Señor… si me das humildad,
no me quites la dignidad

Capítulo 2
- Métodos de los k vecinos más cercanos -

2.1 Introducción

Una de las formas más populares de predecir o clasificar un nuevo dato, basado en observaciones conocidas o pasadas es el enfoque del vecino más cercano. Por ejemplo, el caso del médico que está tratando de predecir el resultado de un procedimiento quirúrgico, puede predecir que el resultado de la cirugía del paciente será aquel del paciente más parecido que conoce que haya sido sometido al mismo procedimiento. Esto puede resultar un tanto extremo. Un solo caso muy similar, en el cual la cirugía falló, puede influir de manera excesiva sobre otros muchos casos ligeramente menos similares, en los cuales la cirugía fue un éxito. Es por eso que el método del vecino más cercano se generaliza al uso de los k vecinos más cercanos. De esta manera, una simple elección entre los k vecinos más cercanos genera la predicción para cada caso. Más aún, se puede extender la regla de decisión a un elección ponderada, en la cual, los vecinos más cercanos al caso tendrán mas peso. Se puede definir el problema de la búsqueda del vecino más cercano de la siguiente manera:

> Dado un conjunto de puntos **P** = {p1 , . . . , p_n} en un espacio métrico **X** con función de distancia d, permitiendo algún preprocesamiento en P de manera eficiente, se desea responder a dos tipos de solicitudes:

- Vecino más cercano: localizar el punto en P más cercano a $q \in X$.
- Rango: Dado un punto $q \in X$, y $r > 0$, regresar todos los puntos $p \in P$ que satisfagan $d(p,q) \leq r$

Se buscan algoritmos de discriminación no paramétrica. Es decir, se buscan enfoques no paramétricos caracterizados por la ausencia de hipótesis a priori sobre la distribución condicional del espacio de definición. Puesto que la base está en el cálculo de distancias, esta puede ser tal vez su mayor inconveniente.

El algoritmo más sencillo para la búsqueda del vecino más cercano es el conocido como fuerza bruta o exhaustivo, que calcula todas las distancias de un individuo a los individuos de la "muestra de entrenamiento" (individuos donantes) y asigna al conjunto de vecinos más cercanos, aquel cuya distancia sea mínima. En la práctica resulta poco aconsejable. Se han ido desarrollando algoritmos eficientes que evitan recorrer exhaustivamente todo el conjunto de entrenamiento. La búsqueda del vecino más cercano es una técnica relevante en áreas como: bibliotecas digitales, bases de datos, buscadores en Internet[1], etc.

2.2 Regla de decisión y selección de la distancia

El método de decisión está relacionado con la noción de proximidad o similitud entre los individuos. El índice de similitud entre individuos más utilizado es la métrica de Minkowski:

$$d(i,j) = (\left|x_{i1} - x_{j1}\right|^q + \left|x_{i2} - x_{j2}\right|^q + ... + \left|x_{ip} - x_{jp}\right|^q)^{1/p} \qquad (2.1)$$

Con p = 2, se tiene la distancia euclídea clásica:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ip} - x_{jp})^2} \qquad (2.2)$$

Se generaliza la distancia euclídea como:

$$d^2(i,j) = (i-j)'A(i-j) \qquad (2.3)$$

en donde A es una matriz positiva definida. La distancia de Mahalanobis toma esta fórmula con la matriz A igual a la inversa de la matriz de varianza covarianza.

Otra métrica conocida es la distancia Manhattan (p = 1):

$$d(i,j) = \left|x_{i1} - x_{j1}\right| + \left|x_{i2} - x_{j2}\right| + ... + \left|x_{ip} - x_{jp}\right| \qquad (2.4)$$

[1] GNAT(Geometric Near-neighbor Access Tree), Brin (1995); fundador del buscador Google

Estas métricas satisfacen los requisitos matemáticos de una función de distancia

- $d(i, j) \geq 0$
- $d(i, i) = 0$
- $d(i, j) = d(j, i)$
- $d(i, j) \leq d(i, h) + d(h, j)$

con $p \to \infty$ se obtiene la distancia de Tchebychev d_∞ :

$$d_\infty (i, j) = \underset{1 \leq k \leq d}{MAX} |i_k - j_k| \tag{2.5}$$

Se puede utilizar un tipo de distancia ponderada para privilegiar a los vecinos más cercanos y disminuir el papel de los vecinos lejanos, considerando las ideas de proximidad y lejanía relacionadas con la distancia d.

S. A. Dudan (1976), define la siguiente regla de los k vecinos más cercanos ponderados de la siguiente manera:

Sean $(x_1^{'},...,x_k^{'})$ los k vecinos más cercanos de x ordenados crecientemente por la distancia. A cada vecino se le asigna un peso w_i:

$$w_i = \frac{d(x, x_k^{'}) - d(x, x_i^{'})}{d(x, x_k^{'}) - d(x, x_1^{'})} \quad \text{si} \quad d(x, x_k^{'}) \neq d(x, x_1^{'}) \tag{2.6}$$

$$w_i = 1 \quad \text{si} \quad d(x, x_k^{'}) = d(x, x_1^{'})$$

El peso de los donantes debe variar inversamente con la distancia, de tal manera que los puntos más cercanos tengan mayor peso.

Se tiene entonces que:

$$w_i = K\,[d\,(x_r, x_d)] \tag{2.7}$$

K es la función Kernel que determina el peso de cada punto basado en la distancia al punto de referencia.

Se pueden considerar las siguientes funciones:

$$K[d(x_d, x_r)] = \begin{cases} \dfrac{1}{d^2} \\ e^{-d} \\ \dfrac{1}{d_0 + d} \\ \dfrac{e^{-d(x_d, x_r)}}{\sum_{i=1}^{d} e^{-d(x_i, x_r)}} \end{cases} \qquad (2.8)$$

En el caso de la última función, la asignación de pesos de esta forma tiene la característica de que $\sum_i w_i = 1$

La selección del Kernel puede ser importante al momento de la imputación, por el valor (peso) que se le asignará a cada vecino.

En la siguiente figura se muestra el comportamiento del Kernel en función de la distancia. Debido a la característica que debe cumplir el Kernel (debe variar inversamente con la distancia) su comportamiento es diferente si la distancia es mayor o menor que uno.

