ALGORITMOS DE APRENDIZAJE: KNN & KMEANS

[Inteligencia en Redes de Telecomuncicación]

Cristina García Cambronero
Universidad Carlos III de Madrid
10003897@alumnos.uc3m.es

Irene Gómez Moreno
Universidad Carlos III de Madrid
100039000@alumnos.uc3m.es

En el siguiente trabajo, vamos a tratar el tema del aprendizaje. En primer lugar definiremos este concepto y veremos los tipos de aprendizaje que existen y las clasificaciones que dentro de este se pueden realizar. A continuación analizaremos mas profundamente el aprendizaje inductivo, explicando mediante conceptos teóricos y ejemplos las características fundamentales del aprendizaje inductivo supervisado (K-NN) y aprendizaje inductivo no supervisado (K-MEANS).

1. INTRODUCCION

Una de las tareas más desafiantes en la ciencia de la computación es construir máquinas o programas de computadoras que sean capaces de aprender. El darles la capacidad de aprendizaje a las máquinas abre una amplia gama de nuevas aplicaciones. El entender también como estas pueden aprender nos puede ayudar a entender las capacidades y limitaciones humanas de aprendizaje.

Algunas definiciones de 'aprendizaje' son:

Cambios adaptivos en el sistema para hacer la misma tarea de la misma población de una manera más eficiente y efectiva la próxima vez [Simon, 83].

Un programa de computadora se dice que aprende de experiencia E con respecto a una clase de tareas T y medida de desempeño D, si su desempeño en las tareas en T, medidas con D, mejoran con experiencia E [Mitchell, 97].

En general, se busca construir programas que mejoren automáticamente con la experiencia.

El aprendizaje no sólo se encarga de obtener el conocimiento, sino también la forma en que éste se representa. A continuación se definen tres conceptos básicos bajo este contexto:

1.1 Conceptos básicos

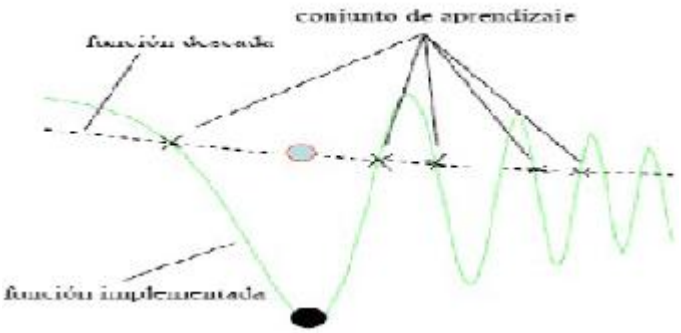

Figure 1:

1. ***Conjunto de datos:*** *Se distinguen dos tipos, el conjunto de entrenamiento y el conjunto de prueba. Para obtener estos, dividimos los datos muestrales en dos partes; una parte se utiliza como conjunto de entrenamiento para determinar los parámetros del clasificador y la otra parte, llamada conjunto de prueba (ó test ó conjunto de generalización) se utiliza para estimar el error de generalización ya que el objetivo final es que el clasificador consiga un error de generalización pequeño evitando el sobreajuste (ó sobre-entrenamiento), que consiste en una sobrevaloración de la capacidad predictiva de los modelos obtenidos: en esencia, no tiene sentido evaluar la calidad del modelo sobre los datos que han servido para construirlo ya que esta práctica nos lleva a ser demasiado optimistas acerca de su calidad.*

   La pérdida de la capacidad de generalización conlleva un comportamiento no deseado (Ver Figura 1).

   El conjunto de entrenamiento suele a su vez dividirse en conjuntos de entrenamiento (propiamente dicho) y conjunto de validación para ajustar el modelo (Ver Figura 2).

   Se suelen utilizar el 80 % de los datos para entrenar a la máquina, el 10 % como conjunto de validación y el 10 % restante para estimar la generalización (pero es sólo un criterio orientativo).

2. ***Modelo:*** *o clasificador, es una conexión entre las variables que son dadas y las que se van a predecir. Usualmente las variables que se van a predecir denominadas*

Figure 2:

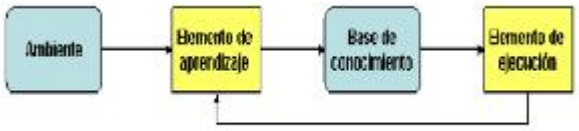

Figure 3: Modelo de Aprendizaje

variables dependientes y las restantes, variables independientes.

