CAPÍTULO IX

Predicción y clasificación basada en distancias parcialmente observadas

Aldo R. Franco Comas*
Andrés M. Alonso Fernández

El procedimiento de vecinos más cercanos, k-NN en inglés, se utiliza para la clasificación de nuevas observaciones empleando la matriz de distancias entre las observaciones a clasificar y las observaciones en la muestra de entrenamiento. En este trabajo, desarrollamos un procedimiento k-NN para situaciones donde no es posible calcular todas las distancias entre las nuevas observaciones y las observaciones en la muestra de entrenamiento. Por otra parte, k-NN depende de la distancia utilizada y proponemos un procedimiento para el aprendizaje de la distancia a partir de las distancias en la muestra de entrenamiento. El procedimiento propuesto se ilustra utilizando conjuntos de datos reales.

Palabras clave: aprendizaje de distancias, clustering, curvas de oferta, k-NN, matrices de distancia.

* Este trabajo ha sido parcialmente financiado por la Agencia Estatal de Investigación (PID2019-108311GB-I00/AEI/10.13039/501100011033) y por la Comunidad de Madrid en el marco del convenio plurianual con la Universidad Carlos III Madrid en su línea de actuación "Excelencia para el Profesorado Universitario" del V Plan Regional de Investigación Científica e Innovación Tecnológica 2016-2020. Los autores agradecen los comentarios y mejoras sugeridas por los editores del libro, Daniel Peña, Pilar Poncela y Esther Ruiz.

1. INTRODUCCIÓN

El algoritmo *k-NN (k- Nearest Neighbors* o k-Vecinos más Cercanos) es probablemente el procedimiento de clasificación y regresión más fácil de entender, incluso por personas que estén alejadas del área de conocimiento del aprendizaje estadístico. En lo que sigue, se presentan las características básicas del procedimiento k-NN que es la herramienta básica de este trabajo.

Los datos de entrada tienen la forma (x_i, y_i) siendo x_i un vector de longitud p donde están representadas p variables o atributos para el caso i-ésimo, mientras que y_i es la variable respuesta o etiqueta. La respuesta puede ser un valor numérico como por ejemplo el salario, la estatura o el peso; también puede ser un valor categórico como el tipo de contrato laboral o la tipología de riesgo de accidentes. Igualmente se necesita una distancia entre un caso x_i y otro x_j denotada por d (x_i, x_j). Para cada nuevo caso x_0, se calculan las distancias d (x_0, x_i) para i = 1,2,...,n, siendo n el número total de casos en el conjunto de entrenamiento y se buscan los k casos más cercanos según esa distancia. En la clasificación k-NN, la salida generalmente es una clase o etiqueta. Al nuevo objeto se le asigna la clase más común entre sus k vecinos más cercanos. Si k = 1, entonces el nuevo objeto simplemente se asigna a la clase del vecino más cercano. En la regresión k-NN, la salida usualmente es el promedio de los valores de los k vecinos más cercanos. Existen otros procedimientos de asignación/clasificación y predicción como vecinos más cercanos ponderados donde se da mayor peso a los casos cercanos en función de la distancia al caso nuevo (ver Biccego y Loog, 2016).

Como ventajas principales del k-NN tenemos: 1) No paramétrico: no hace suposiciones explícitas sobre la forma funcional de los datos, evitando los inconvenientes de alejarse de la distribución o modelo subyacente; 2) Precisión satisfactoria: se obtienen precisiones altas, sin embargo puede no ser competitiva en comparación con otros métodos de aprendizaje supervisado, como pueden ser las redes neuronales o las máquinas de vector soporte (SVM) que utilizan procesos de entrenamiento mucho más costosos computacionalmente; 3) Evoluciona constantemente: dado que es un aprendizaje basado en casos, el algoritmo se adapta inmediatamente a medida que recopilamos nuevos datos de entrenamiento. Esto permite que el algoritmo responda rápidamente a los cambios en la entrada durante el uso en tiempo real; 4) Fácil de implementar para problemas de múltiples clases: la mayoría de los algoritmos son fáciles de implementar para problemas binarios, sin embargo requieren un esfuerzo extra a la hora de ser implementados para múltiples clases, como puede ser el caso de las SVM, mientras que k-NN se ajusta a múltiples clases sin dificultad; 5) Clasificación y regresión: una de las mayores ventajas del k-NN es que se puede usar tanto para problemas de clasificación como de regresión; 6) Variedad de distancias: ofrece una alta flexibilidad a la hora de elegir cualquier distancia al construir el modelo.

Mientras que sus desventajas son: 1) Datos no balanceados: no funciona bien en datos no balanceados, es decir, cuando hay alguna clase claramente menos representada; 2) Sensibilidad a valores atípicos: es sensible a los valores atípicos, ya que simplemente elige a los vecinos según los criterios de distancia; 3) Presencia de valores perdidos: no funciona cuando hay valores perdidos en las variables independientes o regresoras; 4) Necesita carac-

terísticas homogéneas: es recomendable que las características tengan la misma escala para evitar que alguna variable sea determinante por el hecho de utilizar unidades de medida diferente del resto de las variables; 5) Maldición de la dimensión: bajo un amplio conjunto de condiciones, a medida que aumenta la dimensionalidad, la distancia al punto de datos más cercano se acerca a la distancia al punto de datos más lejano (ver Beyer *et al.,* 1999); 6) Computacionalmente costoso: puede ser muy fácil de implementar, pero a medida que crece el conjunto de datos, la velocidad del algoritmo disminuye rápidamente.

Algunas de estas desventajas son compartidas por otros procedimientos de clasificación/regresión. En este trabajo nos centraremos en la desventaja del coste computacional que aparece cuando tenemos un problema donde n es muy grande, como ocurre en el contexto de datos masivos (*big data*), y calcular todas las distancias para predecir no es factible. Esta situación puede darse, por ejemplo, cuando, por razones de tiempo de respuesta, por un tiempo de cómputo excesivo (en conjuntos de datos extremadamente grandes) o pruebas destructivas (análisis de ADN), no es posible calcular todas las distancias de las nuevas observaciones a todas las observaciones en la muestra de entrenamiento. En este trabajo, propondremos una solución para los casos en que no sea posible calcular todas las distancias del caso a clasificar a todos los elementos del conjunto de entrenamiento.

1.1. Objetivos

El objetivo fundamental de este trabajo es proponer un procedimiento k-NN donde no sea necesario calcular todas las distancias cada vez que tengamos que predecir un caso nuevo. Dado que solamente podemos calcular un porcentaje de las distancias, las restantes deben ser imputadas, este es el problema al cual nos enfrentamos. Compararemos distintos enfoques que se han propuesto para completar matrices de distancias (Dhillon, Sra y Tropp, 2005; De Soete, 1984; Lapointe y Kirsch, 1995) en combinación con el procedimiento k-NN en términos del error del vector de distancias y del orden de vecinos resultantes. Los métodos también se compararan por el coste computacional aún sabiendo que esto último depende del ordenador, lenguaje y forma de implementación. Todos los experimentos se realizaron usando R (R Development Core Team, 2008) y los códigos desarrollados están disponibles mediante solicitud a los autores.

1.2. Estructura

El resto del trabajo se estructura en las siguientes secciones. En la sección segunda se hace una revisión de la literatura sobre algoritmos que resuelven el problema de completamiento de matrices de distancias y se proponen dos procedimientos de inicialización. Se realizan comparaciones mediante simulación para determinar el método que obtiene mejores resultados. En la sección tercera se utiliza el método propuesto en el conjunto de datos MNIST (LeCun y Cortes, 2010) para un problema de clasificación de múltiples clases (10 clases) y en datos del mercado eléctrico español para un problema de regresión. La sec-

ción cuarta presenta las conclusiones y extensiones de trabajo. Las definiciones básicas que serán de utilidad para la formulación del problema de completamiento de matrices de distancias y la evaluación de los procedimientos se presentan en un apéndice.

2. METODOLOGÍA: ANTECEDENTES Y PROPUESTA

En esta sección, primero, se realiza una breve revisión bibliográfica de artículos que abordan problemas similares al nuestro como lo es el problema de la métrica más cercana o el problema de la inferencia filogenética. Se propone una solución basada en procedimientos clúster y se compara el desempeño de las distintas opciones mediante un estudio de simulación.

2.1. El problema de la métrica más cercana

Supongamos que tenemos una matriz $\boldsymbol{D}$ de dimensión $n \times n$ donde representamos las distancias entre n individuos. Por tanto, los elementos de $\boldsymbol{D}$ deberían cumplir las desigualdades triangulares pero, por errores de medición o incluso por omisión de algunas mediciones, estas desigualdades no se verifican. El *problema de la métrica más cercana* consiste en encontrar una matriz $\boldsymbol{M}$ cuyos elementos cumplan las correspondientes desigualdades triangulares y que esté próxima a la matriz $\boldsymbol{D}$. Este problema ha sido estudiado en Dhillon, Sra y Tropp (2003 y 2005) y Brickell *et al.* (2008), y puede formularse como sigue:

> Si tenemos n puntos, podemos representar las medidas entre cada par de puntos en una matriz simétrica $\boldsymbol{D}$ cuya entrada (i, j) representa la "distancia" entre los objetos i y j. Buscamos aproximar esta matriz por otra matriz, $\boldsymbol{M}$, cuyas entradas satisfacen las desigualdades triangulares. Es decir, $m_{ik} \leq m_{ij} + m_{jk}$ por cada tupla (i, j, k). En otras palabras se requiere una matriz de distancias $\boldsymbol{M}$ que sea la más cercana a una matriz de disimilitud dada $\boldsymbol{D}$ con respecto a alguna norma entre matrices. Específicamente, se busca una matriz de distancias $\boldsymbol{M}$ tal que,
>
> $$\boldsymbol{M} = \arg \min_{\boldsymbol{X} \in \mathcal{M}_n} \left\{ \left\| \boldsymbol{W} \odot (\boldsymbol{X} - \boldsymbol{D}) \right\| \right\}, \qquad [1]$$
>
> donde $\mathcal{M}_n$ es el conjunto de todas las matrices de distancias $n \times n$, $||\cdot||$ es una norma matricial, $\boldsymbol{W}$ es una matriz $n \times n$ de pesos, simétrica y no negativa, y $\odot$ denota la multiplicación elemento a elemento entre dos matrices. La matriz de pesos $\boldsymbol{W}$ refleja nuestra confianza en las entradas de la matriz $\boldsymbol{D}$. Por ejemplo, cuando cada d_{ij} representa una medida con varianza σ_{ij}^2, podríamos utilizar $w_{ij} = 1/\sigma_{ij}^2$ y si falta una entrada en $\boldsymbol{D}$ podríamos poner cero en el peso correspondiente.