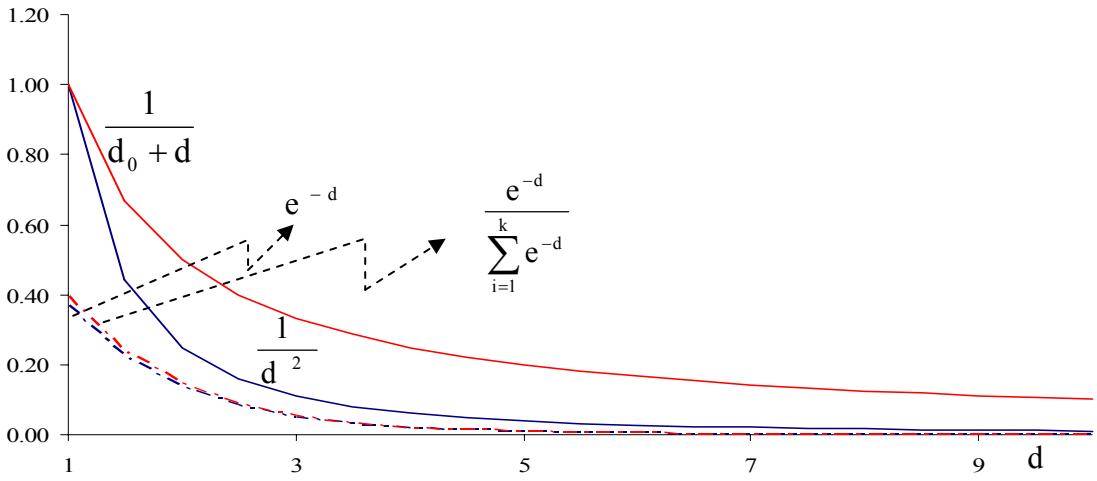

Figura 2.1 Comportamiento del Kernel para distancias > 1

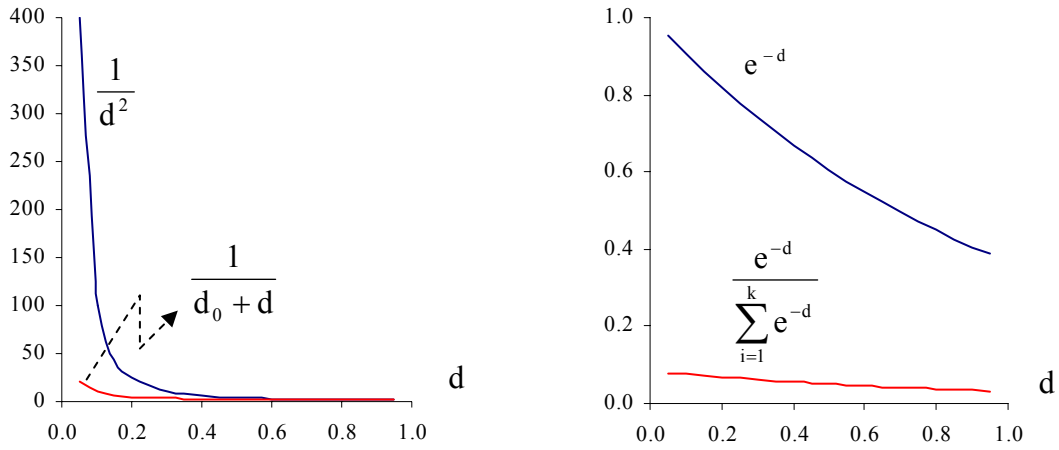

Figura 2.2 Comportamiento del Kernel para distancias < 1

Se deberá tener en cuenta la llamada "maldición de dimensionalidad (curse of dimensionality)". Es un término acuñado por el matemático R. Bellman para describir como aumenta la complejidad de un problema al aumentar la dimensión de las variables involucradas. Al aumentar la dimensión, el espacio está cada vez más vacío complicando cualquier proceso de inferencia. Al aumentar la dimensión aumenta su volumen y la densidad de la variable aleatoria disminuye.

En espacios con muchas dimensiones, el método del vecino más cercano puede presentar algunos problemas, debido a que la vecindad se hace muy grande. Por ejemplo, para el caso de 5000 puntos distribuidos uniformemente en un hipercubo unitario, se desea aplicar el método del vecino más cercano, considerando además que el punto de referencia está en el origen. En una dimensión, se recorrería en promedio una distancia de $\frac{5}{5000} = 0.001$ para tener a los 5 vecinos más cercanos.

En dos dimensiones, se recorrería en promedio una distancia de $\sqrt{0.001}$ para tener un cuadrado que tuviera 0.001 de volumen.

En d dimensiones se recorrería $0.001^{1/d}$ (media geométrica).

También se ve afectado el método, si se incluyen características irrelevantes o ruidosas en los datos. En la siguiente figura se muestran dos puntos x_1, x_2 cercanos a la referencia ubicada en el origen.

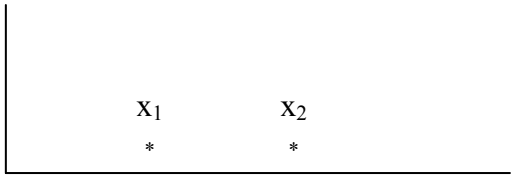

Figura 2.3 Puntos en una dimensión

En una dimensión, el punto x_1 está más cerca de la referencia que el punto x_2, sin embargo, si se le agrega una segunda característica aleatoria, los nuevos puntos más cercanos podrían estar ubicados de otra forma.

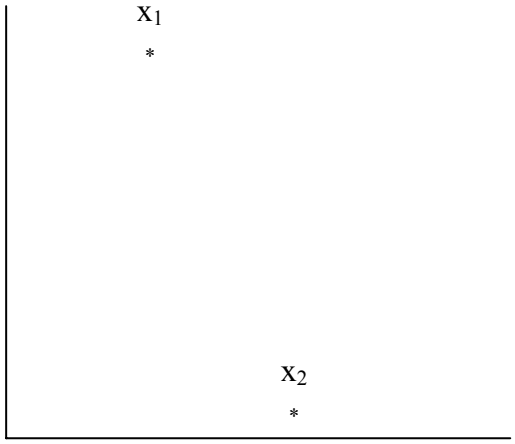

Figura 2.4 Puntos en dos dimensiones

Dado que el interés en esta investigación está en la imputación por vecinos, se revisarán algunos algoritmos de búsqueda de vecinos.

2.3 Métodos de búsqueda

Existen muchos artículos, provenientes de distintas áreas de conocimiento, que proponen algoritmos eficientes para la búsqueda del vecino más cercano[2]. Muchos de los algoritmos encajan dentro de un esquema de búsqueda conocida como esquema de aproximación y eliminación (Ramasubramanian y Paliwal, 2000), y entre estos, algunos de los más conocidos son el k-d tree (Bentley, 1975; Friedman et al., 1977), Fukunaga y Narendra (Fukunaga y Narendra, 1975), el vp-tree (Yianilos, 1993) y el GNAT (Brin, 1995).

[2] cit. en: Moreno, F. (2004)

Existen otros algoritmos que encajan en este esquema, de la familia AESA (Approximating Eliminating Search Algorithm) que resultan eficientes cuando la distancia es costosa.
El esquema general de búsqueda por aproximación y eliminación se resume como:

1. De entre los individuos del conjunto de entrenamiento, se selecciona un candidato a vecino más cercano (aproximación).

2. Se calcula su distancia d al individuo en cuestión.

3. Si la distancia es menor que la distancia del vecino más cercano hasta el momento, d_{nn}, se actualiza el vecino más cercano y se eliminan del conjunto de entrenamiento aquellos individuos que no puedan estar dentro de una hiperesfera de radio d y con centro en la muestra, es decir, se eliminan aquellos individuos que no puedan estar más cerca de la muestra que el vecino más cercano actual.

4. Se repiten los pasos anteriores hasta que no queden individuos por seleccionar en el conjunto de entrenamiento, ya sea porque han sido previamente seleccionados o porque han sido eliminados.

La mayoría de los algoritmos rápidos utilizan una estructura de datos para almacenar el conjunto de entrenamiento (generalmente un árbol).

Otra clasificación de los algoritmos rápidos de búsqueda de vecinos[3] es la siguiente:

Búsqueda rápida local

El objetivo es el de poner en evidencia una estructura sobre el conjunto de aprendizaje (Valores observados, Individuos activos, X0) disminuyendo el número de distancias a calcular.