3. ***Aprendiz:*** *(en ingles "learner"): es cualquier procedimiento utilizado para construir un modelo a partir del conjunto de datos de entrenamiento.*

   Desde el punto de vista técnico, el aprendizaje se define como el proceso mediante el cual un sistema mejora y adquiere destreza en la ejecución de sus tareas, y tiene la capacidad de poseer inferencia inductiva sobre éstas. Un modelo de aprendizaje puede caracterizarse por dos cuerpos de información: ambiente y base de conocimiento, y por dos procedimientos: elemento de aprendizaje (aprendiz) y elemento de ejecución (programa de computación), tal y como se representa en la siguiente figura:

2. CLASIFICACIÓN

Existen diversas tareas que se pueden hacer con sistemas de aprendizaje. Entre ellas podemos en general clasificarlas como sigue:

Descripción: *normalmente es usada como análisis preliminar de los datos (resumen, características de los datos, casos extremos, etc.). Con esto, el usuario se sensibiliza con los datos y su estructura. Busca derivar descripciones concisas de características de los datos (e.g., medias, desviaciones estandares, etc.).*

La Predicción *la podemos dividir en dos:* ***Clasicación y Estimación.***

Clasificación: *Los datos son objetos caracterizados por atributos que pertenecen a diferentes clases (etiquetas discretas). La meta es inducir un modelo para poder predecir una clase dados los valores de los atributos. Se usan por ejemplo, arboles de decisión, reglas, análisis de discriminantes, etc.*

Estimación o Regresión: *las clases son continuas. La meta es inducir un modelo para poder predecir el valor de la clase dados los valores de los atributos. Se usan por ejemplo, arboles de regresión, regresión lineal, redes neuronales, kNN, etc.*

Segmentación: *separación de los datos en subgrupos o clases interesantes. Las clases pueden ser exhaustivas y mutuamente exclusivas o jerarquicas y con traslapes. Se puede utilizar con otras tecnicas de minera de datos: considerar cada subgrupo de datos por separado, etiquetarlos y utilizar un algoritmo de clasificacion. Se usan algoritmos de clustering, SOM (self-organization maps), EM(expectation maximization), k-means, etc.*

Normalmente el usuario tiene una buena capacidad de formar las clases y se han desarrollado herramientas visuales interactivas para ayudar al usuario.

Análisis de dependencias: *El valor de un elemento puede usarse para predecir el valor de otro. La dependencia puede ser probabilística, puede denir una red de dependencias o puede ser funcional (leyes fsicas). Tambien se ha enfocado a encontrar si existe una alta proporcion de valores de algunos atributos que ocurren con cierta medida de confianza junto con valores de otros atributos. Se pueden utilizar redes bayesianas, redes causales, y reglas de asociacion.*

Detección de desviaciones, casos extremos o anomalías: *Detectar los cambios mas signicativos en los datos con respecto a valores pasados o normales. Sirve para filtrar grandes volúmenes de datos que son menos probables de ser interesantes. El problema esta en determinar cuando una desviación es signicativa para ser de interes.*

Aprendizaje de cual es la mejor acción a tomar a partir de experiencia: *Esto involucra búsqueda y exploracion del ambiente. Esto esta relacionado principalmente con aprendizaje por refuerzo, pero tambit'rn con técnicas como aprendizaje de macro-operadores, chunking y EBL.*

Optimización y búsqueda: *Existen una gran cantidad de algoritmos de búsqueda tanto determinística como aleatoria, individual como poblacional, local como global, que se utilizan principalmente para resolver algún problema de optimización. Aquí podemos incluir a los algoritmos genéticos, recocido simulado, ant-colony, tecnicas de busqueda local, etc.*

3. TIPOS DE APRENDIZAJE

Aprendizaje inductivo: *Creamos modelos de conceptos a partir de generalizar ejemplos simples. Buscamos patrones comunes que expliquen los ejemplos. Se basa en el razonamiento inductivo:*

Obtiene conclusiones generales de información específica. El conocimiento obtenido es nuevo. No preserva la verdad (nuevo conocimiento puede invalidar lo obtenido). No tiene una base teórica bien fundamentada.

Aprendizaje analítico o deductivo: *Aplicamos la deducción para obtener descripciones generales a partir de un ejemplo de concepto y su explicación. Se basa en el razonamiento deductivo:*

Obtiene conocimiento mediante el uso de mecanismos bien establecidos. Este conocimiento no es nuevo (ya está presente implícitamente). Nuevo conocimiento no invalida el ya obtenido. Se fundamenta en la lógica matemática.