El problema de la métrica más cercana se diferencia del escalado multidimensional métrico en que no busca una matriz de datos cuyas distancias (usualmente euclídeas) estén cerca de la matriz $\boldsymbol{D}$ ni impone ninguna hipótesis sobre el espacio subyacente más que requerir que sea un espacio métrico.

En Dhillon, Sra y Tropp (2003) se prueba que el problema [1] alcanza su mínimo en $\mathcal{M}_n$. Además, cada mínimo local es un mínimo global. Si, además, la norma es estrictamente convexa y la matriz de pesos no tiene ceros o infinitos fuera de su diagonal, entonces hay un mínimo global único. En principio, es posible utilizar cualquier norma entre matrices para este problema, pero lo analizaremos para las normas L_r. Con lo cual los problemas asociados son:

$$\min_{\mathbf{X}\in\mathcal{M}_n} \left[\sum_{j\neq k} |w_{jk}(x_{jk}-d_{jk})|^r\right]^{1/r} \quad \text{si} \quad 1\leq r<\infty \qquad [2]$$

y

$$\min_{\mathbf{X}\in\mathcal{M}_n} \max_{j\neq k} \left|w_{jk}(x_{jk}-d_{jk})\right| \quad \text{si} \quad r=\infty. \qquad [3]$$

Para resolver los problemas [2] y [3] puede parecer que se debería utilizar programación lineal cuando $r = 1$ ó ∞, y programación convexa para $r > 1$, pero resulta que los requisitos de tiempo y almacenamiento de estos enfoques son prohibitivos. En Brickell *et al.* (2008) se proponen algoritmos para resolver el problema [1] para las normas L_1, L_2, y L_∞ y siguiendo su propuesta, se planteará el algoritmo para $r = 2$ puesto que este caso resulta ser el más simple y juega un papel fundamental en la resolución de los problemas para $r = 1$ y $r = \infty$. El algoritmo *triangle fixing* va recorriendo todas las tuplas (i, j, k) y corrigiendo aquellas que no satisfacen la desigualdad triangular.

En el algoritmo 1, las $e_{ij} = m_{ij} - d_{ij}$ representan los cambios de las disimilaridades originales, d, a las distancias finales, m; $b_{ijk} = d_{ki} + d_{jk} - d_{ij}$ indica en qué medida se viola la desigualdad triangular, y z es tal que $e = -\boldsymbol{A}'z$ con $\boldsymbol{A}$ la matriz que codifica las desigualdades triangulares. Si en una tupla (i, j, k) se incumple la desigualdad triangular, tenemos que $e_{ij} - e_{jk} - e_{ki} > b_{ijk}$. Para resolver este incumplimiento se proyecta ortogonalmente el vector e sobre el conjunto de restricciones $\{x : x_{ij} - x_{jk} - x_{ki} \leq b_{ijk}\}$, lo que devuelve como solución a:

$$\begin{aligned} x_{ij} &\longleftarrow e_{ij} + \mu, \\ x_{jk} &\longleftarrow e_{jk} + \mu, \\ x_{ki} &\longleftarrow e_{ki} + \mu, \end{aligned} \qquad [4]$$

con $\mu = \dfrac{-b_{ijk} + e_{ij} - e_{jk} - e_{ki}}{3}$. Por lo tanto, solo tres componentes de e tienen que ser actualizadas, y sus valores están dados por [4]. El procedimiento se repite hasta que ninguna tupla reciba una actualización significativa.

En las simulaciones utilizaremos $r = 2$. El algoritmo para este valor está implementado en R en la librería *dtw (Dynamic Time Warping)* (Giorgino, 2009).

Llegados a este punto, nos preguntamos ¿qué relación tiene el problema de la métrica más cercana con nuestro problema? Si asumimos que conocemos todas las distancias entre los n objetos en la muestra de entrenamiento podemos crear una nueva matriz $(n + 1) \times (n + 1)$ donde la columna/fila $(n + 1)$ serían las distancias del punto x_0 a las restantes teniendo en cuenta que

Algoritmo 1.

Algoritmo Triangle Fixing para L_2

Entrada: Matriz de disimilitud, $\boldsymbol{D}$, y tolerancia, ε.

Salida: $\boldsymbol{M} = argmin_{\boldsymbol{X} \in \mathcal{M}_N} ||(\boldsymbol{X} - \boldsymbol{D})||_2$.

for $1 \leq i < j < k \leq n$ **do**

| $z_{ijk} \leftarrow 0$

end

for $1 \leq i < j \leq n$ **do**

| $e_{ij} \leftarrow 0$

end

$\delta \leftarrow 1 + \varepsilon$

while $\delta > \varepsilon$ **do**

  foreach *Incumplimiento de la desigualdad triangular para (i, j, k)* **do**

    $b_{ijk} \leftarrow d_{ki} + d_{kj} - d_{ij}$

    $\mu \leftarrow -\frac{1}{3}(b_{ijk} - e_{ij} + e_{jk} + e_{ki})$

    $\theta \leftarrow \text{mín}\ \{-\mu, z_{ijk}\}$

    $e_{ij} \leftarrow e_{ij} - \theta$

    $e_{jk} \leftarrow e_{jk} + \theta$

    $e_{ki} \leftarrow e_{ki} + \theta$

    $z_{ikj} \leftarrow z_{ikj} - \theta$

  end

  $\delta \leftarrow$ *Suma de todos los cambios en los valores de e.*

end

return $M = D + E$

tendremos un ℓ % de distancias conocidas y un $(1 - \ell)$ % de distancias sin calcular y podemos utilizar el algoritmo 1 para imputar esos valores. A estos valores faltantes tenemos que darle algún valor inicial pues el algoritmo triangle fixing necesita valores numéricos. Hemos considerado varias opciones para inicializar estos valores faltantes:

- $d(x_0, x_j) = 0$, siendo j aquel objeto del cual no tenemos la distancia. De este primer valor, no se reportan resultados pues el tiempo de ejecución era extremadamente alto al igual que las diferencias relativas entre matrices.
- $d(x_0, x_j) = \dfrac{\sum_{i=1}^{n} d(x_i, x_j)}{n}, i \neq j.$
- $d(x_0, x_j) = \text{mediana}\ (d(x_i, x_j)), \forall i \neq j.$

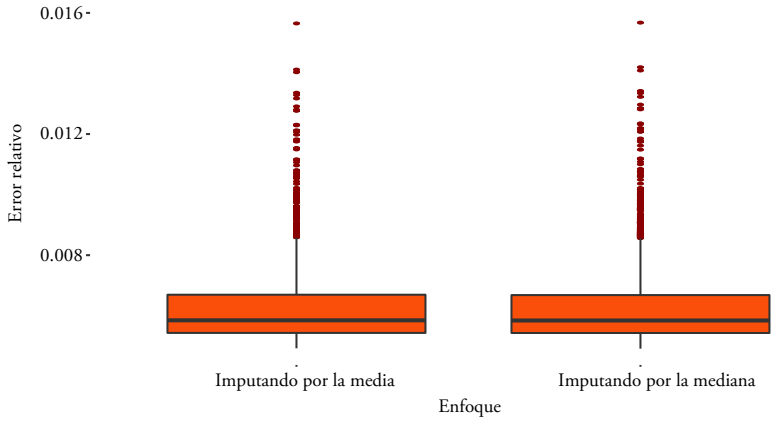
Para comparar las distintas propuestas de valores iniciales de las distancias desconocidas realizamos un estudio de simulación que consta de los siguientes pasos:

1. Se generan $n = 800$ puntos de una distribución normal multivariante con $\mu = 0_p$ y $\Sigma = I_p$, siendo $p = 20$.
2. Se calcula la matriz de distancias, usando la distancia euclidiana.
3. Se genera un nuevo punto x_0 y se calculan las n distancias.
4. Se asumen conocidas las distancias de x_0 a $n\ell = 80$ puntos, ($\ell = 0{,}1$) y las restantes distancias se imputan usando los métodos descritos anteriormente, con lo cual tenemos una matriz de disimilitud, D.
5. Se aplica el algoritmo triangle fixing a D y nos devuelve una matriz M.
6. Se calcula la diferencia relativa entre la matriz M y la matriz de distancias original al igual que la diferencia relativa entre el vector de distancias correspondientes a x_0 en la matriz original y en M. También se registra el tiempo de cómputo.
7. Se repiten los pasos 3 – 6, $N = 1.000$ veces.

Se pueden apreciar en la figura 1 las diferencias relativas entre las matrices que devuelve el algoritmo y las matrices originales. Estas grandes diferencias y altos costes en tiempo (como se puede apreciar en la figura 2) vienen dados por el hecho que estas imputaciones no verifican holgadamente las desigualdades triangulares, con lo cual el algoritmo tiene que iterar muchas veces. Por tanto debemos buscar una forma de mejorar estas aproximaciones iniciales.

Figura 1.

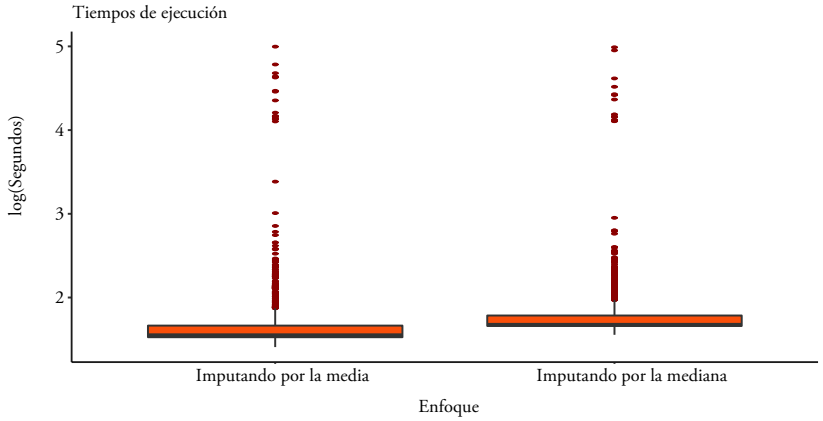
Diagramas de caja de las diferencias relativas entre la matriz de distancias original y la matriz imputada, *M*



Fuente: Elaboración propia.

Figura 2.

Diagramas de caja de los tiempos de ejecución para cada enfoque en escala logarítmica



Fuente: Elaboración propia.

2.1.1. Método de inicialización propuesto

Teniendo en cuenta las restricciones impuestas a nuestro problema, sabemos que las distancias $d(x_i, x_j)$ son conocidas $\forall i, j \in \{1,2,...,n\}$ y podemos calcular las distancias de $d(x_0, x_i)$ cuando $i \in I$ con $|I| \ll n$ pero no podemos calcular las $d(x_0, x_i)$ cuando $i \in I^c$. Los cardinales de los conjuntos I e I^c son aproximadamente iguales a ℓn y $(1 - \ell n)$, respectivamente.