- Método de Friedman, Baskettand y Shusted
- Método de Yunk
- Método de Kittler

[3] cit. en: Comyn, M. (1999)

Estructura del conjunto de aprendizaje

Estructura el conjunto de aprendizaje definiendo una zona restringida en donde se encontrarán los vecinos más cercanos

- Método de Fukunaga/Narendra
- Método de Delannoy

Reducción del conjunto de aprendizaje

- Método de condensación de Hart
- Método de edición de Wilson

2.3.1 K-Dimensional Tree (k-d tree)

Es un árbol utilizado para la búsqueda del vecino más cercano. K representa la dimensión de los datos del espacio de representación. Es un árbol binario que contiene en cada nodo intermedio información acerca de una coordenada que divide en dos el conjunto de datos del subárbol correspondiente al nodo, y en las hojas contiene buckets (puñados) de individuos. Para intentar que el árbol resulte lo más equilibrado posible, se elige el hiperplano de forma que se sitúe en la mediana de los valores de la coordenada discriminante. La coordenada discriminante debe ser aquella que tenga una mayor amplitud .Durante la fase de clasificación se recorre el árbol siguiendo un esquema de ramificación y poda para encontrar el vecino más cercano.

El proceso de búsqueda en el k-d tree es recursivo. Dado un individuo x, en un nodo cualquiera del árbol (que no sea una hoja) se compara la coordenada de x que es discriminante para ese nodo (c) con el valor de corte v (la mediana de las coordenadas discriminantes) y se procede en la dirección más cercana según esa coordenada. Si $x(c)+d_{nn} \leq v$ (d_{nn} es la distancia al vecino más cercano hasta el momento), el hijo derecho de ese nodo no puede contener al vecino más cercano y por tanto no es necesario buscarlo en ese nodo; de forma similar, si $x(c)-d_{nn} \geq v$ el hijo izquierdo no es necesario visitarlo. Si el nodo es una hoja, la muestra se compara con todos lo individuos contenidos en ella.

2.3.2 Algoritmo de Fukunaga/Narendra[4]

Se basa también en la construcción de un árbol a partir del conjunto de aprendizaje. Dicho conjunto se divide en *l* subconjuntos que a su vez son divididos en *l* subconjuntos hasta llegar a cierto nivel. Por lo tanto, cada hoja del árbol contiene un subconjunto de individuos. Cada nodo p del árbol tiene *l* hijos (si no es una hoja) y representa a uno de estos subconjuntos (S_p); contiene además un representante M_p, y un radio R_p definido como el máximo de las distancias de M_p a los elementos del subconjunto.
Se recomienda para la construcción de los subconjuntos la utilización de un algoritmo de agrupamiento (k-medias, k-medianas). Se explica detalladamente más adelante.

2.3.3 Vantage Point Tree (vp-tree)

Construye un árbol binario en el que cada nodo representa un subconjunto S de individuos del conjunto de aprendizaje y utiliza un elemento especial del conjunto llamado pivote (vantage point) para dividir el conjunto S en dos subconjuntos, uno por cada hijo. Cada nodo contiene dos hijos. El hijo izquierdo contiene el subárbol correspondiente al subconjunto de S que está a una distancia del pivote menor que un cierto valor μ y el hijo derecho contiene el subárbol correspondiente al resto de S. Es importante que ambos hijos representen a subconjuntos de tamaño similar, para que el árbol resulte lo más balanceado posible. En cada nodo se almacenan, junto con el pivote y el valor de μ, la cota inferior y superior de las distancias del pivote a los individuos de cada subconjunto. La búsqueda en el árbol se realiza siguiendo el esquema de ramificación y poda, utilizando las cotas almacenadas en cada nodo para dirigir la búsqueda.

[4] cf. Fukunaga, K.; Narendra, P.M. (1975).

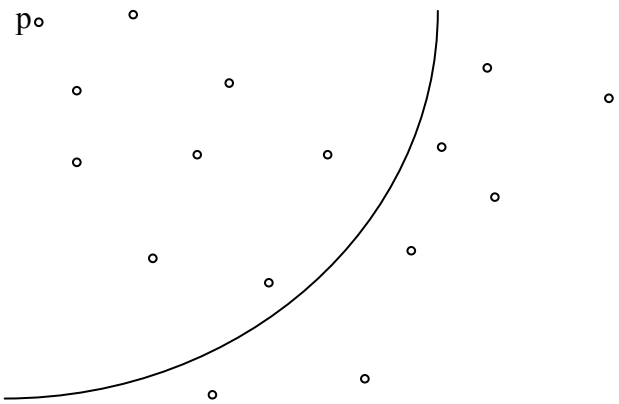

Figura 2.5 Ejemplo de partición de un conjunto

2.3.4 Geometric Near-neighbour Access Tree (GNAT)

Es un árbol que intenta estar balanceado y a la vez reflejar la estructura geométrica del conjunto de individuos. En este algoritmo el número de hijos de cada nodo es variable. Las hojas contienen únicamente un individuo. La construcción del árbol es la siguiente:

1. Se eligen k puntos de partición, p_1, p_2, …, p_k del conjunto de datos. El valor k se denomina grado y puede variar en cada nodo del árbol. Los puntos de partición se pueden elegir al azar. Se elige el primer punto al azar y el segundo como aquel que esté más lejos del primero; el tercero será aquel cuya distancia al más cercano de los anteriores sea la mayor de todos los puntos restantes y así sucesivamente hasta elegir los k puntos de partición.

2. Divide el conjunto inicial en subconjuntos D_{pi}, cada uno asociado a un punto de partición p_i. Cada individuo p estará en el subconjunto asociado al punto de partición más cercano a él.

3. Para cada par de puntos de partición (p_i , p_j), se calcula y almacena el rango de p_i al conjunto D_{pj}, rango(p_i , D_{pj}), que es un par de valores: el mínimo y el máximo de d(p_i, p) para $p \in D_{pj} \cup \{p_j\}$, denotados como min_d($p_i$, D_{pj}) y max_d(pi , D_{pj}) , respectivamente.

4. Construir recursivamente el árbol para cada Dpi, utilizando el mismo o diferente grado.

La construcción del árbol se hace en la fase de preproceso, como sucede con otros algoritmos o estructuras de datos. En este caso, esta fase es especialmente costosa en tiempo y es su principal inconveniente.
El algoritmo de búsqueda se formula de la siguiente manera:

1. Sea P el conjunto de puntos de partición de un nodo (inicialmente, de la raíz del árbol). Cada elemento de P se corresponde con un hijo del nodo en el árbol y representa un conjunto de puntos.

2. Elegir un individuo p de P (que no haya sido elegido anteriormente) y calcular su distancia al individuo x, d(x, p). Se actualiza si es necesario el vecino más cercano hasta el momento y el rango de la búsqueda, r.

3. Para todo $q \in P$, $q \neq p$, si la intersección del rango [d(x, p)-r, d(x, p)+r] con rango (p, D_q) es vacía, eliminar q de P. Esta eliminación se basa en la desigualdad triangular y se explica de la siguiente manera. Sea un individuo $y \in D_q$,
   a) Si d (y, p) < d(x, p) – r, como d(x, y) + d (y, p) ≥ d(x, p), entonces d(x, y) > r, luego y no puede ser el vecino más cercano.
   b) Si d (y, p) > d(x, p) + r, como d (y, x) + d(x, p) ≥ d (y, p), entonces d(x, y) > r, luego y no puede ser el vecino más cercano.

4. Repetir los pasos 2 y 3 hasta que todos los individuos de P se hayan elegido o eliminado. Para los $p_i \in P$ no eliminados, repetir la búsqueda recursivamente en D_{pi} si el nodo de p_i no es una hoja.

En la búsqueda del vecino más cercano el rango va cambiando (siempre es la distancia al vecino más cercano hasta el momento) y es un factor determinante en la eficacia de la búsqueda.