Aprendizaje analógico: *Buscamos soluciones a problemas nuevos basándonos en encontrar similaridades con problemas ya conocidos y adaptando sus soluciones.*

Aprendizaje genético: *Aplica algoritmos inspirados en la teoría de la evolución para encontrar descripciones generales a conjuntos de ejemplos.*

Aprendizaje conexionista: *Busca descripciones generales mediante el uso de la capacidad de adaptación de redes de neuronas artificiales*

A continuación estudiaremos el aprendizaje inductivo:

4. APRENDIZAJE INDUCTIVO

El aprendizaje inductivo es la capacidad de obtener nuevos conceptos, más generales, a partir de ejemplos. Este tipo de aprendizaje conlleva un proceso de generalización/especialización sobre el conjunto de ejemplos de entrada. Los algoritmos implementados son, además , incrementales, es decir, el procesamiento de los ejemplos se realiza uno a uno.

Esta característica, permite visualizar el efecto causado por cada uno de los ejemplos de entrada, en el proceso de obtención del concepto final. Además de la generalización de conceptos, el programa permite clasificar conjuntos de ejemplos a partir de los conceptos obtenidos anteriormente. De este modo, se puede comprobar, para cada ejemplo de un conjunto dado, a qué clase pertenece dicho ejemplo.

De forma más simple podemos ver el aprendizaje inductivo como el proceso de aprender una función. Un ejemplo es un par (x; f(x)), donde x es la entrada y f(x) la salida. El proceso de inferencia inductiva pura (o inducción) es: dada una colección de ejemplos de f, regresar una función h tal que se aproxime a f. A la función h se le llama la hipótesis o modelo.

Este campo lo podemos dividir en dos grandes grupos: los algoritmos supervisados y los no supervisados. Mientras que los algoritmos no supervisados consisten en encontrar la partición más adecuada del conjunto de entrada a partir de similitudes entre sus ejemplos, los algoritmos supervisados intentan extraer aquellas propiedades que permiten discriminar mejor la clase de cada ejemplo, y como consecuencia requieren de una clasificación previa (supervisión) del conjunto de entrenamiento. En este caso, los ejemplos que forman el conjunto de entrenamiento normalmente se componen por pares del estilo <objeto de entrada, clase del objeto>, donde el objeto de entrada suele estar representado por un vector de atributos (o propiedades del objeto). La misión de los algoritmos de aprendizaje automático supervisados es por tanto encontrar el conjunto de atributos que permite predecir con mayor precisión la clase de cada objeto del conjunto de entrenamiento.

Aprendizaje Inductivo Supervisado *Para cada ejemplo se indica a que concepto pertenece.*

El aprendizaje se realiza por contraste entre conceptos (£ Qué características distinguen a los ejemplos de un concepto de otros?).

Un conjunto de heurísticas (Función de preferencia que guía en el proceso) permitirán generar diferentes hipótesis.

Existirá un criterio de preferencia (sesgo) que permitirá escoger la hipótesis mas adecuada a los ejemplos.

Resultado: El concepto o conceptos que mejor describen a los ejemplos.

Aprendizaje Inductivo no Supervisado

No existe una clasificación de los ejemplos.

Se busca descubrir la manera mas adecuada de particionar los ejemplos (buscamos su estructura).

El aprendizaje se guía por la similaridad/disimilaridad entre ejemplos.

Existirán criterios heurísticos de preferencia que guíen la búsqueda.

Resultado: Una partición de los ejemplos y una descripción de la partición.

4.1 Aprendizaje Inductivo Supervisado

4.1.1 KNN

Características generales

Las reglas de clasificación por vecindad están basadas en la búsqueda en un conjunto de prototipos de los k prototipos más cercanos al patrón a clasificar.

No hay un modelo global asociado a los conceptos a aprender.

Las predicciones se realizan basándose en los ejemplos mas parecidos al que hay que predecir.

El coste del aprendizaje es 0, todo el coste pasa al cálculo de la predicción.

Se conoce como mecanismo de aprendizaje perezoso (lazy learning).

Debemos especificar una métrica para poder medir la proximidad. Suele utilizarse por razones computacionales la distancia Euclídea, δ para este fín.

Denominaremos conjunto de referencia (y lo notaremos por R) al conjunto de prototipos sobre el que se buscará el(los) vecino(s) más cercano(s).

4.1.2 Regla 1-NN

La regla de clasificación por vecindad más simple es la regla de clasificación del vecino más cercano o simplemente 1-NN. Se basa en la suposición de que la clase del patrón a etiquetar, X, es la del prototipo más cercano en R, (conjunto de referencia) al que notaremos por X_{NN}. Esta regla puede expresarse como:

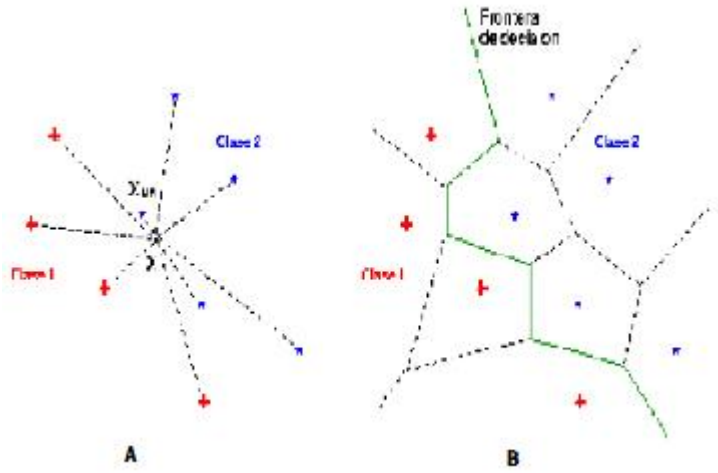

Figure 4:

$$d(X) = w_c si \delta(X, X_{NN}) = min_{i=1-N} \delta(X, X_i)$$

En la figura A (mirar Figura 4), mostramos cómo se clasificaría el patrón X con la regla 1NN para un problema de clasificación de dos clases. Existen cuatro prototipos de clase 1 (representados por cruces) y cinco prototipos de clase dos (representados por asteriscos). El prototipo más cercano es de clase 2, por lo que ésta será la clase asociada a X.

En la figura B mostramos las regiones de Voroni. A cada una de las N regiones con forma poligonal (conocido como partición de Voroni) asociada a cada prototipo, donde los bordes corresponden a los puntos situados a igual distancia entre prototipos.

Cada región de Voronoi puede considerarse como una región de decisión (restringida al prototipo central), por lo que la región de decisión de una clase será la unión de todas las regiones de Voronoi de los prototipos de esa clase. La consecuencia es que las fronteras de decisión serán fronteras lineales a trozos.

4.1.3 Regla k-NN

La regla de clasificación por vecindad más general es la regla de clasificación de los k vecinos más cercanos o simplemente k-NN. Se basa en la suposición de que los prototipos más cercanos tienen una probabilidad a posteriori similar.

Si $K_i(X)$ es el número de muestras de la clase presentes en los k vecinos más próximos a X, esta regla puede expresarse como:

$$d(X) = w_c si K_c(X) = max_{i=1->J} K_i(K)$$

En la Figura 5, mostramos cómo se realizaría la clasificación 3-NN del mismo patrón que se utilizó como ejemplo de clasificación 1-NN en la figura en el ejemplo anterior. En este caso, $K_1(X) = 1$ y $K_2(X) = 2$ por lo que X se etiquetará como de clase 2.

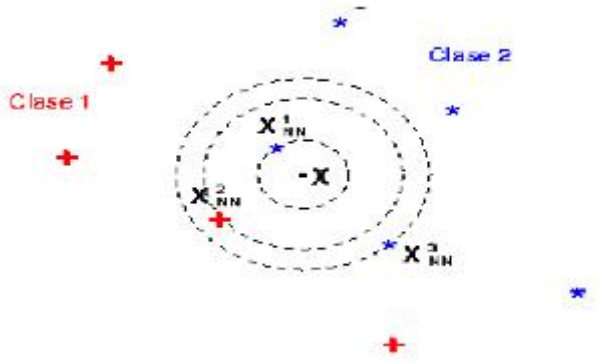

Figure 5:

4.2 Algoritmo

Tenemos tres conjuntos de datos, el conjunto de entrenamiento que utilizaremos para el aprendizaje del clasificador y los conjuntos de validación y de test con los que comprobaremos si el clasificador es capaz de generalizar, es decir si presenta buenos resultados al introducir datos no empleados durante el entrenamiento.

A continuación describiremos los pasos seguidos para la implementación de este algoritmo:

El proceso de aprendizaje de este clasificador consiste en almacenar en un vector el conjunto de entrenamiento, junto a la clase asociada a cada muestra de este conjunto.

En primer lugar, y con motivo del aprendizaje del algoritmo, calcularemos la distancia euclidea de cada muestra de entrenamiento, a todas las demás que tenemos almacenadas en el vector del punto anterior y de las que conocemos la clase a la que corresponden, quedándonos con las K muestras mas cercanas y clasificando la nueva muestra de entrenamiento en la clase más frecuente a la que pertenecen los K vecinos obtenidos anteriormente.

La segunda tarea para diseñar el clasificador, es realizar el mismo proceso con los datos de validación Se calcula el porcentaje de clasificación sobre los ejemplos de este conjunto (desconocidos en la tarea de aprendizaje) para conocer su poder de generalización.

4.3 Elección de K

Supongamos un espacio de representación bidimensional y una serie de prototipos de una misma clase representados en él. Dado un patrón cualquiera X, si consideramos los k prototipos más próximos a X, éstos estarán localizados en un círculo centrado en X. En las figuras 7 y 8 que mostramos en la siguiente página resaltamos los 7 vecinos más cercanos a tres patrones.