Por otra parte, haciendo uso de las desigualdades triangulares, tenemos:

$$d(x_0, x_{i^*}) \leq d(x_0, x_i) + d(x_i, x_{i^*}) \quad \forall i \in I \text{ e } i^* \in I^c \qquad [5]$$

y, puesto que la desigualdad se tiene $\forall i \in I$, se concluye que:

$$d(x_0, x_{i*}) \leq \min_{i \in I} \{d(x_0, x_i) + d(x_i, x_{i*})\}. \qquad [6]$$

De igual forma, tenemos que:

$$d(x_i, x_{i^*}) \leq d(x_0, x_{i^*}) + d(x_0, x_i) \qquad [7]$$

y

$$d(x_0, x_i) \leq d(x_0, x_{i^*}) + d(x_i, x_{i^*}) \qquad [8]$$

por tanto, obtenemos que:

$$|d(x_0, x_i) - d(x_i, x_{i^*})| \leq d(x_0, x_{i^*}), \qquad \forall i \in I \text{ e } i^* \in I^c \qquad [9]$$

lo que implica que:

$$\max_{i \in I} | d(x_0, x_i) - d(x_i, x_{i^*}) | \leq d(x_0, x_{i^*}). \qquad [10]$$

Notar que las desigualdades [6] y [10] nos permiten acotar las distancias $d(x_0, x_{i^*})$ mediante:

$$\max_{i \in I} | d(x_0, x_i) - d(x_i, x_{i^*}) | \leq d(x_0, x_{i^*}) \leq \min_{i \in I} \{d(x_0, x_i) + d(x_i, x_{i^*})\} \qquad [11]$$

e imputaremos dichas distancias con la media de sus cotas:

$$d(x_0, x_{i^*}) = \frac{1}{2}\left(\min_{i \in I} \{d(x_0, x_i) + d(x_i, x_{i^*})\} + \max_{i \in I} | d(x_0, x_i) - d(x_i, x_{i^*}) | \right). \qquad [12]$$

Cuando asignamos los valores imputados usando la expresión [12] para posteriormente aplicar el algoritmo triangle fixing obtenemos que la matriz $\boldsymbol{D}$ coincide con la matriz $\boldsymbol{M}$ que devuelve dicho algoritmo. Con lo cual parece ser que esos valores imputados satisfacen todas las condiciones que debe verificar una matriz de distancia. Tenemos una prueba parcial de esta propiedad y hemos comprobado mediante simulación utilizando ℓ entre 0,1 y 0,4, pero no tenemos una demostración completa de esta conjetura.

Dado el coste computacional del algoritmo triangle fixing, imputaremos las distancias faltantes usando la expresión [12]. Hasta ahora hemos seleccionado el conjunto de puntos, *I*, tomando una muestra aleatoria en el conjunto de entrenamiento. En lo que sigue propondremos un método que nos ayude a mejorar estos resultados basándonos en métodos clúster.

2.2. Clúster o agrupamiento

El clustering o agrupamiento es una técnica de aprendizaje no supervisada que intenta encontrar relaciones entre variables pero no la relación que guardan con respecto a una variable objetivo, es decir sin hacer uso de las etiquetas. El conjunto de datos tiene que ser dividido automáticamente en clústeres, de manera que los objetos dentro del mismo clúster sean similares, mientras que los objetos de diferentes clústeres sean menos semejantes. No existe una definición general de clúster, lo que significa que diferentes enfoques pueden obtener diferentes clústeres del mismo conjunto de datos.

El motivo por el cual el agrupamiento será de gran importancia en este trabajo está dado por el hecho de que hasta ahora hemos seleccionado los puntos a los cuales se le calculan las distancias de una forma totalmente aleatoria. Pero usaremos la idea propuesta en Zhang *et al.* (2016), donde en un conjunto de datos muy grandes se decide aplicar k-NN por submuestras y cada submuestra es definida mediante un clúster. A continuación se puede ver un resumen de dicho algoritmo.

Algoritmo 2.

k-NN basado en clústeres

Entrada: Datos de entrenamiento y dato a clasificar x_0.

Salida: Etiqueta.

1. Producir K clústeres, C_1, C_2, ..., C_K.
2. Calcular la distancia de x_0 a todos los centroides, $d(x_0, C_i)\ \forall i = 1, ..., K$.
3. Buscar el centroide (C_i) más cercano a x_0.
4. Crear un nuevo conjunto de datos, correspondientes a los puntos que pertenecen a C_i.
5. Usar estos datos como datos de entrenamiento para predecir x_0.

Existen muchos métodos de clustering como, por ejemplo, la agrupación jerárquica, los métodos K-medias, *PAM* (*Partitioning Around Medoids*), y *DBSCAN* (*Density-Based Spatial Clustering of Application with Noise*). Estos son de los más conocidos y usados (Schubert *et al.,* 2017), sin embargo dado el impacto de los datos masivos existen trabajos que intentan paralelizar estos procedimientos y escalar a conjuntos de datos más grandes.

2.2.1. K-medias

Uno de los algoritmos más usados es el K-medias, el cual tiene como objetivo crear K grupos a partir de n observaciones, en el que cada una pertenece a un grupo cuyo valor medio es el más cercano. Sean $(x_1, x_2,..., x_n)$ las observaciones, el algoritmo construye una partición de dichas observaciones, $C = \{C_1, ..., C_K\}$, con el fin de minimizar la suma de los cuadrados dentro de cada grupo:

$$\arg \min_{C} \sum_{i=1}^{K} \sum_{x_j \in C_i} \| x_j - \mu_i \|_2^2, \qquad [13]$$

donde μ_i es la media de las observaciones en C_i.

Desafortunadamente, aunque el agrupamiento de K-medias es bastante eficiente en tiempo computacional se sabe que es sensible a los valores atípicos. Por esta razón, a veces se utiliza la agrupación de K-medoides, donde se consideran objetos representativos dentro de cada clúster en lugar de los centroides.

2.2.2. K-medoides

K-medoides es una familia de algoritmos que escogen puntos del conjunto de datos como centros y trabaja con una métrica arbitraria de distancias. Es más robusto ante atípicos que K-medias porque minimiza una suma de disimilaridades (entre pares de puntos) en vez de una suma de distancias euclidianas a una media. Un medoide puede ser definido, como el objeto de un grupo cuya disimilaridad media a todos los objetos en el grupo es mínima.

La implementación práctica más común de K-medoides es el algoritmo Partición Alrededor de Medoides (PAM) cuya idea es similar a K-medias en el sentido de asociar cada punto a un representante más cercano, pero sustituye la media del clúster en [13] por un elemento del clúster. Otros algoritmos K-medoides son CLARA y CLARANS (Schubert y Rousseeuw, 2019). Sin embargo, PAM tiene el inconveniente de que funciona de manera ineficiente para conjuntos de datos grande (Han, Kamber y Tung, 2001). Existe una versión, basada en K-medias, que se denomina fastkmed y que permite el manejo de grandes conjuntos de datos. El algoritmo fastkmed parte de la idea de seleccionar mediodes iniciales basándose en la suma de las distancias relativas al conjunto de datos.

El resultado de varios estudios de similación muestran que dicho método tiene mejor rendimiento que el agrupamiento K-medias. También se reduce significativamente el tiempo de ejecución con respecto al PAM con un rendimiento comparable, PAM es del orden de $O(k(n-k)^2)$ y el fastkmed del orden de $O(nk)$ muy similar al K-medias (Park y Jun, 2009).

Algoritmo 3.

Algoritmo fastkmed

Entrada: Seleccionar K de los n puntos como medoides.

Salida: K medoides.

1. Seleccione los medoides iniciales:
   - Calcular la distancia entre cada par de objetos.
   - Calcular v_j para el objeto j como sigue:

$$v_j = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{i=1}^{n} d_{il}}, \; j = 1, \ldots, n$$

   - Ordenar v_j en orden ascendente y seleccionar los k objetos con menores v_j como medoides iniciales.
   - Obtener la partición inicial asignando cada objeto al medoide más cercano.
   - Calcular la suma de distancias de todos los objetos a sus medoides.
2. Actualizar medoides:
   - Encontrar un nuevo medoide de cada grupo, que es el objeto que minimiza la distancia total a otro objeto en su clúster. Actualizar el medoide actual en cada grupo reemplazando con el nuevo medoide.
3. Asignar objetos a los medoides:
   - Asignar cada objeto al medoide más cercano y obtener el resultado del clúster.
   - Calcular la suma de la distancia de todos los objetos a sus medoides. Si la suma es igual a la anterior, detener el algoritmo y en caso contrario, volver al paso 2.

2.3. Procedimientos aditivo y ultramétrico

El problema de la inferencia filogenética a partir de conjuntos de datos que incluyen entradas incompletas o inciertas es uno de los temas más relevantes en biología sistemática (Lapointe y Makarenkow, 2004). El objetivo es inferir filogenias o relaciones de parentesco a partir de datos evolutivos, incluida información faltante, por ejemplo, cuando las secuencias de nucleótidos o proteínas observadas contienen huecos o entradas faltantes.

Entre los diferentes algoritmos de inferencia filogenética hay dos que se han usado para completar matrices de distancias. Estos métodos han sido propuestos por De Soete (1984) y Lapointe y Kirsch (1995) para ultramétricas, mientras que para matrices de distancias aditivas por Landry, Lapointe y Kirsch (1996). Ambos algoritmos están implementados en R, en la librería *ape* (*Analyses of Phylogenetics and Evolution*) desarrollada por Paradis y Schliep (2018). Estos algoritmos utilizan las imputaciones que se derivan de las definiciones 5.5 y 5.6, respectivamente.

Algoritmo 4.

Algoritmo para distancias ultramétricas

Entrada: Distancia parcial d en el conjunto de n observaciones.

Salida: Distancia completa o parcial d en el conjunto de n observaciones.

1. Contar el número de NA en d, se denota por c.

2. **foreach** $d(i, j)$ *que sea NA de* d **do**

   $MinMax$ es la máxima entrada de d.

   foreach k *tal que* $d(i, k)$ *y* $d(j, k)$ son valores conocidos de d ***do***

      $Max = \max(d(i, k); d(j, k))$

      if $Max = minMax$ **then**

         $Max = minMax$

      end

   end

   if *Si hay al menos un par conocido de entradas* $d(i, k)$ *y* $d(j, k)$ **then**

      $d(i, j) = MinMax$

      $c = c - 1$

   end

end

3. Si cambia c, ir al paso 2.

Algoritmo 5.

Algoritmo para distancias aditivas

Entrada: Distancia parcial d en el conjunto de n observaciones.