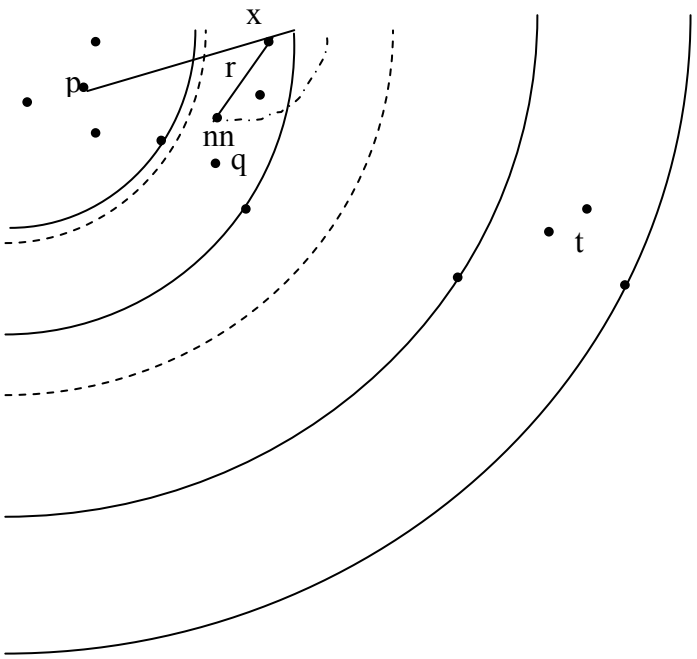

Figura 2.6 Ejemplo de eliminación en el algoritmo GNAT

Dado un individuo p y suponiendo que la distancia al vecino más cercano a x es r , se puede observar que la intersección del rango [d(x , p)-r, d(x , p)+r] (líneas punteadas) con rango (p, D_q) no es vacía,no podría eliminarse de forma segura el nodo de q. La intersección con rango (p, D_t) si resulta vacía, por lo que el nodo asociado a t puede eliminarse porque seguro no contiene al vecino más cercano.

2.4 Algoritmo de Fukunaga / Narendra

El algoritmo de Fukunaga/Narendra es un algoritmo que estructura el conjunto de aprendizaje definiendo una zona restringida en donde se encontrarán los vecinos más cercanos.

El enfoque básico consiste en descomponer jerárquicamente las muestras (entrenamiento) en subconjuntos disjuntos, y después aplicar a los grupos resultantes el método de Branch and Bound.

Se tiene:

$[X1, X_2,..., X_N]$ N muestras n-dimensionales.

Se determinan los k-vecinos más cercanos de una muestra (prueba) **X** medidos por una función de distancia apropiada.

El conjunto se divide en *l* subconjuntos, cada subconjunto es dividido a su vez en *l* subconjuntos y así sucesivamente. El resultado se representa por una estructura de árbol.

Cada nodo p del árbol representa a un grupo de muestras, y esta caracterizado por los siguientes parámetros:

- S p conjunto de muestras asociadas al nodo p
- N p número de muestras asociadas al nodo p
- M p media muestral de S p
- r p MAX d(Xi, M p) distancia más grande desde Mp a un Xi ∈ Sp

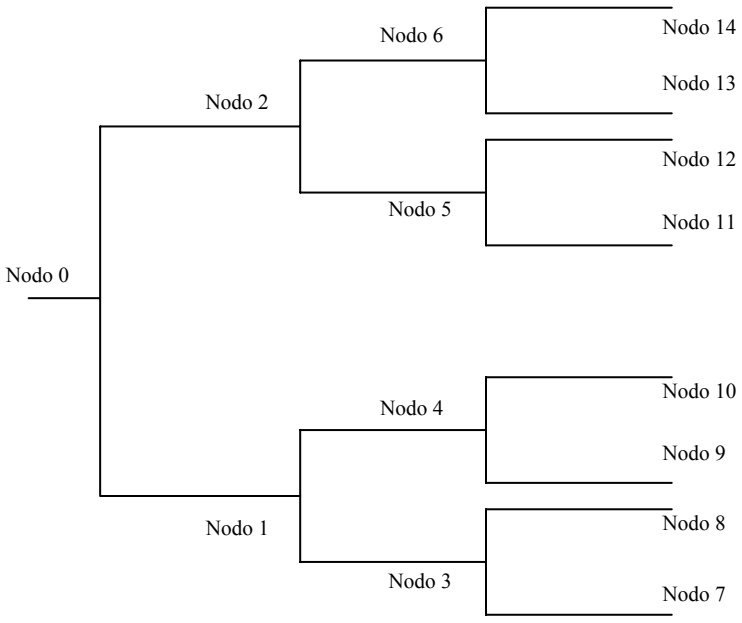

Figura 2.7 Árbol binario (l = 2)

2.4.1 Búsqueda por el método de Branch and Bound

Cada nodo p se puede probar para determinar si el vecino más cercano a X puede estar en S_p con la aplicación de la Regla 1.

Regla 1:

Ningún $X_i \in S_p$ puede ser el vecino más cercano de X si:

$$B + r_p < d(X , M_p) \tag{2.9}$$

B es la distancia al k-ésimo vecino mas cercano de X. Inicialmente $B = \infty$.

Demostración (Regla 1):

Para un $X_i \in S_p$

$$d(X , X_i) + d(X_i , M_p) \geq d(X , M_p) \qquad \text{Desigualdad triangular}$$

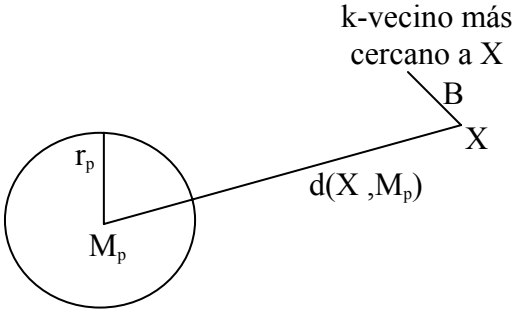

Figura 2.8 Regla 1

Puesto que $d(X_i , M_p) \leq r_p$

$$d(X , X_i) \geq d(X , M_p) - r_p$$

Por lo tanto X_i no puede ser el vecino más cercano a X si:

$$d(X , X_i) \geq d(X , Mp) - r_p > B \tag{2.10}$$

De esta manera, muchos nodos p y los correspondientes grupos de muestras S_p se pueden eliminar sin calcular explícitamente las distancias a las muestras individuales en S_p.
Para un nodo p en el nivel final del árbol, si la regla 1 no se satisface, las distancias a las muestras individuales en S_p desde X se deben calcular. Sin embargo, muchos cálculos de distancias se pueden evitar como sigue.

Regla 2:

X_i no puede se el vecino más cercano a X si:

$$B + d(X_i , Mp) < d(X , Mp) \quad X_i \in S_p \tag{2.11}$$

Demostración (Regla 2):

Para un $X_i \in S_p$

$$d(X , X_i) + d(X_i , M_p) \geq d(X , M_p) \qquad \text{Desigualdad triangular}$$

$$d(X , X_i) \geq d(X , M_p) - d(X_i , M_p)$$

Por lo tanto X_i no puede ser el vecino más cercano a X si:

$$d(X , Mp) - d(Xi , Mp) > B \tag{2.12}$$

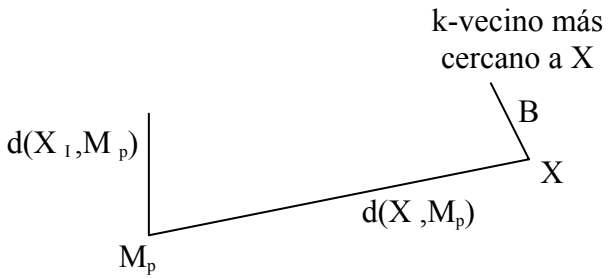

Figura 2.9 Regla 2

Se presenta a continuación en pseudo código, el proceso de búsqueda del vecino más cercano empleando el algoritmo de Fukunaga/Narendra.

```
Función buscar [Fukunaga/Narendra]
entrada          t (árbol)
                 x (individuo receptor)
entrada/salida   d_nn (distancia al vecino más cercano)
For  p = Hijo (t)
        M_p = representante de p
        R_p = radio de p
        D_p = d (M_p, x)
        Si      d(x, M_p) < d_nn
                d_nn =  d (M, x)
                nn = p
        Fin
        While queden hijos de t por visitar hacer
                p = min q=HijoNoVisitado(t) d_q
                visitado[p] = cierto
                If      d_p < d_nn + R_p        % No se puede podar
                        If      Hoja (p)
                                For individuo x_i ∈ S_p hacer
                                        If      dp ≤ d (x_i, Mp) + d_nn
                                                d_xi = d(x, x_i)
                                                If      d_xi < d_nn
                                                        d_nn = d_xi
                                                        nn = x_i
                                                Endif
                                        Endif
                                Endfor
                        Else
                                buscar (p, x, d_nn)
                        Endif
                Endif
        Endwhile
Endfor
```