Observamos que el área del círculo que encierra un número fijo de puntos, k, es menor en regiones densamente pobladas que en regiones donde los puntos están más dispersos. Este sencillo planteamiento es la base de la estimación mediante los k vecinos más próximos. En espacios multidimensionales, el círculo se convierte en una hiperesfera,

```
function T_KNN = KNN(X_tr1, T_tr1,X,k)

% Inicializamos los vectores que vamos a utilizar.
distancias = 100*ones(k,1);
k_etiquetas = zeros(k,1);
T_KNN = zeros(length(X),1);

for j = 1:length(X)
    muestra = X(j,:);
    distancias = sqrt(sum((repmat(muestra,length(X_tr1),1) - X
    % Esto me lo ordena de menor a mayor.
    [valor,pos]=sort(distancias,'ascend');
    % Me quedo con las k muestras mas pequeñas!!
    k_etiquetas = T_tr1(pos(1:k));
    % Decimos de que clase es, viendo la clase de sus vecinos.
    numneg = sum(k_etiquetas == 1);
    numpos = k - numneg;
    if numpos > numneg
      T_KNN(j) = 1;
    else
      T_KNN(j) = -1;
    end
    % Vuelvo a inicializar todo!!
    distancias = 100*ones(k,1);
    k_etiquetas = zeros(k,1);
end
```

Figure 6: pseudocódigo del algoritmo KNN

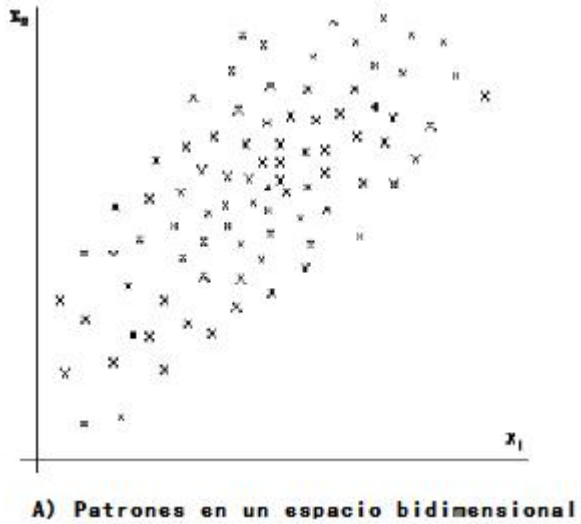

Figure 7:

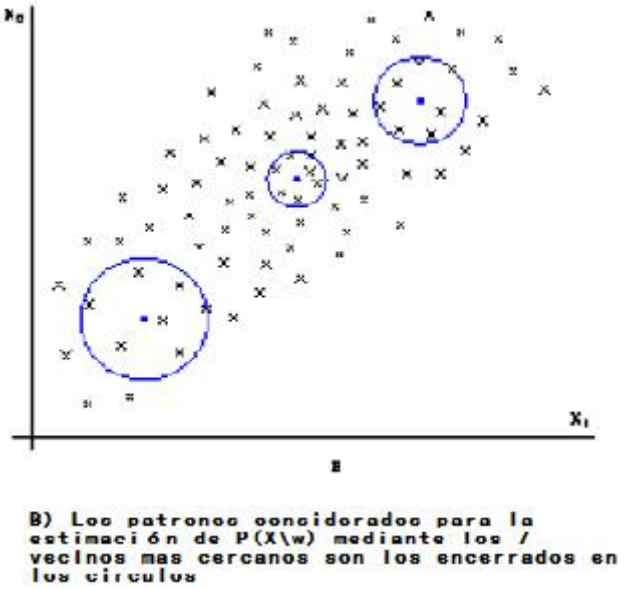

Figure 8:

Si $K_i(X)$ $i = 1, 2,..., J$ es el número de prototipos de clase w_i que se encuentran en una vecindad de X, el valor de

$$\widehat{p}(X|w_i)$$

puede calcularse como la proporción de los N_i prototipos de la clase que se encuentran en esa vecindad:

$$\widehat{p} = \frac{K_i(X)}{N_i}$$

Y si tenemos en cuenta que:

La región muy poblada, la vecindad a considerar tendrá un radio menor. La región poco poblada, la vecindad a considerar tendrá un radio mayor.

Debemos de modificar la ecuación anterior de manera que se pondere inversamente con el área de la región considerada. Por tanto obtenemos:

$$\widehat{p} = \frac{K_i(X)}{N_i V(X)}$$

El problema que se plantea ahora es el de si existe un valor óptimo para k o en su defecto si se pueden dar algunas reglas para fijar este valor. Observamos que la elección de k está determinado por la densidad de los puntos y debería hacerse en relación a Ni.