Salida: Distancia completa o parcial d en el conjunto de n observaciones.

1. Contar el número de NA en d, se denota por c.

2. **foreach** $d(i, j)$ que sea *NA de* d **do**

   $MinMax$ es la máxima entrada de d.

   foreach k, l tal que $d(i, k)$, $d(j, k)$, $d(i, l)$, $d(j, l)$ y $d(k, l)$ son valores conocidos de d ***do***

      $Max = \max(d(i, k) + d(j, l); d(i, l) + d(j, l) - d(k, l))$

      if $Max < minMax$ **then**

         $Max = minMax$

      end

   end

   if Si hay al menos las entradas $d(i, k)$, $d(j, k)$, $d(i, l)$, $d(j, l)$ *y* $d(k, l)$ **then**

      $d(i, j) = MinMax$

      $c = c - 1$

   end

end

3. Si cambia c, ir al paso 2.

Estos algoritmos tienen la ventaja de no necesitar ningún tipo de imputación inicial. De hecho los valores faltantes deben definirse como NA para poder hacer uso de estos métodos.

2.4. Estudio de simulación

En esta sección se compararán los diferentes métodos propuestos y para ello usaremos dos métricas, el MAE y el índice de Jaccard, es decir, el error absoluto medio entre la matriz real y la matriz obtenida y la concordancia entre el conjunto de vecinos real y el derivado a partir de la matriz de distancias obtenida. Las definiciones formales pueden verse en el apéndice. Haremos simulaciones donde compararemos los resultados observación a observación, es decir, en ningún momento utilizaremos las etiquetas que pueden tener estas observaciones. Este ejercicio se hará en la próxima sección para el conjunto de datos MNIST usando la mejor estrategia que salga de estas simulaciones.

El motivo por el cual usaremos ambas métricas viene dado por el hecho de que el índice de Jaccard presenta un inconveniente. Por ejemplo, supongamos que los vecinos más cercanos para una observación de validación x_0 son las observaciones $A = \{10, 15, 20, 30\}$ de la muestra de entrenamiento y obtenemos que un algoritmo nos devuelve los siguientes puntos más cercanos $B_1 = \{10, 15, 20, 35\}$ y otro algoritmo los puntos $B_2 = \{10, 15, 20, 40\}$ con lo cual se obtiene que $J(A, B_1) = J(A, B_2) = 3/5$. Vemos que el índice de Jaccard coincide y no permite diferenciar entre ambos resultados. Con el MAE seleccionaríamos el resultado con menor error de imputación de las distancias.

Primero, estudiaremos el número de clústeres que debemos escoger de tal manera que el índice de Jaccard sea mayor. Es importante destacar que este problema es diferente al que resuelven métodos de selección del número de clústeres presentes en un conjunto de datos.

Para encontrar un número "óptimo" de clústeres mediante simulación se realizan los siguientes pasos:

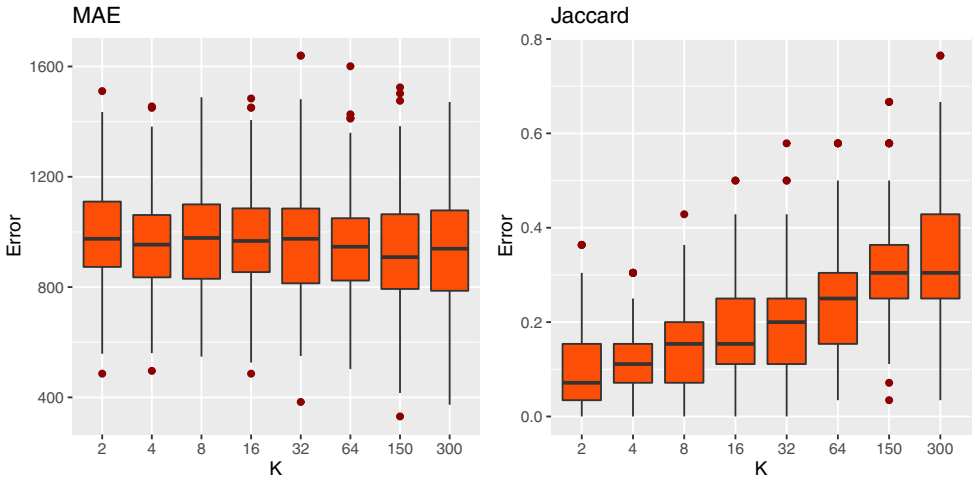
1. Se generan n = 3.000 puntos de una distribución normal multivariante con $\mu = 0_p$ y $\Sigma = I_p$, siendo $p = 50$.
2. Se calcula la matriz de distancias, usando la distancia euclidiana.
3. Se aplica fastkmed a dicha matriz de distancias para diferentes valores de $K = (2, 4, 8, 16, 32, 64, 150, 300)$, donde K es el número de clústeres. De tal manera se obtienen K mediodes $\{M_1, M_2, ..., M_K\}$.
4. Se genera un nuevo punto x_0.
5. Se calculan las distancias $d(x_0, M_i)$ con $i = 1, ..., K$ y se ordenan de menor a mayor.
6. En este punto ya hemos calculado K distancias con lo cual las restantes se harán seleccionando de cada clúster más cercano hasta que se hayan calculado el número máximo de distancias $n\ell = 300$, pues $\ell = 0{,}1$.
7. Para los diferentes valores de K expuestos se calcula el índice de Jaccard y el MAE entre el conjunto real de puntos más cercano y el conjunto de puntos más cercano que se obtiene imputando, para $k = 15$ vecinos.
8. Se repiten los pasos 4 - 7, $N = 200$ veces.

Se puede observar en la figura 3 que a medida que el número de clústeres aumenta el índice de Jaccard mejora, con lo cual a partir de este momento decidiremos usar como número de clústeres el valor correspondiente a $K = n\ell/2$ ($K = 150$ en la figura), pues presenta menor variabilidad que $n\ell$ que sería el valor correspondiente a $K = 300$. También observamos que el MAE decrece lentamente con el número de clústeres.

A continuación, compararemos los procedimientos de imputación usando puntos al azar o usando clústeres:

Figura 3.

Búsqueda de un valor "óptimo" del número de clústeres, *K*



Fuente: Elaboración propia.

1. Se generan n = 5.000 puntos de una distribución normal multivariante con $\mu = 0_p$ y $\Sigma = I_p$, siendo p =50.

2. Se calcula la matriz de distancias, usando la distancia euclidiana.

3. Se genera un nuevo punto x_0.

4. Se calculan las distancias de x_0 a los n puntos.

5. Se calculan las distancias de x_0 a $n\ell$ puntos al azar y también a la misma cantidad usando $K = n\ell/2$ clústeres. En ambos casos las restantes distancias se imputan usando la expresión [12].

6. Se ordenan dichas distancias y se extrae cuáles son los k puntos más cercanos, el máximo valor que toma k es $\sqrt{n}/2$.

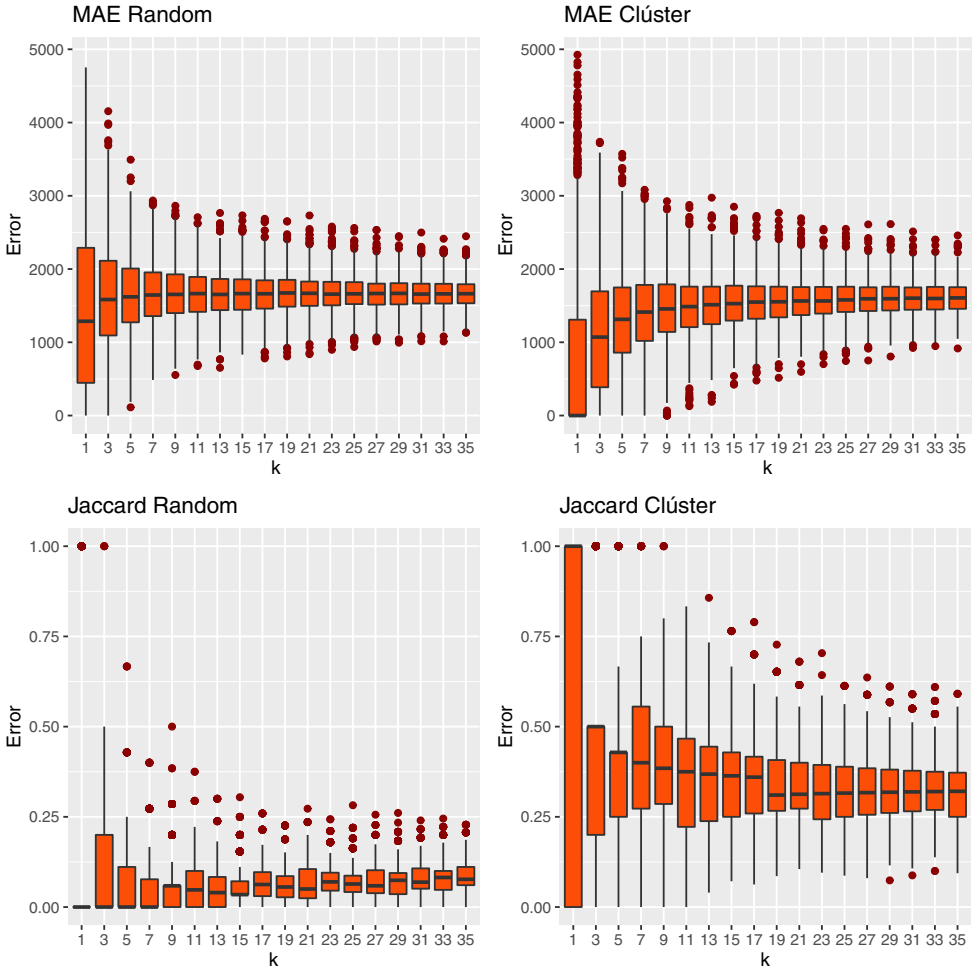
7. Para los diferentes valores de k y los dos procedimientos, se calcula el índice de Jaccard y el MAE entre el conjunto de puntos más cercano real y el conjunto de puntos más cercano que se obtiene mediante imputación.

8. Se repiten los pasos 4 – 7, N = 1.000 veces.

En la figura 4 se puede apreciar claramente que calcular distancias al azar tiene un desempeño inferior al procedimiento basado en clústeres tanto para el MAE como para el

Figura 4.