La búsqueda de los k vecinos más cercanos de forma exhaustiva es costosa, pero no mucho más que la búsqueda exhaustiva del vecino más cercano, en el caso general de que el valor de k sea menor que el tamaño del conjunto de entrenamiento.
Se pueden encontrar los k vecinos más cercanos realizando los siguientes cambios en el algoritmo:

- Cuando se calcula una distancia, se almacena en un vector o lista que contiene los k vecinos más cercanos hasta el momento.

- En las condiciones de eliminación o poda, se debe utilizar la distancia al último de los k vecinos más cercanos encontrados hasta el momento.

2.4.2 Aplicación del algoritmo

Se implantó el algoritmo de Fukunaga/Narendra para la búsqueda de vecinos más cercanos. El siguiente ejemplo muestra el funcionamiento del algoritmo y los resultados que se obtienen en un reporte generado por el sistema desarrollado.

Tabla 2.1 Archivo de prueba

	Etiqueta	Tipo
1	Type de client	N
2	Age du client	N
3	Situation familiale	N
4	Ancienneté	N
5	Domiciliation du salaire	N
6	Domiciliation de l'épargne	N
7	Profession	N
8	Moyenne en cours	N
9	Moyenne des mouvements	N
10	Cumul des débits	N
11	Autorisation de découvert	N
12	Interdiction de chéquier	N
13	C 1	C
14	C 2	C

Se cuenta con 28 individuos donantes y 9 individuos receptores. Se buscarán para cada individuo receptor, tres individuos donantes (sus vecinos más cercanos). Como primer paso, se posicionaron los individuos donantes y receptores en un mismo espacio factorial.

Se desarrolló para esto un análisis de correspondencias múltiples (ACM) y se conservaron los dos primeros ejes factoriales. En este análisis de correspondencias múltiples, los individuos donantes se etiquetan como activos(A) y los individuos receptores, como ilustrativos (I). En el caso de las variables, se consideraron como variables activas: Type de client, Age du client, Ancienneté, Profession. El resto de las variables fueron etiquetadas como ilustrativas. Tanto los individuos ilustrativos como las variables ilustrativas no participan en la determinación de los ejes factoriales. El resultado de la búsqueda de los vecinos más cercano se muestra en el reporte **R1**.

Se han marcado en negrillas, los individuos receptores y a continuación los individuos donantes (los vecinos más cercanos). De manera sombreada se muestra para cada vecino más cercano (individuo su distancia al individuo receptor.

X1[18]:**I1** -> X0[8]: **A9** (0;0.000000) X0[1]:**A2** (0;0.183019) X0[5]:**A6** (0;0.268175)

En este ejemplo, para el individuo receptor **I1**, sus 3 vecinos más cercanos son: **A9**, **A2** y **A6**. Sus respectivas distancias son: 0.000000, 0.183019, 0.268175. Se puede ver en la Figura 2.10 marcada con un círculo la coincidencia del individuo **I1** con el individuo **A9**. Se ha encerrado en un cuadro en el reporte **R1** el texto **Constraints on Vars.: NO,** lo que significa que esta búsqueda de vecinos se ha realizado sin restricciones sobre las variables, tema que se tratará más adelante.

Existe información adicional relacionada con el desempeño de algoritmo, información que esta en función de la estructura del árbol generado[5] para la búsqueda de los vecinos , así como el número de distancias calculadas en los nodos internos como nodos terminales (hojas). También se presenta el tiempo de ejecución y el número de individuos donantes empleados. Como parte del sistema, se puede asignar nombre y trayectoria de almacenamiento al archivo de texto del Reporte como al archivo binario, generado para usos posteriores. En este ejemplo: C:\ Tesis \CREDIT_knn.txt y C:\ Tesis \CREDIT.knn respectivamente.

[5] En este caso, el ACM y la construcción del árbol se ha desarrollado con ayuda del paquete estadístico SPAD.

R1 Resultado de la búsqueda de vecinos (generado por el sistema)

ndic file: C:\Tesis\CMWVAL01.dic
ndon file: C:\ Tesis \CMWVAL01.don
ngus file: C:\ Tesis \CMWVAL01.gus
ngri file: C:\ Tesis \CMWVAL02.gri
k = 3
Constraints on Vars.: NO
Internal structures construction time: 0.000 s
#Terminal nodes: 18
#Donors: 18
#Receivers: 9
knn search time: 0.000 s
#usedDonnors: 14
22 distances calculated per receiver
3 distances calculated per receiver in terminal nodes
18 distances calculated per receiver in internal nodes
3 terminal nodes explored per receiver
knn binary file: C:\ Tesis \CREDIT.knn

X0[0]:**A1** -> X0[11]:	**A12** (0;0.315339)	X0[9]:A10 (0;0.504163)	X0[17]:A18 (0;0.504163)
X0[1]:**A2** -> X0[8]:	**A9** (0;0.183019)	X0[5]:A6 (0;0.347523)	X0[3]:A4 (0;0.631717)
X0[2]:**A3** -> X0[14]:	**A15** (0;0.587897)	X0[12]:A13 (0;0.720776)	X0[4]:A5 (0;1.042659)
X0[3]:**A4** -> X0[4]:	**A5** (0;0.073036)	X0[5]:A6 (0;0.320739)	X0[12]:A13 (0;0.358694)
X0[4]:**A5** -> X0[3]:	**A4** (0;0.073036)	X0[12]:A13 (0;0.347523)	X0[5]:A6 (0;0.372744)
X0[5]:**A6** -> X0[8]:	**A9** (0;0.268175)	X0[3]:A4 (0;0.320739)	X0[1]:A2 (0;0.347523)
X0[6]:**A7** -> X0[9]:	**A10** (0;0.084277)	X0[17]:A18 (0;0.084277)	X0[11]:A12 (0;0.552586)
X0[7]:**A8** -> X0[0]:	**A1** (0;0.679809)	X0[9]:A10 (0;0.749336)	X0[17]:A18 (0;0.749336)
X0[8]:**A9** -> X0[1]:	**A2** (0;0.183019)	X0[5]:A6 (0;0.268175)	X0[3]:A4 (0;0.587897)
X0[9]:**A10** -> X0[17]:	**A18** (0;0.000000)	X0[6]:A7 (0;0.084277)	X0[11]:A12 (0;0.471009)
X0[10]:**A11** -> X0[13]:	**A14** (0;0.438027)	X0[8]:A9 (0;0.988322)	X0[1]:A2 (0;1.038447)
X0[11]:**A12** -> X0[0]:	**A1** (0;0.315339)	X0[9]:A10 (0;0.471009)	X0[17]:A18 (0;0.471009)
X0[12]:**A13** -> X0[14]:	**A15** (0;0.161887)	X0[4]:A5 (0;0.347523)	X0[3]:A4 (0;0.358694)
X0[13]:**A14** -> X0[10]:	**A11** (0;0.438027)	X0[6]:A7 (0;0.847141)	X0[17]:A18 (0;0.928224)
X0[14]:**A15** -> X0[12]:	**A13** (0;0.161887)	X0[4]:A5 (0;0.509128)	X0[3]:A4 (0;0.516338)
X0[15]:**A16** -> X0[0]:	**A1** (0;0.690058)	X0[16]:A17 (0;0.720776)	X0[7]:A8 (0;0.857237)
X0[16]:**A17** -> X0[15]:	**A16** (0;0.720776)	X0[0]:A1 (0;1.140978)	X0[11]:A12 (0;1.250819)
X0[17]:**A18** -> X0[9]:	**A10** (0;0.000000)	X0[6]:A7 (0;0.084277)	X0[11]:A12 (0;0.471009)