Puede demostrarse que si el estimador (el anterior) es insesgado y consistente si se verifican las siguientes condiciones sobre k:

$\lim_{N_i \to \infty} k(N_i) = \infty$

$\lim_{N_i \to \infty} \frac{k(N_i)}{N_i} = \infty$

Así, una elección adecuada de $k(N_i)$ es: $N_i = cte\ \sqrt{N_i}$

4.4 Consideraciones Computacionales

La aplicación de las reglas de clasificación por vecindad requiere, en la práctica, el conocimiento de ciertas restricciones de convergencia y del coste computacional asociado a su aplicación.

Antes de nada hay que tener en cuenta que el coste computacional asociado a las reglas 1-NN y k-NN es equiparable. La diferencia estriba en que en la búsqueda de los k vecinos, en todo momento se debe mantener una estructura auxiliar que mantenga ordenados los k vecinos más cercanos encontrados hasta ese punto.

Así, sea cual sea el valor de k la búsqueda requiere explorar todo el conjunto de referencia. Esto significa que el coste de la búsqueda depende linealmente de N. Ahora debemos considerar el coste del cálculo de cada distancia. El cálculo de la distancia Euclídea tiene un coste lineal respecto a d (coste global: O(Nd)) mientras que el coste de una distancia de Mahalanobis es cuadrático respecto a d (coste global: O(Nd2)). Adicionalmente debemos considerar el espacio de almacenamiento requerido: deben consultarse todos los prototipos, por lo que los requerimientos de espacio son del orden O(Nd).

Si consideramos que la optimalidad está garantizada cuando el conjunto de referencia es lo suficientemente grande, la aplicación de las reglas k-NN en su formulación original (fuerza bruta) resulta, en la práctica, inaplicable si se dispone de conjuntos de referencia numerosos y con datos de alta dimensionalidad.

De todo lo anterior se deduce que dos son los factores que determinan el coste computacional de las reglas k-NN:

1. La dimensionalidad de los datos, d.

2. El tamaño del conjunto de referencia, N.

Existen diferentes estrategias que permiten la aplicación de la regla del vecino más cercano para conjuntos de referencia numerosos con un coste computacional menor que el asociado a la fuerza bruta. Estas estrategias pueden agruparse en dos clases, de acuerdo a la filosofía en que se basan:

***1.Reducir el conjunto de referencia:** Se basan en la selección de un subconjunto del conjunto de referencia que tenga las mismas propiedades que el conjunto original para, una vez reducido, aplicar la regla del vecino más cercano en su formulación original (fuerza bruta) sobre el nuevo conjunto.*

Si M es el número de prototipos del conjunto reducido, el orden de complejidad sigue siendo lineal respecto a M, con la salvedad de que ahora $M < N$ (en muchos casos, y para ser más precisos, $M << N$). No obstante, la selección de un conjunto de referencia reducido que refleje todas las características del conjunto general no está garantizado.

***2. Mejorar la eficiencia computacional de la búsqueda del vecino más cercano:** Estos métodos trabajan con el conjunto completo de referencia y su objetivo es reducir el número de cálculos, eliminando aquellos que se consideran "inútiles" o "innecesarios". En estos casos pueden conseguirse órdenes de complejidad del orden de O(log N) en el mejor de los casos. La utilización de estos métodos no representa ningún peligro de pérdidad de generalidad ya que están orientados, únicamente, a reducir el número de cálculos.*

Entre estos métodos están los basados en ordenaciones y jerárquicos. Los métodos basados en ordenaciones organizan los patrones del conjunto de referencia de acuerdo a alguna coordenada, y la búsqueda se realiza de acuerdo a alguna propiedad de la métrica asociada al espacio.

Los métodos jerárquicos realizan una descomposición jerárquica del conjunto de referencia que implica la organización de los patrones en una estructura de tipo árbol, para después aplicar una técnica de poda y ramificación (branch and bound) para explorar el árbol.

No obstante, hay que considerar que todos estos métodos tienen asociado un coste adicional a la búsqueda: el coste del preprocesamiento, entendido éste como el coste asociado a la selección del conjunto reducido o la construcción del árbol de búsqueda. En consecuencia, para determinadas aplicaciones se ha de considerar que el coste de preprocesamiento puede ser alto y que la elección de estos métodos como alternativa a la fuerza bruta debe considerar, necesariamente, este coste y llegar a un compromiso entre los costes de procesamiento y búsqueda.

4.5 Ventajas

El coste del aprendizaje es nulo.

No necesitamos hacer ninguna suposición sobre los conceptos a aprender.

Podemos aprender conceptos complejos usando funciones sencillas como aproximaciones locales.