Comparación del procedimiento de imputación con selección al azar frente a selección usando clústeres



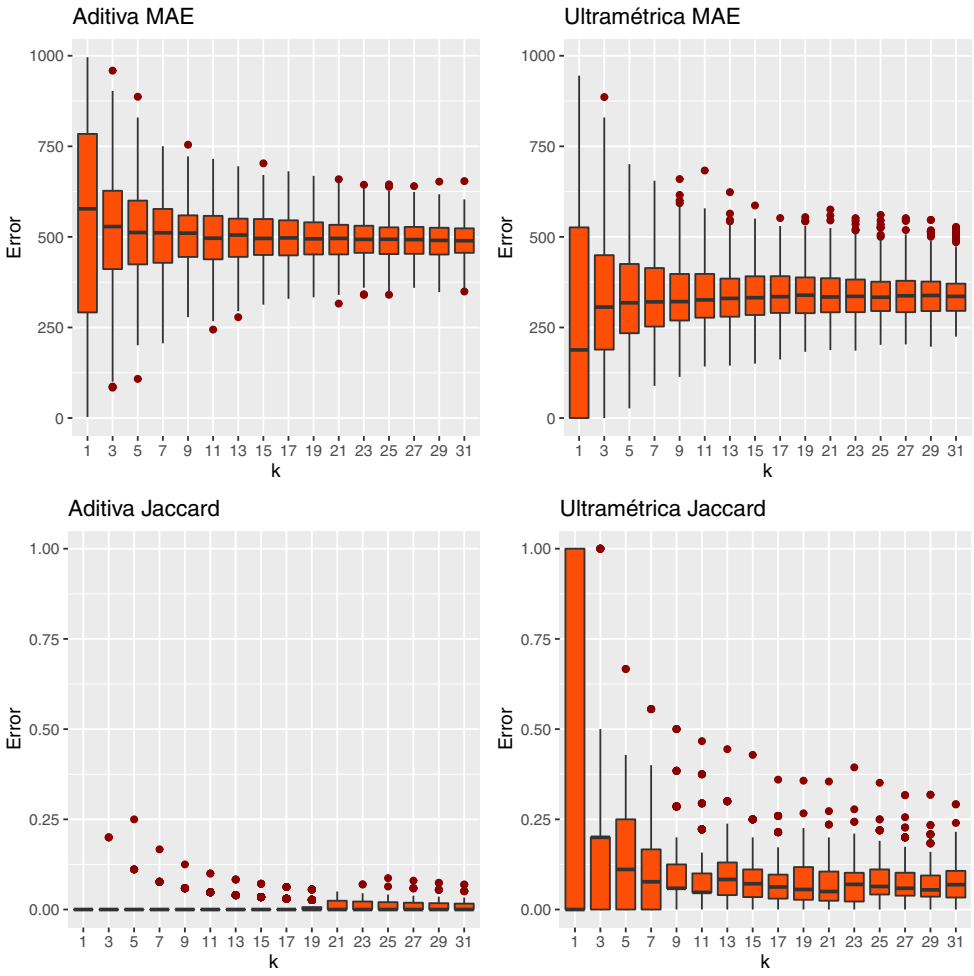
Fuente: Elaboración propia.

índice de Jaccard. También hemos realizado la comparación entre los métodos ultramétrico y aditivo, viendo claramente que el ultramétrico obtiene mejores resultados, ver figura 5.

Finalmente, se muestra una comparación entre el procedimiento ultramétrico y el procedimiento propuesto basado en clústeres, donde se puede apreciar que nuestro método tiene un mejor desempeño (ver figura 6).

Figura 5.

Comparación de los procedimientos aditivo y ultramétrico

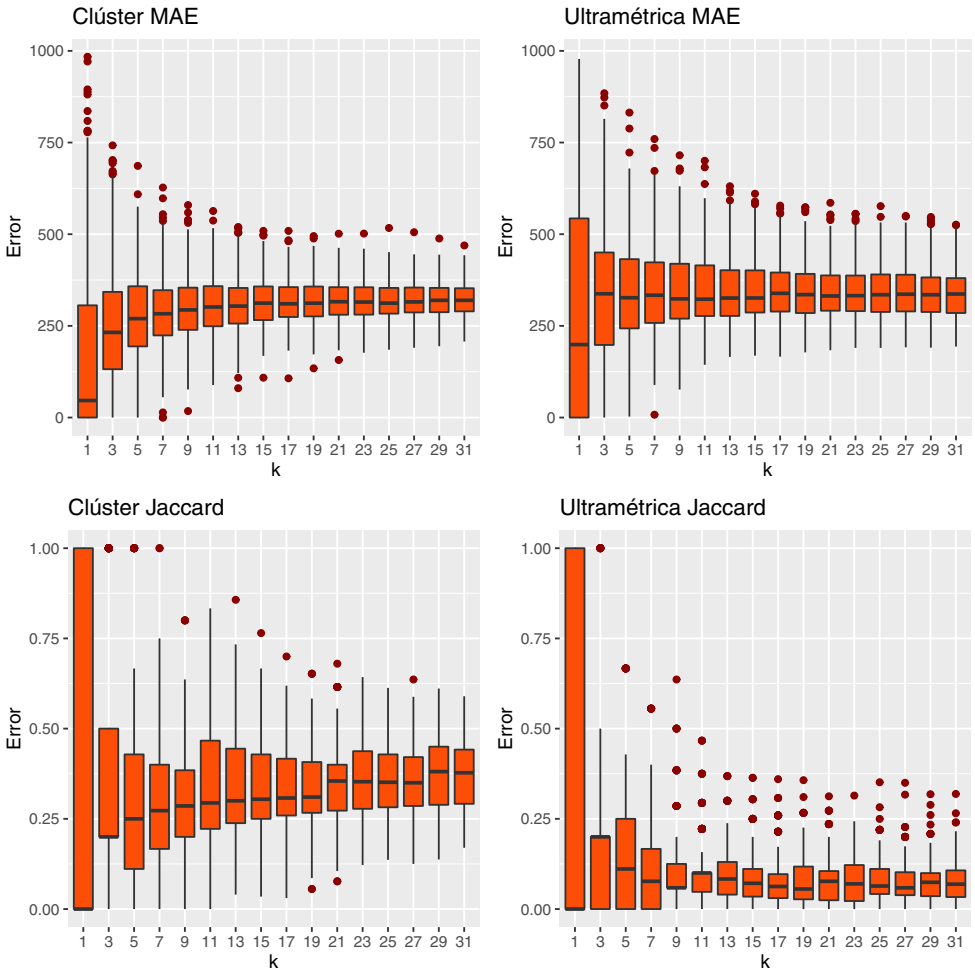


Fuente: Elaboración propia.

Se observa que, respecto al MAE, las diferencias entre ambos enfoques es pequeña. Sin embargo, analizando el índice de Jaccard se puede ver que el método propuesto es superior al resto. Al método propuesto, en adelante, lo denominaremos como "Imputación mediante clústeres".

Figura 6.

Comparación de los procedimientos ultramétrico y de imputación mediante clúster



Fuente: Elaboración propia.

3. EJEMPLOS CON CONJUNTOS DE DATOS REALES

En esta sección, ilustraremos el uso del procedimiento de imputación mediante clústeres en dos conjuntos de datos reales:

- MNIST: Es una base de datos de dígitos escritos a mano. Son 42.000 imágenes en blanco y negro, cada una con 28 pixeles de ancho y de alto.

- COEMS: Es una base de datos que contiene las curvas de oferta de electricidad horarias del mercado secundario en España en el período desde el 1 de enero de 2014 al 31 de diciembre de 2019.

3.1. MNIST

En esta sección, comprobaremos el procedimiento propuesto con el k-NN clásico usando un dataset real de clasificación. Usaremos la base de datos MNIST que consiste en imágenes en blanco y negro normalizadas cuyas dimensiones son de 28 × 28 píxeles en niveles de escala de grises y representan dígitos escritos a mano. El problema de clasificación consiste en, dada una nueva imagen, predecir qué número tiene escrito.

Figura 7.

Imágenes seleccionadas del conjunto de datos de entrenamiento del MNIST

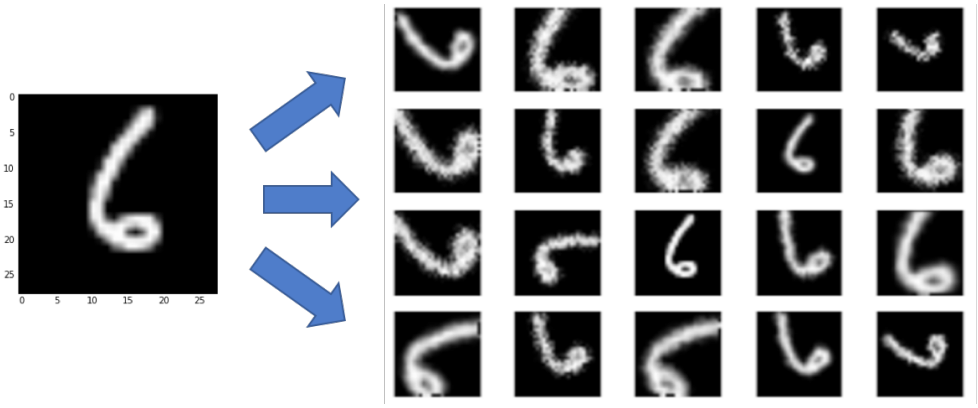
Fuente: https://commons.wikimedia.org/wiki/File:MnistExamples.png

La base de datos fue descargada de https://www.kaggle.com/c/digit-recognizer/data https://www.kaggle.com/c/digit-recognizer/data, consta de 42.000 observaciones, de las cuales dejaremos el 75 % para entrenar y validar y las restantes para probar nuestro método. Esta división se hizo usando un reparto estratificado entre las muestras de entrenamiento y de prueba (Kuhn *et al.,* 2019).

Las razones para utilizar un muestreo estratificado en lugar de un muestreo aleatorio simple son mantener una distribución similar de las etiquetas originales en el nuevo subconjunto; las mediciones se vuelven más manejables cuando la población se agrupa en estratos y, a menudo, es deseable tener estimaciones de los parámetros de la población para todos los subgrupos (Hyndman y Athanassopoulos, 2018).

Figura 8.

Ejemplo de *data augmentation* para una imagen seleccionada del MNIST

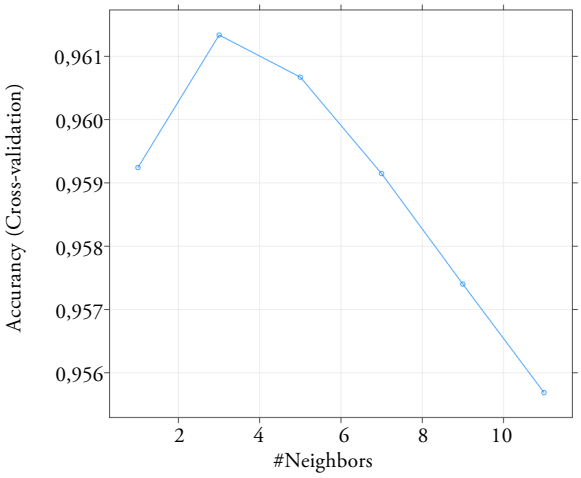


Fuente: https://www.snorkel.org/doks-theme/assets/images/2017-08-11-tanda/data_aug_basic.png

De esta manera, el conjunto de datos está balanceado con lo cual evitamos un punto débil que presenta k-NN. En caso de que fuera un conjunto de datos no balanceado se recomienda hacer *data augmentation*. Esta técnica es muy común a la hora de clasificar imágenes y no

Figura 9.