X1[18]:**I1** -> X0[8]:	**A9** (0;0.000000)	X0[1]:A2 (0;0.183019)	X0[5]:A6 (0;0.268175)
X1[19]:**I2** -> X0[11]:	**A12** (0;0.320542)	X0[0]:A1 (0;0.635686)	X0[17]:A18 (0;0.638165)
X1[20]:**I3** -> X0[12]:	**A13** (0;0.000000)	X0[14]:A15 (0;0.161887)	X0[4]:A5 (0;0.347523)
X1[21]:**I4** -> X0[8]:	**A9** (0;0.499551)	X0[6]:A7 (0;0.598497)	X0[13]:A14 (0;0.635474)
X1[22]:**I5** -> X0[14]:	**A15** (0;0.305195)	X0[12]:A13 (0;0.434579)	X0[2]:A3 (0;0.635474)
X1[23]:**I6** -> X0[8]:	**A9** (0;0.320739)	X0[1]:A2 (0;0.363016)	X0[5]:A6 (0;0.587897)
X1[24]:**I7** -> X0[11]:	**A12** (0;0.343347)	X0[0]:A1 (0;0.635098)	X0[17]:A18 (0;0.750854)
X1[25]:**I8** -> X0[5]:	**A6** (0;0.438027)	X0[3]:A4 (0;0.500851)	X0[8]:A9 (0;0.572318)
X1[26]:**I9** -> X0[11]:	**A12** (0;0.327475)	X0[0]:A1 (0;0.385220)	X0[15]:A16 (0;0.762965)

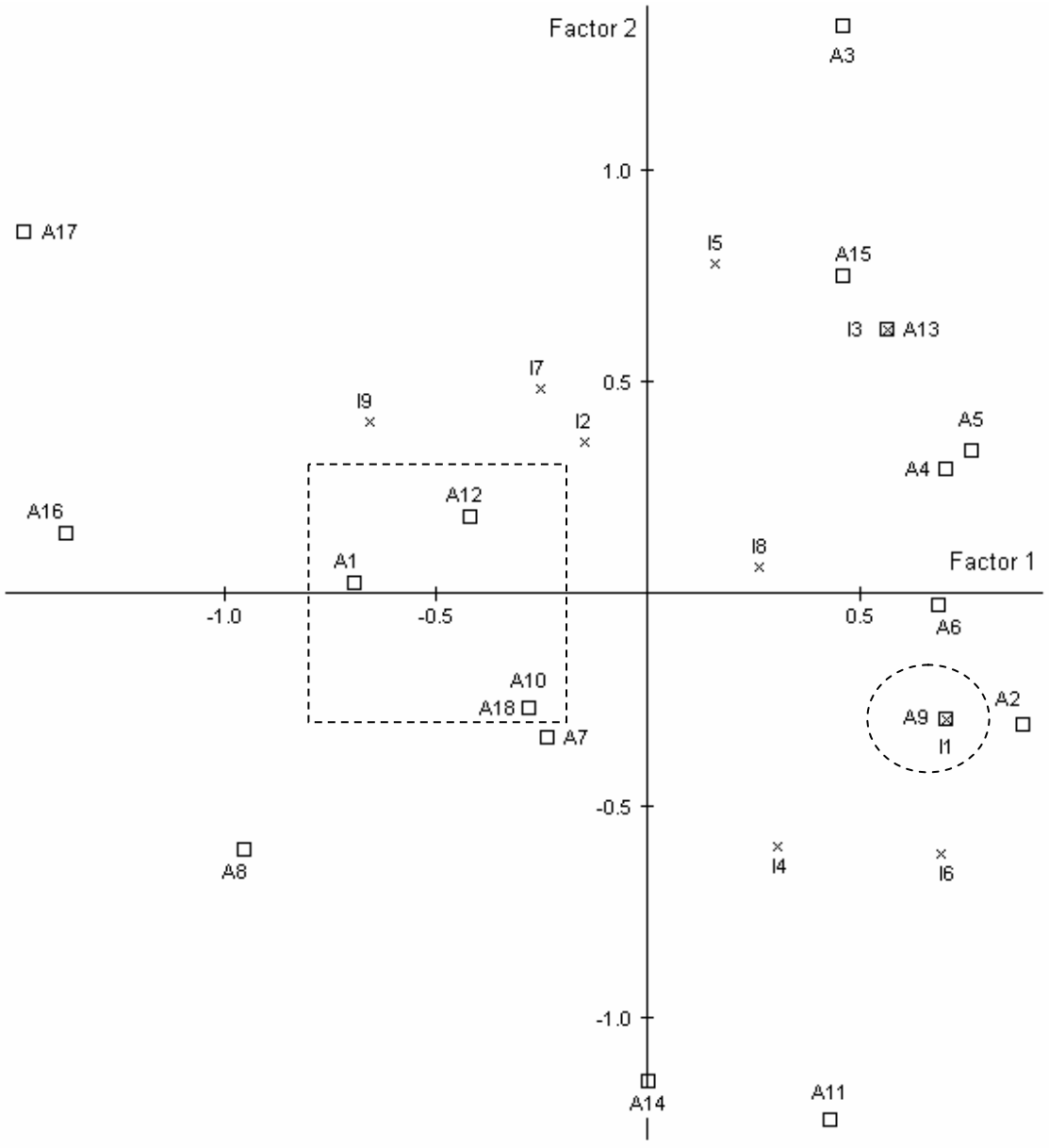

Figura 2.10 Individuos donantes (A) y receptores (I)

2.4.3 Restricciones

Otra modificación o mejora al algoritmo implantado consiste en la imposición de restricciones a la búsqueda de vecinos. Esta imposición de restricciones tiene las siguientes características:

- Permite seleccionar a los vecinos semejantes más cercanos definido por una o unas variables establecidas previamente (región, sexo, nivel de estudios etc.)

- Evita resultados incongruentes al momento de efectuar imputaciones por vecinos más cercanos con características diferentes (imputar a hombres, características de mujeres).

Esta modificación se hace antes del cálculo de las distancias. Las restricciones se imponen a las variables de la siguiente manera:

- restringidas con prioridad

- restringidas sin prioridad

Ordenadas con prioridad significa que la búsqueda de cumplimiento de las restricciones se hace en el orden en el que fueron seleccionadas las variables, y en el momento en el que alguna restricción no se cumpla se detiene la búsqueda restringida y se continua con la búsqueda normal de vecinos más cercanos. En el reporte se indica entre paréntesis el número de restricciones que se cumplieron.