Podemos extender el mecanismo para predecir un valor continuo (regresión).

Es muy tolerante al ruido.

4.6 Inconvenientes

El coste de encontrar los k mejores vecinos es grande (estructuras especializadas kd-trees).

No hay un mecanismo para decidir el valor optimo para k (depende de cada conjunto de datos).

Su rendimiento baja si el numero de descriptores crece.

Su interpretabilidad es nula (no hay una descripción de los conceptos aprendidos).

4.7 Algoritmo inductivo no supervisado

4.7.1 Clustering

Clustering es el proceso de agrupar datos en clases o clusters de tal forma que los objetos de un cluster tengan una

similaridad alta entre ellos, y baja (sean muy diferentes) con objetos de otros clusters.

Características:

1. *Escalabilidad: normalmente corren con pocos datos.*

2. *Clusters de formas arbitrarias: lo basados en distancias numéricas tienden a encontrar cluster esféricos.*

3. *Capacidad de manejar diferentes tipos de atributos: numéricos (lo más común), binarios, nominales, ordinales, etc.*

4. *Capacidad de añadir restricciones.*

5. *Manejo de ruido: muchos son sensibles a datos erróneos.*

6. *Poder funcionar eficientemente con alta dimensionalidad.*

7. *Requerimientos mínimos para especificar parámetros, como el número de clusters.*

8. *Independiendentes del orden de los datos.*

9. *Que los clusters sean interpretables y utilizables*

Aplicabilidad: *El objetivo del clustering como vimos en el apartado anterior es identificar la clasificación intrínseca de un conjunto de datos no etiquetados. Los algoritmos de clasificación de datos tienen numerosas aplicaciones en distintos ámbitos: Biología, Comercialización, Biblioteca, Seguros, Sismología, Urbanismo... Una interpretación de lo anterior es el caso de que los distintos seres vivos se clasificaron en varias especies. Otra aplicación de interés es el estudio de los sismos. La reagrupación de los epicentros de los sismos observados permite determinar las zonas de riesgos, y poder ayudar a evitar catástrofes.* ***Definición de cliuster:***

Cluster, o grupo, es un conjunto de objetos que son "similares" entre ellos y "diferentes" de los objetos que pertenecen a los otros grupos.

La palabra "cluster" viene del inglés y significa agrupación. Desde un punto de vista general, el cluster puede considerarse como la búsqueda automática de una estructura o de una clasificación en una colección de datos "no etiquetados".

Clustering basado en probabilidades

Ahora los objetos tienen cierta probabilidad de pertenecer a un grupo o cluster. Desde el punto de vista bayesiano, lo que buscamos es el grupo de clusters más probables dados los datos.

La base de un clustering probilístico está basado en un modelo estadístico llamado finite mixtures (mezcla de distribuciones). Una mezcla es un conjunto de k distribuciones, representando k clusters.

La mezcla más sencilla es cuando tenemos puros atributos numéricos con distribuciones gaussianas con diferentes medias y varianzas.

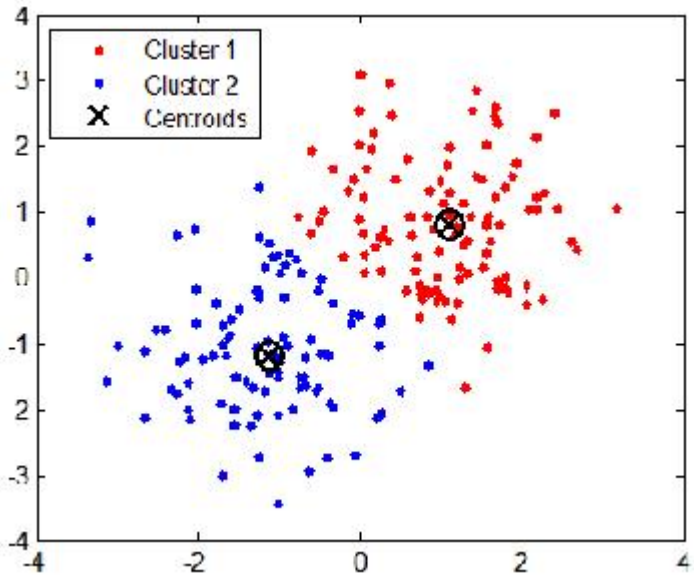

Figure 9:

La idea es, dado un conjunto de datos, determinar las k distribuciones normales (medias y varianzas) y las probabilidades particulares de cada distribución (pueden ser diferentes).

Existe una gran cantidad de algoritmos de clustering como: K-Means, CobWeb, Algoritmo EM (Expectation Maximization). Nosotros nos centraremos en el estudio del algoritmo de K-MEANS.