Selección del parámetro *k* del k-NN en datos de entrenamiento y validación del MNIST



Fuente: Elaboración propia.

es más que tomar una imagen y crear nuevas muestras rotando, dilatando, trasladando o agregando ruido blanco a la original, ver figura 8.

Para la búsqueda del valor óptimo de k, se utilizó el procedimiento de validación cruzada con cinco submuestras del conjunto de entrenamiento y validación donde se obtiene una precisión del 96,14 % siendo el mejor valor $k = 3$, ver figura 9.

Usando ahora este valor de k predecimos las etiquetas para los valores de la muestra de prueba usando el k-NN donde se obtiene una precisión del 96,98 % con la siguiente matriz de confusión:

Tabla 1.

Matriz de confusión del MNIST usando k-NN (k = 3)

	Referencia									
Predicción	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	1.024	0	4	2	1	4	6	2	1	6
1	0	1.161	8	1	7	1	0	6	10	2
2	2	4	1.002	4	0	2	2	3	4	2
3	1	0	5	1.043	0	8	1	0	18	10
4	0	1	0	0	985	0	0	1	3	8
5	0	1	1	20	0	920	6	0	16	6
6	5	0	0	1	8	11	1.019	0	5	0
7	0	3	21	5	3	0	0	1.083	5	10
8	0	1	1	6	1	0	0	0	941	1
9	1	0	2	5	13	2	0	5	12	1.002

Fuente: Elaboración propia.

Observando la matriz anterior podemos notar que los casos donde más falla el procedimiento son en etiquetas similares, por ejemplo el 2 con el 7, y el 8 con el 3 y con el 5.

Para este ejercicio, trabajamos con $\ell = 0{,}25$, lo que significa que solo podemos calcular el 25 % de las distancias. Las restantes fueron imputadas usando el método propuesto, donde el número de clústeres es K = 3938. De esta forma se obtiene una precisión del 96,42 % y la matriz de confusión que aparece en la tabla 2.

La matriz de confusión es similar a la obtenida utilizando todos los datos (Tabla 1). Este ejercicio muestra que el procedimiento propuesto tiene un comportamiento similar al que se obtiene calculando todas las distancias.

Tabla 2.

Matriz de confusión del MNIST usando k-NN (k = 3) e imputación mediante clústeres

	Referencia									
Predicción	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	1.022	0	9	3	2	4	11	0	4	4
1	1	1.166	10	4	10	2	1	10	11	1
2	1	1	990	7	0	0	1	6	7	0
3	0	1	2	1.030	0	7	0	0	14	3
4	1	2	2	0	971	0	2	6	4	6
5	0	0	0	14	0	920	2	0	19	2
6	6	0	4	3	7	8	1.016	0	11	1
7	2	0	21	13	4	2	0	1.067	5	12
8	0	0	3	7	0	1	1	0	923	2
9	0	1	3	6	24	4	0	11	17	1.016

Fuente: Elaboración propia.

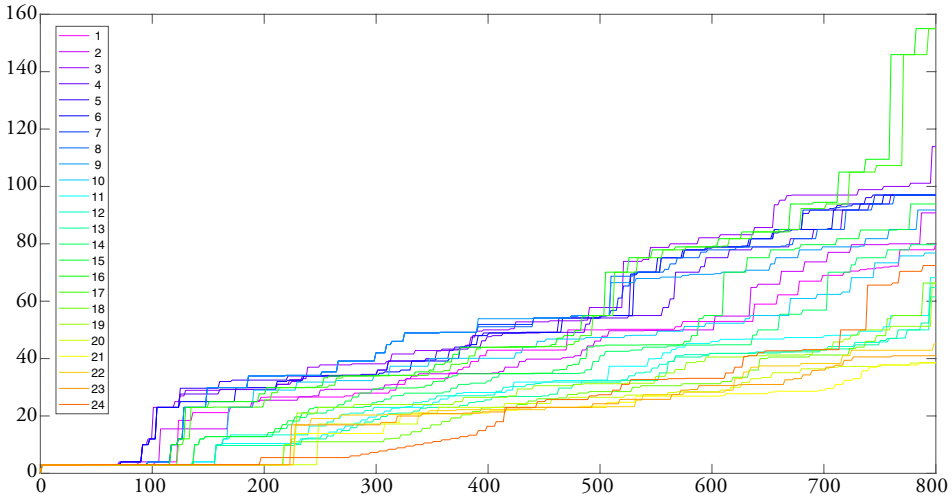
3.2. COEMS

El mercado eléctrico en España permite a los productores de electricidad ofrecer bloques de energía a diferentes precios, generalmente relacionados con sus costos marginales, en momentos específicos del día. El operador del sistema, Red Eléctrica de España (REE), recoge todas las ofertas (bloques de energía) y sus correspondientes precios de todos los participantes para formar la curva de oferta con la que se obtendrá el precio marginal de cada hora. La energía final producida por cada participante será remunerada a este precio marginal. Existen diferentes mercados: diario, intradiario, secundario y terciario. Estos mercados se diferencian tanto en el momento del día en que se realiza como en el horizonte de aplicación. Los datos están disponibles en [https://www.esios.ree.es/es/curvas-de-ofertas].

En este ejemplo estamos interesados en el mercado secundario donde el horizonte de aplicación (y por tanto, de predicción) son las 24 horas del día siguiente. Usaremos la información disponible hasta el momento antes de que se realice el mercado, por lo que podemos usar las curvas de oferta de ese mercado del día D-1, para predecir las curvas de oferta del día siguiente, D. También, en ese momento, están disponibles los precios del día D (mercado diario) y las predicciones de demanda y de producción de parques eólicos pero su incorporación en el modelo predictivo será objeto de investigación futura. A modo de ejemplo, en la figura 10 mostramos las curvas de oferta para las 24 horas del 1 de enero de 2014 en el rango de 0 a 800 MW. Utilizamos la gráfica de arco iris propuesta por Hyndman y Shan (2010), ya que nos permite darnos cuenta de que las curvas que corresponden a las horas

Figura 10.

Curvas de oferta de electricidad del mercado secundario, 1 de enero de 2014



Fuente: Elaboración propia.

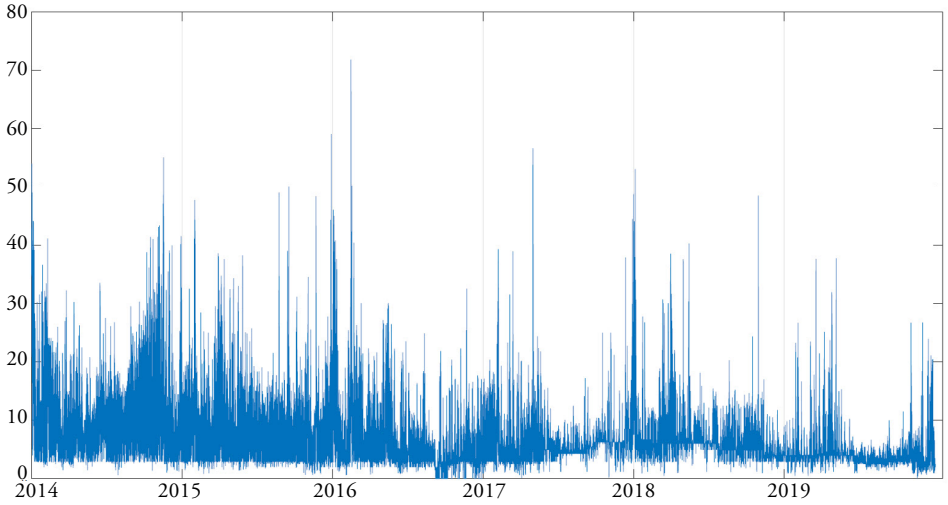
contiguas están cercanas lo que significa que los datos históricos serán útiles para predecir las curvas de oferta del día siguiente.

Estas curvas de ofertas son funciones escalonadas, no decrecientes y continuas por la izquierda. Tanto el número de escalones (ofertas con distinto precio) como la máxima oferta acumulada son dependientes del día/hora. En este ejemplo, usaremos 800 MW como valor máximo de la oferta acumulada para todas las curvas de oferta.

Otra representación interesante de estas curvas son las series de precios horarias para cantidades fijas. Es decir, si fijamos la cantidad de electricidad que queremos "comprar", qué precios tendremos que pagar por esa cantidad. Por ejemplo, supongamos que queremos obtener la serie temporal de precios al que podríamos comprar 400MW, para ello tendríamos que cortar las curvas de la figura 10 fijando el valor 400 en el eje horizontal y determinar los precios correspondientes. En la figura 11, se representa esta serie temporal. Esta representación de series de tiempo nos permite estudiar la dependencia temporal utilizando herramientas como las autocorrelaciones simples y parciales. En la figura 12, mostramos las funciones de autocorrelaciones simples de estas series temporales. Se observa que todas las series tienen una clara estacionalidad diaria pero no solo eso sino que la dependencia temporal varía en función de la cantidad a comprar. Por ejemplo, la dependencia es más fuerte en pequeñas o medianas cantidades (menos de 200MW), aumenta en cantidades intermedias (400MW a 600MW) y disminuye en cantidades grandes (más de 600MW). Estas diferencias en la dinámica sugieren una posible relación no lineal entre las curvas en D-1 y en D.

Figura 11.

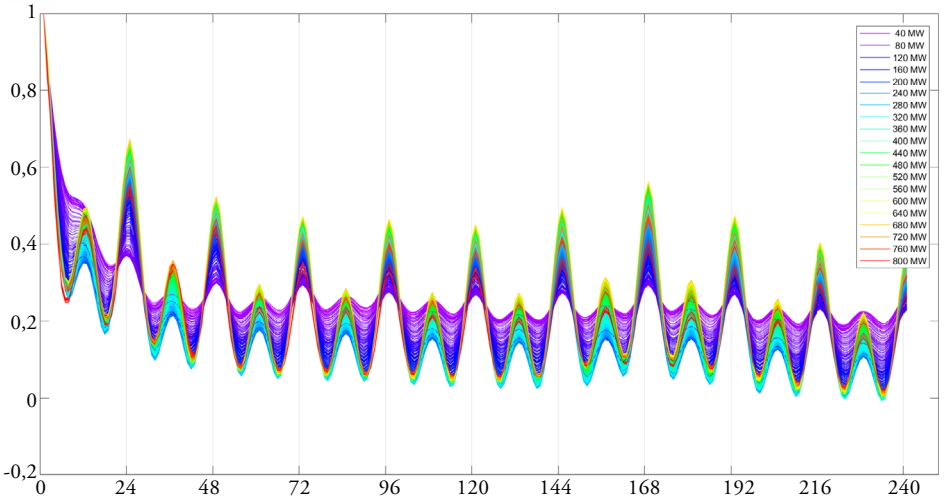
Serie temporal de los precios de 400MW en el mercado secundario, 2014-2019



Fuente: Elaboración propia.