X1[18]:I1 -> X0[10]: A11 (2;0.988322) X0[8]:A9 (1;0.000000) X0[12]:A13 (1;0.928259)

Ordenadas sin prioridad significa que la búsqueda del cumplimiento de las restricciones continúa en el caso de que alguna restricción no se cumpla. Si ninguna de las restricciones se cumple, la búsqueda se realiza de forma normal.

Para el ejemplo que se esta presentando, se impusieron restricciones sin prioridad sobre las variables:

- Situation familiale
- Age du client

como se muestra en el reporte **R2**.

Constraints on Vars.: 3, 2
Sorted Constraints: No

X1[18]:**I1** -> X0[10]: **A11** (2;0.988322) X0[8]:A9 (1;0.000000) X0[12]:A13 (1;0.928259)

En este ejemplo, para el individuo receptor **I1**, sus 3 vecinos más cercanos son: **A11**, **A9**y **A13**. Sus respectivas distancias son: 0.988322, 0.000000, 0.928259.
Se puede ver que solo el individuo donante A11 cumple las dos restricciones impuestas y no es necesariamente el individuo más cercano (ver Figura 2.10).

Adicionalmente se reportan los vecinos más cercanos para cada individuo donante (sin considerar a él mismo) con el fin de evaluar posteriormente el desempeño de las imputaciones (Cross-validation). Así por ejemplo tenemos que para el individuo A1 sus vecinos más cercanos son:
A12, A10 y A18 para una búsqueda de vecinos más cercanos sin restricciones marcado con un cuadro en la Figura 2.10.

X0[0]:**A1** -> X0[11]: **A12** (0;0.315339) X0[9]:A10 (0;0.504163) X0[17]:A18 (0;0.504163)

R2 Resultado de la búsqueda de vecinos con restricciones (generado por el sistema)

ndic file: C:\Tesis\CMWVAL01.dic

ndon file: C:\ Tesis \CMWVAL01.don

ngus file: C:\ Tesis \CMWVAL01.gus

ngri file: C:\ Tesis \CMWVAL02.gri

k = 3

Constraints on Vars.: 3, 2

Sorted Constraints: No

Internal structures construction time: 0.000 s

#Terminal nodes: 18

#Donors: 18

#Receivers: 9

knn search time: 0.010 s

#usedDonnors: 12

43 distances calculated per receiver

10 distances calculated per receiver in terminal nodes

32 distances calculated per receiver in internal nodes

16 terminal nodes explored per receiver

knn binary file: C:\Tesis\CREDIT.knn

X0[0]:**A1** -> X0[9]:	**A10** (2;0.504163)	X0[15]:A16 (2;0.690058)	X0[11]:A12 (1;0.315339)
X0[1]:**A2** -> X0[4]:	**A5** (2;0.661579)	X0[8]:A9 (1;0.183019)	X0[5]:A6 (1;0.347523)
X0[2]:**A3** -> X0[12]:	**A13** (1;0.720776)	X0[3]:A4 (1;1.071118)	X0[5]:A6 (1;1.381969)
X0[3]:**A4** -> X0[5]:	**A6** (1;0.320739)	X0[12]:A13 (1;0.358694)	X0[14]:A15 (1;0.516338)
X0[4]:**A5** -> X0[1]:	**A2** (2;0.661579)	X0[5]:A6 (1;0.372744)	X0[8]:A9 (1;0.635391)
X0[5]:**A6** -> X0[11]:	**A12** (2;1.127872)	X0[8]:A9 (1;0.268175)	X0[3]:A4 (1;0.320739)
X0[6]:**A7** -> X0[17]:	**A18** (2;0.084277)	X0[7]:A8 (2;0.762965)	X0[9]:A10 (1;0.084277)
X0[7]:**A8** -> X0[17]:	**A18** (2;0.749336)	X0[6]:A7 (2;0.762965)	X0[0]:A1 (1;0.679809)
X0[8]:**A9** -> X0[1]:	**A2** (1;0.183019)	X0[5]:A6 (1;0.268175)	X0[4]:A5 (1;0.635391)
X0[9]:**A10** -> X0[0]:	**A1** (2;0.504163)	X0[15]:A16 (2;1.165322)	X0[17]:A18 (1;0.000000)
X0[10]:**A11** -> X0[13]:	**A14** (1;0.438027)	X0[8]:A9 (1;0.988322)	X0[6]:A7 (1;1.123427)
X0[11]:**A12** -> X0[5]:	**A6** (2;1.127872)	X0[0]:A1 (1;0.315339)	X0[9]:A10 (1;0.471009)
X0[12]:**A13** -> X0[3]:	**A4** (1;0.358694)	X0[5]:A6 (1;0.661579)	X0[2]:A3 (1;0.720776)
X0[13]:**A14** -> X0[14]:	**A15** (2;1.955419)	X0[10]:A11 (1;0.438027)	X0[8]:A9 (1;1.109326)
X0[14]:**A15** -> X0[13]:	**A14** (2;1.955419)	X0[3]:A4 (1;0.516338)	X0[8]:A9 (1;1.071118)
X0[15]:**A16** -> X0[0]:	**A1** (2;0.690058)	X0[9]:A10 (2;1.165322)	X0[16]:A17 (1;0.720776)
X0[16]:**A17** -> X0[15]:	**A16** (1;0.720776)	X0[0]:A1 (1;1.140978)	X0[7]:A8 (1;1.550914)
X0[17]:**A18** -> X0[6]:	**A7** (2;0.084277)	X0[7]:A8 (2;0.749336)	X0[9]:A10 (1;0.000000)

X1[18]:**I1** -> X0[10]:	**A11** (2;0.988322)	X0[8]:A9 (1;0.000000)	X0[12]:A13 (1;0.928259)
X1[19]:**I2** -> X0[11]:	**A12** (2;0.320542)	X0[5]:A6 (2;0.922892)	X0[0]:A1 (1;0.635686)
X1[20]:**I3** -> X0[5]:	**A6** (2;0.661579)	X0[11]:A12 (2;1.078098)	X0[12]:A13 (1;0.000000)
X1[21]:**I4** -> X0[10]:	**A11** (2;0.661579)	X0[8]:A9 (1;0.499551)	X0[6]:A7 (1;0.598497)
X1[22]:**I5** -> X0[12]:	**A13** (2;0.434579)	X0[2]:A3 (1;0.635474)	X0[3]:A4 (1;0.731436)
X1[23]:**I6** -> X0[10]:	**A11** (2;0.679809)	X0[8]:A9 (1;0.320739)	X0[13]:A14 (1;0.871697)
X1[24]:**I7** -> X0[8]:	**A9** (2;1.233876)	X0[11]:A12 (1;0.343347)	X0[0]:A1 (1;0.635098)
X1[25]:**I8** -> X0[12]:	**A13** (2;0.635474)	X0[5]:A6 (1;0.438027)	X0[3]:A4 (1;0.500851)
X1[26]:**I9** -> X0[17]:	**A18** (2;0.773123)	X0[6]:A7 (2;0.857237)	X0[7]:A8 (2;1.054096)

2.4.4.Costo

El principal problema con el método de los k vecinos más cercanos es que el costo computacional en el proceso de búsqueda es proporcional con el tamaño del conjunto de entrenamiento (individuos donantes).

Para un conjunto $P = \{p_1, p_2, \ldots, p_n\} \subset \mathbf{R}^d$, el algoritmo ingenuo (naive) para calcular el vecino más cercano de un individuo q, consiste en verificar todos los puntos del conjunto P y regresar el más cercano. Tiene una complejidad $O(dn)$. Un algoritmo sofisticado[6] tiene una complejidad $O(\exp(d)\log n)$ con $\exp(d)$ una función que crece tan rápido como 2^d. Cuando $d \geq \log n$ resulta mejor opción el método de la fuerza bruta. El costo del algoritmo empleado es de $n\log_2(n)$.

