorio. Entonces, el estimador de máxima verosimilitud queda

$$\theta^* = \text{argmax}_\theta \{\ln P(D \mid \theta)\}$$

y el logaritmo de la función de verosimilitud es

$$\ln P(D \mid \theta) = \sum_{j=1}^{n} \ln f(x_j) = \sum_{j=1}^{n} \ln\left(\sum_{i=1}^{k} f(x_j \mid \mu_i, \Sigma_i)P(C_i)\right) \tag{6}$$

***Expectation Maximization* (EM)**

Maximizar directamente el logaritmo de la función de verosimilitud en función de θ es difícil. En su lugar, podemos utilizar el enfoque de EM para lograr el estimador de máxima verosimilitud. Se trata de un algoritmo iterativo con dos pasos que comienza dando un valor inicial a θ. A medida que avanza, se van sucediendo diferentes estimaciones para θ.

En el algoritmo, cada *cluster* se caracteriza por una distribución normal multivariante (3), con parámetros μ_i, Σ_i y $P(C_i)$. Para cada *cluster* C_i se necesita estimar el vector de medias

$$\mu_i = (\mu_{i1}, \ldots, \mu_{id})^T$$

y la matriz de varianzas-covarianzas de dimensión $d \times d$

$$\Sigma_i = \begin{pmatrix} (\sigma_{i1})^2 & 0 & \ldots & 0 \\ 0 & (\sigma_{i2})^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & (\sigma_{id})^2 \end{pmatrix} \tag{7}$$

tiene todos los valores nulos fuera de la diagonal debido a que se asume que todas las variables de los datos son independientes.

Maximizar directamente el logaritmo de la función de verosimilitud (6) es complicado porque el *mixture term* aparece dentro del logaritmo. El problema que se encuentra es que, para cada punto x_j, se desconoce de qué normal o conjunto de normales proviene. Supongamos que cada dato tiene asociado un valor que indica el *cluster* que lo ha generado. Al conocer esta información, es mucho más sencillo maximizar el logaritmo de la función de verosimilitud.

El atributo categórico correspondiente a la etiqueta del *cluster* puede modelarse como un vector aleatorio $\hat{C} = (\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_k)$, donde cada $\hat{C}_i$ es una variable aleatoria de Bernoulli. Si un dato es generado por la i-ésima gaussiana, entonces $\hat{C}_i = 1$, en otro caso $\hat{C}_i = 0$. El parámetro $P(C_i)$ indica la probabilidad $P(\hat{C}_i = 1)$. Como cada punto puede pertenecer a tan solo un *cluster*, si $\hat{C}_a = 1$ para un punto, entonces $\hat{C}_i = 0$ para todo $i \neq a$. Por lo tanto $\sum_{i=1}^{k} P(C_i) = 1$.

Se define para cada dato x_j su vector *clusters* $c_j = (c_{j1}, \ldots, c_{jk})^T$. Tan solo una de las componentes de c_j tiene valor 1. Si $c_{ji} = 1$, quiere decir que $C_i = 1$, es decir, que el punto x_j es generado por la i-ésima gaussiana. La función de probabilidad de C viene dada por

$$P(C = c_j) = \prod_{i=1}^{k} P(C_i)^{c_{ji}}$$

debido a que en c_j solamente el j-ésimo elemento puede ser 1 ($c_{jj} = 1$), y el resto de componentes es 0 (cumple la definición de variable aleatoria multinomial de Bernoulli). Dada la información sobre los *clusters* c_j para cada punto x_p, la función de densidad de probabilidad condicionada de $\mathbf{X}$ es

$$f(x_p \mid c_j) = \prod_{i=1}^{k} f(x_p \mid \mu_i, \Sigma_i)^{c_{ji}}$$

Solo un *cluster* puede generar un punto x_p, supongamos que es C_a, en ese caso $c_{pa} = 1$ y la expresión anterior podría simplificarse a $f(x_p \mid c_j) = f(x_p \mid \mu_a, \Sigma_a)$. El par (x_p, c_j) es una muestra aleatoria del conjunto de distribuciones de los vectores aleatorios $\mathbf{X}$= $(X_1, \ldots, X_d)$ y $\hat{C} = (\hat{C}_1, \ldots, \hat{C}_k)$ correspondientes a los d atributos y k *clusters*. La función de densidad conjunta de $\mathbf{X}$ y $\hat{C}$ viene dada como

$$f(x_p, c_j) = f(x_p \mid c_j)P(c_j) = \prod_{i=1}^{k} (f(x_p \mid \mu_i, \Sigma_i)P(C_i))^{c_{ji}}$$

El logaritmo de la función de verosimilitud de los datos, conociendo la información sobre el *cluster*, queda entonces

$$
\begin{aligned}
\ln\left(P(D \mid \theta)\right) &= \ln \prod_{j=1}^{n} f(x_j, c_j \mid \theta) \\
&= \sum_{j=1}^{n} \ln f(x_j, c_j \mid \theta) \\
&= \sum_{j=1}^{n} \ln \left(\prod_{i=1}^{k} (f(x_j \mid \mu_i, \Sigma_i) P(C_i))^{c_{ji}}\right) \\
&= \sum_{j=1}^{n} \sum_{i=1}^{k} c_{ji} \left(\ln f(x_j \mid \mu_i, \Sigma_i) + \ln P(C_i)\right)
\end{aligned}
\tag{8}
$$

A continuación, se detallan los pasos que forman el algoritmo EM:

- **Inicialización**

  Para cada *cluster* C_i con $i = 1, \ldots, k$ se inicializa aleatoriamente la media μ_i seleccionando el valor μ_{ia} para cada dimensión X_a en el rango de X_a. La matriz de varianzas-covarianzas Σ_i se inicializa como I_d. Por último, la probabilidad del *cluster* previa se inicializa como $P(C_i) = \frac{1}{k}$, es decir, que todos los *clusters* tienen la misma probabilidad.

- ***Expectation step***

  En este paso se calcula la esperanza del logaritmo de la función de verosimilitud sobre los datos, vista en (8). La esperanza se realiza sobre la información desconocida del *cluster* c_j y se tratan los parámetros μ_i, Σ_i, $P(C_i)$ y x_j como valores fijos. Debido a la linearidad de las esperanzas, el valor de la esperanza del logaritmo de la función de verosimilitud es el siguiente:

$$
E\left[\ln P(D \mid \theta)\right] = \sum_{j=1}^{n} \sum_{i=1}^{k} E[c_{ji}] \Big(\ln f(x_j \mid \mu_i, \Sigma_i) + \ln P(C_i)\Big)
$$

  Por otro lado, la esperanza $E[c_{ij}]$ puede calcularse como

$$
\begin{aligned}
E\left[c_{ji}\right] &= 1 \cdot P(c_{ji} = 1 \mid x_j) \;+\; 0 \cdot \; P(c_{ji} = 0 \mid x_j) = P(c_{ji} = 1 \mid x_j) = P(C_i \mid x_j) \\
&= \frac{P(x_j \mid C_i) P(C_i)}{P(x_j)} \;=\; \frac{f(x_j \mid \mu_i, \Sigma_i) P(C_i)}{f(x_j)} = w_{ij}
\end{aligned}
\tag{9}
$$

  Entonces, en el segundo paso se utilizan los valores de $\theta = \{\mu_i, \Sigma_i, P(C_i)\}_{i=1}^{k}$ para estimar las probabilidades a posteriori o pesos w_{ij} para cada punto del *dataset* y para cada *cluster*. Si utilizamos (9), la esperanza del logaritmo de la función de verosimilitud puede reescribirse como sigue:

$$
E\left[\ln P(D \mid \theta)\right] = \sum_{j=1}^{n} \sum_{i=1}^{k} w_{ji} \left(\ln f(x_j \mid \mu_i, \Sigma_i) + \ln \; P(C_i)\right) \tag{10}
$$

- ***Maximization step***

  Se vuelven a estimar los parámetros μ_i, Σ_i y $P(C_i)$ para cada C_i derivando el logaritmo de la función de máxima verosimilitud respecto a cada uno de los parámetros e igualando las derivadas obtenidas a cero. A continuación, se muestra el desarrollo de la derivada respecto

del parámetro θ_i del *cluster* C_i de la función (6):

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\ln\left(P(D\theta)\right) &= \frac{\partial}{\partial\theta_i}\left(\sum_{j=1}^{n}\ln\left(f(x_j)\right)\right)\\ &= \sum_{j=1}^{n}\left(\frac{1}{f(x_j)}\frac{\partial f(x_j)}{\partial\theta_i}\right)\\ &= \sum_{j=1}^{n}\left(\frac{1}{f(x_j)}\sum_{a=1}^{k}\frac{\partial}{\partial\theta_i}\left(f(x_j\mid\mu_a,\Sigma_a)P(C_a)\right)\right)\\ &= \sum_{j=1}^{n}\left(\frac{1}{f(x_j)}\frac{\partial}{\partial\theta_i}\left(f(x_j\mid\mu_i,\Sigma_i)P(C_i)\right)\right)\end{aligned}$$

El último paso se debe a que θ_i es el parámetro del i-ésimo *cluster*, entonces para el resto de *clusters* son constantes respecto a θ_i. Se sabe que $|\Sigma_i| = \frac{1}{|\Sigma_i^{-1}|}$; por lo tanto, la ecuación (3) puede reescribirse como

$$f(x_j\mid\mu_i,\Sigma_i) = (2\pi)^{(\frac{-d}{2})}|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\{g(\mu_i,\Sigma_i)\} \tag{11}$$

donde

$$g(\mu_i,\Sigma_i) = -\frac{1}{2}(x_j-\mu_i)^T\Sigma_i^{-1}(x_j-\mu_i) \tag{12}$$

Por lo tanto, la derivada del logaritmo de la función de verosimilitud puede escribirse de la siguiente forma

$$\frac{\partial}{\partial\theta_i}\ln\left(P(D\mid\theta)\right) = \sum_{j=1}^{n}\left(\frac{1}{f(x_j)}\frac{\partial}{\partial\theta_i}((2\pi)^{\frac{-d}{2}}|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\left\{g(\mu_i,\Sigma_i)\right\}P(C_i))\right) \tag{13}$$

Por último, queda incluir la siguiente derivada parcial a la ecuación anterior

$$\frac{\partial}{\partial\theta_i}\exp\left\{g(\mu_i,\Sigma_i)\right\} = \exp\left\{g(\mu_i,\Sigma_i)\right\}\frac{\partial}{\partial\theta_i}g(\mu_i,\Sigma_i) \tag{14}$$

- **Estimación de la media μ_i**

  Para derivar el logaritmo de la función de máxima verosimilitud respecto de μ_i, basta sustituir en las ecuaciones anteriores $\theta_i = \mu_i$. Dado que en la ecuación (13) el único término que involucra μ_i es $\exp\left\{g(\mu_i,\Sigma_i)\right\}$, se tiene que:

$$\frac{\partial}{\partial\mu_i}g(\mu_i,\Sigma_i) = \Sigma_i^{-1}(x_j-\mu_i) \tag{15}$$

  En virtud de la ecuación (14), la derivada parcial del logaritmo de la función de verosimilitud queda:

$$\begin{aligned}\frac{\partial}{\partial\mu_i}\ln P(D\mid\theta) &= \sum_{j=1}^{n}\left(\frac{1}{f(x_j)}(2\pi)^{\frac{-d}{2}}|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\{g(\mu_i,\Sigma_i)\}P(C_i)\Sigma_i^{-1}(x_j-\mu_i)\right)\\ &= \sum_{j=1}^{n}\left(\frac{f(x_j\mid\mu_i,\Sigma_i)P(C_i)}{f(x_j)}\Sigma_i^{-1}(x_j-\mu_i)\right)\\ &= \sum_{j=1}^{n}w_{ij}\Sigma_i^{-1}(x_j-\mu_i)\end{aligned}$$

  utilizando la ecuación (11) y sabiendo que la probabilidad posterior $P(C_i\mid x_j) = \frac{f_i(x_j)\cdot P(C_i)}{\sum_{a=1}^{k}f_a(x_j)\cdot P(C_a)}$ se llega a lo siguiente

$$w_{ij} = P(C_i\mid x_j) = \frac{f(x_j\mid\mu_i,\Sigma_i)P(C_i)}{f(x_j)}$$

Al igualar a cero la derivada parcial del logaritmo de la función de verosimilitud respecto de la media, obtenemos la estimación buscada:

$$\sum_{j=1}^{n} w_{ij}(x_j - \mu_i) = 0 \Longrightarrow \sum_{j=1}^{n} w_{ij}x_j = \mu_i \sum_{j=1}^{n} w_{ij} \Longrightarrow \boxed{\hat{\mu_i} = \frac{\sum_{j=1}^{n} w_{ij}x_j}{\sum_{j=1}^{n} w_{ij}}} \tag{16}$$

- **Estimación de la matriz de varianzas-covarianzas Σ_i**

  Para estimar la matriz de varianzas-covarianzas se actúa de forma análoga a la estimación de la media, basta con derivar la ecuación (13), esta vez respecto de Σ_i^{-1} aplicando la regla del producto de la derivada al término $|\Sigma_i^{-1}|^{\frac{1}{2}} \exp\{g(\mu_i, \Sigma_i)\}$. Se va a utilizar también el resultado que dice que para cualquier matriz A cuadrada, $\frac{\partial |A|}{\partial A} = |A|(A^{-1})^T$. Se obtiene el siguiente resultado

$$\frac{\partial |\Sigma_i^{-1}|^{\frac{1}{2}}}{\partial \Sigma_i^{-1}} = \frac{1}{2}|\Sigma_i^{-1}|^{\frac{-1}{2}}(|\Sigma_i^{-1}|)' = \frac{1}{2}|\Sigma_i^{-1}|^{\frac{-1}{2}}|\Sigma_i^{-1}|((\Sigma_i^{-1})^{-1})^T = \frac{1}{2}|\Sigma_i^{-1}|^{\frac{1}{2}}\Sigma_i \tag{17}$$

A continuación, se utiliza el resultado que dice que dada una matriz cuadrada $A \in \mathbb{R}^{d\times d}$ y vectores $a, b \in \mathbb{R}^d$ entonces $\frac{\partial a^T A b}{\partial A} = ab^T$. Por tanto, la derivada de $\exp\{g(\mu_i, \Sigma_i)\}$ respecto a Σ_i^{-1} utilizando la igualdad (14) viene dada por

$$\frac{\partial}{\partial \Sigma_i^{-1}}(\exp\{g(\mu_i, \Sigma_i)\}) = -\frac{1}{2}\exp\{g(\mu_i, \Sigma_i)\}(x_j - \mu_i)(x_j - \mu_i)^T \tag{18}$$

Al hacer uso de la regla del producto y de las ecuaciones (17) y (18) se obtiene

$$\begin{aligned}\frac{\partial}{\partial \Sigma_i^{-1}}\left(|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\{g(\mu_i, \Sigma_i)\}\right) &= \frac{1}{2}|\Sigma_i^{-1}|^{\frac{1}{2}}\Sigma_i \exp\{g(\mu_i, \Sigma_i)\} \\ &\quad - \frac{1}{2}|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\{g(\mu_i, \Sigma_i)\}(x_j - \mu_i)(x_j - \mu_i)^T \\ &= \frac{1}{2}|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\{g(\mu_i, \Sigma_i)\}\left(\Sigma_i - (x_j - \mu_i)(x_j - \mu_i)^T\right)\end{aligned} \tag{19}$$

Al sustituir la igualdad (19) en la ecuación (13) se llega a que la derivada del logaritmo de la función de verosimilitud respecto a Σ_i^{-1} es

$$\begin{aligned}\frac{\partial}{\partial \Sigma_i^{-1}}\ln(P(D \mid \theta)) &= \frac{1}{2}\sum_{j=1}^{n}\frac{(2\pi)^{-\frac{d}{2}}|\Sigma_i^{-1}|^{\frac{1}{2}}\exp\{g(\mu_i, \Sigma_i)\}P(C_i)}{f(x_j)}\left(\Sigma_i - (x_j - \mu_i)(x_j - \mu_i)^T\right) \\ &= \frac{1}{2}\sum_{j=1}^{n}\frac{f(x_j \mid \mu_i, \Sigma_i)P(C_i)}{f(x_j)}\left(\Sigma_i - (x_j - \mu_i)(x_j - \mu_i)^T\right) \\ &= \frac{1}{2}\sum_{j=1}^{n} w_{ij}\left(\Sigma_i - (x_j - \mu_i)(x_j - \mu_i)^T\right)\end{aligned}$$

Al igualar a cero la derivada obtenida, se llega al estimador de Σ_i

$$\sum_{j=1}^{n} w_{ij}\left(\Sigma_i - (x_j - \mu_i)(x_j - \mu_i)^T\right) = 0$$

$$\boxed{\hat{\Sigma_i} = \frac{\sum_{j=1}^{n} w_{ij}(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{n} w_{ij}}} \tag{20}$$

- **Estimación de la *Prior Probability* ($P(C_i)$)**

  Para lograr el estimador de máxima verosimilitud de $P(C_i)$, se debe calcular la derivada parcial de (13) respecto de $P(C_i)$. Se añade también el multiplicador de Lagrange α para introducir la restricción de $\sum_{a=1}^{k} P(C_a) = 1$ y se obtiene la siguiente expresión

$$\frac{\partial}{\partial P(C_i)}\left(\ln(P(D \mid \theta)) + \alpha\left(\sum_{a=1}^{k} P(C_a) - 1\right)\right) \tag{21}$$

Al derivar la expresión (13) respecto de $P(C_i)$ queda

$$\frac{\partial}{\partial P(C_i)} \ln\left(P(D \mid \theta)\right) = \sum_{j=1}^{n} \frac{f(x_j \mid \mu_i, \Sigma_i)}{f(x_j)}$$

por lo tanto, la expresión (21) es:

$$\left(\sum_{j=1}^{n} \frac{f(x_j \mid \mu_i, \Sigma_i)}{f(x_j)} \right) + \alpha$$

Al igualar la derivada a cero y multiplicando a ambos lados por $P(C_i)$, se obtiene que

$$\sum_{j=1}^{n} \frac{f(x_j \mid \mu_i, \Sigma_i)}{f(x_j)} P(C_i) = -\alpha P(C_i)$$

$$\sum_{j=1}^{n} w_{ij} = -\alpha P(C_i) \tag{22}$$

al sumar para todos los *clusters* se llega a la siguiente expresión

$$\sum_{i=1}^{k} \sum_{j=1}^{n} w_{ij} = -\alpha \sum_{i=1}^{k} P(C_i) \tag{23}$$

dado que $\sum_{i=1}^{k} w_{ij} = 1$, se tiene que $n = -\alpha$. Si se introduce esta última expresión en la ecuación (22) se obtiene el estimador de máxima verosimilitud para $P(C_i)$

$$\boxed{\hat{P(C_i)} = \frac{\sum_{j=1}^{n} w_{ij}}{n}} \tag{24}$$

Una vez determinados los tres estimadores para los parámetros de cada distribución normal de cada *cluster*, se observa que todos dependen de los pesos w_{ij} (la probabilidad posterior del *cluster* $P(C_i \mid x_j)$). Por ello, las fórmulas para los estimadores de máxima verosimilitud de los parámetros no son fórmulas cerradas, si no que se encuentran sujetas al cálculo de w_{ij}. Así, el algoritmo EM calcula en cada iteración w_{ij} en el paso de *expectation* y a continuación, en el *maximization-step* se vuelven a estimar los parámetros μ_i, Σ_i y $P(C_i)$.

Algoritmo 2 Pseudocódigo del algoritmo *Expectation-Maximization* multivariante

Entrada: *dataset* D, número de *clusters* k, tolerancia ϵ

$t \leftarrow 0$
//Inicialización aleatoria dentro del rango X_a
Inicialización de $\mu_1^0, \ldots, \mu_k^0$
$\Sigma \leftarrow I_d,\ \forall i = 1, \ldots, k$
$P(C_i) \leftarrow \frac{1}{k},\ \forall i = 1, \ldots, k$
repetir
  $t \leftarrow t + 1$
  para $i = 1, \ldots, k$ *y* $j = 1, \ldots, k$ **hacer**
    //Probabilidad posterior $P(C_i \mid x_j)$
    $w_{ij} \leftarrow \frac{f(x_j|\mu_i^{t-1}\Sigma_i) \cdot P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a^{t-1}, \Sigma_a) \cdot P(C_a)}$
  fin para
  para $i = 1, \ldots, k$ **hacer**
    //Se vuelve a estimar la media
    $\mu_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} x_j}{\sum_{j=1}^{n} w_{ij}}$
    //Se vuelve a estimar la matriz de varianzas-covarianzas
    $\Sigma_i \leftarrow \frac{\sum_{j=1}^{n} w_{ij} (x_j - \mu_i^t)(x_j - \mu_i^t)^T}{\sum_{j=1}^{n} w_{ij}}$
    //Se vuelve a estimar la probabilidad a priori
    $P(C_i) \leftarrow \frac{\sum_{j=1}^{n} w_{ij}}{n}$
  fin para
hasta que $\sum_{i=1}^{k} \|\mu_i^t - \mu_i^{t-1}\| \leq \epsilon$
devolver Probabilidad de pertenecer cada dato a cada *cluster*: $P(C_i)$

El pseudocódigo para el algoritmo EM multivariante encontrado en ([31, pg. 389]) se muestra en el Algoritmo 2. Tras la inicialización de μ_i, Σ_i y $P(C_i)$ para todo $i = 1, \ldots, k$, los pasos de *expectation* y *maximization* se repiten hasta que el algoritmo converge. Para el test de convergencia, se comprueba que $\sum_i \|\mu_i^{t+1} - \mu_i^t\|^2 \leq \epsilon$, siendo $\epsilon > 0$ el umbral de convergencia y t denota la iteración en la que se encuentra el algoritmo. En otras palabras, el algoritmo continua hasta que las medias de los *clusters* varían muy poco.

En cuanto a **la complejidad computacional**, en el primer paso se calculan las probabilidades posteriores, para lo que se necesita invertir la matriz Σ_i y calcular su determinante $|\Sigma_i|$, lo cual requiere una complejidad computacional $\mathcal{O}(d^3)$ (con d el rango de la matriz y el número de variables). Dado que se realiza para cada *cluster*, la complejidad se eleva hasta $\mathcal{O}(kd^3)$. Por otro lado, evaluar la función de densidad $f(x_j \mid \mu_i, \Sigma_i)$ requiere complejidad temporal $\mathcal{O}(d^2)$, por tanto en total requiere complejidad temporal de $\mathcal{O}(knd^2)$ para n datos y k *clusters*. En cuanto al segundo paso, la complejidad temporal la domina la actualización de Σ_i que requiere de $\mathcal{O}(knd^2)$ al actualizar las de todos los *clusters*. Finalmente, la complejidad temporal del algoritmo EM es $\mathcal{O}(t(kd^3 + nkd^2))$ siendo t el número de iteraciones realizadas por el algoritmo. En el caso de utilizar una matriz de varianzas-covarianzas diagonal, el cálculo de la inversa y su determinante requieren $\mathcal{O}(d)$. El cálculo de la función de densidad requiere de $\mathcal{O}(d)$ para cada punto, luego la complejidad temporal total del primer paso es $\mathcal{O}(knd)$. El *maximization-step* sigue requiriendo de una complejidad temporal $\mathcal{O}(knd^2)$ por tener que volver a estimar Σ_i. La complejidad total para el algoritmo con un matriz diagonal de varianzas-covarianzas es por tanto $\mathcal{O}(tnkd^2)$.

***k-means* como caso especial de EM:** A pesar de suponer que los *clusters* siguen una distribución normal o combinación de ellas, el enfoque de EM puede aplicarse a otros modelos para la función de distribución $P(x_j \mid C_i)$. Por ejemplo, *k-means* puede considerarse como un caso especial del algoritmo EM, obteniendo lo siguiente

$$P(x_j \mid C_i) = \begin{cases} 1 & \text{si } \; C_i = \operatorname{argmin}_{C_a}\{\|x_j - \mu_a\|^2\} \\ 0 & \text{en otro caso} \end{cases}$$

Si se utiliza que

$$P(C_i \mid x_j) = \frac{f_i(x_j) \cdot P(C_i)}{\sum_{a=1}^{k} f_a(x_j) \cdot P(C_a)}$$

la *posterior probability* $P(C_i \mid x_j)$ viene dada por

$$P(C_i \mid x_j) = \frac{P(x_j \mid C_i)P(C_i)}{\sum_{a=1}^{k} P(x_j \mid C_a)P(C_a)}$$

Como puede verse, si $P(x_j \mid C_i) = 0$, entonces $P(C_i \mid x_j) = 0$. En otro caso, si $P(x_j \mid C_i) = 1$, entonces $P(x_j \mid C_a) = 0$ para todo $a \neq i$, y $P(C_i \mid x_j) = \frac{1 \cdot P(C_i)}{1 \cdot P(C_i)} = 1$. Al juntar todo, se puede determinar la *posterior probability* como

$$P(C_i \mid x_j) = \begin{cases} 1 & \text{si } \; x_j \in C_i \iff C_i = \operatorname{argmin}_{C_a}\{\|x_j - \mu_a\|^2\} \\ 0 & \text{en otro caso} \end{cases} \tag{25}$$

Como observación final, cabe destacar que para *k-means* los parámetros de los *clusters* son μ_i y $P(C_i)$. Puede ignorarse la matriz de varianza covarianza.

2.2. Aprendizaje supervisado

En el aprendizaje supervisado se parte de un conjunto $D = \{(x_i, y_i)_{i=1}^n\}$ y se crea una función de hipótesis h_w con la que dado un dato cualquiera x del que se conocen las variables predictoras, $h_{w,b}(x)$ permite determinar su variable de respuesta y. En los algoritmos supervisados, se busca minimizar las diferencias entre las etiquetas reales y las predichas por la función $h_{w,b}$. Para ello se crea una función de coste J que cuantifica dichas diferencias y se persigue minimizar.

El problema del sobreajuste

Al tratar de crear funciones que predicen nuevos datos a partir de datos conocidos pueden surgir dos problemas clásicos: sobreajuste (*overfitting*) y subajuste (*underfitting*). El primero ocurre cuando los modelos predicen con mucha precisión los datos del conjunto de entrenamiento pero de forma incorrecta los nuevos datos. Suele ocurrir cuando los modelos son muy complejos (por ejemplo, si la función de hipótesis es un polinomio de grado muy elevado), consiguen adaptarse y aprender muy bien del *dataset*. En cambio, el *underfitting* ocurre cuando los modelos no predicen

de forma precisa el conjunto de entrenamiento y tampoco los nuevos datos. Suele darse cuando los modelos son demasiado simples (por ejemplo si la función de hipótesis es una recta y los datos siguen una tendencia parabólica).

Se llama sesgo a la diferencia entre la predicción del modelo y el valor real; un alto valor del sesgo indica que el modelo no se aproxima lo suficiente a los datos y ocurre lo que se conoce como *underfitting*. Otro concepto utilizado en minería de datos es la varianza de un modelo. Hace referencia a la sensibilidad que tiene respecto a los datos de entrenamiento, la dependencia que presenta. Estos conceptos se explican con mayor detalle en [16]. Si un modelo posee baja varianza, indica que al modificar los datos del conjunto de entrenamiento se producen cambios pequeños en las futuras estimaciones. Un modelo con alta varianza indica que si se modifican los datos del conjunto de entrenamiento se producirán grandes cambios en las estimaciones futuras, es decir el fenómeno mencionado antes: *overfitting*. El problema con estos parámetros es que si disminuye la varianza aumenta el sesgo y viceversa. Por tanto, se trata de llegar a un equilibrio.

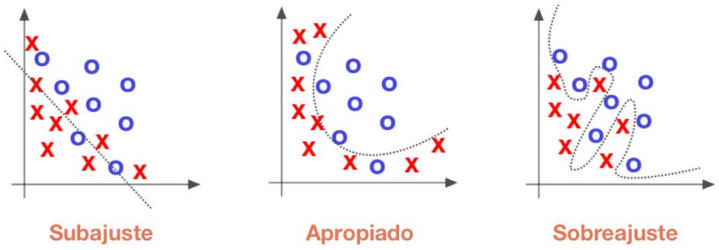


Figura 4: Ejemplo de los fenómenos de *underfitting*, el caso en que se encontró el equilibrio y *overfitting*. Fuente figura: [16]

Como se observa en la Figura 4, en la primera gráfica, la función de hipótesis no se aproxima demasiado a la frontera entre los datos, y se produce subajuste. En la segunda gráfica se aproxima más pero queda algún dato erróneamente clasificado. Finamente en la última gráfica se observa como el modelo se ha adaptado perfectamente a los datos y los separa a la perfección. Aparentemente puede parecer que la tercera situación es la idónea para cualquier modelo, pero si se intentan predecir nuevos valores a partir del modelo no se obtendrán resultados demasiado satisfactorios ya que al haberse adaptado tanto al *dataset*, habrá aprendido también del ruido presente, lo que produce peores estimaciones para nuevos datos.

Regularización

El *overfitting* puede solucionarse introduciendo un nuevo término en la función de coste del modelo $J(h(x))$, el término regularizador

$$J_1(h(x)) = J(h_{w,b}(x)) + \lambda \text{ Reg}(h_{w,b}(x)) \tag{26}$$

donde λ es el factor de regularización que determina la importancia de la regularización del coste total ($\lambda \geq 0$). En algunas implementaciones de los algoritmos de aprendizaje supervisado λ se regula mediante otro parámetro C que es su inverso. Cuanto mayor sea el valor de λ, mayor fuerza tendrá el término regularizador y menor peso tendrá el término del error en la función de coste que se pretenderá minimizar, por lo tanto, menos preciso será el modelo y menor *overfitting* se producirá. La complejidad del modelo suele ser a suma de los valores absolutos de los parámetros o la suma de los cuadrados de los parámetros de los modelos. En el primer caso se trata de la norma q con $q = 1$ (que suele llamarse *hing loss* o *lasso*) y en el segundo se trata de la norma dos (suele llamarse *cuadratic loss* o *ridge*).

2.2.1. Problema de clasificación

A continuación, se va a realizar una introducción al problema de clasificación, basándonos en [7, pg. 13]. El problema de clasificación se engloba dentro del aprendizaje supervisado y consiste en predecir etiquetas de entre un conjunto discreto de datos que se desconocen. Sea $D = \{(x_i, y_i)_{i=1}^n\}$ el conjunto de observaciones con n puntos de dimensión d ($X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$), y sus correspondientes etiquetas ($Y = \{y_i\}_{i=1}^n \subset \mathbb{R}$). El objetivo es determinar un modelo $h_{w,b}$ que permita realizar predicciones razonables de etiquetas y' sobre puntos sin etiquetar x'. A los datos sin etiquetas se les asignarán las etiquetas de datos etiquetados con los que guarden semejanza, es decir, datos cercanos, que vienen de la misma distribución o se encuentran en la misma zona del umbral de decisión. El aprendizaje supervisado puede ser difícil ya que los conjuntos de entrenamiento pueden

tener variables que no ayudan a la clasificación, mientras que variables importantes pueden ser desconocidas; las similitudes entre los datos no siempre son fáciles de definir, y las observaciones a veces no siguen distribuciones sencillas. Además, los modelos de aprendizaje pueden ser complicados de determinar debido a que las clases, a veces, no son linealmente separables y esto dificulta el poder separar los datos mediante reglas simples o ecuaciones no demasiado complejas.

Normalmente, las etiquetas tienen un significado semántico relacionado con alguna aplicación específica; en el presente trabajo son ciclistas profesionales o amateurs. El modelo aprendido se llama modelo de entrenamiento. El conjunto de datos no etiquetados se llama conjunto de datos de prueba.

En algunos modelos la clasificación es binaria, por lo que el número de etiquetas $k = 2$. En este caso, las etiquetas suelen ser -1 y 1 o 0 y 1. El modelo se construye a partir de los datos de entrenamiento y luego se utiliza para predecir las etiquetas de los datos de prueba. El output de cualquier algoritmo de clasificación puede ser de dos tipos:

- **Predicción de etiquetas:** En este caso, se predice la etiqueta para cada instancia.
- **Puntuación numérica:** En la mayoría de los casos, el modelo asigna una puntuación a cada instancia que mide la propensión de la instancia a pertenecer a una clase en particular. Esta puntuación se puede convertir fácilmente en una predicción de etiqueta mediante el uso del valor máximo, o un valor máximo ponderado por el costo de la puntuación numérica en diferentes clases. Una ventaja de usar puntuación es que diferentes instancias de prueba pueden ser comparadas y clasificadas por su propensión a pertenecer a una clase particular. Tales puntuaciones son particularmente útiles en las situaciones en que una de las clases es rara y la puntuación numérica ofrece una forma de determinar el top de ranking de posibles pertenecientes a dicha clase.

Cuando el conjunto de datos de entrenamiento es pequeño, el rendimiento de los modelos de clasificación es a veces pobre. En tales casos, el modelo puede describir las características aleatorias específicas del conjunto de datos de entrenamiento, y puede que no se generalice a la estructura de grupo de una prueba no vista previamente. En otras palabras, dichos modelos podrían predecir con precisión las etiquetas de las instancias utilizadas para construirlos, pero funcionan mal en instancias de pruebas no conocidas. Se trataría en este caso de un sobreajuste.

Los algoritmos más conocidos para la clasificación de datos incluyen árboles de decisión, clasificación basada en reglas, modelos probabilísticos, clasificaciones basadas en clases, máquinas de soporte vectorial y redes neuronales. La fase de modelado suele estar precedida por una fase de selección de variables para identificar las características más formativas para la clasificación.

2.2.1.1. Máquinas de soporte vectorial (SVM).

En esta sección se va a estudiar el método de clasificación de las máquinas de soporte vectorial (en inglés, *Support Vector Machines*) en base a las notas de [31, pgs. 565-592]. Es un método de clasificación basado en el máximo margen lineal discriminante, es decir, el objetivo del método es encontrar el hiperplano que maximice el margen (distancia) entre las clases. Se puede emplear el truco del núcleo, que corresponde a buscar un hiperplano separador en algún espacio no lineal de dimensión superior a la de los puntos. En esta sección se asumirá el caso de clasificación binaria donde cada uno de los puntos d-dimensionales del *dataset* $D = \{(x_i, y_i)_{i=1}^n\}$ tendrá como etiqueta 1 o -1.

Discriminantes lineales y márgenes

Hiperplanos

Una función lineal discriminante en d dimensiones viene dada por una aplicación lineal h, que se define como sigue

$$h(x) = w^T x + b \tag{27}$$

donde w es un vector d-dimensional llamado peso y b un escalar llamado *bias*. Para los puntos pertenecientes al hiperplano se tiene que $h(x) = 0$. Por tanto, el hiperplano H se define como el conjunto de puntos x que satisfacen la ecuación: $w^T x = -b$. Los puntos del hiperplano que intersecan con los ejes son $\frac{-b}{w_i}$ (suponiendo que $w_i \neq 0$).

Hiperplanos separadores

Un hiperplano divide el espacio en el que se encuentra en dos subespacios. Se dice que un *dataset* es separable linealmente si en los subespacios que induce el hiperplano se encuentran puntos de una

única clase. Por tanto, si un *dataset* es linealmente separable se puede encontrar una aplicación lineal (o discriminante lineal) h, que permite predecir la etiqueta para cualquier punto x de acuerdo a la siguiente regla de decisión

$$y = \begin{cases} 1 & \text{si } h(x) \geq 0 \\ -1 & \text{si } h(x) < 0 \end{cases} \tag{28}$$

Dados x_1 y x_2 dos puntos pertenecientes al hiperplano, de la ecuación (27) se obtiene que $w^T(x_1 - x_2) = 0$. Lo que significa que w define una dirección ortogonal al hiperplano H y b define el desplazamiento del hiperplano en el espacio d-dimensional.

Distancia de un punto al hiperplano

Sea x un punto del espacio d-dimensional que no pertenece al hiperplano y sea x^p su proyección ortogonal sobre el hiperplano. Se define entonces $\mathbf{r} = x - x^p$, que puede reescribirse como

$$\begin{aligned} x &= x^p + \mathbf{r} \\ x &= x^p + r\frac{w}{\|w\|} \end{aligned} \tag{29}$$

donde r es un escalar que determina el vector $x^p - x$ en términos del vector unitario $\frac{w}{\|w\|}$.

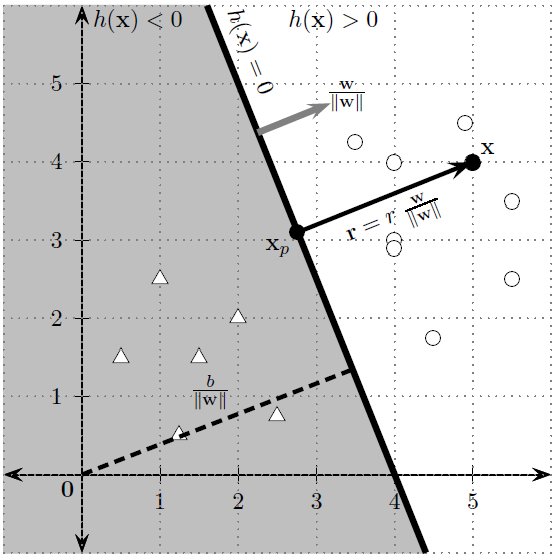


Figura 5: Geometría de un hiperplano separador en 2D. La clase +1 se muestra mediante círculos y la clase -1 en triángulos, fuente: [31, pg. 567]

.

Al utilizar la igualdad (29) en la ecuación (27) se tiene

$$\begin{aligned} h(x) &= h\left(x^p + r\frac{w}{\|w\|}\right) \\ &= w^T\left(x^p + r\frac{w}{\|w\|}\right) + b \\ &= \underbrace{w^T x^p + b}_{h(x^p)} + r\frac{w^T w}{\|w\|} \\ &= \underbrace{h(x^p)}_{0} + r\|w\| \\ &= r\|w\| \end{aligned}$$

Al utilizar este resultado se llega a una expresión para $r = \frac{h(x)}{\|w\|}$, pero la distancia debe ser no negativa. Para conseguirlo, basta multiplicar por la etiqueta de la clase del punto

$$\delta = yr = \frac{y\ h(x)}{\|w\|} \tag{30}$$

En el caso particular en que $x = 0$, $r = \frac{b}{\|w\|}$ como se muestra en la Figura 5.

Margen y vectores soporte de un hiperplano

Dado un conjunto de datos clasificados $D = \{(x_i, y_i)_{i=1}^n\}$ y un hiperplano definido por $h(x) = 0$, el margen del clasificador lineal es la mínima distancia δ^* entre los puntos de D y el hiperplano separador

$$\delta^* = \min_{x_i} \left\{ \frac{y_i(w^T x_i + b)}{\|w\|} \right\} \tag{31}$$

Los vectores (puntos) que se encuentran a distancia δ^* se llaman vectores soporte del hiperplano. Dichos vectores cumplen la siguiente condición

$$\delta^* = \frac{y^*(w^T x^* + b)}{\|w\|}$$

donde y^* es la etiqueta del punto x^*. El numerador proporciona la distancia absoluta y δ^* es la distancia relativa en términos de w.

Hiperplano canónico

Se considera la ecuación del hiperplano (27) y se multiplica a ambos lados de la ecuación por un escalar s, llegando a un hiperplano equivalente.

$$sh(x) = sw^T x + sb = (sw)^T x + (sb) = 0$$

Para obtener el hiperplano canónico, se debe escoger s de tal forma que cumpla la siguiente ecuación

$$sy^*(w^T x^* + b) = 1$$

es decir, que la distancia absoluta de un vector de soporte al hiperplano debe ser 1. De ahora en adelante, se supondrá que los hiperplanos son canónicos, es decir, que se han reescalado para que cumplan que $y^* h(x^*) = 1$ para los vectores soporte y el margen tiene el valor $\delta^* = \frac{1}{\|w\|}$.

Para cualquier hiperplano canónico, dado un vector de soporte x_i^* (con etiqueta y_i^*), se tiene que $y_i^* h(x_i^*) = 1$ y para los puntos que no son vectores de soporte $y_i h(x_i) > 1$. Luego por definición, si un punto no es un vector de soporte debe encontrarse más lejos del hiperplano que los vectores de soporte. Al generalizar lo observado se llega a que

$$y_i(w^T x_i + b) \geq 1, \ \forall x_i \in D \tag{32}$$

Caso lineal y separable

Dado un *dataset* D, se asume por el momento que los puntos son linealmente separables, es decir, que existe un hiperplano H que clasifica perfectamente cada punto de D. Se encuentra en este caso el siguiente problema: existen infinitos hiperplanos que cumplen dicha condición para datos linealmente separables, veamos cuál se debe escoger.

Hiperplano con máximo margen

La idea fundamental detrás de SVM es seleccionar el hiperplano canónico especificado por el vector peso w y el escalar *bias* b que logre el máximo margen de entre todos los posibles hiperplanos separadores. Se define $\delta_{w,b}^*$ como el margen del hiperplano $h_{w,b}(x) = 0$. El objetivo es encontrar el hiperplano óptimo $h_{w,b}^*(x) = 0$

$$h_{w,b}^* = \text{argmax}_{h_{w,b}} \{\delta_{h_{w,b}}^*\} = \text{argmax}_{w,b} \left\{ \frac{1}{\|w\|} \right\}$$

Pero el hiperplano buscado $h_{w,b}^*$, además de maximizar $\delta_{w,b}^*$, debe cumplir las inecuaciones surgidas en (32). Por tanto se tiene el siguiente problema de optimización

- **Función objetivo:** $\text{mín}_{w,b} \left\{ \frac{\|w\|^2}{2} \right\}$
- **Restricciones lineales:** $y_i(w^T x_i + b) \geq 1, \quad \forall x_i \in D$

Nota: El problema de maximizar $\frac{1}{\|w\|}$ es equivalente al de minimizar $\|w\|$, se divide entre dos para simplificar el desarrollo.

Para resolver el problema planteado se puede optar por aplicar algoritmos de optimización estándar para el problema de minimización convexa primal. Pero es más común optar por el problema dual que se obtiene mediante los multiplicadores de Lagrange y que se detalla a continuación.

Se introducen los multiplicadores de Lagrange $\alpha_1, \ldots, \alpha_n$, uno para cada restricción, basándose en las condiciones de Karush-Kuhn-Tucker (KKT):

- $\alpha_i(y_i(w^T x_i + b) - 1) = 0$
- $\alpha_i \geq 0$

Al introducir las n restricciones, la nueva función objetivo llamada Lagrangiana es

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 \ - \sum_{i=1}^{n} \alpha_i(y_i(w^T x_i \ + b) - 1) \tag{33}$$

donde $w = (w_1, \ldots, w_d)$ y $\alpha = (\alpha_1, \ldots, \alpha_n)$. L debe minimizarse respecto a w y a b, y maximizarse respecto a α_i. Al tomar las derivadas de L respecto de w y b e igualando a cero se obtiene:

$$\frac{\partial}{\partial w} L(w, b, \alpha) = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \Longrightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{34}$$

$$\frac{\partial}{\partial b} L(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{35}$$

La ecuación (34) implica que w puede expresarse como combinación lineal de los datos x_i con coeficientes $\alpha_i y_i$. Al introducir estas nuevas igualdades en la ecuación (33) se obtiene la función objetivo del problema dual Lagrangiano, que se expresa en términos de los multiplicadores de Lagrange como sigue

$$\begin{aligned} L_{dual}(\alpha) &= \frac{1}{2} w^T w \ - w^T \underbrace{\left(\sum_{i=1}^{n} \alpha_i y_i x_i \right)}_{w} - b \underbrace{\sum_{i=1}^{n} \alpha_i y_i}_{0} + \sum_{i=1}^{n} \alpha_i \\ &= -\frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

El problema dual viene dado entonces por

- **Función objetivo:** $\max_\alpha L_{dual}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$
- **Restricciones lineales:** $\alpha_i \geq 0, \forall i \in D$, y $\sum_{i=1}^{n} \alpha_i y_i = 0$

L_{dual} es un problema de programación cuadrática convexa, el cual puede resolverse con técnicas estándar de optimización.

Vector de peso y *bias*

Una vez quedan determinados los multiplicadores de Lagrange α_i con $i = 1, \ldots, n$ se puede calcular el vector de peso w y el escalar *bias* b. Recordando las condiciones KKT (27)

$$\alpha_i(y_i(w^T x_i + b) - 1) = 0$$

se deben distinguir dos casos:

- $\alpha_i = 0$
- $y_i(w^T x_i + b) - 1 = 0 \Longrightarrow y_i(w^T x_i + b) = 1$

Este es un resultado muy importante, ya que si $\alpha_i > 0$ entonces $y_i(w^T x_i + b) = 1$ y x_i deberá ser un vector de soporte. En caso de que $y_i(w^T x_i + b) > 1$ entonces $\alpha_i = 0$, es decir que si el punto no es un vector de soporte, entonces su multiplicador de Lagrange asociado será cero. Por tanto, w queda determinado por

$$w = \sum_{i, \alpha_i > 0} \alpha_i y_i x_i \tag{36}$$

luego w es combinación lineal de los vectores soporte. Para calcular b primero se calcula la solución b_i para cada vector de soporte como sigue

$$\begin{aligned} \alpha_i(y_i(w^T x_i + b) - 1) &= 0 \\ y_i(w^T x_i + b) &= 1 \\ b_i = \frac{1}{y_i} - w^T x_i &= y_i - w^T x_i \end{aligned} \tag{37}$$

Se puede determinar b como la media del *bias* de cada vector de soporte

$$b = \text{avg}_{\alpha_i > 0}\{b_i\} \tag{38}$$

Clasificador SVM

Dado el hiperplano óptimo $h_{w,b}(x) = w^T x + b = 0$, para cualquier nuevo punto z, se predice su clase mediante

$$\hat{\text{y}} = \text{sign}(h_{w,b}(z)) = \text{sign}(w^T z + b) \tag{39}$$

donde la función sign devuelve 1 si su argumento es positivo y -1 en caso de ser negativo.

Caso lineal pero no separable: *Soft margin SVM*

En esta sección se considera el caso en que las clases están mezcladas y una separación perfecta no es posible (se muestra un ejemplo en la Figura 6).

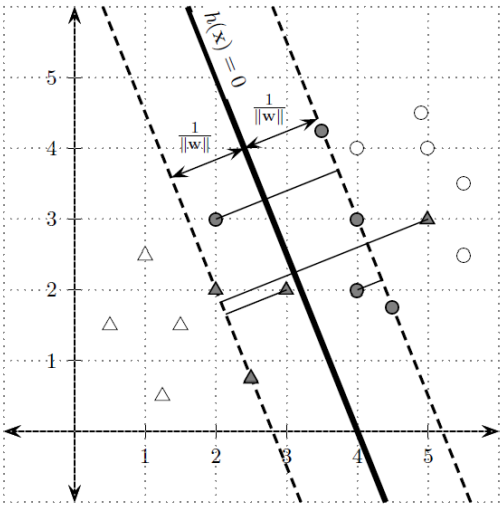


Figura 6: Hiperplano con margen suave: los puntos que se encuentran sobre la raya discontinua son los vectores de soporte, (el margen es de $\frac{1}{\|w\|}$). Los datos con *slack* positivo se encuentran sombreados pero a una distancia menor que de $\frac{1}{\|w\|}$. Fuente figura: [31, pg. 576]

El método *soft margin SVM* puede salvar la situación en la que los puntos sean no separables introduciendo nuevas variables *slack* ξ_i en la ecuación (32)

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

siendo $\xi_i \geq 0$ la variable *slack* correspondiente al punto x_i, la cual indica cuánto viola el punto la condición de separación. El valor de *slack* diferencia tres tipos de puntos: $\xi = 0$ (el punto se encuentra a una distancia mayor o igual que $\frac{1}{\|w\|}$ del hiperplano), $0 < \xi < 1$ (el punto se encuentra a menos de $\frac{1}{\|w\|}$ pero en la zona del subespacio correcta para la clase a la que pertenece según $h_{w,b}(x)$) y $\xi \geq 1$ (el punto está mal clasificado y se encuentra en el subespacio erróneo).

El objetivo de la clasificación SVM en el caso de puntos no separables es encontrar el hiperplano con mayor margen que a su vez minimice los términos *slack*. La nueva función objetivo viene dada por

- **Función objetivo:** $\text{mín}_{w,b,\xi_i} \left\{ \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\}$

- **Restricciones lineales:** $y_i(w^T x_i + b) \geq 1 - \xi_i,\ \text{con } \xi_i \geq 0\ \forall x_i \in D$

donde C y k son constantes que se incorporan para controlar el coste de clasificación errónea. El término $\sum_{i=1}^{n}(\xi_i)^k$ determina la pérdida, es decir, estima la diferencia del caso separable. C se determina empíricamente, se denomina constante de regularización y controla el equilibrio entre maximizar el margen o minimizar la pérdida. Por ejemplo, si $C \longrightarrow 0$ entonces el componente de pérdida desaparece prácticamente y la función objetivo se centra en maximizar el margen. Por otro lado, si $C \longrightarrow \infty$ el margen deja de tener tanto peso en la función objetivo y se centrará en reducir la pérdida. La constante k determina el tipo de pérdida y normalmente adopta los valores 1 o 2. Cuando $k = 1$ se llama *hinge loss* y si $k = 2$ se llamará pérdida cuadrática.

Para determinar el hiperplano separador se actúa de manera análoga a la forma anterior. Se calcula el Lagrangiano y el problema dual.

- Problema dual para el caso $k = 1$

  Función objetivo: $\text{máx}_\alpha\, L_{dual}(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$

  Restricciones lineales: $0 \leq \alpha_i \leq C, \forall i \in D,\ \ \sum_{i=1}^{n}\alpha_i y_i = 0$

- Problema dual para el caso k=2

  Función objetivo: $\text{máx}_\alpha\, L_{dual}(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i^T x_j + \frac{1}{2C}\delta_{ij})$

  Restricciones lineales: $\alpha_i \geq 0,\ \ \sum_{i=1}^{n}\alpha_i y_i = 0$

donde δ_{ij} es la función de *Knocher* que toma el valor $\delta_{ij} = 1$ si $i = j$ y $\delta_{ij} = 0$ si $i \neq j$. Se determinan los multiplicadores de Lagrange $\alpha_1, \ldots, \alpha_n$ y después se calculan w, b y las variables *soft* ξ_i que fijan el hiperplano obtenido.

- En el caso en que $k = 1$, para el cálculo de b y w se llega a la situación anterior $w = \sum_{i,\alpha_i>0}\alpha_i y_i x_i$ y $b_i = \frac{1}{y_i} - w^T x_i$, además $\xi_i = \frac{1}{2C}\alpha_i$
- Cuando $k = 2$ se calcula $w = \sum_{i,\alpha_i>0}\alpha_i y_i x_i$, $b = \text{avg}_{i,\alpha_i>0}\{y_i - w^T x_i\}$ que caracterizan el hiperplano buscado, y finalmente ξ_i mediante la igualdad $(C - \alpha_i)\xi_i = 0$

Una vez calculado el hiperplano y las variables *slack*, los nuevos puntos se clasifican según la norma de la sección anterior (39) introduciendo las nuevas variables ξ_i.

Caso no lineal: *kernel* SVM

El caso lineal de SVM puede aplicarse a *datasets* cuya frontera de decisión no es lineal si se aplica el truco del *kernel*. Conceptualmente, consiste en proyectar el espacio d-dimensional de puntos $\{x_i\}$ en el espacio de puntos de $\{\phi(x_i)\}$ que será de mayor dimensión y le llamaremos espacio de variables. Esta proyección se consigue a través de una transformación no lineal ϕ. Al obtener dicha flexibilidad, es más probable que los puntos $\phi(x_i)$ sean separables linealmente en el espacio de variables. Se debe notar que a pesar de que la frontera de decisión sea lineal en el espacio de variables, no lo será en el espacio d-dimensional. A través de la función *kernel* se pueden realizar operaciones en el espacio de variables. Para aplicar el truco del *kernel* a la clasificación no lineal SVM, se debe mostrar que las operaciones solo requieren de la función *kernel* que se define como

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Dado el *dataset* original $D = \{(x_i, y_i)_{i=1}^n\}$, si se aplica ϕ a cada punto, se puede obtener un nuevo *dataset* en el espacio de variables $D_\phi = \{(\phi(x_i), y_i)_{i=1}^n\}$. La función objetivo mostrada al principio del caso lineal y no separable en el espacio de variables es

- **Función objetivo:** $\text{mín}_{w,b,\xi_i}\left\{\frac{\|w\|^2}{2} + C\sum_{i=1}^{n}(\xi_i)^k\right\}$
- **Restricciones lineales:** $y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i$ con $\xi_i \geq 0, \forall x_i \in D$

donde w es el vector de peso y ξ_i son las variables *slack*, todo ello en el espacio de variables.

- *Hinge Loss*

Para el caso en que $k = 1$, el Lagrangiano dual en el espacio de variables queda

$$\begin{aligned} \max_{\alpha} \; L_{dual}(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \end{aligned} \tag{40}$$

sujeto a las restricciones $0 \leq \alpha_i \leq C$ y $\sum_{i=1}^{n} \alpha_i y_i = 0$. Nótese que el Lagrangiano dual depende únicamente de los productos entre dos vectores en el espacio de variables $\phi(x_i)^T \phi(x_j) = K(x_i, x_j)$ y entonces se puede resolver el problema de optimización mediante la matriz del kernel $K = \{K(x_i, x_j)\}_{i,j=1,\ldots,n}$. Una de las técnicas más comunes para resolver la función objetivo del problema dual es el gradiente ascendente que se tratará más adelante.

- Pérdida cuadrática

  Para la pérdida cuadrática, el Lagrangiano dual corresponde a un cambio de *kernel*. Se define la nueva función *kernel* K_q como sigue

  $$K_q(x_i, x_j) = x_i^T x_j + \frac{1}{2C} \delta_{ij} = K(x_i, x_j) + \frac{1}{2C} \delta_{ij}$$

  $K_q(x_i, x_j)$ solo afecta a las entradas de la diagonal de la matriz del *kernel* K, porque $\delta_{ij} = 1$ si y solo si $i = j$, y cero en otro caso. Luego el dual Lagrangiano queda

  $$\max_{\alpha} L_{dual}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_q(x_i, x_j) \tag{41}$$

  sujeto a las restricciones de que $\alpha_i \geq 0$ y que $\sum_{i=1}^{n} \alpha_i y_i = 0$. El problema de optimización puede solucionarse como el caso de *hinge loss* con un cambio de *Kernel*.

Veamos ahora cómo calcular el vector de peso w y el término *bias* b. w puede calcularse en el espacio de variables como

$$w = \sum_{\alpha_i > 0} \alpha_i y_i \phi(x_i) \tag{42}$$

Como w utiliza directamente $\phi(x_i)$, en general, no siempre será posible determinar explícitamente w, aunque más adelante veremos que no es necesario para clasificar los puntos. Para calcular b veamos primero que puede calcularse mediante operaciones del *kernel*, para ello calculamos b como la media de los vectores de soporte

$$b = \frac{1}{n_{sv}} \left(\sum_{\alpha_i > 0} y_i - \sum_{\alpha_i > 0} w^T \phi(x_i) \right)$$

donde n_{sv} es el número de vectores soporte $\alpha_i > 0$. Si se sustituye en la ecuación anterior la expresión (42) de w, se obtiene la siguiente nueva expresión de b:

$$\begin{aligned} b &= \frac{1}{n_{sv}} \left(\sum_{\alpha_i > 0} y_i - \sum_{\alpha_i > 0} \sum_{\alpha_j > 0} \alpha_i y_i \phi(x_i)^T \phi(x_j) \right) \\ &= \frac{1}{n_{sv}} \left(\sum_{\alpha_i > 0} y_i - \sum_{\alpha_i > 0} \sum_{\alpha_j > 0} \alpha_i y_i K(x_i, x_j) \right) \end{aligned}$$

Notar que al igual que w, b es una función del producto de vectores en el espacio de variables $\phi(x_i)^T \phi(x_j) = K(x_i, x_j)$. Veamos ahora como el clasificador *kernel* SVM predice la etiqueta de cualquier nuevo punto z

$$\begin{aligned}\hat{y} &= \text{sign}(w^T\phi(z) + b) \\ &= \text{sign}\left(\sum_{\alpha_i>0} \alpha_i y_i \phi(x_i)^T \phi(z) + b\right) \\ &= \text{sign}\left(\sum_{\alpha_i>0} \alpha_i y_i K(x_i, z) + b\right)\end{aligned}$$

Notar de nuevo que $\hat{y}$ solo utiliza el producto en el espacio de variables para calcularse.

2.2.1.2. Algoritmo k-Nearest Neighbor (KNN)

En esta sección se detalla el algoritmo de *k-Nearest Neighbor* basándose en [7, pgs. 13-15] y [22, pgs. 102-108]. KNN es un algoritmo de clasificación basado en instancias (en inglés, *instance-base algorithm*). Estos algoritmos se caracterizan por memorizar el conjunto de entrenamiento. Los *lazy learner* son un caso especial de ellos en el que el proceso de aprendizaje tiene coste cero. Los algoritmos basados en instancias se rigen por el siguiente principio: "Instancias similares tienen etiquetas similares". La forma de actuar del algoritmo KNN es la siguiente: para un dato, se determinan los k datos más próximos (vecinos), se observan las etiquetas de estos y al punto dado se le asigna la etiqueta que tienen la mayoría de sus vecinos.

Una de las dos tareas más difíciles del algoritmo KNN es determinar el parámetro k que indica el número de vecinos que deben observarse para determinar la etiqueta del elemento que queremos clasificar. Un valor pequeño de k no nos hará obtener un modelo de clasificación robusto, debido a que la clasificación se encontrará vinculada a posibles variaciones de los puntos del *dataset*. Por otro lado, un valor demasiado grande provoca la pérdida de sensibilidad local que caracteriza el algoritmo. La mejor estrategia para determinar k será aquella que se adapte a los datos y suele determinarse mediante heurísticos. Un enfoque bastante común es testear distintos valores de k con el conjunto de entrenamiento y mediante la selección de alguna medida de bondad del ajuste (precisión) concretar el mejor valor de k.

La otra tarea pendiente antes de poder utilizar el algoritmo es proporcionar una noción de similitud válida. Para ello, se debe seleccionar una métrica de distancia apropiada para las variables de las que se dispone. Existen muchas variaciones del clasificador del vecino más cercano y algunas funcionan mejor que otras. Esto es debido a la selección de las métricas (aunque también del parámetro k). Por ejemplo, la distancia Euclídea funciona bien cuando las variables son valores reales y se encuentran en la misma escala. Por ello, si se opta por esta función es conveniente normalizar las variables antes de aplicar el método.

La distancia entre dos puntos d-dimensionales x_i y x_j se define mediante una matriz $A \in \mathbb{R}^{d\times d}$

$$\text{Dist}(x_i, x_j) = \sqrt{(x_i - x_j)A(x_i - x_j)^T} \tag{43}$$

Notar que esta función coincide con la métrica euclídea si $A = I_d$. Las distintas elecciones de A inducen a distintas métricas y una mejor selección de A puede llevar a que el método KNN tenga mejor sensibilidad respecto a las distribuciones locales y globales de los datos.

Para obtener mejores resultados con el clasificador KNN, la función de distancia debería tener en cuenta las distintas distribuciones de las diferentes clases como se explica en [2, pgs. 331-334]. Una forma de lograrlo es asignar peso a las direcciones más discriminantes en la función de distancia mediante una selección apropiada de la matriz A que aparece en (43). A determina la forma que tendrá el vecindario para cada dato. Una distorsión del vecindario proporcionado por la distancia euclídea "circular" corresponderá a *soft weighting*, más detalles en [2, pgs. 333-334]. Esta forma de seleccionar la matriz A hace que el clasificador KNN sea mas robusto sobretodo en los *datasets* con pocos elementos, donde es muy susceptible de realizar *overfitting*. Por tanto, la idea es: alargar la forma del vecindario en la dirección más discriminante de las clases y reducir el vecindario en las direcciones menos discriminantes, mediante la elección de A.

Un ejemplo de matriz que logra dicho objetivo es la empleada en la distancia de Mahalanobis. En este caso, el nivel de elongación logrado por A es inversamente proporcional a la varianza de las distintas direcciones.

Las métricas seleccionadas en este trabajo son las siguientes:

- Métrica de Mahalanobis

$$\text{Dist}(x_i, x_j) = \sqrt{(x_i - x_j)\Sigma^{-1}(x_i - x_j)^T} \tag{44}$$

La matriz Σ es la matriz de varianzas-covarianzas entre los pares de dimensiones. Esta métrica se adapta bien a las distintas escalas de las variables. Es una métrica útil porque se auto-escala para los distintos rangos de los atributos que describen las distintas características de los datos. En esta métrica no habrá un atributo que domine la función de distancia. La debilidad de esta función de distancia es que no varía la forma de los vecindarios en función de las distribuciones subyacentes de las clases.

- Métrica euclídea

$$\text{Dist}(x_i, x_j) = \sqrt{(x_i - x_j) I_{d\times d} (x_i - x_j)^T} \tag{45}$$

Es una de las métricas más empleadas, pero en el caso de KNN no es la más precisa debido a que no tiene en cuenta distintas formas de vecindario ni tampoco tiene en cuenta los rangos en los que se encuentran las distintas variables de cada punto.

- Métrica de Minkowski

$$\text{Dist}(x_i, x_j) = \left(\sum_{k=1}^{n} |x_{ik} - y_{ik}|^p \right)^{\frac{1}{p}} \tag{46}$$

donde la k-ésima coordenada del elemento x_i se denota por x_{ik}.

El pseudocódigo del algoritmo KNN encontrado en [19] lo muestra el Algoritmo 3.

Algoritmo 3 Pseudocódigo del algoritmo KNN

Entrada: $D = \{(x_i, y_i)_{i=1}^n\}$ y x' un nuevo elemento a clasificar.
1: **para todo** $(x_i, y_i) \in D$ **hacer**
2: Calcular $d_i = Dist(x_i, x')$
3: **fin para**
4: Ordenar d_i para $i = 1, \ldots, n$ en orden ascendente y guardar en D'
5: Sea $D_{x'}$ el conjunto de los primeros k elementos de D'
6: Asignar a x' la clase más frecuente en $D_{x'}$
7: **devolver** La etiqueta de x': y'

Complejidad computacional: Sea cual sea el valor de k, cada vez que se quiera clasificar un dato nuevo se deberá recorrer el *dataset* para calcular la distancia a la que se encuentra de cada elemento y quedarse con los k mas cercanos. Por lo tanto, la complejidad final será $\mathcal{O}(nl)$, siendo l la complejidad del cálculo de la distancia para cada punto. Veamos qué complejidad se tiene en las distancias mencionadas [5]:

- Complejidad de la distancia euclídea y la distancia de Minkowski: es lineal respecto al número de variables o características d. Por tanto, en este caso la complejidad de KNN es $\mathcal{O}(nd)$
- Complejidad de la distancia de Mahalanobis: es cuadrática respecto al número de variables o características d. Por lo tanto, la complejidad total de KNN con esta distancia será: $\mathcal{O}(nd^2)$

Esta complejidad puede verse mejorada al aplicar métodos cuyo objetivo es la eliminación de cálculos innecesarios. Estos métodos pueden llegar a conseguir órdenes de complejidad de $\mathcal{O}(\log n)$ en el mejor de los casos. Entre ellos se encuentran algunos basados en ordenaciones y jerárquicos (ramificación y poda) como se ve en [5].

El algoritmo KNN presenta algunos inconvenientes:

- En espacios de dimensión baja el algoritmo funciona muy bien, pero cuando se incrementa el número de variables surge el problema de "la maldición de las dimensiones". Se trata de un fenómeno en el que al aumentar la dimensión y mantener el número de datos, dos elementos que antes podían encontrarse cercanos ya no lo estarán y al aumentar exageradamente el número de dimensiones todas las distancias pueden tender a ser muy grandes y entonces el método de KNN deja de ser preciso. Para solucionar dicho problema puede aplicarse regularización o un método para reducir las dimensiones como se explica en [29].
- No existe un mecanismo concreto para encontrar el valor óptimo de k, depende de cada conjunto de datos.

- El coste de encontrar los k vecinos es demasiado grande cuando n es muy grande.

Por otro lado, también tiene ventajas:

- El coste del aprendizaje es cero.
- Es fácil adaptar el método para regresión (valores continuos).

2.2.1.3. Método de regresión logística

En esta sección se tratará el método de regresión logística con información de [2, pgs. 306-312]. Este método se engloba dentro de los métodos de clasificación probabilísticos. Tratan de cuantificar la relación entre las variables y la clase con probabilidades de pertenencia a cada clase y una función discriminante. Los parámetros del modelo probabilístico se estiman con los datos de entrenamiento. Se limita el estudio al caso de clasificación binaria donde $y_i \in \{0, 1\}$. En la regresión logística se cuenta con la función de sigmoid o función logística $s : \mathbb{R} \longrightarrow (0, 1)$, donde $s(x) = \frac{1}{1+e^{-x}}$, para modelar las probabilidades de pertenencia a cada clase.

Se cuenta también con una función $h_{w,b} : \mathbb{R}^d \longrightarrow (0, 1)$ llamada función hipótesis, $h_{w,b}(x) = P(y = 1 \mid x, w, b)$, que representa la probabilidad de que x pertenezca a la clase 1 dados los parámetros w y b, con $w \in \mathbb{R}^d$ y $b \in \mathbb{R}$. Los valores w_i son los coeficientes de la dimensión i para $i = 1, \ldots, d$ y b es el término *offset*. Dado que nos encontramos ante un problema de clasificación binaria, los puntos solo pueden pertenecer a la clase 1 o a la clase 0. Por tanto, $P(y = 1 \mid x, w, b) + P(y = 0 \mid x, w, b) = 1$, y así queda determinada $P(y = 0 \mid x, w, b)$.

Para cualquier dato $x = (x_1, \ldots, x_d)$, la probabilidad de que pertenezca a la clase 1 ($y = 1$) o a la clase 0 ($y = 0$) queda modelada por

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(b+\sum_{i=1}^{d} w_i x_i)}} \tag{47}$$

$$P(y = 0 \mid x) = \frac{1}{1 + e^{b+\sum_{i=1}^{d} w_i x_i}} \tag{48}$$

Notar que la ecuación (48) deriva de la ecuación (47) y del hecho de que la clasificación es binaria y por tanto, la suma de ambas debe ser 1.

Para predecir la clase de un punto dado se utiliza la función de hipótesis $h_{w,b}$, con la siguiente regla de decisión

$$y = \begin{cases} 1 & \text{si } h_{w,b}(x) \geq 0{,}5 \\ 0 & \text{si } h_{w,b}(x) < 0{,}5 \end{cases} \tag{49}$$

Se deduce que el punto x pertenece a la clase 1 si $h_{w,b}(x) \geq 0{,}50$, es decir, si $w^T x + b \geq 0$ (en vista de la función de sigmoid s). Análogamente, el punto x pertenece a la clase 0 si $h_{w,b}(x) < 0{,}50$, es decir, si $w^T x + b < 0$. Por tanto, si se debe predecir la clase a la que pertenece cierto punto, basta con comprobar el signo de $w^T x + b$. Así, la frontera de decisión se encuentra como solución a $w^T x + b = 0$.

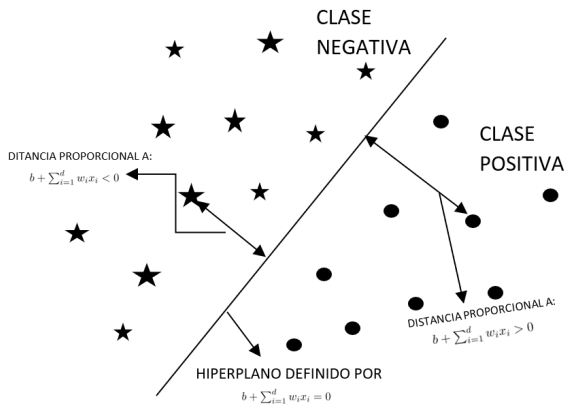


Figura 7: Ilustración del modelo de regresión logística en términos de separadores lineales.

El método de regresión logística puede verse como un clasificador probabilístico, pero también como un clasificador lineal (se utiliza un hiperplano para separar dos clases). Los parámetros

$(b, w) = (b, w_1, \ldots, w_d)$ pueden verse como los coeficientes del hiperplano separador $b+\sum_{i=1}^{d} w_i x_i = 0$. El término w_i es el coeficiente lineal de la dimensión i y el término b es el término constante (*bias*). Como se ha visto antes el valor de $b + \sum_{i=1}^{d} w_i x_i$ puede ser negativo o positivo, en función de la zona que delimita el hiperplano a la que pertenezca.

El término $w^T x + b$ sin la exponencial de la función logística es proporcional a la distancia del punto al hiperplano separador. Cuando el dato se encuentra en el hiperplano se asigna 0,50 probabilidad de pertenecer a ambas clases. Por lo tanto, la función logística exponencia las distancias para convertirlas en probabilidades. Esta idea se plasma en la Figura 7.

- **Estimar los parámetros** w, b

  Para estimar los parámetros de la función se suele utilizar el estimador de máxima verosimilitud. Sean D_+ y D_- los subconjuntos del conjunto de entrenamiento que pertenecen a la clase 1 y 0 respectivamente. Sea el k-ésimo elemento del *dataset* denotado por $x_k = (x_{k1}, \ldots, x_{kd})$. Entonces la función de verosimilitud $\mathcal{L}(w, b)$ para el conjunto de datos del *dataset* es

$$\mathcal{L}(w, b) = \prod_{x_k \in D_+} \frac{1}{1 + e^{-(b+\sum_{i=1}^{d} w_i x_{ki})}} \prod_{x_k \in D_-} \frac{1}{1 + e^{(b+\sum_{i=1}^{d} w_i x_{ki})}} \tag{50}$$

  La función de verosimilitud es el resultado del producto de las probabilidades de los elementos del conjunto de entrenamiento de acuerdo con el modelo de regresión logística. El objetivo es maximizar dicha función para determinar el valor óptimo de los parámetros w y b. De nuevo, se utilizarán logaritmos para simplificar la optimización de la función de similitud

$$\mathcal{LL}(w, b) = \log\left(\mathcal{L}(w, b)\right) = -\sum_{x_k \in D_+} \log\left(1 + e^{-(b+\sum_{i=1}^{d} w_i x_{ki})}\right) - \sum_{x_k \in D_-} \log\left(1 + e^{(b+\sum_{i=1}^{d} w_i x_{ki})}\right) \tag{51}$$

  Existen muchos métodos para optimizar la función de verosimilitud. Un planteamiento común es emplear el método del gradiente ascendente para llegar al valor del parámetro óptimo de manera iterativa.

  El método del gradiente ascendente se fundamenta en la noción de la derivada en $\mathbb{R}$, donde el gradiente representa la pendiente de la función en cada punto. Dado que la función del logaritmo de verosimilitud es cóncava, para aproximarnos al máximo de la función solo debemos avanzar en la dirección del gradiente. Para ello se utiliza un parámetro α que se conoce como ratio de aprendizaje (representa el paso en cada iteración). La elección de este parámetro es crucial, ya que si α fuera demasiado grande podría darse el caso en que nos saltáramos el máximo y el método divergiera. En cambio, si el valor de α es demasiado pequeño el algoritmo podría tardar demasiado en converger. Con una correcta elección de α se tiene garantizada la convergencia del método del gradiente debido a que la función es cóncava.

  El vector gradiente se obtiene como resultado de derivar el logaritmo de la función de verosimilitud respecto a cada parámetro

$$\nabla\mathcal{LL}(w, b) = \left(\frac{\partial\mathcal{LL}(w, b)}{\partial b}, \frac{\partial\mathcal{LL}(w, b)}{\partial w_1}, \ldots, \frac{\partial\mathcal{LL}(w, b)}{\partial w_d}\right) \tag{52}$$

  Al calcular la derivada parcial a ambos lados en la ecuación (51) respecto de w_i, se obtiene

$$\begin{aligned} \frac{\partial\mathcal{LL}(w, b)}{\partial w_i} &= \sum_{x_k \in D_+} \frac{x_{ki}}{1 + e^{(b+\sum_{i=1}^{d} w_i x_{ki})}} - \sum_{x_k \in D_-} \frac{x_{ki}}{1 + e^{-(b+\sum_{i=1}^{d} w_i x_{ki})}} \\ &= \sum_{x_k \in D_+} P(x_k \in D_-) x_{ki} - \sum_{x_k \in D_-} P(x_k \in D_+) x_{ki} \\ &= \sum_{x_k \in D_+} P(\text{Error } (1)) x_{ki} - \sum_{x_k \in D_-} P(\text{Error } (2)) x_{ki} \end{aligned} \tag{53}$$

  Entonces los términos $P(x_k \in D_-)$ y $P(x_k \in D_+)$ representan la probabilidad de realizar una predicción incorrecta de x_k en las clases 0 y 1. Error (1) significa clasificar el dato x_k en

la clase 0 si pertenece a la clase 1, y Error (2) significa clasificar el elemento x_k en la clase 1 cuando pertenece a la clase 0.

El caso particular de derivar respecto a b queda

$$\begin{aligned}\frac{\partial \mathcal{LL}(w,b)}{\partial b} &= \sum_{x_k \in D_+} \frac{1}{1+e^{(b+\sum_{i=1} w_i x_{ki})}} - \sum_{x_k \in D_-}^{d} \frac{1}{1+e^{-(b+\sum_{i=1}^{d} w_i x_{ki})}} \\ &= \sum_{x_k \in D_+} P(x_k \in D_-) - \sum_{x_k \in D_-} P(x_k \in D_+) \\ &= \sum_{x_k \in D_+} P(\text{Error (1)}) - \sum_{x_k \in D_-} P(\text{Error (2)})\end{aligned}$$

Entonces, el error del modelo se emplea para identificar las direcciones de ascenso. Además, el factor multiplicativo x_{ki} afecta a la magnitud de la componente i-ésima de la dirección del gradiente con x_k. Así, los parámetros w_i y b se actualiza en cada iteración como sigue

$$w_i \longleftarrow w_i + \alpha \left(\sum_{x_k \in D_+} P(x_k \in D_-) x_{ki} - \sum_{x_k \in D_-} P(x_k \in D_+) x_{ki} \right) \tag{54}$$

$$b \longleftarrow b + \alpha \left(\sum_{x_k \in D_+} P(x_k \in D_-) - \sum_{x_k \in D_-} P(x_k \in D_+) \right) \tag{55}$$

Los métodos de regularización se emplean para evitar el *overfitting*. Un ejemplo típico de término de regularización que se añade a la función de verosimilitud $\mathcal{LL}(w)$ es $\lambda \sum_{i=1}^{d} \frac{w_i^2}{2}$, donde λ es el parámetro que equilibra. La única diferencia respecto a la actualización del gradiente es que el término λw_i necesita ser añadido a la componente i-ésima del gradiente.

2.2.2. Problema de regresión

En el problema de clasificación se intenta aprender la dependencia del valor discreto de cierta variable en función de los valores de otras. En el caso de regresión, la variable a predecir (variable de respuesta) tiene valores continuos. Las demás variables son las variables predictoras. El objetivo de los métodos que se presentan es construir un modelo a partir de una muestra de datos (cuyas variables predictoras y de respuesta son conocidas), que sea capaz de predecir el valor de la variable de respuesta para nuevos datos cuyas variables predictoras son conocidas.

El problema, como muestra [22, pgs. 315-318], se soluciona al encontrar una función $h_{w,b} : \mathbb{R}^d \longrightarrow \mathbb{R}$ de la forma $h_{w,b}(x) = w^T x + b = b + w_1 x_1 + \ldots + w_d x_d$ que sirva para estimar el valor que toma la variable predictora de cara a las nuevas instancias. La función $h_{w,b}$ se denomina función hipótesis, como en el caso de la regresión logística.

2.2.2.1. Regresión lineal

Modelos lineales

Son modelos de predicción que estiman una variable de respuesta en función de las variables predictoras mediante una función lineal

$$y = b + \sum_{i=1}^{d} w_i x_i \tag{56}$$

donde $(b, w) = (b, w_1, \ldots, w_d)$ es el vector de parámetros a estimar para el modelo, e y es la variable de respuesta asociada al dato $x = (x_1, \ldots, x_d)$.

Geométricamente, este modelo describe un hiperplano d-dimensional contenido en un espacio de dimensión $d+1$, determinado por los parámetros w_i $(i \geq 1)$ que suelen llamarse pesos (y que cuantifican la importancia de la variable i-ésima, x_i, en el ajuste) y por el parámetro b llamado *bias*. El objetivo del modelo es determinar los parámetros que caracterizan la localización de dicho hiperplano así como su ángulo, de tal forma que aproxime lo mejor posible los elementos del *dataset*.

Los modelos con esta estructura lineal tienen bastante protagonismo en el ámbito del análisis de datos, debido a que la estimación de los parámetros es simple si se selecciona la función adecuada y en parte porque la estructura del modelo también es sencilla y fácil de interpretar. Una característica a resaltar es la naturaleza aditiva del modelo, que otorga la posibilidad de modificar el valor de una variable sin afectar a las demás.

Regresión lineal multivariante

Los datos de los que se dispone en el estudio se encuentran en $\mathbb{R}^{66}$, por tanto, se aplicará regresión multivariante. Se debe encontrar la función lineal

$$h_{w,b}(x) = w^T x + b = b + w_1 x_1 + \dots w_d x_d$$

que mejor se adapte al *dataset*. En este caso, la función de coste (la suma de los errores cuadráticos, notar que la constante $\frac{1}{2}$ se introduce para simplificar el gradiente) es

$$J(w,b) = \frac{1}{2} \sum_{1 \leq i \leq n} (y_i - h_{w,b}(x_i))^2 \tag{57}$$

Para minimizar la función de coste, se puede optar por dos alternativas: el algoritmo del gradiente descendente mencionado en la sección anterior (método iterativo) o el cálculo de las ecuaciones normales (método analítico).

Método iterativo: gradiente descendente

El gradiente de la función de coste es $\nabla J(w,b) = (\frac{\partial J(w,b)}{\partial b}, \frac{\partial J(w,b)}{\partial w_1}, \dots, \frac{\partial J(w,b)}{\partial w_d})$ con

$$\frac{\partial J(w,b)}{\partial w_j} = - \sum_{1 \leq i \leq n} x_{ij}(y_i - h_{w,b}(x_i)) \quad \forall j \in \{1, \dots, d\} \tag{58}$$

$$\frac{\partial J(w,b)}{\partial b} = - \sum_{1 \leq i \leq n} (y_i - h_{w,b}(x_i)) \tag{59}$$

con $x_i = (x_{i1}, \dots, x_{id}) \in D$ e y_i es la etiqueta de x_i. El objetivo del descenso por gradiente debido a que la función es convexa, es buscar el mínimo de la función iterando en la posición opuesta al gradiente (esto se consigue restando en cada actualización, misma idea que en regresión logística). Por tanto, se van actualizando los parámetros w y b en cada iteración como sigue

$$w_j := w_j - \alpha\Big(\frac{\partial J(w,b)}{\partial w_j}\Big) \tag{60}$$

$$b := b - \alpha\Big(\frac{\partial J(w,b)}{\partial b}\Big) \tag{61}$$

El pseudocódigo de la regresión lineal multivariante queda plasmado en el Algoritmo 4. En cuanto a la convergencia del algoritmo, queda garantizada debido a la convexidad de la función J como explican en [34] (bajo una correcta elección de α).

Algoritmo 4 Pseudocódigo regresión lineal multivariante.

Entrada: $D = \{(x_i, y_i)_{i=1}^n\}$, tolerancia ϵ
  Inicializar los parámetros $(b, w_1, \dots, w_d)$ de manera aleatoria
  repetir
    $w_j^t \leftarrow w_j^{t-1} - \alpha(\frac{\partial J(w^{t-1}, b^{t-1})}{\partial w_j})$
    $b^t \leftarrow b^{t-1} - \alpha(\frac{\partial J(w^{t-1}, b^{t-1})}{\partial b})$
  hasta que $|J(w^{t-1}, b^{t-1}) - J(w^t, b^t)| \leq \epsilon$
  devolver Parámetros w^t y b^t

Método analítico: ecuaciones normales

Para minimizar la función de coste en función de w y b, basta con derivar respecto a cada uno de los parámetros e igualar a cero

$$\frac{\partial J(b,w)}{\partial w_j} = - \sum_{1 \leq i \leq n} x_{ij}(y_i - h_{w,b}(x_i)) = 0 \qquad \forall j \in \{1, \dots, d\} \tag{62}$$

$$\frac{\partial J(b,w)}{\partial b} = - \sum_{1 \leq i \leq n} (y_i - h_{w,b}(x_i)) = 0 \tag{63}$$

Mediante el uso de matrices, el resultado queda de forma más compacta. Se define la matriz $X \in \mathbb{R}^{n\times(d+1)}$ (64) donde cada fila tiene en la primera posición un 1 y el resto es un elemento del *dataset* e y es un vector columna que almacena los valores de la variable respuesta de cada dato. Para facilitar la notación se define el parámetro $\theta = (b, w)$.

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nd} \end{bmatrix} \tag{64}$$

$$Y = (y_1, \dots, y_n)^T$$

x_{ij} es el valor de la variable j-ésima del elemento i-ésimo del *dataset* D, y_i la variable de respuesta del dato i-ésimo de D.

Al derivar e igualar a 0 se obtiene el siguiente sistema de ecuaciones

$$X^T(X\theta - Y) = 0$$

Al despejar se obtiene que el mínimo en la función J se alcanza en

$$\begin{aligned} (X^T X)\theta &= X^T Y \\ \theta &= (X^T X)^{-1} X^T Y \end{aligned}$$

Comparación de los métodos

Cada método tiene sus ventajas e inconvenientes. La principal ventaja del método del gradiente descendente es que funciona bien incluso para d muy grande. Como desventajas, que hay que seleccionar un α, hace falta iterar muchas veces y escalar las variables. Por otro lado, en cuanto a las ecuaciones normales, tienen la ventaja de que no hay que seleccionar ningún parámetro y no hace falta iterar. Basta con calcular $(X^t X)^{-1}$ aunque, si d es muy grande puede hacer que el método sea muy lento, pues calcular la inversa de una matriz requiere complejidad $\mathcal{O}(d^3)$ siendo d la dimensión de la matriz.

El problema de *overfitting* es bastante común en regresión lineal, por ello se introduce un término regularizador a la función de coste J que depende del parámetro regularizador λ.

$$J(\theta) = \frac{1}{2} \sum_{1\leq i\leq n} (y_i - h_{w,b}(x_i))^2 + \lambda \|w\|_q$$

En el caso en que $q = 1$ se llama *lasso* y corresponde a la norma 1: $\lambda\|w\|_1 = \lambda \sum_{i=1}^{n} |w_i|$. En función de la fuerza de regularización (determinada por λ) podría ocurrir que ciertas coordenadas de w fueran 0, lo cual convierte a la penalización *lasso* como un útil seleccionador de variables. Si $q = 2$, se obtiene el término regularizador siguiente: $\lambda \sum_{i=1}^{n} w_i^2$, que se conoce como *ridge*.

2.2.2.2. Regresión polinómica

En la sección anterior se estudió la regresión lineal, que es una herramienta muy utilizada y muy potente a la hora de inferir sobre datos nuevos. Pero cuando los datos no guardan relación lineal no funciona tan bien. Una forma de solucionar dicho problema es utilizando la regresión lineal añadiendo términos polinómicos a la función $h_{w,b}$ planteada en la sección anterior, como se detalla en [22, pgs. 315-318].

Ahora para regresión polinómica se añaden dimensiones a los datos mediante el producto de los distintos atributos iniciales de los que se dispone. Por ejemplo, dado el x anterior se podría obtener el nuevo dato para regresión polinómica $\hat{x} = (x_1, \dots, x_d, x_1^2, x_2^2, \dots, x_d^2)$. En este caso simplemente se han elevado al cuadrado las variables, pero podría escogerse cualquier combinación de los atributos iniciales. Por tanto, haciendo la transformación de las variables pertinentes, se trata de un caso de regresión lineal, pues se deberán encontrar los parámetros w y b siguientes

$$h_{w,b}(x) = w^T \hat{x} + b = b + w_1 x_1 + w_2 x_2 + \dots + w_d x_d + w_{d+1} x_1^2 + \dots + w_{d+d} x_d^2 \tag{65}$$

En este caso, las transformaciones de las variables se han realizado de acuerdo con el ejemplo anterior (elevando cada una de ellas al cuadrado). La diferencia con la regresión lineal es el aumento de dimensiones, que dependerá de los términos polinómicos que se quiera añadir. Otra diferencia a

resaltar es la necesidad de normalizar las variables al elevarlas al cuadrado (en este caso) o aplicar las transformaciones que se hayan seleccionado, para evitar dar mayor importancia a unas variables frente a otras. De la misma forma puede generalizarse para cualquier grado de polinomio. En este trabajo, debido a los pocos datos de los que se dispone (22) y el gran número de variables (66), solo se ha llevado a a práctica la regresión polinómica con los polinomios de grado dos y tres (se desaconseja en esta situación utilizar polinomios de grado muy alto). Este método continúa estando dentro de la regresión lineal debido a que los parámetros a estimar w y b siguen siendo lineales. Las principales desventajas de la regresión polinomial son que conforme se incrementa el grado del polinomio, el número de variables también lo hace y sobretodo si el *dataset* o las dimensiones iniciales son muchas puede hacer que los datos sean difíciles de manejar. Otro de los inconvenientes es que suele tender a realizar *overfitting* incluso aunque se encuentre en un espacio 1-dimensional (más detalles en [10] y [26]). Por eso no se recomienda utilizar polinomios de grado muy elevado.

2.3. Representación de datos en altas dimensiones

Debido a que el resultado del experimento es un *dataset* con puntos en $\mathbb{R}^{66}$, no es posible visualizarlos y por ello se ha recurrido a una técnica de reducción de dimensiones conocida como t-SNE.

t-SNE hace posible visualizar datos con altas dimensiones asignando a cada punto otro en un mapa de bi- o tri-dimensional. La técnica es una variación de *Stochastic Neighbor Embedding* (SNE) [11], mucho más sencillo de optimizar y con mejores visualizaciones debido a que reduce la tendencia de los puntos a agruparse en el centro del mapa. SNE utiliza la probabilidad condicionada de similitud entre puntos en lugar de la distancia euclídea.

Los métodos de reducción de dimensiones convierten *datasets* de altas dimensiones $D = \{(x_i)_{i=1}^n\}$ en datos bi- o tri-dimensionales $V = \{(v_i)_{i=1}^n\}$ que puedan mostrarse en una gráfica. Normalmente se refiere a la representación en dimensiones menores V como mapa y las representaciones de los puntos en baja dimensión v_i como puntos del mapa. Al reducir las dimensiones se persigue conservar al máximo la estructura de los datos iniciales en el mapa de menor dimensión. Se han propuesto muchos métodos para conseguir este objetivo, entre ellos cabe resaltar otro muy conocido: *Principal Components Analysis* (PCA) [12]; es una técnica lineal que convierte observaciones de variables posiblemente correlacionadas en un conjunto de valores de variables sin correlación lineal llamadas componentes principales.

SNE comienza convirtiendo la distancia euclídea de alta dimensión entre los puntos iniciales en $p_{j|i}$ que representa la similitud entre los puntos x_i y x_j. $p_{j|i}$ es la probabilidad condicionada de que x_i y x_j sean vecinos, si los vecinos se eligen en proporción a su densidad de probabilidad bajo una distribución gaussiana centrada en x_i. Para los vecinos cercanos, $p_{j|i}$ será relativamente alto, mientras que para los puntos más alejados, $p_{j|i}$ será prácticamente infinitesimal. Matemáticamente, la probabilidad condicionada $p_{j|i}$ viene dada por

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|^2/2\sigma_i^2}} \tag{66}$$

donde σ_i es la varianza de la gaussiana centrada en el punto x_i.

Se fija el valor de $p_{i|i} = 0$. Para los puntos del mapa, sean v_i y v_j los representantes de x_i y x_j respectivamente; es posible calcular una probabilidad condicionada similar, que se denota por $q_{j|i}$. Se fija el valor de la varianza de la gaussiana utilizada en el cálculo de $q_{j|i}$ como $\frac{1}{\sqrt{2}}$. (El hecho de fijar la varianza de baja dimensión a un valor u otro afecta en que el mapa se encontrará reescalado de una versión a otra, pero las similitudes y diferencias se mantienen en proporción). Se puede modelar la similitud entre los puntos v_j y v_i como sigue

$$q_{j|i} = \frac{e^{-\|v_i - v_j\|^2}}{\sum_{k \neq i} e^{-\|v_i - v_k\|^2}}$$

De nuevo, se asigna el valor 0 a $q_{i|i}$. Si los puntos v_i e v_j modelan correctamente la similitud entre los puntos iniciales de los que provienen (x_i y x_j), la probabilidad condicionada $p_{j|i}$ debe ser igual a $q_{j|i}$. En vista de esta observación, el objetivo de SNE es encontrar una representación en el mapa que minimice la diferencia entre $q_{j|i}$ y $p_{j|i}$. Una medida natural para cuantificar lo bien que modela $q_{j|i}$ a $p_{j|i}$ es la divergencia de *Kullback-Leibler*. Mide la similitud o diferencia entre dos distribuciones de probabilidad P y Q y se define como

$$KL(P\|Q) = \sum_i P_i \log\left(\frac{P_i}{Q_i}\right)$$

con $P_i = (p_{j|i})_{j \neq i}$ y $Q_i = (q_{j|i})_{j \neq i}$ probabilidades. SNE minimiza la suma de la divergencia de *Kullback-Leiber* sobre todos los puntos a través del método del gradiente descendiente. La función de coste J de SNE es

$$J = \sum_i \mathrm{KL}(P_i \| Q_i) = \sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{67}$$

donde P_i representa la distribución de probabilidad de los puntos iniciales, mientras que Q_i es la distribución de probabilidad de los puntos en el mapa. Como la divergencia de *Kullback-Leiber* no es simétrica, errores de distinto tipo en las distancias entre pares en el mapa de dimensión baja no tienen el mismo peso. En particular, se penaliza mucho tomar puntos muy separados del mapa de alta dimensión y representarlos como puntos cercanos en el mapa de baja dimensión (lo que equivale a modelar con una baja $q_{j|i}$ una gran $p_{j|i}$). En cambio, se penaliza poco el error de representar en un mapa de baja dimensión puntos alejados que se encuentran juntos en el mapa de alta dimensión. Entonces, SNE se centra más en conservar la estructura local de los datos del mapa.

El parámetro que queda por seleccionar es la varianza σ_i de la gaussiana centrada en cada punto del espacio de alta dimensión, x_i. Desafortunadamente, no existe un valor óptimo que funcione bien para todos los puntos del mapa porque depende de la densidad de los datos en cada punto. En las regiones con más puntos concentrados, es decir, con mayor densidad, es más apropiado utilizar valores bajos de σ_i que altos, que suelen utilizarse para regiones con pocos puntos. Para cada valor de σ_i se induce una distribución de probabilidad, P_i, sobre el resto de puntos del mapa. SNE realiza una búsqueda binaria para el valor de σ_i, que produce cierta P_i con una *perplexity* fija que es introducida por el usuario. Se puede definir la *perplexity* como sigue:

$$\mathrm{Perp}(P_i) = 2^{H(P_i)}$$

siendo $H(P_i)$ la entropía de Shannon de P_i que se mide en bits.

$$H(P_i) = -\sum_{j \neq i} p_{j|i} \log_2 p_{j|i}$$

La *perplexity* puede interpretarse como una medida de suavidad (*smooth measure*) del número adecuado de vecinos. El desarrollo de SNE se considera robusto en cuanto a la variabilidad del parámetro de *perplexity*. Los valores usuales para dicho parámetro se encuentran entre 5 y 50.

La minimización de la función de coste de la ecuación (67) se realiza mediante el gradiente descendiente y el gradiente de la función de coste es

$$\frac{\partial J(v_1, \ldots, v_n)}{\partial v_i} = 2 \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(v_i - v_j)$$

El gradiente descendente se inicializa con una muestra aleatoria de una *isotropic gaussian* (una distribución gaussiana cuya matriz de varianzas-covarianzas es $\Sigma = 2\sigma I$) con varianza pequeña y centrada en el origen. La actualización en cada iteración del conjunto de representantes en el mapa se realiza como

$$V := V - \alpha \frac{\partial J(v_1, \ldots, v_n)}{\partial V} \tag{68}$$

T-Distributed Stochastic Neighbor Embedding (t-SNE)

A pesar de proporcionar buenas visualizaciones de los datos, SNE presenta dos inconvenientes principales: en primer lugar, la dificultad para optimizar la función de coste y en segundo lugar, el conocido *crowding problem* (problema de hacinamiento). La técnica t-SNE pretende aliviar estas dificultades. La función de coste empleada en t-SNE difiere de la de SNE en dos aspectos: (1) emplea una versión simétrica de la función de coste de SNE con el gradiente simple que fue introducido por Cook [6]; (2) utiliza la distribución t-Student en lugar de la gaussiana para calcular la similitud entre dos puntos en el mapa. t-SNE emplea esta distribución en el espacio de dimensión baja para mejorar el problema de optimización y el problema de hacinamiento.

- Versión simétrica de la función de coste de SNE

  En vez de minimizar la suma de las divergencias de *Kullback-Leibler* entre las probabilidades condicionadas $p_{j|i}$ y $q_{j|i}$, puede minimizarse la divergencia de *Kullback-Leibler* entre la distribución P en el espacio de alta dimensión y Q en el espacio de baja dimensión, quedando entonces la función de coste de t-SNE como

$$J = \mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

donde de nuevo se asigna el valor cero a p_{ii} y a q_{ii}. Esta versión de SNE se le llama simétrica porque $p_{ij} = p_{ji}$ y $q_{ij} = q_{ji}$ $\forall i, j$. En ella la similitud entre pares de baja dimensión (q_{ij}) se calcula mediante la fórmula

$$q_{ij} = \frac{e^{-\|v_i - v_j\|^2}}{\sum_{k \neq l} e^{-\|v_k - v_l\|^2}} \tag{69}$$

Parece natural que la forma de calcular la similitud en alta dimensión (p_{ij}) sea

$$p_{ij} = \frac{e^{-\|x_i - x_j\|^2/2\sigma^2}}{\sum_{k \neq l} e^{-\|x_k - x_l\|^2/2\sigma^2}}$$

pero esta formulación no es correcta debido a que causa problemas cuando x_i es un *outlier* (es decir $\|x_i - x_j\|^2$ son muy grandes para $x_i, \forall j$). Para evitar estos problemas, se define como la simetría de la probabilidad condicionada, es decir: $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$ con n el número de puntos en el espacio inicial. La ventaja principal de esta versión simétrica de SNE es la forma simplificada que tiene el gradiente, que se calcula mucho mas rápido:

$$\frac{\partial J(v_1, \ldots, v_n)}{\partial v_i} = 4 \sum_j (p_{ij} - q_{ij})(v_i - v_j) \tag{70}$$

En el artículo [30] se realizan experimentos en los que la versión simétrica de SNE produce mapas tan buenos como los producidos por SNE clásico, incluso a veces los mejoraba.

- Problema de hacinamiento:

  En este problema lo que ocurre es que el área del mapa disponible para colocar las representaciones de los puntos originales no es suficientemente grande en comparación con el área disponible en el espacio inicial. Como consecuencia, los puntos que se encuentren a una distancia moderada se deberán de encontrar muy lejos en la representación bidimensional o tridimensional y como no es posible, se encuentran más cerca y muchas veces no es posible preservar las distancias. En SNE, el *spring* que hay entre el punto v_i a cada uno de los puntos distantes ejerce una fuerza atractiva muy pequeña. A pesar de que dicha fuerza sea muy pequeña, dado que hay un gran número de fuerzas similares empujando los puntos al centro del mapa se acumulan y evitan que se formen huecos entre los *clusters* naturales. Según [6], agregar una ligera repulsión (impulso) puede solucionar este problema.

- Distribución t-Student en la representación de los puntos

  En el espacio de altas dimensiones, se convierten distancias en probabilidades mediante la distribución gaussiana. Se podría seleccionar la misma para el espacio de baja dimensión, pero una de las características de esta distribución es que tiene una cola "corta" y eso hace que se produzca el problema del hacinamiento (esta idea se muestra en la Figura 8). Este problema se puede intentar solucionar seleccionando una distribución con una cola más "larga", como es la distribución t-Student con un grado de libertad (distribución de Cauchy). En el espacio de baja dimensión, podemos utilizar una distribución de probabilidad que tenga unas colas mas largas para así aliviar el problema del hacinamiento. Al utilizar dicha distribución se obtiene q_{ij} definido como sigue

$$q_{ij} = \frac{(1 + \|v_i - v_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|v_k - v_l\|^2)^{-1}} \tag{71}$$

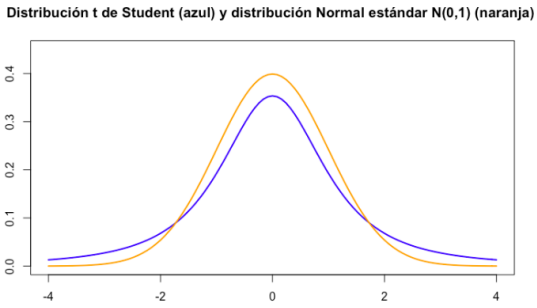


Figura 8: Comparación de las funciones de densidad de la distribución normal N(0,1) y la distribución t-Student, fuente: [23].

Se emplea esta distribución porque tiene la propiedad de que $(1 + \|v_i - v_j\|^2)^{-1}$ se aproxima a $\|v_i - v_j\|$ para grandes distancias en el mapa de baja dimensión, lo que hace que las representaciones de probabilidad conjunta se muestran invariantes (prácticamente) a variaciones en la escala del mapa para puntos que se encuentran alejados. Una justificación teórica para la selección de la distribución es que se encuentra estrechamente relacionada con la distribución gaussiana, ya que la t-Student es una suma infinita de gaussianas. Una justificación computacional es que para evaluar la densidad de un punto bajo la distribución t-Student es mucho más rápido que evaluarlo en una gaussiana, dado que no involucran exponenciales. Por otro lado, el gradiente de la divergencia de *Kullback-Leibler* entre P y la probabilidad conjunta de la t-Student Q viene dado por:

$$\frac{\partial J(v_1, \ldots, v_n)}{\partial v_i} = 4 \sum_j (p_{ij} - q_{ij})(v_i - v_j)(1 + \|v_i - v_j\|^2)^{-1} \tag{72}$$

El pseudocódigo de t-SNE encontrado en [11] se muestra en el Algoritmo 5.

Algoritmo 5 Algoritmo simplificado de t-SNE

Entrada: D, $Perp$, número de iteraciones T y ratio de aprendizaje α

1: Se calculan las similitudes $p_{i|j}$ con $Perp$ y la ecuación (66)
2: Se asigna $p_{ij} \leftarrow \frac{p_{j|i} + p_{i|j}}{2n}$
3: Se inicializa la solución $V = \{v_1, \ldots, v_n\}$ como una muestra n-dimensional de $\mathcal{N}(0, 10^{-4}I)$
4: **para** $t = 1$ hasta T **hacer**
5: Se calculan las afinidades de baja dimensión q_{ij} mediante la ecuación (71).
6: Se calcula el gradiente $\frac{\partial J(v_1, \ldots, v_n)}{\partial V}$ con la ecuación (72)
7: $V = V - \alpha \frac{\partial J(v_1, \ldots, v_n)}{\partial V}$
8: **fin para**
9: **devolver** Representación en baja dimensión $V = \{v_1, v_2, \ldots, v_n\}$

t-SNE es capaz de captar a grandes rasgos la estructura local de los puntos en la alta dimensión, mientras representa la estructura global de los datos, así como la presencia de *clusters* en las distintas escalas. Pone mayor énfasis en (1) modelar puntos no similares a través de la media de las grandes distancias, y (2) modelar puntos similares mediante la media de las distancias pequeñas. Como resultado de dichas características de t-SNE, la función de coste es mucho más sencilla de optimizar que la función de coste de SNE. En particular, t-SNE introduce fuerzas de largo alcance en el mapa de baja dimensión que pueden unir dos puntos similares que se separan al principio de la optimización. SNE no dispone de estas fuerzas de largo alcance y como resultado necesita utilizar varias simulaciones para obtener resultados razonables. En su lugar, las fuerzas de largo alcance facilitan la identificación de óptimos locales sin la necesidad de realizar simulaciones.

En cuanto a la **complejidad temporal** de t-SNE, es cuadrática respecto al número de elementos del espacio inicial de alta dimensión: $\mathcal{O}(n^2)$ como se explica en [14].

3. Caso de estudio

El ejercicio de resistencia induce fatiga periférica y deteriora la eficiencia mecánica muscular alterando sus respuestas metabólicas. La actuación de un ciclista profesional requiere de un exce-

lente rendimiento de resistencia y durante la competición los ciclistas deben ser capaces de soportar niveles muy altos de fatiga y esfuerzo relacionados con: el metabolismo, acidosis, temperatura o hipoxia. El factor de potencia umbral (FTP) se ha descrito como la capacidad máxima de los ciclistas para desarrollar una potencia constante en un tiempo fijo (de 20 a 60 minutos). El test FTP se ha demostrado que es muy fiable y fácil de aplicar, y que puede determinar el nivel de rendimiento de los ciclistas y extrapolarlo para estudiar posibles correlaciones mediante tests de laboratorios de medidas cardiovasculares y respiratorias. Utiliza en sus mediciones los aparatos que miden la potencia instalados en las bicicletas.

Por otro lado, el test de Bosco (BT) es un enfoque específico para medir indirectamente las funciones neuromusculares y el rendimiento de potencia. Muchos estudios han observado que el BT es una herramienta muy eficiente para demostrar que la fatiga central y periférica perjudican directamente la capacidad neuromuscular del músculo para contraerse. Fue creado por el italiano Carmelo Bosco [4], y consiste en una serie de saltos verticales en los que se pretenden evaluar las características morfohistológicas (fibras musculares), funcionales (altura y potencia mecánica) y neuromusculares (energía elástica, reflejo miotático y resistencia a la fatiga) de los músculos extensores de los miembros inferiores en base a las alturas obtenidas en los saltos. Está compuesto de los saltos: *squat jump*, *counter movement jump*, *squat jump con carga*, *Abalakov*, *drop jump* y saltos continuados durante varios segundos (pueden ser 15, 20 o 30 segundos normalmente).

3.1. Presentación de los datos

Los participantes en el estudio fueron 22 ciclistas masculinos, de ellos 10 eran profesionales y 12 amateurs. Se realizaron distintas mediciones durante tres días consecutivos. El primer día se tomaron medidas físicas basales, se llevo a cabo el test de Bosco y se realizaron dos pruebas de ciclismo como estudio previo del rendimiento de los individuos. En la primera prueba los deportistas debían pedalear a máxima potencia durante 30 segundos y en la segunda debían hacerlo durante 5 minutos (esta segunda prueba se llama *5' all out*). El segundo día los participantes desempeñaron el FTP y se les tomaron medidas de recuperación. El último día se registraron medidas fisiológicas de recuperación y se anotaron marcadores biológicos de sangre, así como medidas físicas de recuperación.

En el experimento que se analiza en este trabajo ([33]) se seleccionaron los siguientes saltos para llevar a cabo el test de Bosco:

- *Squat jump* (SJ) o salto en sentadilla. Consiste en realizar un salto vertical. El individuo debe partir de una posición de media sentadilla (rodillas flexionadas a 90^{o}), con el tronco erguido y las manos en la cintura.
- *Counter movement jump* (CMJ) o salto a contramovimiento. Se realiza como sigue: el sujeto parte desde una posición erguida y con las manos en las caderas. A continuación, se realiza un salto hacia arriba por medio de una flexión seguida lo más rápidamente de una extensión de piernas.
- *Continuous counter movement jump* o salto a contramovimiento continuo. Consiste en lo mismo que el anterior pero repitiéndolo durante los segundos establecidos; en el caso del experimento fueron 30 segundos (CMJ30).

En cuanto al test FTP, consistió en un trayecto con pendiente pronunciada con una duración de 20 minutos aproximadamente (+/- 1 minuto). Al llegar a la cima de la subida, dos investigadores paraban inmediatamente a los ciclistas y repetían las mediciones fisiológicas. Durante las mediciones, los ciclistas se encontraban descansando sentados y no comieron ni bebieron nada. Después de 20 minutos de recuperación, las medidas volvieron a los rangos de las tomadas con anterioridad y se repitió el test de Bosco con los mismos saltos que se realizaron antes. Finalmente, 24 horas después del test FTP, los participantes fueron citados de nuevo en el laboratorio para medir variables mecánicas (SJ, CMJ y CMJ30) y fisiológicas (albúmina y creatina quinasa).

3.1.1. Representación de los datos

Debido a que los datos del experimento se encuentran en $\mathbb{R}^{66}$, no es posible visualizarlos y por ello se ha recurrido a la técnica de t-SNE explicada en la Sección 2.3.

A continuación, se muestra el resultado de aplicar t-SNE a los datos del experimento [33].

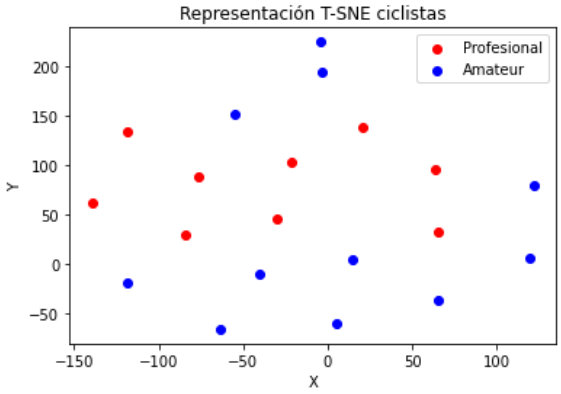


Figura 9: Resultado de aplicar el algoritmo t-SNE a los datos del experimento [33] que se van a analizar durante todo el trabajo.

Como se observa en la gráfica, el espacio resultante de aplicar t-SNE a los datos obtenidos en el experimentos no es linealmente separable. Otra observación que puede realizarse es la aparente ausencia de *outliers*.

3.1.2. Explicación de las variables

A continuación, se detallan las variables extraídas en el experimento en el que se basa el trabajo ([33]).

- **Variables medidas el primer día:** Edad (en años), altura (en centímetros cm) y peso (en kilogramos kg). Las siguientes variables fueron tomadas estando los individuos tanto sentados como de pie (ambas en reposo): ritmo cardiaco (HR), presión sistólica (SBP) y diastólica (DBP) y saturación de oxígeno en sangre (SaO2). Las siguientes medidas fueron tomadas en reposo tan solo una vez: albúmina[2] (Albu), concentración de lactato[3] (CL) e índice de elasticidad. Se realizó el test de Bosco (explicado anteriormente) en reposo midiendo los siguientes saltos: salto a contramovimiento (CMJ), a contramovimiento continuo durante 30 segundos (CMJ $30''$) y el salto en sentadilla (SJ).

  Además, al desempeñar la prueba en la que los individuos debían pedalear a máxima potencia durante $30''$ se obtuvieron las variables: potencia media del individuo (*Wingate Mean*) y pico de potencia (*Wingate Peak*). Finalmente, cuando realizaron la prueba de $5'$ *all out* se midió la potencia media que tuvieron ($5'$ *all out*).

- **Variables medidas el segundo día durante el test FTP:** Potencia media durante los 20 minutos de prueba (*power mean*), potencia media relativa (*relative power*) y la proporción de tiempo que los individuos estuvieron pedaleando erguidos durante la prueba (*standing position climbing*).

- **Variables medidas el segundo día después de la prueba y 24h después de la prueba:** Las siguientes variables fueron medidas en el momento en que terminaron el test FTP, y al cabo de 3, 5, 10 y 20 minutos: la saturación de oxígeno en sangre (SaO2), nivel de creatina quinasa[4] (CK), ritmo cardíaco (HR), nivel de glucosa en sangre (GLUC) medida en

[2] **Albúmina (Albu):** Es la principal proteína de la sangre y una de las más abundantes en el ser humano. Se sintetiza en el hígado. La concentración normal en humanos ronda el 3,5-5 g/dL. Funciona como un transportador de la sangre y lo contiene el plasma. Más información en https://es.wikipedia.org/wiki/Alb%C3%BAmina

[3] **Concentración de lactato (CL):** Según [9]: "El lactato es un compuesto orgánico formado por carbono, hidrogeno y oxigeno que se encuentra en los músculos, sangre y algunos órganos. Forma parte de diferentes procesos para la producción de energía." La concentración de lactato en sangre de un adulto normal es de 1 o 2 mmol/l en reposo, pero puede aumentar hasta 20 mmol/l durante un esfuerzo intenso.

[4] **Creatina quinasa o creatina cinasa (CK):** Es un tipo de proteína conocida como enzima. Se encuentra principalmente en músculos esqueléticos y el corazón y en menor cantidad en el cerebro. Interviene en la producción de energía y cuando se requiere producir gran cantidad de energía interviene con mayor contundencia. Suele

mg/dL, presión sistólica (SBP) y diastólica (DBP) y la temperatura en grados centígrados (Temp). También se repitió el test de Bosco con los mismos saltos: CMJ, SJ y CMJ 30″. El test de Bosco se realizó al terminar la prueba y 24 horas después. La albúmina (Albu) también se midió 24 horas después de la prueba.

3.2. Análisis descriptivo

A continuación se muestran algunas características del *dataset*, con el fin de comprender mejor los datos de estudio. En primer lugar, se han determinado las matrices de correlación con las variables más destacadas, bien porque se sospecha que guardan alguna relación, bien porque han resultado interesantes a lo largo del estudio. Un coeficiente de correlación se considera relevante a partir de 0,8 como explican en [32].

Las correlaciones más relevantes encontradas en la matriz, dentro del conjunto de las variables que mayor precisión obtuvieron en los algoritmos de clasificación ($\geq 0,8$) son: *Post SBP* 0′ y *Post SBP* 3′, *Post SBP* 0′ y *Post DBP* 0′. Además entre las variables medidas durante las pruebas realizadas a los ciclistas se observa una alta correlación entre las variables *Relative Power* y *Power Mean*, esperable debido a que la potencia relativa no es más que la potencia media partido por el peso del individuo. Entonces la variable *Relative Power* es el resultado de la operación de otras dos variables, por tanto, se podría haber prescindido de esta variable para el estudio. También se observa una correlación alta entre *Wingate Mean* y *Wingate Peak*. Pero la más sorprendente ha sido la encontrada entre *Post 24h CK* y las variables *Power Mean* y *Relative Power*.

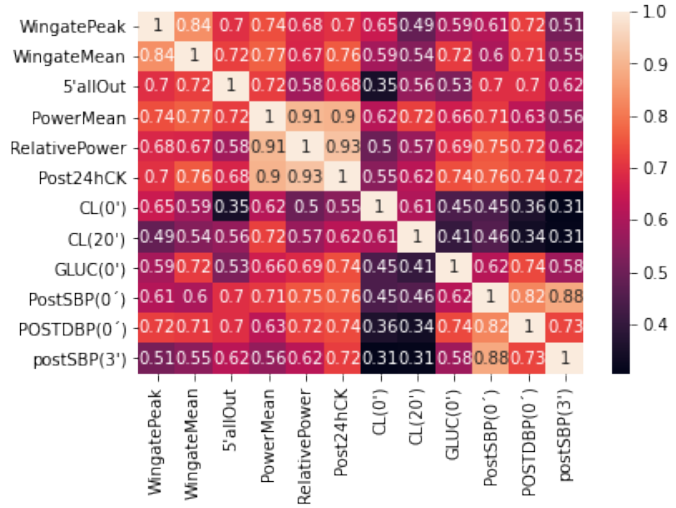


Figura 10: Matriz de correlación de las variables que mayor precisión obtuvieron al aplicar regresión logística y SVM.

Tabla 1: Medias de las variables más interesantes según el grupo al que pertenecen los individuos.

	Post 24h CK	*CL 0′*	*CL 20′*	*Power Mean*
General	357,86	11,52	6,56	312,82
Profesionales	442,0	14,96	8,85	344,5
Amateurs	332,0	8,5	5,1	292

medirse en *mg/dl* y el rango normal de referencia en mujeres es de 0,5-1,0 *mg/dl* y en hombre es de 0,7-1,2 *mg/dl*. Suele medirse para detectar lesiones y enfermedades musculares. Más detalles en: https://es.wikipedia.org/wiki/Creatina_quinasa y https://medlineplus.gov/spanish/pruebas-de-laboratorio/creatina-cinasa/

3.3. Aplicación de los distintos algoritmos

3.3.1. Algoritmos de aprendizaje supervisado

Para poder evaluar la bondad de ajuste de los modelos de aprendizaje supervisado se utiliza la estrategia siguiente explicada en [2, pgs. 334-337]: dividir el *dataset* en dos subconjuntos: de entrenamiento y de pruebas. Esta estrategia es muy común y la utilizaremos tanto en los algoritmos de clasificación como en los de regresión. En función de cómo se seleccionan el conjunto de entrenamiento y el conjunto de pruebas se encuentran las siguientes variantes:

- *Holdout* : El *dataset* se divide aleatoriamente en dos subconjuntos disjuntos, que corresponden al conjunto de entrenamiento y el conjunto de pruebas respectivamente. Normalmente, la mayoría (aproximadamente dos tercios o tres cuartos del total de los datos) forman el conjunto de entrenamiento, y el resto forman el de pruebas. Uno de los inconvenientes de esta estrategia es que cuando algunas clases se encuentran representadas muchas más veces en el *dataset* que otras, deja de ser robusta para la validación. Este método se ha seleccionado para validar los algoritmos de regresión.

- *Cross-Validation* : El *dataset* se divide en m subconjuntos disjuntos llamados *folds* del mismo tamaño $\frac{n}{m}$ (una elección común de m es 10). Se escoge uno de los subconjuntos y se utiliza como conjunto de pruebas y el resto ($m-1$ conjuntos) se unen y se utilizan como conjunto de entrenamiento. Este procedimiento se repite seleccionando cada uno de los m subconjuntos como conjunto de pruebas y se calcula la media de la precisión en cada elección de los conjuntos. Existen algunos casos particulares interesantes de *cross-validation*, como son:

  - *Leave-one-out*: Ocurre cuando $m = n$ (el número de subconjuntos en el que se divide el *dataset* es el número de elementos). En este caso, $n - 1$ datos son utilizados para el conjunto de entrenamiento y un dato para el conjunto de pruebas. Es particularmente útil esta estrategia para el caso en que el *dataset* es pequeño, en cambio, es muy costoso cuando el número de elementos del *dataset* es grande. Este método será el empleado en nuestro caso para validar los algoritmos de clasificación, pues $|dataset| = 22$ es pequeño.
  - *Stratified cross-validation:* Utiliza una representación proporcional de las clases en los distintos *folds* y normalmente tiene resultados menos pesimistas.

- Bootstrap: En este método se toman muestras uniformes con reemplazamiento del *dataset* para crear el conjunto de entrenamiento. Por lo tanto, es posible que haya datos repetidos. Estas muestras se extraen n veces; luego, el tamaño del conjunto de entrenamiento es el mismo que el del conjunto de pruebas.

Algoritmos de clasificación

Una vez la división en conjunto de entrenamiento y de pruebas se haya realizado, se debe seleccionar una forma de cuantificar la bondad de ajuste del modelo. Existen varios criterios y algunos solamente son válidos para los algoritmos de clasificación que retornan puntuaciones numéricas y no la clase a la que pertenece cierto dato (como es el caso de los algoritmos explicados en la Sección 2.2.1). Dentro de las formas de cuantificar la bondad de ajuste de los modelos de clasificación que retornan la clase a la que pertenece el dato, hay dos que son los más utilizados:

- *Cost-sensitive accuracy*: el objetivo es determinar la precisión del método pero con ponderaciones. Se penaliza más clasificar erróneamente algunas clases que otras. Es especialmente interesante en los casos en que las clases se encuentran muy poco equilibradas, pero no es el caso de este trabajo ya que |profesionales|=10 y |amateurs|=12. Por lo tanto, en este trabajo se optará por la medida explicada a continuación.

- Precisión (en inglés, *accuracy*): se trata de la relación entre los datos que se clasifican correctamente y los datos totales del *dataset*.

En las siguientes secciones se van a aplicar los algoritmos explicados en la Sección 2.2.1, y se mostrarán los resultados obtenidos. También se comparará la selección de los distintos parámetros de cada algoritmo para determinar cuál funciona mejor a través de la precisión del modelo y con la estrategia *leave-one-out*.

Algoritmo de máquinas de soporte vectorial

Para poner en práctica lo explicado en la Sección 2.2.1, apartado de SVM, se ha utilizado la librería `Scikit-Learn`. Se trata de una librería de código abierto para Python que fue desarrollada por David Cournapeaus en 2007. El *dataset* del que disponemos tiene pocos datos (tan solo 22

muestras), por tanto, es recomendable utilizar el método de SVM sin *kernels* o con un *kernel* lineal. Se ha escogido la aplicación de SVM con *kernel* lineal, de la clase `sklearn.svm.SVC`[5] y para la validación de los resultados se ha empleado la estrategia *leave-one-out*. Se ha aplicado el algoritmo al *dataset* con todas las variables disponibles con el fin de clasificar a los ciclistas en profesionales y amateurs. Para ello se han asignado valores a C desde 0,01 hasta 100 con paso 1,01 siendo $C = \frac{1}{\lambda}$. Se ha obtenido máxima precisión tanto en el conjunto de entrenamiento como en el conjunto del test.

En vista de los resultado obtenidos, la precisión del modelo es aparentemente perfecta para todos los casos, por tanto se ha analizado la precisión del modelo con cada una de las variables con el fin de determinar aquellas que mejor clasifican. Una vez obtenidas las precisiones, se han buscado variables que hagan que el modelo de SVM con *kernel* lineal tenga una precisión por encima del 0,8, es decir, que clasifica bien al menos el 80 % de los datos. Se han obtenido las variables predictoras incluidas en la Tabla 2.

Tabla 2: Variables predictoras que mejor clasifican con SVM y *kernel* lineal a los individuos en profesionales y amateurs, con sus respectivas precisiones en el conjunto de pruebas.

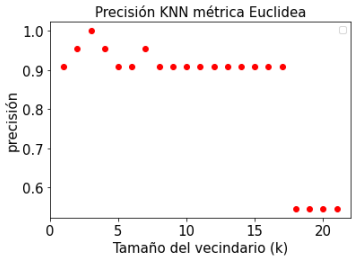
Variable	Precisión
Wingate Peak	1,0
Wingate Mean	0,82
5′ all out	0,86
Power Mean	0,91

Variable	Precisión
Relative Power	1,0
Post 24h CK	0,91
CL 0′	0,86
CL 20′	0,82

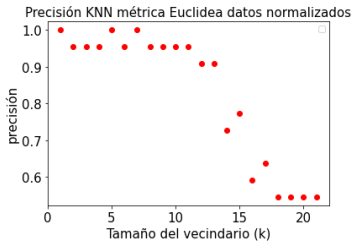
Variable	Precisión
Post SBP 0′	0,91
Post DBP 0′	0,95
post SBP 3′	0,86

Algoritmo *k-Nearest Neighbor* (KNN)

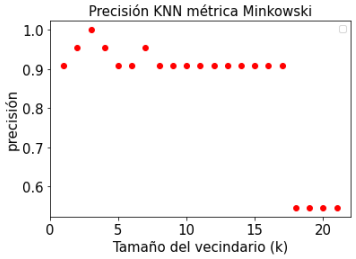
Para aplicar este algoritmo a los datos se ha utilizado la clase `KNeighborsClassifier`[6] de la librería `Scikit-Learn`. Se han realizado pruebas con tres métricas distintas (métrica Euclídea, métrica de Mahalanobis y métrica de Minkwoski) y se ha medido la precisión en función de los tamaños de los vecindarios de cada una, mediante la estrategia *leave-one-out*.



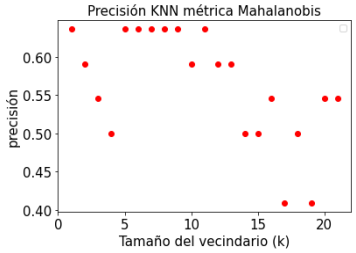
(a) Precisión de KNN con la métrica Euclídea y los datos sin normalizar según el tamaño del vecindario



(b) Precisión de KNN con la métrica Euclídea y con los datos normalizados, según el tamaño del vecindario



(a) Evaluación de la precisión del algoritmo KNN con la métrica de Minkowski, en función del tamaño del vecindario



(b) Evaluación de la precisión del algoritmo KNN con la métrica de Mahalanobis en función del tamaño del vecindario

En vista de los resultados representados por las gráficas se obtienen las siguientes conclusiones:

- En cuanto a la métrica euclídea con los datos sin normalizar, se observa una precisión menor en general frente a la de los datos normalizados, aunque para vecindarios grandes parece

[5]Documentación accesible en https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[6]Documentación disponible en https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

mejor opción. Como resultado, el vecindario óptimo para la métrica euclídea con los datos normalizados es $k = 5$ (aunque se obtiene una precisión máxima también con el tamaño de vecindario $k = 1$, pero puede ser inestable, demasiado local). Para el caso de la métrica euclídea con los datos sin normalizar, el vecindario óptimo parece $k = 3$ (es el único valor de k con el que se logra precisión máxima).

- Respecto a la métrica de Minkowski, se observa que el valor óptimo del vecindario es $k = 3$ (por la misma razón que en la métrica euclídea se evita tomar $k = 1$).
- Finalmente, para la métrica de Mahalanobis el valor óptimo del vecindario es $k = 5$, aunque en ningún momento se alcanza la precisión máxima (ocurre en el resto de métricas); su valor máximo es 0,636.

Método de regresión logística

Se ha aplicado el método de regresión logística con la clase `LogisticRegression`[7] de la librería `Scikit-Learn`. Se ha creado un modelo con las siguientes características: se ha utilizado, como se mencionó en la Sección 2.2.1, la penalización *ridge*, donde el término regularizador es $\lambda \sum_{i=0}^{d} \frac{w_i^2}{2}$ y el parámetro regulador λ se controla mediante $C = \frac{1}{\lambda}$. La convergencia del gradiente descendente se evalúa con $\epsilon = 0,0004$. Se ha medido la precisión del modelo con la técnica de *leave-one-out* en función del parámetro C.

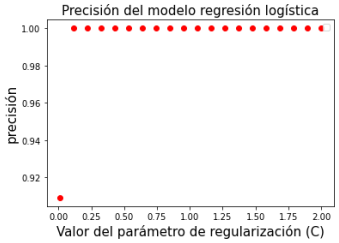


Figura 13: Evaluación de la precisión del algoritmo regresión logística según los valores de C.

En vista de los resultados de la precisión del método, se sospecha que hay variables que determinan la clase de los individuos, ya que sin apenas regularización el modelo tiene una precisión máxima. Por lo tanto, se ha procedido a aplicar regresión logística con cada una de las variables con el objetivo de identificar aquellas que mejor clasifiquen a los ciclistas del experimento en profesionales y amateurs. Se muestran a continuación, las variables que mejor clasifican (aquellas con una precisión mayor que 0,8).

Tabla 3: Variables predictoras que mejor clasifican con el método de regresión logística a los individuos en profesionales y amateurs con sus respectivas precisiones en el conjunto de pruebas.

Variable	Precisión
Wingate Peak	8,82
Wingate Mean	0,91
$5'$ *all out*	0,91
Power mean	1,0

Variable	Precisión
Relative Power	0,91
Post 24h CK	1
CL $0'$	0,86
CL $20'$	0,82

Variable	Precisión
GLUC	0,82
Post DBP $0'$	0,86
Post SBP $3'$	0,86

Cabe destacar que todas las variables que han resultado tener precisión mayor o igual que 0,8 son las mismas que la obtuvieron al aplicar SVM salvo *GLUC* $0'$ que no aparecía y en su lugar se encontraba *Post SBP* $0'$.

Algoritmos de regresión

En esta sección se van a exponer los resultados obtenidos con los distintos algoritmos de regresión explicados en la Sección 2.2.2, así como las principales medidas de error de los modelos entrenados. Para ello, se denota por $\hat{y}_i$ a la predicción de la variable de respuesta de x_i por el modelo, e y_i es la verdadera variable de respuesta de x_i. Para medir la bondad de ajuste de los distintos modelos de regresión se seguirá la estrategia *holdout* seleccionando el 30 % de las muestras para el conjunto de pruebas y el 70 % restante para entrenar los modelos. Para cuantificar la bondad de ajuste se utilizarán las siguientes medidas de error válidas para regresión:

[7] Documentación disponible en https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

- Error cuadrático medio (RMSE): Es la raíz cuadrada de la distancia cuadrada promedio entre la variable de respuesta real y la predicha.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Se trata de una medida absoluta de ajuste. Puede interpretarse como la desviación estándar de la varianza no explicada y se encuentra en las mismas unidades que la variable respuesta. El ajuste será mejor cuanto menor sea el valor de RMSE.

- Error absoluto medio (MAE): Se trata de la media de la diferencia en valor absoluto del valor de la variable de respuesta real y el valor de la variable respuesta estimado.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Una alternativa a MAE es el error porcentual absoluto medio (MAPE) que otorga el carácter relativo.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Puede entenderse como una versión ponderada de MAE, donde el peso de la muestra es inversamente proporcional al objetivo.

- Error cuadrático medio (MSE): Se trata de la métrica más sencilla para evaluar el modelo, pero también la menos útil, porque mide el error cuadrado promedio de las predicciones. Cuanto mayor sea el valor de MSE peor será el modelo y mejor cuanto menor sea.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Su principal inconveniente es que si la predicción es muy mala, al elevarlo al cuadrado el error empeora aún más y puede que se sobreestime la maldad del modelo. Podría ocurrir incluso con un modelo que estimara muy bien, por tanto, cuesta juzgar el funcionamiento de un modelo en base a MSE.

En cambio, si en todas las estimaciones se tiene un error de entre 0 y 1, ocurre lo contrario; se subestima la maldad de modelo. En cambio, el error de porcentaje cuadrático medio (MSPE) puede ser más interesante (otorga el caracter relativo que falta en MSE).

$$MSPE = \frac{100}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2$$

Para cada elemento del *dataset*, se divide el error absoluto por su variable de respuesta, obteniendo el error relativo. Se puede entender el MSPE como una versión ponderada de MSE.

- Coeficiente de correlación R-cuadrado (R^2): Indica la bondad de ajuste del modelo y sus valores se encuentran entre 0 y 1. Si R^2 tiene un valor próximo a 0, entonces el modelo no tiene buenas predicciones, no será fiable y si su valor se encuentra próximo a 1, la predicción es perfecta.

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

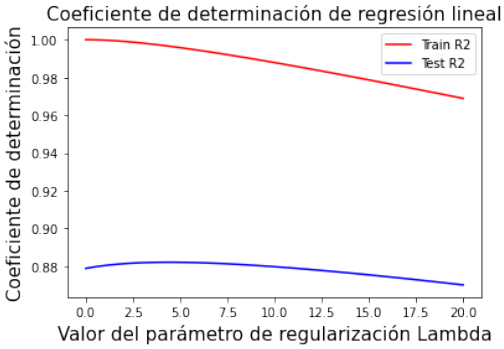
El inconveniente que tiene R^2 es que solo aumenta su valor conforme se agregan predictores al modelo de regresión (no penaliza el echo de tener variables no significativas). Una alternativa a R^2 es R^2 ajustado ($\overline{R}^2$), pero se debe utilizar en modelos cuyas variables de respuesta son dos o mas, ya que en nuestro caso, con solo 1 variable de respuesta queda muy parecido al R^2 ya que se calcula según la siguiente expresión donde n es el número de datos y k el número de variables de respuesta

$$\overline{R}^2 = 1 - \frac{n-1}{n-k-1}(1-R^2) = 1 - \frac{n-1}{n-2}(1-R^2).$$

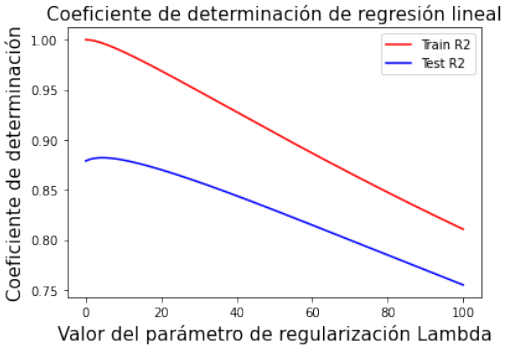
Algoritmo de regresión lineal

Para utilizar el algoritmo de regresión lineal con regularización se utiliza de la librería `Scikit-Learn` la clase `sklearn.linear_model.Ridge`[8]. Se aplica la regresión lineal para predecir el valor de la variable *Mean Power* (potencia media del ciclista en la prueba FTP), con el resto de variables del *dataset*. Se utiliza el modelo de regresión lineal con la regularización *ridge* explicada en la Sección 2.2.2. Para determinar el mejor valor del parámetro de regularización λ, se calculan algunas de las medidas de error explicadas anteriormente en función de los valores de λ (se han tomado valores discretos desde 0 hasta 200), tanto del conjunto de entrenamiento como del conjunto de pruebas y finalmente se seleccionará el parámetro que tenga mejores valores en las medidas de bondad de ajuste del conjunto de pruebas.

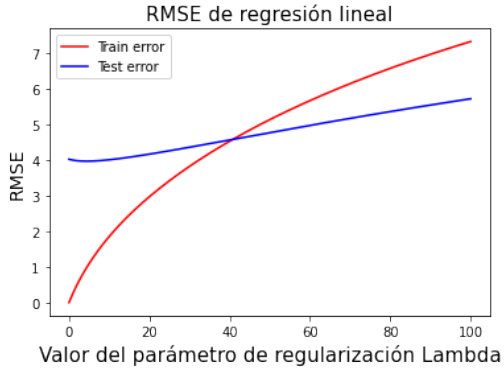
Como se muestra en la Figura 14a, el coeficiente R^2 del conjunto de pruebas se incrementa conforme aumenta λ, pero a partir de cierto valor vuelve a decaer. El valor máximo es $R^2 = 0{,}882$ y se obtiene cuando $\lambda = 4{,}44$. En la Figura 14b se observa como tanto el coeficiente de determinación del conjunto de pruebas como el del conjunto de entrenamiento decaen para valores grandes de λ (esto puede ser indicador del *underfitting*).



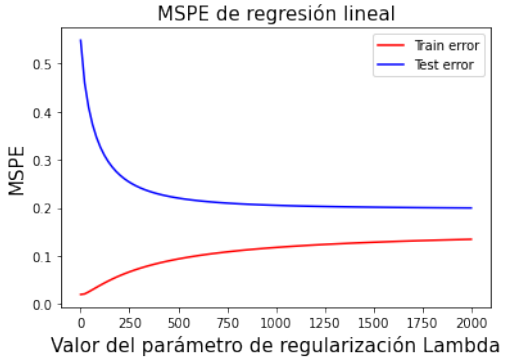
(a) Evaluación de R^2 del modelo para valores pequeños de λ.



(b) Evaluación de R^2 del modelo para valores grandes de λ.



(c) Evaluación de RMSE en función del parámetro de regularización λ.



(d) Evaluación de MSPE en función del parámetro de regularización λ.

Las gráficas de los errores MAPE, MAE, RMSE y MSE tienen un comportamiento muy parecido. Debido a falta de espacio se ha decidido realizar el análisis e incluir solo la de RMSE[9] (Figura 14c). En los cuatro casos, para valores pequeños de λ se obtiene un error pequeño en el conjunto de entrenamiento, mientras que el error del conjunto de pruebas es bastante superior. Este hecho manifiesta que puede estar ocurriendo *overfitting*, pues para valores pequeños de λ el modelo se ajusta bien a los datos de entrenamiento (bajo valor de RMSE) y, en cambio, se ajusta peor a los datos de pruebas (alto RMSE). A medida que el valor de λ aumenta, se observa un rápido crecimiento del valor de RMSE del conjunto de entrenamiento, mientras que el valor del RMSE del conjunto de pruebas disminuye al principio y luego crece a menor velocidad que el del conjunto de entrenamiento, lo que quiere decir que se corrige el posible *overfitting* detectado hasta llegar a

[8]Documentación disponible en https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

[9]Pueden encontrarse el resto en: https://github.com/ClaudiaCarpintero/TFG.git

valores de λ previos al que coinciden los valores de RMSE de ambos conjuntos. El punto en el que los errores son iguales para el conjunto de entrenamiento y el conjunto de pruebas varía en función del error que se trate: en el caso de RMSE ocurre en $\lambda \approx 42{,}42$ ($RMSE \approx 4{,}609$), en MAPE ocurre cuando $\lambda \approx 18{,}18$ ($MAPE \approx 0{,}389$), en MAE ocurre cuando $\lambda \approx 18{,}18$ ($MAE \approx 1{,}167$) y en MSE sucede en $\lambda \approx 42{,}42$ ($MSE \approx 21{,}244$). A partir de cierto punto (previo al punto de intersección), tanto el RMSE del conjunto de entrenamiento como el del conjunto de pruebas comienzan a crecer y esto se mantiene para los valores mayores de λ. Se debe a que a partir de cierto valor de λ el término regularizador es muy fuerte y comienza a producirse *underfitting*. Es interesante que el error en el conjunto de test llega a ser menor que el error en el conjunto de entrenamiento.

Por otro lado, en la Figura 14d, puede observarse como también el conjunto de entrenamiento comienza con un valor bajo de MSPE cuando λ es pequeño, mientras que el valor de MSPE del conjunto de pruebas es mayor (de nuevo al no haber apenas regularización al principio se produce *overfitting*). A medida que aumenta λ se incrementa el valor del MSPE del conjunto de entrenamiento y decrece el del conjunto de pruebas. Finalmente, a partir de cierto valor de λ se estabilizan ambos valores del MSPE pero aparentemente no llegan a igualarse.

Tabla 4: Se muestran los mejores valores de la medida de error (según la columna) junto al valor de λ para el que se alcanzan.

	RMSE	MAE	MAPE	MSE	MSPE	R^2	$\overline{R}^2$
Valor	3,96	1,13	0,38	15,72	0,19	0,88	0,88
λ	4,04	10,10	10,10	4,04	200	4,44	4,44

Nota: En la tabla 4 aparece que el menor valor de MSPE es 0,19, pero el MSPE del conjunto de pruebas decrece conforme se aumenta λ, se ha seleccionado el punto en que $\lambda = 200$ porque a partir de ese punto la disminución de MSPE es muy pequeña.

Algoritmo de regresión polinómica

En esta sección se estimará de nuevo el valor de la potencia de los ciclistas (en concreto, la variable de respuesta potencia media *Power mean*) utilizando regresión polinomial. Esta vez se han seleccionado las variables que mejor clasificaron en los algoritmos de SVM y regresión logística. Dado que algunas de las variables se encuentran relacionadas con la potencia, solo se seleccionarán las seis siguientes variables predictoras: *Post 24h CK*, *CL* $0'$, *GLUC* $0'$, *Post SBP* $0'$, *Post DBP* $0'$ y *Post SBP* $3'$. Con las anteriores variables se ha aplicado regresión polinomial para un polinomio de grado dos y tres (notar que el caso de grado uno es la regresión lineal del anterior apartado). Además se han estudiado las mismas medidas de error que en regresión lineal con el objetivo de poder comparar los tres modelos y determinar la bondad de ajuste. Para ello se ha empleado de nuevo el método *holdout*, seleccionando el 30 % de las muestras para el conjunto de pruebas y el 70 % restante para el conjunto de entrenamiento (mismos datos en ambos conjuntos que en los tomados para regresión lineal). Para la obtención de los resultados se ha vuelto a utilizar la clase `sklearn.linearmodel.Ridge` de la librería `Scikit-Learn` para la aplicación de regresión lineal una vez las variables se encuentran transformadas.

Tabla 5: Se muestran los mejores valores de la medida de error (según la columna) junto al valor de λ para el que se alcanzan.

Polinomio de grado dos	RMSE	MAE	MAPE	MSE	MSPE	R^2	$\overline{R}^2$
Valor	8,38	3,6	4×10^{-11}	70,33	0,84	0,82	0,82
λ	6,06	2,02	0	6,06	200	6,06	6,06

Tabla 6: Se muestran los mejores valores de la medida de error (según la columna) junto al valor de λ para el que se alcanzan.

Polinomio de grado tres	RMSE	MAE	MAPE	MSE	MSPE	R^2	$\overline{R}^2$
Valor	8,87	3,83	$2{,}41 \times 10^{-11}$	78,71	1,08	0,80	0,80
λ	2,02	2,02	0	2,02	200	2,02	2,02

Nota: En las tablas 5 y 6 aparece que el menor valor de MSPE es 1,08 y 0,84 respectivamente, pero lo cierto es que el MSPE del conjunto de pruebas decrece conforme se aumenta λ, se ha seleccionado el punto en que $\lambda = 200$ porque a partir de ese punto la disminución de MSPE es muy pequeña.

En cuanto al comportamiento de las distintas medidas de bondad de ajuste, no se han plasmado en la memoria[10] por cuestiones de espacio, pero a continuación se va a detallar su comportamiento:

- Las gráficas de MAPE, MAE, MSE y RMSE tienen el mismo comportamiento (como ya ocurría en regresión lineal) y entre las gráficas de regresión polinómica con polinomio de grado dos y tres se diferencian en que los errores crecen más rápidamente con la regularización al aplicar polinomios de grado dos. En cualquier caso, el comportamiento de los errores en regresión polinómica se diferencia respecto a de regresión lineal en que en este caso los errores nunca se intersecan. Además, hay que destacar la rapidez con la que aumenta el error en ambos conjuntos conforme se introduce la regularización (en comparación con el caso de regresión lineal), en especial el error en el conjunto de pruebas. Este hecho puede ser indicativo de que una regularización en este caso no es recomendable. Ambos errores crecen rápidamente pero siempre el error del conjunto del test es mayor que el del conjunto de entrenamiento. Con valores pequeños de regularización estos valores se acercan, pero cuando se pasa de cierto valor de λ vuelven a separarse.

- En cuanto a la gráfica de R^2 con polinomio de grado dos: el coeficiente del conjunto de entrenamiento disminuye conforme aumenta el peso de la regularización, en cambio, el del conjunto de pruebas aumenta para pequeños valores de C y luego vuelve a disminuir.

  En cuanto a la gráfica del coeficiente de determinación con polinomios de grado tres: ambos coeficientes disminuyen conforme se introduce regularización, pero el del conjunto de entrenamiento más rápidamente, por lo que para $\lambda \approx 90$ el del conjunto de test es mayor.

- Finalmente, la gráfica del error MSPE tiene el mismo comportamiento que la gráfica de MSPE de regresión lineal, solo que los errores nunca se acercan tanto.

3.3.2. Variables detectadas

En esta sección se presentan aquellas variables que durante la aplicación de los algoritmos de clasificación han resultado ser de mayor interés:

- **Variables relacionadas con la potencia obtenida en las pruebas:** Claramente los sujetos que se encuentren más entrenados (profesionales) deberían tener unos valores mayores de estas variables. De hecho, según [18], la variación del rendimiento específico (en nuestro caso representado por la potencia) es el marcador más relevante para diferenciar entre deportistas fatigados y recuperados (se entiende que un individuo no entrenado alcance antes o con mayor intensidad la fatiga frente a otro entrenado).

- **Post 24h CK:** Los niveles altos de creatina quinasa se pueden relacionar con el estado de entrenamiento y, si en reposo se mantienen elevados, puede ser un indicativo del síndrome de sobreentrenamiento explicado en la Sección 1. El comportamiento de los niveles de CK tras una actividad intensa de ejercicio es el siguiente: a las 6-24 horas de finalizar el entrenamiento alcanza sus niveles más altos y a las 48-72 horas vuelve a valores normales. En [21] se explica cómo los niveles de respuesta de CK tras realizar ejercicio pueden ser un indicador para diferenciar sujetos que se entrenan regularmente y sujetos que no, pues los factores más determinantes que modifican los niveles de CK son: la intensidad del ejercicio, la modalidad del ejercicio (en nuestro caso ambas son constantes para los individuos) y el estado de entrenamiento de cada persona. Los individuos que entrenan habitualmente suelen presentar una respuesta mayor (debido a su capacidad de amortiguar o revertir el incremento) frente a los que no entrenan, como avala la Tabla 1.

- **CL $0'$ y *CL* $20'$:** El lactato es un intermediario en el metabolismo de la glucosa, se produce de forma continua al realizar ejercicio y también en reposo. Alcanza sus mayores valores (valor conocido como "umbral del lactato") tras la práctica de la actividad física. El lactato produce una disminución en el rendimiento deportivo y hay estudios que lo vinculan directamente con la fatiga, que se evidencia al llegar al umbral.

  El comportamiento de los niveles de concentración de lactato en sangre es el siguiente: si el individuo se encuentra realizando ejercicio suave o moderado, los niveles de la concentración de lactato no se ven alterados, pero a medida que se intensifica el ejercicio, estos aumentan rápidamente (más detalles en [27]).

[10] Pero están disponibles en el enlace: https://github.com/ClaudiaCarpintero/TFG.git

3.3.3. Algoritmos de aprendizaje no supervisado: problema de *clustering*

En la sección presente se mostrarán los resultados obtenidos al aplicar los algoritmos explicados en la Sección 2.1. Se va a tratar también el problema de la validación del *clustering* (en inglés, *cluster validation*). La técnica de *clustering* se encuentra dentro del aprendizaje no supervisado, por tanto, normalmente se desconoce el número de *clusters* que subyacen en el *dataset* y también a qué *cluster* pertenece cada elemento. La validación del *clustering* consiste en intentar medir cómo de bueno es el agrupamiento obtenido. Para ello hay dos tipos de criterios: de validación externa y de validación interna. Los primeros se aplican cuando se conoce la estructura real del *dataset*, es decir, el número de *clusters* y la pertenencia de cada punto. El más utilizado es el de las matrices de confusión. Se pueden aplicar, también, algunos de los criterios de validación interna (utilizados sobretodo cuando se desconoce la estructura de los datos) como pueden ser [2, pgs. 195-201]:

- La suma de los cuadrados de las distancias de los puntos a los centros de los *cluster* que pertenecen.
- Las distancias intra e inter *clusters*.
- El coeficiente *Silhouette*.
- Medidas probabilísticas.

k-means

Las variables se encuentran en rangos muy distintos; por ejemplo, la edad se encuentra en el rango $[19, 37]$, *Wingate Peak* (que hace referencia a la potencia máxima) se encuentra en el intervalo $[598, 1280]$, la proporción de tiempo que los ciclistas estuvieron de pie pedaleando está entre 0,08 y 0,62, etc. Debido a las diferencias entre las magnitudes de los intervalos de las variables podría ocurrir que aquellas que se encuentran en los rangos más grandes tuvieran más peso y por tanto las de rangos menores casi no aportaran información al *clustering*. Para solucionar este problema se procede a la normalización de las variables antes de aplicar el algoritmo. Existen varias técnicas de normalización de variables como, por ejemplo:

- *MinMax Scaler* donde $X_{normalizado} = \frac{X - X_{\text{mín}}}{X_{\text{máx}} - X_{\text{mín}}}$
- Escalado estándar con $X_{normalizado} = \frac{X - X_{media}}{X_{\text{desviación típica}}}$

Este último ha sido el seleccionado en este trabajo. De esta manera, se fuerza a que cada variable provenga de una distribución $N(\mu_v, \sigma_v)$ con μ_v y σ_v la media y varianza de todos los valores de la variable v.

Como se muestra en [15], Kleinberg enuncia tres propiedades que cualquier algoritmo de *clustering* debería cumplir. Aunque pueden parecer intuitivas, él mismo demuestra que es imposible que cualquier método las verifique a la vez. Estas propiedades son: invarianza a la escala (en inglés, *scale-invariance*), consistencia (*consistency*) y riqueza (*richness*). En el caso de *k-means*, se cumple la primera y es lo que permite que la solución del agrupamiento no se vea distorsionada por la normalización de las variables.

El algoritmo de *k-means* que se ha aplicado a los datos de [33] se encuentra en la librería `Scikit-Learn`. De acuerdo a la implementación del método de *k*-`means` en la librería mencionada[11], se ha creado un modelo con las siguientes características: agrupa el *dataset* en dos *clusters* ($k = 2$) y como inicialización del algoritmo se emplea *k-means++* (explicado en la Sección 2.1.1).

Se pueden observar las siguientes relaciones entre el agrupamiento realizado por el algoritmo y el determinado por la división de los deportistas entre profesionales y amateurs.

(a) Resultado de *k-means* con todas las variables sin normalizar.

	Cluster 1	*Cluster 0*
Profesionales	8	2
Amateur	0	12

(b) Resultado de *k-means* con todas las variables normalizadas.

	Cluster 0	*Cluster 1*
Profesionales	10	0
Amateur	0	12

Como se puede apreciar, agrupa a los individuos en profesionales y amateurs bastante bien, ya que cada posición de la anterior representación hace referencia al número de profesionales y amateurs que agrupa en cada *cluster* (indicado por la columna) el algoritmo en cada ejecución.

[11] Documentación accesible en https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

En vista de los resultados obtenidos con los algoritmos de clasificación, en los que las variables que mayor precisión tenían a la hora de clasificar a los individuos eran: *Wingate Peak*, *Wingate Mean*, 5′ all out, *Power Mean*, *Relative Power*, *Post 24h CK*, *CL* 0′, *CL* 20′, *Post SBP* 0′, *Post DBP* 0′ y *Post SBP* 3′, se ha decidido volver a aplicar el algoritmo *k-means* con algunas de estas variables. Se han obtenido los siguientes resultados:

(a) Resultado de *k-means* con *Post 24h CK*.

Post 24h CK	*Cluster 1*	*Cluster 0*
Profesionales	10	0
Amateurs	0	12

(b) Resultado de *k-means* con *CL* 0′.

CL 0′	*Cluster 1*	*Cluster 0*
Profesionales	8	2
Amateusr	1	11

La tabla de la izquierda representa la agrupación que devolvió el algoritmo *k-means* tomando como única variable el valor de *Post 24h CK* (variable que obtuvo una precisión de 1 en los algoritmos de clasificación). La tabla de la derecha representa la agrupación que resultó de aplicar *k-means* con el valor de *CL* 0′ (variable que obtuvo una precisión de 0,863 en los algoritmos de clasificación) como única variable.

Finalmente, al aplicar el algoritmo *k-means* con las variables que mayor precisión obtuvieron en los algoritmos de clasificación ($\geq$ 0,8) y al aplicar el algoritmo con el valor de la variable *Power Mean* (variable con una precisión de 1 en los algoritmos de clasificación) se obtuvieron los mismos agrupamientos que los obtenidos con todas las variables normalizadas, es decir, la segunda tabla que aparece. Se han obtenido las tablas que representan agrupamientos con algunas otras de las variables con alta precisión en los modelos de clasificación, pero por falta de espacio no aparecen en la memoria[12].

***Expectation-Maximization* (EM)**

En la librería `Scikit-Learn` se encuentran implementados los modelos compuestos de gaussianas (*gaussian mixture models*) que se explicaron en la Sección 2.1.2. La clase que utilizamos se llama `GaussianMixture`[13] y el modelo creado tiene las siguientes características: *n_components* (el número de *clusters* en los que se desea agrupar) es dos, *tol* (nivel de tolerancia del algoritmo) se ha elegido 0,001. Otro parámetro a destacar es *covariance_type*, que determina cómo son las matrices de varianzas-covarianzas de cada *cluster*. Los valores que puede adoptar dicho parámetro son:

- *full*: Cada *cluster* tiene su propia matriz de varianzas-covarianzas.
- *diag*: Los *clusters* tienen todos matrices de varianzas-covarianzas diagonales (es el tipo de matriz con los que se han realizado los cálculos en la Sección 2.1.2).
- *tied*: Todos los *clusters* tienen la misma matriz de varianza-covariaza.
- *spherical*: Cada componente tiene su valor de varianza.

Al aplicar el algoritmo EM con los cuatro tipos de matrices de varianzas-covarianzas se obtuvieron las probabilidades reflejadas en la Tabla 9d, los agrupamientos de EM con las matrices *diag* y *full* se muestran en la Tabla 9a, el de la matriz *tied* en la Tabla 9b y finalmente el de la matriz *spherical* en la Tabla 9c.

Como se observa en la Tabla 9d los mejores modelos para nuestro *dataset* son aquellos cuyas matrices de varianzas-covarianzas son *diag* y *full* dado que sus probabilidades son las más parecidas a las proporciones de individuos profesionales y amateurs (10/22 y 12/22). Lo cual se corrobora en vista de la Tabla 9a.

Por otro lado, se ha repetido el algoritmo EM con la matriz de varianzas-covarianzas *diag* con cada una de las variables que resultaron ser más precisas en los algoritmos de clasificación (SVM y regresión logística) en la Sección 3.3.1, así como con cada una de ellas y con el conjunto de variables utilizadas como predictoras en el algoritmo de regresión polinómica. Los mejores resultados se han obtenido con los agrupamientos resultantes al emplear EM con la variable *Post 24h CK* y con todo el conjunto de variables. Cabe destacar que el tercer agrupamiento que mejor se ajusta a la realidad es el producido al tomar únicamente la variable *GLUC* 0′ como se muestra en la Tabla 10b.

[12] Pueden encontrarse en https://github.com/ClaudiaCarpintero/TFG.git

[13] La documentación puede encontrarse en https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

Tabla 9: Resultados obtenidos al aplicar el algoritmo EM con todas las variables y las distintas matrices.

(a) *Clustering* resultante al aplicar EM con las matrices de varianzas-covarianzas *diag* y *full*.

diay y *full*	*Cluster 1*	*Cluster 0*
Profesionales	10	0
Amateur	0	12

(b) *Clustering* resultante al aplicar EM con la matriz de varianzas-covarianzas *tied*.

tied	*Cluster 1*	*Cluster 0*
Profesionales	3	7
Amateur	0	12

(c) *Clustering* resultante al aplicar EM con la matriz de varianzas-covarianzas *spherical*.

spherical	*Cluster 1*	*Cluster 0*
Profesionales	8	2
Amateur	0	12

(d) Resultados de las probabilidades resultantes al aplicar las distintas matrices al algoritmo EM de los *clusters* de producir puntos.

Prob	*diag*	*full*	*tied*	*spherical*
Cluster 0	0,45	0,54	0,13	0,36
Cluster 1	0,54	0,45	0,86	0,63

Tabla 10: *clusterings* resultantes con EM al utilizar menos variables.

(a) *Clustering* resultante al aplicar EM con las variables que mayor precisión obtuvieron con los algoritmos de clasificación, que es el mismo que al utilizar solo *Post 24h CK*.

	Cluster 1	*Cluster 0*
Profesionales	10	0
Amateur	0	12

(b) *Clustering* resultante al aplicar EM con las variables empleadas para predecir la potencia media en regresión polinómica y al utilizar solo $GLUC\,0'$.

	Cluster 1	*Cluster 0*
Profesionales	8	0
Amateur	2	12

3.4. Conclusiones

3.4.1. Análisis de los resultados

Para finalizar este trabajo se van a realizar algunas reflexiones en base a los resultados obtenidos a lo largo del análisis, siendo consciente en todo momento de que el *dataset* del que se disponía era pequeño y por tanto las conclusiones deben tomarse con cautela.

Al calcular las matrices de correlación en el comienzo del análisis, se ha observado que hay una relación importante entre la proteína creatina quinasa y el rendimiento de los individuos, lo cual no es de extrañar debido a los numerosos estudios que hay sobre ella y su utilización como marcador biológico para la fatiga y rendimiento de los deportistas. Esta relación se ha visto reafirmada al aplicar los algoritmos de clasificación y calcular las precisiones con dicha variable (0,909 y 1 en los algoritmos de SVM y regresión logística, respectivamente) y al observar los resultados del agrupamiento en base a dicha variable de deportistas en amateurs y profesionales por medio de los algoritmos de *k-means* y EM. Al calcular, al comienzo del estudio (en el análisis descriptivo) las medias de algunas variables, se observaba como la media de *Post 24h CK* en el conjunto de amateurs era notablemente inferior (332,0) frente a la media en el conjunto de profesionales (442,0). Esto hace pensar sobre la posible bondad de la creatina quinasa como indicador del rendimiento en los individuos.

Además de esta variable han surgido otras cuyo estudio puede ser interesante: la concentración de lactato tanto en el momento de finalizar la actividad como 20 minutos después, al finalizar la recuperación (que ya es protagonista de numerosos estudios como biomarcador en el rendimiento de deportistas), algunas medidas sobre el rendimiento directo de los individuos (en nuestro caso han sido las potencias, tanto media como máxima, en las pruebas) o algunas medidas sobre la presión sanguínea (tanto sistólica como diastólica) de los individuos en momentos inmediatos a la finalización de las pruebas (valores al terminar la prueba y tres minutos después).

A nivel teórico se ha podido observar como en el algoritmo de KNN la métrica más adecuada para nuestro estudio fue la euclídea con los datos normalizados, ya que se obtuvo en líneas generales una mayor precisión para varios vecindarios. En cuanto a la decisión de los tamaños de los vecindarios se ha corroborado lo explicado en la sección de teoría: el número de vecinos no debe ser demasiado pequeño (se obtenían precisiones inferiores) porque dependería demasiado de los datos, ni demasiado grande (a partir de cierto valor todas las métricas empeoraban sus precisiones), pues pierde el carácter local que caracteriza al método.

También se ha observado como, al tener una o varias variables que identifican lo suficiente a los individuos de cada clase, los resultados de los algoritmos de clasificación empleados no se han

visto empeorados aún en presencia de variables que podían dificultar su clasificación por introducir ruido en la clasificación (edad, peso, altura, etc.)

Para finalizar, al realizar el estudio de los errores en los métodos de regresión: primero, en las gráficas de los errores de regresión lineal, se ha corroborado el comportamiento explicado en la Sección 2.2 de la regularización aunque parece interesante el hecho de que algunas medidas de error se intersequen y otras no.

Comparando las tablas que muestran los mejores valores para las distintas medidas de bondad del modelo de regresión polinómica con ambos polinomios, puede observarse que la regresión polinomial con polinomio de grado dos ofrece mejores resultados que la regresión polinomial con polinomio de grado tres, salvo para el caso de MAPE donde es mayor su valor con polinomios de grado dos. Por tanto, salvando esta medida de error, parece que es mejor la regresión polinomial con polinomios de grado dos frente a los de grado tres.

Siguiendo en esta línea, si se comparan las medidas de error entre el modelo de regresión lineal y el modelo de regresión polinómica con polinomio de grado dos, se tienen mejores medidas de error para la regresión lineal. Por tanto, este es el mejor modelo para estimar la potencia media de los individuos, es decir, el rendimiento. Se ha comprobado que incrementar la complejidad del modelo no necesariamente implica una mejora en las estimaciones. En regresión polinómica al definir modelos más complejos empeoraba la capacidad de predecir datos nuevos (lo que es indicativo de que se produce *overfitting*).

Futuras líneas de trabajo

Cabe añadir también la importancia que tienen los marcadores tanto biológicos como genéticos. Cada vez hay mas líneas de investigación sobre el tema ya que pueden ser herramientas muy útiles en la detección de enfermedades o problemas de salud en general (en nuestro caso el sobreentrenamiento puede detectarse mediante los niveles de CK como muestra [18]).

En un futuro sería interesante estudiar la importancia de otras variables (que por falta de espacio no se han estudiado en este trabajo) como los incrementos de temperatura, glucosa o CK ya que los deportistas son capaces de aumentar esos indicadores más que el resto de la población.

Por otro lado, sería bueno realizar este estudio en otros experimentos en los que los individuos practiquen otros deportes para determinar si las variables más informadas encontradas dependen del tipo de esfuerzo requerido por el deporte (claramente la prueba de ciclismo de 20 minutos requería de un esfuerzo de resistencia, pero podría estudiarse en individuos que practiquen halterofilia que requiere fuerza explosiva) y también en experimentos con mayor número de individuos.

3.4.2. Conclusiones del trabajo

Al realizar este trabajo, ha sido reconfortante ver que tras cuatro años estudiando el grado de matemáticas he podido relacionar muchos de los conocimientos adquiridos con una aplicación real y tan importante como es el deporte.

Para la realización de este trabajo he necesitado las aptitudes adquiridas a lo largo de las asignaturas de programación como la destreza con distintos lenguajes (este trabajo se ha realizado con Python), la capacidad de análisis de resultados para lograr sacar de ellos conclusiones, los conocimientos sobre optimización de funciones sin los cuales muchas de las técnicas empleadas me hubieran sido imposibles de comprender, la capacidad de abstracción obtenida con las asignaturas más teóricas del grado para poder entender algunas partes teóricas de este trabajo (SVM y t-SNE), y la madurez académica adquirida al finalizar una carrera universitaria sin la cual la realización de este trabajo hubiera sido imposible.

En este proyecto he tenido la oportunidad de adquirir mayor habilidad con el lenguaje de programación de Python, conocer algunas técnicas de Aprendizaje Automático, y averiguar si esta parte de las matemáticas realmente me interesaba.

4. Referencias

[1] Manuel Rodríguez Abreu and Alejandro Núñez Llobregat. Aproximación teórica sobre la fatiga y el sobreentrenamiento, 2010.

[2] Charu C. Aggarwal. *Data Mining: textbook*. Springer, Yorktown Heights New York USA, 1 edition, 4 2015.

[3] Fernando Berzal. Clustering basado en particiones. Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada.

[4] C Bosco, P Luhtanen, and P V Komi. A simple method for measurement of mechanical power in jumping. *Eur J Appl Physiol Occup Physiol.*, 50(2):273–282, 1983.

[5] Cristina García Cambronero and Irene Gómez Moreno. Algoritmos de aprendizaje: Knn & kmeans, curso: Inteligencia en redes de telecomuncicación. Universidad Carlos III.

[6] James Cook, Ilya Sutskever, Andriy Mnih, and Geoffrey Hinton. Visualizing similarity data with a mixture of maps. Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, 2007.

[7] Oliver Cramer. *Dimensionality Reduction with Unsupervised Nearest Neighbors*, volume 51. Springr, Carl von Ossietzky University Oldenburg 26111 Oldenburg Germany, 6 2013.

[8] S. Dasgupta. The hardness of k-means clustering. Technical report, Department of Computer Science and Engineering University of California, San Diego, 2008. Technical Report CS2008-0916.

[9] Miguel Sanz Gutiérrez. Todo lo que debes saber sobre el lactato, 1 2020.

[10] José Martínez Heras and Jose Martinez Heras. Regresión Polinómica en Python con scikit-learn. Blog IArtificial: https://www.iartificial.net/regresion-polinomica-en-python-con-scikit-learn/.

[11] Geoffrey E Hinton and Sam Roweis. *Stochastic Neighbor Embedding*, volume 15, pages 833–840. The MIT Press, 2003.

[12] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Education Psychology*, 24(6):417–441, 1933.

[13] M. Inaba, N. Katoh, and H. Imai. Applications of weighted voronoi diagrams and randomization to variance-based k -clustering. *Scopus*, pages 332–339, 1994.

[14] Saurabh Jaju. Comprehensive Guide on t-SNE algorithm with implementation in R & Python, 6 2017. https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/#h2_5.

[15] Jon Kleinberg. An Impossibility Theorem for Clustering. Department of Computer Science, Cornell University, Ithaca NY 14853.

[16] Kristen Lewis, James O'Neil, and Nataliya Stepulev. Introducción a Sobreajuste y Subajuste para Machine Learning. Blog aprendeIA: https://aprendeia.com/sobreajuste-y-subajuste-en-machine-learning/.

[17] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In Sandip Das and Ryuhei Uehara, editors, *WALCOM: Algorithms and Computation*, pages 274–285, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[18] Diego Marqués-Jiménez, Julio Calleja-González, Iñaki Arratibel, and Nicolás Terrados. Marcadores bioquímicos relevantes del proceso de recuperación en futbol. *revista de la Federación Española de Medicina del Deporte y de la Confederación Iberoamericana de Medicina del Deporte*, 33(176):404–412, 2016.

[19] Abdelmalik Moujahid, Iñaki Inza, and Pedro Larrañaga. Clasificadores K-NN, 2007. Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad del País Vasco–Euskal Herriko Unibertsitatea.

[20] Ricardo Moya. ¿Qué es el Machine Learning?, 1 2016.

[21] Mario Muñoz. Creatina kinasa (CK) alta en deportes de fuerza, 1 2017.

[22] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning*. Packt Publishing Ltd., 35 Livery Street, Birmingham B3 2PB, UK, 3 edition, 12 2015.

[23] Paula Rodó. Propiedades de la distribución t de Student, 11 2019.

[24] Jorge Romero. Análisis Clúster, 6 2019. Blog: https://jorgeromero.net/analisis-cluster/.

[25] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[26] Alberto García Serrano. Regresión polinómica para Machine Learning. Blog El laberinto de Falken: https://www.ellaberintodefalken.com/2019/12/regresion-polinomica-polinomial.html.

[27] Sergio Steven Conde Sánchez, Diana Carolina Zambrano Ríos, and Andrés Matta. Lactato como biomarcador del rendimiento deportivo en deportistas de halterofilia del valle del Cauca. *Encuentro Semilleros de Investigación*, 1(1):32–36, 2018.

[28] Cristina Tîrnăucă, Domingo Gómez-Pérez, José L. Balcázar, and José L. Montaña. Global optimality in k-means clustering. *Information Sciences*, 439-440:79–94, 2018.

[29] Rafael Hernandez Ucan. Método de los K vecinos más cercanos. Blog: https://medium.com/soldai/.

[30] Laurens vander Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2587, 11 2008.

[31] Mohammed J. Zaki and Wagner Meira Jr. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 5 2014.

[32] Dlama Zira. What is the minimum value of correlation coefficient to prove the existence of the accepted relationship between scores of two of more tests?, 06 2021. Blog.

[33] Álvarez Herms J, Odriozola A, Julià-Sánchez J, and Cinca-Morros S. The limits of performance during prolonged exercise. Sin publicar.

[34] Álvaro Vielba. Métodos de optimización convexa en aprendizaje automatico. Master's thesis, Universidad de Valladolid, 3 2019.