Figura 12.

Funciones de autocorrelación simple de los precios de ofertas desde 40 MW a 800MW en el mercado secundario, 2014-2019



Fuente: Elaboración propia.

Un procedimiento simple de predicción basado en k-NN consiste en buscar días pasados que hayan tenido un comportamiento similar al día D-1 y tomar como predicciones las curvas del día siguiente de esos días similares. El procedimiento se formula como sigue:

- Dadas las 24 curvas de oferta del día D-1, $\boldsymbol{C_t} = \{C_{t-23}, C_{t-22}, ..., C_t\}$, queremos predecir las 24 curvas del día D, $\boldsymbol{C_{t+24}} = \{C_{t+1}, C_{t+2}, ..., C_{t+24}\}$.
- Sea $s^* = argmin s_{\leq t-24} d(\boldsymbol{C_t}, \boldsymbol{C_s})$, es decir, $\boldsymbol{C_{s^*}}$ son las 24 curvas consecutivas más cercanas a $\boldsymbol{C_t}$.
- La predicción se obtiene mediante $\hat{\boldsymbol{C}}_t = \{C_{s^*+1}, C_{s^*+2}, ..., C_{s^*+24}\}$.

En la formulación anterior faltaría por definir la distancia, d, entre $\boldsymbol{C_t}$ y $\boldsymbol{C_s}$. Esto nos condujo a la pregunta de cuál distancia es la importante para la predicción, d ($\boldsymbol{C_t}$, $\boldsymbol{C_s}$) o d ($\boldsymbol{C_{t+24}}$, $\hat{\boldsymbol{C}}_{t+24}$). Evidentemente, la segunda de estas distancias es más relevante porque mide el error de predicción. Sin embargo, esta distancia no se puede calcular *a priori* porque no conocemos $\boldsymbol{C_{t+24}}$ en el momento de la predicción. La cuestión es si podemos aprender de d ($\boldsymbol{C_t}$, $\boldsymbol{C_s}$) para predecir $d(\boldsymbol{C_{t+24}}, \boldsymbol{C_{s+24}})$. Notar que la predicción de $\boldsymbol{C_{t+24}}$ mediante el procedimiento k-NN será $\boldsymbol{C_{s+24}}$. En lo que sigue proponemos un procedimiento para el aprendizaje de la distancia (*distance learning*). Vamos a simplificar el problema y predecir $d(\boldsymbol{C_{t+h}}, \boldsymbol{C_{s+h}})$, es decir, la distancia entre las curvas de la hora h.

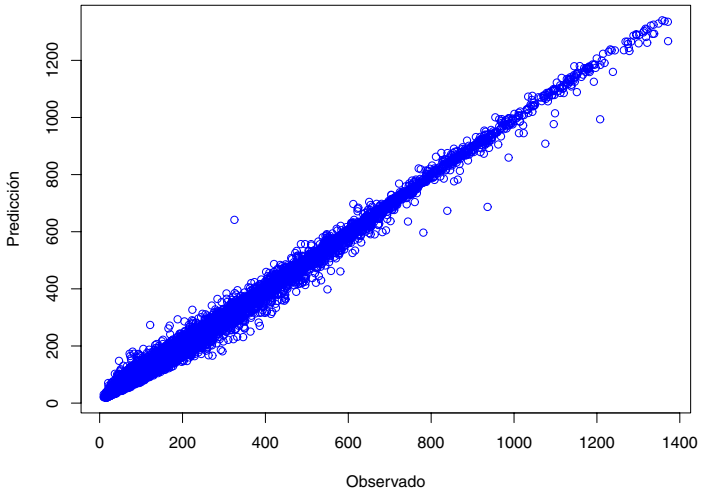
- Para dos tiempos en la muestra de entrenamiento, s y t con $s \leq t - 24$, podemos calcular las siguientes distancias:
  - $d(C_t, C_s)$, $d(C_{t-1}, C_{s-1})$ ..., $d(C_{t-23}, C_{s-23})$
  - $d(C_t, C_{s+h})$, $d(C_{t-1}, C_{s+h})$ ..., $d(C_{t-23}, C_{s+h})$
- Estas 48 variables serán el *input* de un procedimiento de bosque aleatorio (*random forest*) para predecir la distancia $d(C_{t+h}, C_{s+h})$.

La selección de las variables está motivada por la alta dependencia temporal entre las curvas tanto a corto plazo como en retardos múltiplos de 24. La elección del procedimiento de bosque aleatorio se basa en su versatilidad para modelizar relaciones no lineales e interacciones entre variables. Por simplicidad, hemos utilizado la distancia euclídea entre cada par de curvas, es decir, $d^2(C_t, C_s) = \int_0^{800} (C_t(q) - C_s(q))^2 dq$.

Para el entrenamiento del procedimiento se usan las 43.824 curvas horarias de los años 2014 a 2018 y, como conjunto de prueba, las 8.760 horas del año 2019. Se utilizan 25 réplicas bootstrap con 20000 observaciones para la selección del número de variables a incluir en el modelo. El modelo final utiliza 500 árboles y obtiene un ajuste satisfactorio en el conjunto de entrenamiento tal como se ilustra en la figura 13. Las variables más relevantes se muestran en la figura 14 que como vemos son tanto variables que captan la dependencia de corto plazo ($X25 = d(C_t, C_{s+1})$, X1 = ($d(C_t, C_s)$ y $X26 = d(C_{t-1}, C_{s+1})$) como variables que captan la dependencia estacional diaria ($X48 = C_{t-23}, C_{s+1}$) y $X47 = d(C_{t-22}, C_{s+1})$).

Figura 13.

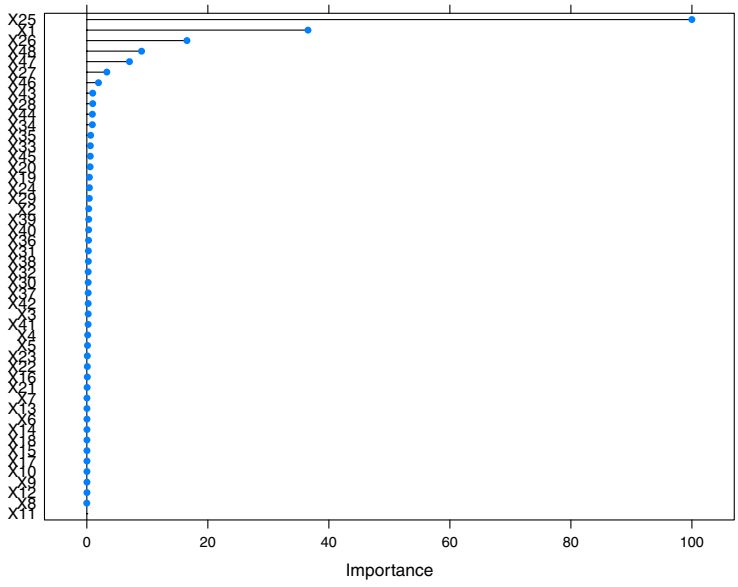
Ajuste del procedimiento de bosque aleatorio para *h* = 1, 2014-2018



Fuente: Elaboración propia.

Figura 14.

Importancia de las variables en el modelo de bosque aleatorio para *h* = 1



Fuente: Elaboración propia.

El mismo procedimiento se realiza para los distintos horizontes de predicción, $h = 1, 2, ..., 24$, es decir, se entrenan 24 modelos de bosque aleatorio. Estos modelos nos permiten realizar la predicción para el día siguiente utilizando las prediciones de las distancias para los distintos horizontes de predicción. Se utiliza como modelo de referencia aquel que elige

Tabla 3.

Medias de las distancias euclídeas entre la curva real y su predicción
(Errores estándar entre parentésis)

Método	*h = 1*	*h = 12*	*h = 24*
Referencia	88.586 (0.322)	79.068 (0.275)	73.555 (0.288)
k-NN + RF (1)	**59.245** (0.227)	87.614 (0.224)	73.347 (0.236)
k-NN + RF (12)	95.813 (0.265)	**64.543** (0.213)	87.332 (0.229)
k-NN + RF (24)	72.806 (0.224)	87.185 (0.205)	**57.811** (0.147)

Fuente: Elaboración propia.

la menor distancia euclídea agregada entre el último día disponible y días anteriores en la muestra de entrenamiento. En la tabla 3 se muestran las medias de las distancias euclídeas entre la curva real y su predicción en la muestra de prueba (8.760 horas del año 2019). En la tabla, Referencia corresponde al modelo de referencia, y k-NN + RF(h) corresponde a la combinación de k-NN con el procedimiento de bosque aleatorio entrenado para el horizonte h. Como vemos, el procedimiento propuesto mejora al modelo de referencia en todos los horizontes de predicción y se concluye que es relevante el entrenamiento para cada horizonte de predicción por separado.

Tabla 4.

Desempeño del procedimiento k-NN + RF(h) en el caso de matrices parcialmente observadas. Medias de las distancias euclídeas entre la curva real y su predicción
(Errores estándar entre parentésis)

Método	*h = 1*	*h = 12*	*h = 24*
k-NN + RF (1) ℓ = 25 %	**60.217** (0.227)	93.154 (0.239)	84.137 (0.244)
k-NN + RF (12) ℓ = 25 %	122.196 (0.312)	**64.405** (0.213)	102.161 (0.250)
k-NN + RF (24) ℓ = 25 %	87.253 (0.243)	101.508 (0.244)	**57.457** (0.150)

Fuente: Elaboración propia.

Por último, hemos repetido el ejercicio anterior para el caso de matrices parcialmente observadas, en particular, solo se calculan el 25 % ($\ell = 0{,}25$) de las distancias entre las curvas del conjunto de entrenamiento y las curvas del último día disponible. Los resultados se muestran en la tabla 4 y podemos concluir que los resultados son similares a los obtenidos usando todas las distancias. En este caso, es más evidente la importancia del entrenamiento del modelo para cada horizonte de predicción.

4. CONCLUSIONES Y EXTENSIONES

En este trabajo hemos estudiado modificaciones al procedimiento k-NN cuando no es factible calcular todas las distancias entre las nuevas observaciones y todas las observaciones en el conjunto de entrenamiento.

Hemos estudiado la combinación del algoritmo triangle fixing con varias propuestas de valores iniciales resultando que la propuesta basada en acotaciones obtiene mejores resultados y es computacionalmente factible en grandes conjuntos de datos.

Hemos comprobado que la selección de observaciones al azar en el conjunto de entrenamiento es inferior a una selección basada en clústeres tanto en índice de Jaccard como en el MAE. El procedimiento de imputación basado en clústeres también resulta superior a los procedimientos aditivos y ultramétricos que han sido propuestos en relación con la inferencia filogenética.

Finalmente, hemos utilizado el procedimiento propuesto en dos conjuntos de datos reales y hemos observado que los resultados son similares a los obtenidos con un k-NN que utiliza todas las distancias.

Como futuras líneas de investigación, tenemos:

- Implementar estos algoritmos en un lenguaje como C o C++ haciendo uso del paquete Rcpp (Eddelbuettel y Balamuta. 2018) lo que permitiría realizar experimentos en conjuntos de datos de mayor dimensión.

- Buscar un número "óptimo" de clústeres que tenga en cuenta la función de pérdida del problema a resolver con k-NN, es decir, que maximice la precisión en el caso del problema de clasificación o que minimice el error de predicción en el problema de regresión. También puede ser interesante estudiar la selección conjunta del parámetro k del k-NN y del parámetro K del procedimiento de imputación mediante clústeres.

- Estudiar extensiones al aprendizaje de distancia incorporando la componente temporal.

Referencias

Beyer, K., Goldstein, J., Ramakrishnan, R. y Shaft, U. (1999). When is "nearest neighbor" meaningful? *Database Theory- ICDT'99,* pp. 217-235. Springer Berlin Heidelberg.

Bicego, M. y Loog, M. (2016). Weighted k-nearest neighbor revisited. *23rd International Conference on Pattern Recognition (ICPR),* pp. 1642-1647. IEEE.

Brickell, J., Dhillon, I., Sra, S. y Tropp, J. A. (2008). The metric nearness problem. *SIAM Journal on Matrix Analysis and Applications,* 30, pp. 375-396.

Buneman, P. (1971). *The Recovery of Trees from Measures of Dissimilarity,* pp. 387-395. Edinburgh: University Press.

Dauben, J. W. (1990). *Georg Cantor: His Mathematics and Philosophy of the Infinite.* Princeton University Press.

De Soete, G. (1984). Additive-tree representations of incomplete dissimilarity data. *Quality and Quantity,* 18, pp. 387-393.

Dhillon, I., Sra, S. y Tropp, J. (2003). The metric nearness problem with applications. Technical report, University of Texas at Austin.

Dhillon, I., Sra, S. y Tropp, J. A. (2005). Metric nearness: Problem formulation and algorithms. *Advances in Neural Information Processing,* 17, pp. 361-368.

Eddelbuettel, D. y Balamuta, J. J. (2018). Extending R with C++: A Brief Introduction to Rcpp. *The American Statistician,* 72, pp. 28-36.

Giorgino, T. (2009). Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of Statistical Software,* 31, pp. 1-24.

Han, J., Kamber, M. y Tung, A. K. H. (2001). Spatial clustering methods in data mining. *Geographic data mining and knowledge discovery,* pp. 188-217.

Hartigan, J. A. (1967). Representation of similarity matrices by trees. *Journal of the American Statistical Association,* 62, pp. 1140-1158.

Heltshe, J. F. (1988). Jackknife estimate of the matching coeficient of similarity. *Biometrics,* 44, pp. 447-460.

Hyndman, R. J. y Athanasopoulos, G. (2018). *Forecasting: principles and practice. OTexts*.

Hyndman, R. J. y Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting,* 22, pp. 679-688.

Hyndman, R. J. y Shang, H. L. (2010). Rainbow plots, bagplots and boxplots for functional data. *Journal of Computational & Graphical Statistics,* 19, pp. 29-45.

Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C. y Hunt, T. (2019). *caret: Classification and Regression Training.*

Landry, P.-A. y Lapointe, F.-J. (1997). *Estimation of missing distances in path-length matrices: Problems and solutions,* pp. 209-218.

Landry, P.-A., Lapointe, F.-J. y Kirsch, J. A. W. (1996). Estimating phylogenies from lacunose distance matrices: Additive is superior to ultrametric estimation. *Molecular Biology and Evolution,* 13, pp. 818-823.

Lapointe, F. J. y Kirsch, J. A. W. (1995). Estimating Phylogenies from Lacunose Distance Matrices, with Special Reference to DNA Hybridization Data. *Molecular Biology and Evolution,* 12, pp. 266-266.

Lapointe, F.-J. y Makarenkov, V. (2004). A weighted least-squares approach for inferring phylogenies from incomplete distance matrices. *Bioinformatics,* 20, pp. 2113-2121.

LeCun, Y. y Cortes, C. (2010). MNIST handwritten digit database.

Paradis, E. y Schliep, K. (2018). ape 5.0: An environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics,* 35, pp. 526-528.

Park, H.-S. y Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications,* 36, pp. 3336-3341.

R Development Core Team (2008). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing.

Real, R. y Vargas, J. (1996). The Probabilistic Basis of Jaccard's Index of Similarity. *Systematic Biology,* 45, pp. 380-385.

Rudin, W. (1991). *Functional Analysis. International series in pure and applied mathematics.* McGraw-Hill.

Schubert, E. y Rousseeuw, P. J. (2019). Faster k-medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms. En: *Similarity Search and Applications,* (pp. 171-187). Springer International Publishing.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P. y Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS),* 42, pp. 1-21.

Zhang, S., Deng, Z., Cheng, D., Zong, M. y Zhu, X. (2016). Eficient kNN Classification Algorithm for Big Data. *Neurocomputing,* 195, pp. 143-148.

APÉNDICE

Definiciones básicas

Definición 5.1. *Para un conjunto de elementos, $\mathcal{X}$, una distancia o métrica es cualquier función $d(a, b) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ donde $\mathbb{R}$ es el conjunto de los números reales, que verifique las siguientes condiciones (ver en Rudin, 1991):*

- No negatividad: $\forall a,b \in X : d(a,b) \geq 0$.
- Coincidencia: $d(a,b) = 0 \Leftrightarrow a = b$.
- Simetría: $\forall a,b \in X : d(a,b) = d(b,a)$.
- Desigualdad triangular: $\forall a, b, c \in X : d(a,b) \leq d(a,b) + d(c,b)$.

Definición 5.2. *Sea $\boldsymbol{M} = \{m_{ij}\}$ una matriz $n \times m$, se dice no negativa cuando todas sus entradas son no negativas, es decir $m_{ij} \geq 0$ $\forall i,j$.*

Definición 5.3. *Sea $\boldsymbol{M}$ una matriz $n \times n$, se dice que es matriz de disimilitud si es una matriz no negativa, simétrica y con ceros en su diagonal principal (Dhillon, Sra y Tropp, 2003).*

Definición 5.4. *Sea $\boldsymbol{M} = \{m_{ij}\}$ una matriz $n \times n$, se dice que es una matriz de distancias, si es una matriz de disimilitud cuyas entradas satisfacen la desigualdad triangular. Esto es, $\boldsymbol{M}$, es matriz de distancias si y solo si para toda terna de índices (i, j, k) se verifica que (Dhillon, Sra y Tropp, 2005):*

$$m_{ik} \leq m_{ij} + m_{jk}.$$

Definición 5.5. *Una matriz de distancias, $\boldsymbol{M} = \{m_{ij}\}$, se dice aditiva, cuando sus entradas verifican tanto la desigualdad triangular como la cuadrangular (Buneman, 1971):*

$$m_{ij} + m_{kl} \leq \text{máx}[m_{ik} + m_{jl}; m_{il} + m_{jk}] \qquad \forall i, j, k, l.$$

Definición 5.6. *Una matriz de distancias, $\boldsymbol{M} = \{m_{ij}\}$, se dice ultramétrica, cuando es aditiva y sus entradas verifican la desigualdad ultramétrica (Hartigan, 1967):*

$$m_{ij} \leq \text{máx}[m_{ik}; m_{jk}] \qquad \forall i, j, k.$$

Definición 5.7. *Se llama cardinalidad al número de elementos de un conjunto A y se denota por $|A|$ (Dauben, 1990).*

Definición 5.8. *El índice o coeficiente de Jaccard entre dos conjuntos A y B se define como la cardinalidad de la intersección de ambos conjuntos dividida por la cardinalidad de su unión (Real y Vargas, 1996):*

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{M_{11}}{M_{01}+M_{10}+M_{11}}, \qquad [14]$$

donde, M_{11} representa el número total de casos donde A y B tienen un valor de 1, M_{01} el número total de casos donde el valor de A es 0 y el valor de B es 1, M_{10} el número total de casos donde el valor de A es 1 y el valor de B es 0 y M_{00} representa el número total de casos donde A y B tienen un valor de 0.

El índice de Jaccard siempre toma valores entre 0 y 1, correspondiendo este último a la igualdad entre ambos conjuntos. Existe otra medida que se llama el coeficiente de *coincidencia simple* (SMC) (Heltshe, 1988) o *coeficiente de similitud de Rand* que es muy similar al Jaccard pues se define como:

$$SMC=\frac{M_{00}+M_{11}}{M_{01}+M_{10}+M_{11}},$$

pero en este trabajo preferimos usar el índice de Jaccard porque nos centramos en la coincidencia de una de las categorías de los conjuntos *A* y *B*.

Definición 5.9. *Se define el error absoluto medio (MAE), como* (Hyndman y Koehler, 2006):

$$MAE=\frac{\sum_{i=1}^{k}|\widehat{O}_i-O_i|}{k},$$

donde $\widehat{O}$ son los valores estimados y O_i los valores reales u observados, siendo k la cantidad de valores a estimar.

Definición 5.10. *Sea* $\mathbf{O}$ *la matriz original y* $\mathbf{P}$ *la matriz pronosticada, se define la diferencia relativa entre estas matrices como:*

$$\frac{\|\mathbf{O}-\mathbf{P}\|_2}{\|\mathbf{O}\|_2},$$

donde $\|\boldsymbol{A}\|_2=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$.

Definición 5.11. *Sea o el vector original y p el vector pronosticado, se define la diferencia relativa entre estos vectores como:*

$$\frac{\|o-p\|_2}{\|o\|_2},$$

donde $\|o\|_2=\sqrt{\sum_{i=1}^{n}o_i^2}$.