Se ha construido un archivo con 25500 individuos y 30 variables para evaluar el comportamiento del algoritmo empleado. Las gráficas empleadas para este propósito son: Distancias calculadas por receptor y tiempo de búsqueda por receptor en función de:

- Nodos terminales
- Ejes factoriales
- Número de vecinos más cercanos
- Número de donantes

[6] Cazals, F. (1997)

Búsqueda del vecino más cercano, empleando:

- 25000 donantes
- 500 receptores
- 5 ejes factoriales

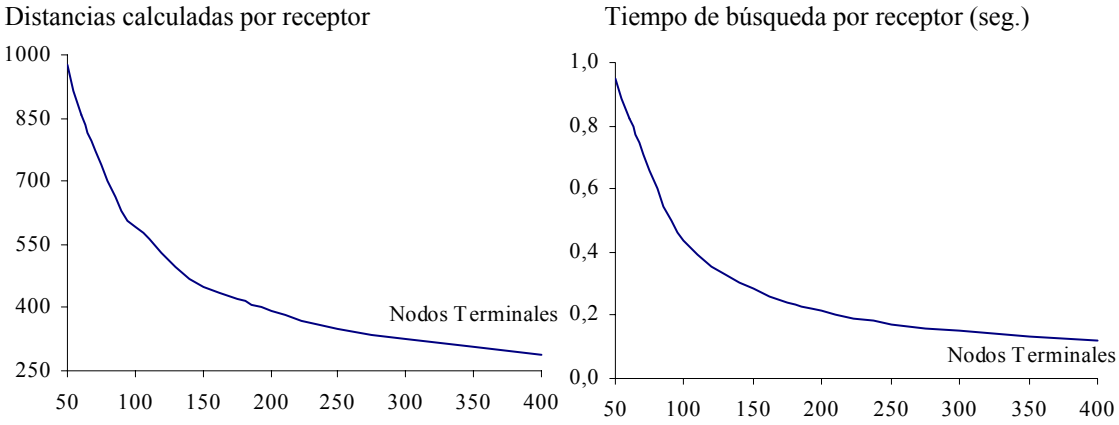

Figura 2.11 Comportamiento en función del número de nodos terminales

Búsqueda del vecino más cercano, empleando:

- 25000 donantes
- 500 receptores
- 250 nodos terminales

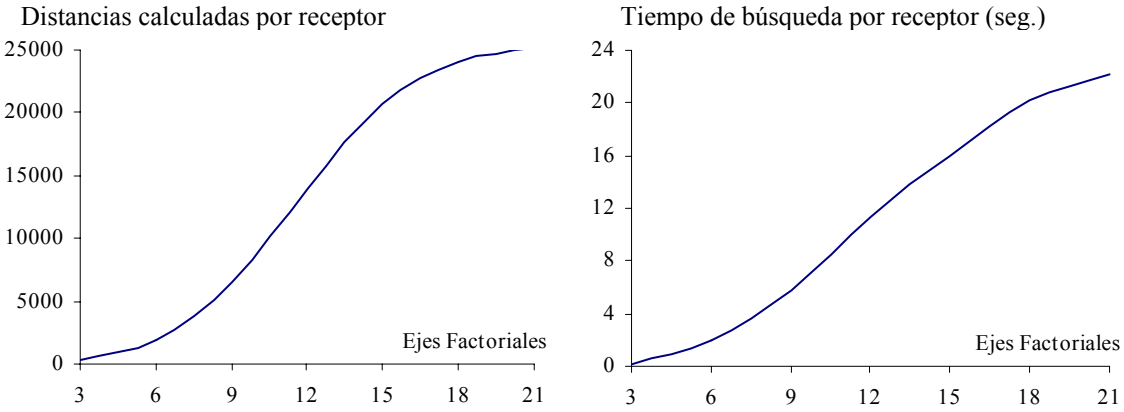

Figura 2.12 Comportamiento en función de ejes factoriales

Los tiempos de búsqueda están basados en el desempeño del algoritmo sobre un ordenar portátil con procesador Pentium III, memoria RAM de 512 y sistema operativo XP.

Búsqueda de los k vecinos más cercanos, empleando:

- 10000 donantes
- 500 receptores
- 250 nodos terminales
- 4 ejes factoriales

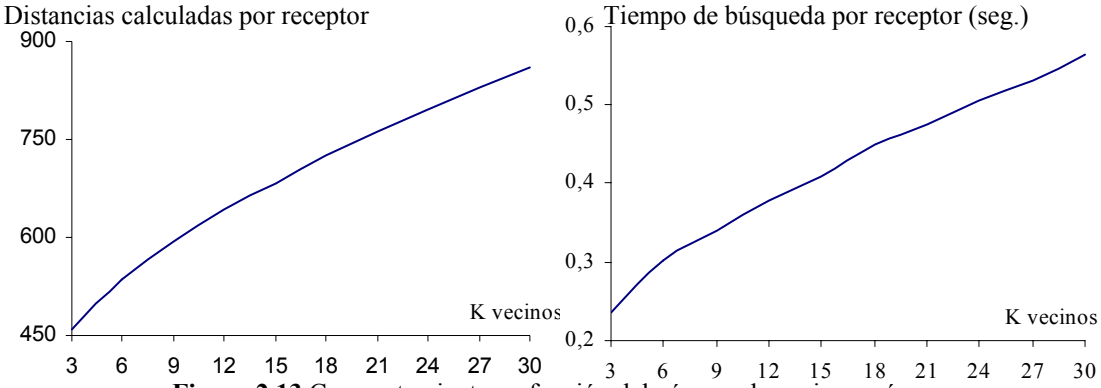

Figura 2.13 Comportamiento en función del número de vecinos más cercanos

Búsqueda del vecino más cercano, empleando:

- 500 receptores
- 250 nodos terminales
- 4 ejes factoriales

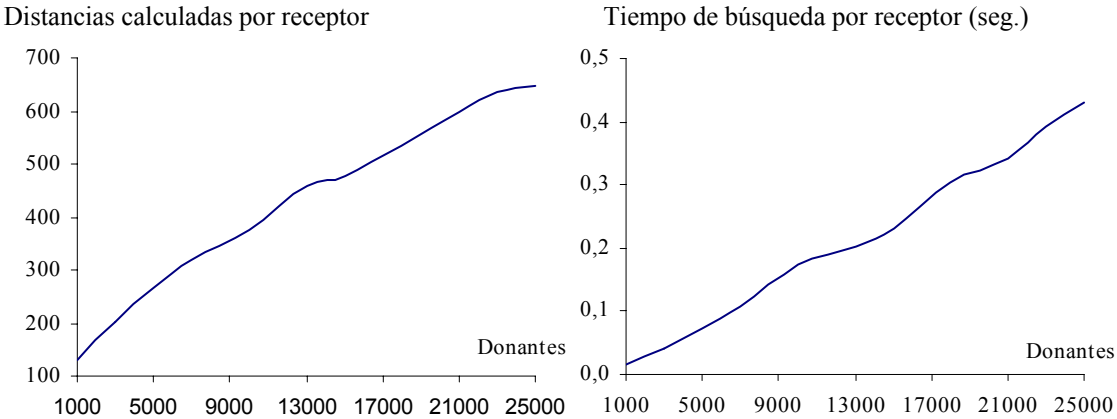

Figura 2.14 Comportamiento en función del número de donantes

En las gráficas 2.13 y 2.14 se puede observar un comportamiento escalable del algoritmo de busqueda, respecto del número de donantes y número de vecinos más cercanos.