4.7.2 K-Means

El algoritmo K-means, creado por MacQueen en 1967 es el algoritmo de clustering más conocido y utilizado ya que es de muy simple aplicación y eficaz. Sigue un procedimiento simple de clasificación de un conjunto de objetos en un determinado número K de clústeres, K determinado a priori.

El nombre de K-means viene porque representa cada uno de los clusters por la media (o media ponderada) de sus puntos, es decir, por su centroide. La representación mediante centroides tiene la ventaja de que tiene un significado gráfico y estadístico inmediato.Cada cluster por tanto es caracterizado por su centro o centroide (ver figura 9) que se encuentra en el centro o el medio de los elementos que componen el cluster. Kmeans es traducido como K-medias.

O un conjunto de objetos Dn = (x1, x2..., xn), para todo el i, xi reales y k, $\nu 1$, los centros de los K cluster. El algoritmo del K-means se realiza en 4 etapas:

Etapa 1: *Elegir aleatoriamente K objetos que forman asi los K clusters iniciales. Para cada cluster k, el valor inicial del centro es = x_i, con los x_i unicos objetos de Dn pertenecientes al cluster.*

$$\widehat{s} = argmin||u_k - x||^2$$

Etapa 2: *Reasigna los objetos del cluster. Para cada objeto x, el prototipo que se le asigna es el que es más próximo al objeto, según una medida de distancia, (habitualmente la medida euclidiana).*

Etapa 3: *Una vez que todos los objetos son colocados, recalcular los centros de K cluster. (los baricentros).*

Etapa 4: *Repetir las etapas 2 y 3 hasta que no se hagan más reasignaciones. Aunque el algoritmo termina siempre, no se garantiza el obtener la solución óptima. En efecto, el algoritmo es muy sensible a la elección aleatoria de los K centros iniciales. Esta es la razón por la que, se utiliza el algoritmo del K-means numerosas veces sobre un mismo conjunto de datos para intentar minimizar este efecto, sabiendo que a centros iniciales lo más espaciados posibles dan mejores resultados.*

4.7.3 Inconvenientes:

1. Uno de los inconvenientes principales del K-means, además del hecho de que sea necesario realizar en sucesivas ocasiones el algoritmo para así tener el resultado más óptimo posible, es la necesidad de inicializar el número de prototipos al principio de la ejecución. Esto perjudica la eficacia del algoritmo ya que en la práctica, no se conoce a priori el número de cluster final.

Este defecto le perjudicará al compararlo con otros algoritmos, ya que en muchos la inicialización del número de clusters no es necesaria.

2. k-means es susceptible a valores extremos porque distorsionan la distribución de los datos.

Pseudocódigo: (Ver Figura 10)

5. REFERENCES

[1] *Eduardo Morales. Apartado Aprendizaje*
http://ccc.inaoep.mx/~emorales/Cursos/Aprendizaje/principa

[2] *Aprendizaje inductivo (KNN y K-MEANS)*
http://www.lsi.upc.es/~bejar/apren/teoria.html

[3] *Técnicas Avanzadas de Aprendizaje*
http://www.lsi.upc.edu/~bejar/aaac/teoria.html

[4] *Capítulo sobre clustering*
http://docs.jboss.org/jbossas/jboss4guide/r4/html/cluster.cha

[5] *Aplicación algoritmo K-MEANS*
http://celtico-celtico.blogspot.com/2008/02/filtro-k-means-k-m

[6] *Clasificación de Datos, KNN*
http://74.125.77.132/search?q=cache:MGUOiJMtAFcJ:www.


```
function [nuevos] = k_mean(X_tr,k)
perm = randperm(length(X_tr));
% Cojo los kc primeros centroides
centroides = X_tr(perm(1:k),:);
pos_centroides = zeros(length(X_tr),1);
nuevos = zeros(k,2);
p = 0;iguales = 0;
while(iguales == 0)
  if p==0
    nuevos(1:k,:) = centroides(1:k,:);
    p = 1;
  end
  % Los centroides actuales serán los que
  % se habían obtenido en la iteración anterio
  centroides(1:k,:) = nuevos(1:k,:);
  % Paso 2
  for i = 1:length(X_tr)
    distancia = sum(((repmat(X_tr(i,:),k,1))
    -centroides).^2,2);
    [minimo pos_min] = min(distancia);
    pos_centroides(i) = pos_min;
  end
  % Paso 3
  for j = 1:size(centroides,1)
    vector = j == pos_centroides;
    nuevos(j,:) = sum(X_tr(vector,:))./
    sum(vector);
  end
  % Si los centroides calculados son iguales a
  % los anteriores ya tenemosla condición de
  % parada
  if(isequal(centroides,nuevos))
    iguales = 1;
  end
end
```

Figure 10: