2013-06

Un algoritmo de clasificación incremental basado en los k vecinos más similares para datos mezclados

Sánchez-Díaz, Guillermo; Escobar-Franco, Uriel; Morales-Manilla, Luis; Aguirre-Salado, Carlos A.; Franco-Arcega, Anilu; Piza-Dávila, Hugo I.

Sánchez-Díaz, G.; Escobar-Franco, U.; Morales-Manilla, L; Piza-Dávila, H.I; Aguirre-Salado, C.A. y Franco-Arcega A. (2012) Un algoritmo de clasificación incremental basado en los k vecinos más similares para datos mezclados. Revista Facultad de Ingeniería, núm.67, junio. Medellín, Colombia: Universidad Antioquía.

Rev. Fac. Ing. Univ. Antioquia N. º XX pp. XXX 200X

Incremental k most similar neighbor classifier for mixed data

Un algoritmo de clasificación incremental basado en los k vecinos más similares para datos mezclados

Guillermo Sánchez-Díaz[1], Uriel E. Escobar-Franco[2], Luis R. Morales-Manilla[2], Iván Piza-Dávila[3], Carlos Aguirre-Salado[1] y Anilu Franco-Arcega[4]

[1] Universidad Autónoma de San Luis Potosí, Facultad de Ingeniería. Av. Dr. Manuel Nava no. 8. Zona Universitaria. San Luis Potosí, SLP, México. C.P. 78290

[2] Universidad Politécnica de Tulancingo, División de Ingenierías. Ingenierías # 100, Col. Huapalcalco, Tulancingo, Hgo. México. C.P. 43629

[3] Instituto Tecnológico y de Estudios Superiores de Occidente, Departamento de Electrónica, Sistemas e Informática. Periférico Sur Manuel Gómez Morín 8585, Tlaquepaque, Jal. México. C.P. 45604

[4] Universidad Autónoma del Estado de Hidalgo, Centro de Investigación en Tecnologías de Información y Sistemas. Carr. Pachuca-Tulancingo Km. 4.5, Zona Universitaria, Pachuca, Hgo., México. C.P. 42084

(Recibido el 23 de Abril de 2012. Aceptado el...)

Abstract

This paper presents an incremental k-most similar neighbor classifier, for mixed data and similarity functions that are not necessarily distances. The algorithm presented is suitable for processing large data sets, because it only stores in main memory the k most similar neighbors processed until step t, traversing only once the training data set. Several experiments with synthetic and real data are presented.

Keywords: Pattern recognition, supervised classification, incremental algorithms, artificial intelligence

Resumen

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

En este trabajo, se presenta un algoritmo de clasificación incremental basado en los k vecinos más similares, el cual permite trabajar con datos mezclados y funciones de semejanza que no necesariamente son distancias. El algoritmo presentado es adecuado para procesar grandes conjuntos de datos, debido a que sólo almacena en la memoria principal de la computadora los k vecinos más similares procesados hasta el paso t, recorriendo una sola vez el conjunto de datos de entrenamiento. Se presentan resultados obtenidos con diversos conjuntos de datos sintéticos y reales.
Palabras claves: algoritmo k-NN, Reconocimiento de patrones, clasificación supervisada, algoritmos incrementales, inteligencia artificial

Introducción

La regla del vecino más cercano k-Nearest Neighbor (k-NN) [23] ha sido ampliamente utilizada como una técnica no paramétrica en reconocimiento de patrones, debido a su sencillez y buen desempeño.

El algoritmo k-NN tradicional clasifica un nuevo objeto almacenando todo el conjunto de datos de entrenamiento en memoria, y calculando la distancia de estos objetos con el nuevo objeto que se desea clasificar. Posteriormente, realiza un ordenamiento de las distancias calculadas, para finalmente, obtener los k-objetos cuyas distancias tengan el menor valor. La clase asignada a ese nuevo objeto será la clase mayoritaria de los k-objetos más similares.

Por la aplicación que ha tenido, se han desarrollado diferentes alternativas de clasificadores basados en la regla k-NN. Algunos de ellos, aparecen en el estado del arte como clasificadores rápidos k-NN [25,6]. Estos algoritmos han sido desarrollados para procesar grandes conjuntos de datos, aplicados sobre diversos problemas, como el análisis de valores en línea, control de tráfico aéreo, detección de intrusos, entre otros. Sin embargo, varios de estos problemas, están definidos por conjuntos de datos con alta dimensionalidad, donde la función de comparación puede resultar computacionalmente muy costosa, por lo que se recomienda reducir el número de comparaciones

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

realizadas con los objetos de entrenamiento [17].

Para resolver diversos problemas de clasificación, se requiere procesar conjuntos de datos de entrenamiento muy grandes, los cuales, en ocasiones no es factible almacenarlos en la memoria principal de la computadora. Algunos ejemplos de estos conjuntos de datos muy grandes son: imágenes hiperespectrales de alta resolución con al menos 256 bandas [11, 10], algunos grupos de transacciones bancarias [10] o de compañías muy grandes, entre otras. Para este tipo de problemas, se han desarrollado algunos algoritmos k-NN incrementales estáticos [5, 7]. Sin embargo, si el conjunto de datos de entrenamiento es modificado (agregando o eliminando objetos del mismo), entonces el algoritmo k-NN debe ejecutarse nuevamente. Esta deficiencia limita el uso de estos algoritmos estáticos cuando en la práctica, las actualizaciones suelen ser inevitables. Por ejemplo, en métodos de minería de datos basados en muestreo, comúnmente se requieren algoritmos que permitan actualizaciones entre el conjunto de puntos de muestreo y los datos originales para evaluar la calidad de la muestra [3]. Como una alternativa para resolver este problema, se ha desarrollado un algoritmo k-NN que permite realizar actualizaciones sobre el conjunto de datos de manera incremental [8]. En este tipo de entornos donde los conjuntos de datos no son estáticos, los algoritmos k-NN que permiten las actualizaciones, pueden llevar a cabo periódicamente el muestreo, para proporcionar una medida exacta de la calidad de los datos de manera eficiente.

No obstante, la mayoría de estos algoritmos clasificadores k-NN propuestos en la literatura, han sido diseñados para descripciones numéricas de sus objetos, los cuales hacen uso de distancias. De esta manera, la mayoría de estos métodos hacen uso de las propiedades métricas para reducir el número de comparaciones entre los objetos.

Sin embargo, en muchas aplicaciones del mundo real, los objetos se describen por variables numéricas y categóricas (datos mezclados) [15]. En algunos casos, la función de semejanza no necesariamente satisface las propiedades métricas. Por esta razón, no siempre es factible aplicar la mayoría de los

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

clasificadores k-NN propuestos para procesar objetos con descripciones mezcladas.

Como una alternativa, se han propuesto el algoritmo de los k vecinos más similares (k-MSN), (descrito en [22] y [18]) y un algoritmo rápido denominado fast k-MSN, el cual trabaja con datos mezclados, utilizando una estructura de árbol [22]. A pesar de dar solución a algunas de las deficiencias expuestas en esta sección, estos algoritmos no permiten realizar actualizaciones en la muestra de entrenamiento. En el caso del segundo algoritmo mencionado, se tendría que generar el árbol nuevamente con el nuevo objeto agregado.

Diferentes aplicaciones en minería de datos requieren algoritmos k-NN y k-MSN que permitan realizar actualizaciones en el conjunto de datos de entrenamiento. Además de evitar que se cargue todo el conjunto de datos en la memoria principal (cuando el conjunto sea muy grande).

En este artículo, se propone una solución a algunos de los problemas encontrados anteriormente, con las siguientes contribuciones:

- Se propone un algoritmo incremental para procesar grandes, y muy grandes conjuntos de datos mezclados.
- El algoritmo propuesto permite insertar nuevos objetos al conjunto de entrenamiento, obteniendo los k vecinos más similares realizando solamente la comparación de los objetos insertados con el objeto a clasificar, sin tener que procesar nuevamente todo el conjunto de datos de entrenamiento actualizado.
- Al ser un algoritmo incremental, no necesita cargar ni almacenar en la memoria principal todo el conjunto de datos de entrenamiento. Solamente conserva los k vecinos más similares calculados hasta el paso *t*. De esta forma, el algoritmo permite procesar conjuntos de datos cuyo tamaño rebase la memoria principal.
- Para clasificar un nuevo objeto, se recorre una sola vez la muestra de entrenamiento.

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

Definición del problema

Sea U un universo de objetos no necesariamente finito. Cada objeto $o_i \in U$ está descrito por un conjunto de atributos $R=\{x_1, x_2, \ldots, x_n\}$, y distribuidos en c-clases $\{S_1, S_2, \ldots, S_c\}$. Cada atributo $x_i \in R$ puede tomar valores en un conjunto $M_{i,} i=1, \ldots, n$ el cual determina la naturaleza del mismo, pudiendo ser numérico o categórico. Sea además $TM=\{o_1, o_2, \ldots, o_m\}$, $TM \subseteq U$, el conjunto de entrenamiento de los objetos de U.

Cuando se manejan datos mezclados, un concepto fundamental es la analogía o semejanza entre objetos, el cual puede formalizarse a través de una función de similaridad o disimilaridad [14].

Un criterio de comparación $C_i: M_i \times M_i \rightarrow L_i$ es asociado a cada variable $x_{i,} i=1, \ldots, n$, donde $C_i(x_i(o), x_i(o))=min\{y\}, y \in L_i$, si C_i es un criterio de disimilaridad entre valores de la variable x_i o $C_i(x_i(o), x_i(o))=max\{y\}, y \in L_i$, si C_i es un criterio de similaridad entre valores de la variable x_i. C_i es una evaluación del grado de similaridad (o disimilaridad) entre cualquiera dos valores de la variable $x_{i,} i=1, \ldots, n$ donde L_i es un conjunto totalmente ordenado.

Entre cada par de objetos de U, se puede calcular una magnitud. Esta magnitud se obtiene aplicando una función de semejanza FS (que pudiera inclusive ser parcial), la cual puede ser definida para cualquier subconjunto de R. Cuando se manejan datos mezclados, existen funciones de semejanza que no cumplen la desigualdad triangular [21].

Dado un objeto $o \in U, o \notin TM$, del cual se quiere obtener la clase a la que pertenece, el problema abordado en este trabajo es de clasificación supervisada sobre grandes volúmenes de datos, los cuales pueden ser de naturaleza cualitativa o cuantitativa (datos mezclados), permitiendo además realizar incrementos en la muestra de entrenamiento TM.

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

Trabajo relacionado

Actualmente, la tecnología desarrollada permite almacenar grandes cantidades de información. Para aplicar el clasificador k-NN cuando el conjunto de datos de entrenamiento es grande, se han propuesto diferentes técnicas [1, 4, 13, 16, 20]. En particular, en este artículo, se considerará una organización específica en el desarrollo de algoritmos de clasificación k-NN: algoritmos incrementales y no incrementales.

Dentro de los algoritmos no incrementales se pueden mencionar dos subtipos: 1) algoritmos de aproximación y 2) algoritmos que generan el resultado exacto.

El algoritmo original propuesto en [23], así como el algoritmo k-MSN (descrito en [22] y [18]) entran en la segunda categoría mencionada. La principal desventaja que presenta estos algoritmos es su complejidad cuadrática, la cual cuando se manejan conjuntos de datos muy grandes hace que el clasificador se vuelva muy lento. Además, estos algoritmos requieren que todos los datos se encuentren en la memoria principal. Finalmente, si se agrega un dato a la muestra de entrenamiento, se debe ejecutar todo el proceso del clasificador nuevamente. Este algoritmo puede extenderse para generar los k vecinos más similares, manejando un diferente operador de comparación entre objetos, así como el orden entre las comparaciones obtenidas (el cual se puede denotar como *k-MSN*).

Los algoritmos de aproximación han tenido un notable desarrollo, debido a que reducen el tamaño de la muestra de entrenamiento antes de aplicar el clasificador k-NN [4]. Además, de los algoritmos que particionan el espacio de los datos en regiones, usando estructuras de árboles [13, 20].

Otros algoritmos de aproximación, basan su funcionamiento en el uso de reglas de poda derivadas de la desigualdad triangular, para hacer eliminaciones en algunas comparaciones de la muestra de entrenamiento [6].

La principal desventaja que tienen estas técnicas es que al utilizar estructuras basadas en métricas, son adecuadas para procesar solamente datos numéricos. Además, algunos de estos algoritmos son

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

estáticos, y no permiten agregar nuevos objetos a la muestra de entrenamiento sin tener que generar nuevamente la estructura de árbol o producir nuevamente las reglas de poda.

Recientemente, se propuso un algoritmo que usa una estructura de árbol, para procesar conjuntos grandes de datos mezclados, el cual calcula una aproximación de los k vecinos más similares (k-MSN) [22]. Esta técnica consta de dos fases: a) construcción del árbol con los datos de entrenamiento y b) fase de clasificación recorriendo el árbol generado. Este algoritmo, a pesar de poder procesar grandes conjuntos de datos mezclados, tiene la limitación de que si es agregado un objeto a la muestra de entrenamiento, se debe nuevamente volver a construir todo el árbol con los datos de entrenamiento, como es indicado en su fase (a). Además, en su proceso de construcción del árbol, el primer nodo de éste contiene al conjunto de datos de entrenamiento completo. Por lo tanto, este algoritmo requiere que todos los datos se encuentren en la memoria principal.

El segundo grupo que incluye a los incrementales, se puede clasificar en dos subtipos: estáticos y dinámicos. Los algoritmos incrementales estáticos utilizan estructuras de árbol, índices y ordenamientos de los datos para aplicar en una fase posterior la regla de clasificación de los k vecinos más cercanos [5, 7]. Estos algoritmos pueden procesar grandes conjuntos de datos multidimensionales, teniendo la limitación de usar ciclos anidados para procesar el conjunto de datos de entrenamiento. Entonces, si se realiza alguna actualización en el conjunto mencionado, los ciclos empleados para generar sus estructuras deben calcularse nuevamente para obtener el resultado con los nuevos objetos agregados. Por otro lado, dentro de los algoritmos dinámicos existe la técnica knnJoin^{+} [8], la cual permite procesar grandes conjuntos de datos multidimensionales, dando la facilidad de agregar y eliminar objetos del conjunto de datos de entrenamiento, sin tener que volver a procesar el conjunto de entrenamiento actualizado. La limitación que presenta este algoritmo, es que fue diseñado para procesar exclusivamente datos numéricos.

Existen problemas en bosques usando sensores remotos [26] así como la predicción y estimación de

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

especies para inventario forestal [12, 19], que están descritos por grandes conjuntos de datos, con variables que pueden ser tanto numéricas como categóricas. Además algunos de estos problemas realizan un monitoreo constante, lo cual requiere de agregar o eliminar nuevos datos en la muestra de aprendizaje, ya que puede ser cambiante.

Tomando en cuenta las problemáticas expuestas anteriormente, en este trabajo se propone el desarrollo de un clasificador incremental basado en los k vecinos más similares, que permita procesar datos mezclados. El algoritmo propuesto representa una alternativa de solución a algunos de los problemas que han sido reportados en el estado del arte.

El algoritmo incremental propuesto

El algoritmo incremental propuesto (que se denotará por *inc-k-MSN*) procesa objeto por objeto del conjunto de datos de entrenamiento, inclusive si a este conjunto se le van agregando nuevos objetos, *inc-k-MSN* no almacena el conjunto de datos de entrenamiento en la memoria principal, solamente guarda el objeto a clasificar, así como los k vecinos más similares calculados hasta el paso t (que normalmente sería cuando se ha procesado el objeto t de la muestra de entrenamiento).

Por cada objeto del conjunto de entrenamiento que *inc-k-MSN* procesa, se generan los k vecinos más similares de manera parcial. Al procesarse todos los objetos del conjunto de datos el algoritmo calcula los mismos k vecinos más similares que el algoritmo original genera, con la diferencia de que *inc-k-MSN* podrá continuar anexando nuevos objetos en el conjunto de datos, siguiendo con la misma filosofía de procesamiento que con los objetos iniciales del conjunto de entrenamiento. Este hecho garantiza que la precisión de la clasificación no decrezca.

El procedimiento que realiza el algoritmo propuesto, le permite procesar y manejar conjuntos de datos de entrenamiento que rebasen la capacidad de la memoria principal. El algoritmo *inc-k-MSN* sigue la filosofía de procesar objeto por objeto los datos de entrenamiento, almacenando resultados parciales, y

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

generando el mismo resultado que el algoritmo convencional una vez procesados todos los objetos que conformen los datos de entrenamiento, como lo realiza el algoritmo INC-ALVOT [24].

La principal diferencia entre los algoritmos estáticos reportados y el propuesto en este trabajo, radica en que los algoritmos estáticos necesitan tener todo el conjunto de datos de entrenamiento para poder obtener los k vecinos más similares. De manera diferente, *inc-k-MSN* basa la generación de los k vecinos más similares entre cada objeto del conjunto de entrenamiento y el objeto a clasificar, permitiéndole utilizar los k vecinos más similares previamente generados con los objetos ya procesados, con los objetos restantes del conjunto de entrenamiento.

Este funcionamiento le permite al algoritmo propuesto manejar nuevos objetos añadidos en el conjunto de datos de entrenamiento, como cualquier otro objeto ya incluido en el conjunto mencionado. Esta característica hace diferente al algoritmo propuesto de los algoritmos estáticos reportados, debido a que si hubiera algún incremento en la muestra de entrenamiento, estos algoritmos para encontrar los k vecinos más similares, deben procesar nuevamente toda la muestra de entrenamiento.

En la descripción del algoritmo inc-k-MSN, se utiliza una estructura simple denotada como *msn* (most similar neighbor), la cual contiene dos elementos: $o_d \in TM$; y $FS(o, o_d)$ que indica el valor de la similaridad calculada entre el objeto o y el objeto o_d .

Cada *msn* almacena la información de cada uno de los k vecinos más similares obtenidos por el algoritmo *inc-k-MSN*. Al especificarse k vecinos más similares a obtener, se manejaran k estructuras de msn ($msn_i, i = 1, \ldots, k$).

Entonces, cada $msn_i = (o_d, FS(o, o_d))$ contendrá al vecino más similar representado por el objeto o_d con respecto al objeto a clasificar o .

A continuación, se describe el algoritmo *inc-k-MSN* propuesto.

Entrada: o *(objeto a clasificar);* o_t *(objeto tomado de* TM *);* $msn_i, i = 1, \ldots, k$ *(k vecinos*

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

más similares obtenidos antes de procesar el objeto o_t *)*

Salida: $msn_i, i=1,\ldots,k$ *(k vecinos más similares obtenidos después de procesar al objeto* o_t *);*

Para el objeto $o_t \in TM$

Calcular la función de semejanza $FS(o, o_t)$, y generar $msn=(o_t, FS(o, o_t))$

Caso 1: Se utiliza una medida de similaridad.

Si $msn.FS(o, o_t) > msn_i.FS(o, o_q), t \neq q, i=1,\ldots,k$, entonces

$Inserta(msn, i)$;

$Elimina_{\square}^{msn}$;

Caso 2: Se utiliza una medida de disimilaridad

Si $msn.FS(o, o_t) < msn_i.FS(o, o_q), t \neq q, i=1,\ldots,k$, entonces

$Inserta(msn, i)$;

$Elimina_{\square}^{msn}$;

$Inserta(msn, i)$ inserta en la lista de los k vecinos más similares el msn creado, en la posición i de la lista de los k vecinos más similares, desplazando hacia el final de la lista a todos los elementos restantes. $Elimina_{\square}^{msn}$ elimina el último msn de la lista de los k vecinos más similares (debido a que se convirtió en el elemento $k+1$ de la lista mencionada, por la incorporación en la misma del msn creado).

De manera computacional, la lista de los k vecinos más similares puede inicializarse de dos maneras: en la primera se considera que la lista está vacía y se van agregando nuevos objetos conforme van llegando, hasta llegar a contener *k*-elementos. En la segunda, se inicializan los valores de $msn_i, FS(o, o_t), i=1,\ldots,k$, con un valor muy grande (si se está manejando una medida de

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

disimilaridad o distancia), o con un valor muy pequeño o negativo (si se está manejando una medida de similaridad).

El algoritmo incremental presentado, va generando los k vecinos más similares de manera parcial conforme va procesando cada objeto o_t del conjunto de datos de entrenamiento, modificando solamente la lista de los k vecinos más similares si se cumpliera la condición marcada en el algoritmo. De esta manera, se van generando de manera incremental los k vecinos más similares. Al procesarse el último objeto del conjunto de datos de entrenamiento, los k vecinos más similares serán los mismos que los obtenidos por el algoritmo clásico.

Discusión del algoritmo

El algoritmo propuesto es capaz de procesar una cantidad considerable de funciones de similaridad así como distancias, para generar los k vecinos más similares. Sin embargo, el algoritmo no está concebido para procesar funciones que se calculan a partir de todos los objetos, como el caso de la mediana. Esto se debe, a que el algoritmo no almacena todas las comparaciones realizadas con los objetos de la muestra de entrenamiento. Tampoco realiza un ordenamiento de todas las similaridades calculadas. Otra limitación del algoritmo propuesto, radica en que no fue concebido para trabajar con funciones de similaridad no simétricas.

El algoritmo propuesto recorre una sola vez al conjunto de datos de entrenamiento, el cual contiene *m*-objetos. En el peor caso, la lista de los k vecinos más similares generada se recorre completamente cada vez que se procesa un nuevo objeto de entrenamiento. Entonces, la complejidad de este algoritmo es $O(m*k)$. Pero al procesar conjuntos de datos grandes y muy grandes el valor de *k* suele ser insignificante comparándolo con el valor de *m*. Por tanto, la complejidad del algoritmo sería $O(m)$ (cuando $k << m$).

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

Al aplicar el algoritmo *k-msn* a todos los objetos del conjunto de datos de entrenamiento, se han generado los k vecinos más similares de este conjunto. Estos k vecinos pueden ser los mismos que los calculados con $m-1$ objetos considerados en el paso anterior. Esto se cumple si el objeto $p=m$ no forma parte de los k vecinos ya calculados con los $m-1$ objetos. En otro caso, el objeto $p=m$ ahora forma parte de los k vecinos ya calculados con los $m-1$ objetos. Entonces, al agregarse el objeto $p=m$ a la lista de los k vecinos, se desplazan hacia el final de la lista todos los vecinos cuyo valor sea menor que el del objeto insertado (al usarse una disimilaridad o distancia) o cuyo valor sea mayor que el del nuevo objeto (al utilizarse una similaridad), eliminando al último objeto desplazado de la lista de los k vecinos.

Por otro lado, hasta este paso se ha aplicado el algoritmo *inc-k-MSN,* $m-1$ veces sobre el conjunto de datos, calculando los k vecinos con los $m-1$ objetos considerados en el paso anterior, faltando por procesar un último objeto del conjunto de datos de entrenamiento (el objeto $p=m$). Al aplicarse nuevamente el algoritmo *inc-k-MSN* para procesar el objeto $p=m$, pueden darse dos casos: a) el objeto $p=m$ no cumple la condición para formar parte de los k vecinos ya calculados con los $m-1$ objetos. Entonces, los k vecinos son los mismos considerando tanto $m-1$ como m objetos; b) el objeto $p=m$ si cumple la condición para formar parte de los k vecinos ya calculados con los $m-1$ objetos. De esta manera, el objeto $p=m$ será agregado a la lista de los k vecinos ya calculados con los $m-1$ objetos. Entonces, cuando se adiciona el objeto $p=m$ a la lista de los k vecinos actual, se desplazan hacia el final de la lista todos los vecinos cuyo valor sea menor que el del objeto insertado (al usarse una disimilaridad o distancia) o cuyo valor sea mayor que el del nuevo objeto (al utilizarse una similaridad), eliminando al último objeto desplazado de la lista de los k vecinos. Por lo tanto, los k vecinos más similares generados como resultado de aplicar el algoritmo *inc-k-MSN m*-veces al conjunto de datos de entrenamiento, son los mismos k vecinos generados cuando se aplica el algoritmo *k-msn* al mismo conjunto de datos de entrenamiento.

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

Experimentación

En esta sección, se muestra un ejemplo paso a paso del algoritmo incremental propuesto, para ilustrar su funcionamiento. Posteriormente, se presenta una tabla comparativa con algunos conjuntos de datos de entrenamiento reportados en [22]. Finalmente, se muestran los resultados obtenidos al aplicarse el algoritmo propuesto y el algoritmo clásico *k-msn* a conjuntos de datos grandes de entrenamiento para comparar el desempeño del algoritmo,

El ejemplo ilustrativo usa la base de datos Irisdata [2], la cual contiene 150 objetos, descritos por 4 variables, distribuidos en 3 clases (setosa: objetos de la posición 1 a la 50 en la muestra de entrenamiento, versicolor: objetos 51 a 100 y virginica: objetos 101 a 150). Se extrajo el objeto 1 de la muestra de entrenamiento para utilizarlo como objeto a clasificar. Los objetos restantes 2 al 150 permanecieron en la muestra de entrenamiento. La descripción del objeto a clasificarse es: o *={5.1, 3.5, 1.4, 0.2}*. En la tabla 1, se muestran los resultados de obtener los k vecinos más similares al procesar a los objetos $o_t, t = 2, \ldots, 150$. En este experimento, el objeto o_{39} es el último de la muestra de entrenamiento que modifica los k vecinos más similares obtenidos. Después de procesar todos los objetos de la muestra de entrenamiento, el algoritmo *inc-k-MSN* obtiene los mismos k vecinos que genera el algoritmo *k-MSN*.

El objeto a clasificarse para ilustrar el algoritmo propuesto es: o ={5.1 3.5 1.4 0.2}. En la tabla 1, se muestran los resultados obtenidos por *inc-k-MSN* al procesar cada objeto $o_t \in TM$, que modifica la lista de k vecinos más similares, así como el resultado generado por el algoritmo al procesar todos los objetos de la muestra de entrenamiento. Para este caso en particular, la función de semejanza utilizada fue una distancia Euclidiana. Después de procesar todos los objetos, *inc-k-MSN* obtiene los mismos k vecinos más similares que el algoritmo *k-MSN* secuencial.

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

Tabla 1. k vecinos más similares obtenidos por *inc-k-MSN* al procesar los objetos $o_t, t=2,\ldots,150$. Mostrando los correspondientes $msn_i=\left(o_d, FS\left(o, o_d\right)\right)$, con el objeto a clasificar o *={5.1, 3.5, 1.4, 0.2}*

$\left(o_t, FS\left(o, o_t\right)\right)$	***K vecinos generados al procesar el objeto*** t
(o_1 ={4.9, 3.0, 1.4, 0.2}, 0.5385)	(o_1 , 0.5385), (null, ∞), (null, ∞), (null, ∞), (null, ∞)
(o_2 ={4.7, 3.2, 1.3, 0.2}, 0.5099)	(o_2 , 0.5099), (o_1 , 0.5385), (null, ∞), (null, ∞), (null, ∞)
(o_3 ={4.6, 3.1, 1.5, 0.2}, 0.6480)	(o_2 , 0.5099), (o_1 , 0.5385), (o_3 , 0.6480), (null, ∞), (null, ∞)
(o_4 ={5.0, 3.6, 1.4, 0.2}, 0.1414)	(o_4 , 0.1414),(o_2 , 0.5099), (o_1 , 0.5385), (o_3 , 0.6480), (null, ∞)
(o_5 ={5.4, 3.9, 1.7, 0.4}, 0.6164)	(o_4 , 0.1414),(o_2 , 0.5099), (o_1 , 0.5385), (o_5 , 0.6164), (o_3 , 0.6480)
(o_6 ={4.6, 3.4, 1.4, 0.3}, 0.5196)	(o_4 , 0.1414),(o_2 , 0.5099), (o_6 , 0.5196), (o_1 , 0.5385), (o_5 , 0.6164)
(o_7 ={5.0, 3.4, 1.5, 0.2}, 0.1732)	(o_4 , 0.1414),(o_7 , 0.1732), (o_2 , 0.5099), (o_6 , 0.5196), (o_1 , 0.5385)
(o_9 ={4.9, 3.1, 1.5, 0.1}, 0.4690)	(o_4 , 0.1414),(o_7 , 0.1732), (o_9 , 0.4690), (o_2 , 0.5099), (o_6 , 0.5196)
...	...
(o_{28} ={5.2, 3.4, 1.4, 0.2}, 0.099)	(o_{17} , 0.0999),(o_4 , 0.1414), (o_{27} , 0.1414), (o_{28} , 0.1414), (o_7 , 0.1732)
(o_{39} ={5.1, 3.4, 1.5, 0.2}, 0.099)	(o_{17} , 0.0999),(o_4 , 0.1414), (o_{39} , 0.1414), (o_{27} , 0.1414), (o_{28} , 0.1414)

Para verificar el comportamiento del algoritmo propuesto, se tomaron exclusivamente en consideración las pruebas que muestran el tiempo de ejecución del algoritmo *fast k-MSN*, reportado como el más eficiente [22]. En este trabajo, se usaron 4 conjuntos de datos sintéticos, con: 2000 (1800 de entrenamiento y 200 para clasificación), 3000 (2700 de entrenamiento y 300 para clasificación), 4000

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

(3600 de entrenamiento y 400 para clasificación) y 7200 (6480 de entrenamiento y 720 para clasificación) objetos. Todos los conjuntos de datos tienen 2 variables y fue usado un valor de $k=1$. De igual manera, fueron generados conjuntos de datos sintéticos con las mismas características mencionadas anteriormente, para poder realizar una comparación entre los tiempos de ejecución lo más fiable posible. Estos resultados se muestran en la tabla 2.

Tabla 2. Tiempo de ejecución expresado en segundos, obtenido por los algoritmos *fast k-MSN* e *inc-k-MSN*, con $k=1$

Número de objetos total a procesar inc-k-MSN	***Número de objetos datos de entrenamiento***	***Número de objetos datos de control***	***Tiempo de ejecución en segundos fast k-MSN***	***Tiempo de ejecución en segundos inc-k-MSN***
2000	1800	200	0.485	0.6
3000	2700	300	0.812	1.2
4000	3600	400	1.139	2.0
7200	6480	720	7.63	6.48
726480	6480	720000	7630 (2.1 hrs)	6480 (1.8 hrs)

El algoritmo *inc-k-MSN* fue implementado en Java. Tales pruebas fueron realizadas en una PC con procesador Intel Pentium con 2 cores, 3 Gb de memoria RAM, bajo el sistema operativo Mandriva Linux 2010.

Finalmente en las últimas pruebas realizadas para verificar el desempeño del algoritmo al procesar grande conjuntos de datos de entrenamiento, se utilizaron 2 conjuntos, los cuales se especifican a

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

continuación: a) 10 conjuntos, desde 1,000 hasta 100,000 objetos con 3 atributos (el primer conjunto tiene 1,000 objetos, los restantes conjuntos de entrenamiento se van formando al incrementar de 10,000 en 10,000 objetos) y b) 10 conjuntos, desde 1,000,000 hasta 10,000,000 de objetos con 3 atributos (el primer conjunto tiene 1,000,000 de objetos, los restantes conjuntos de entrenamiento se van formando al incrementar de 1,000,000 en 1,000,000 de objetos). Los resultados de estos experimentos son mostrados en las figuras 1 y 2.

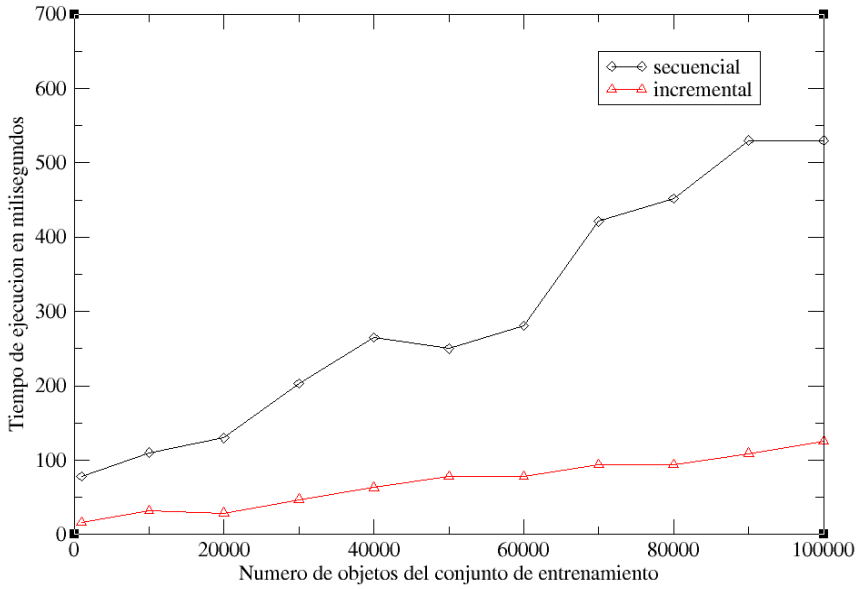
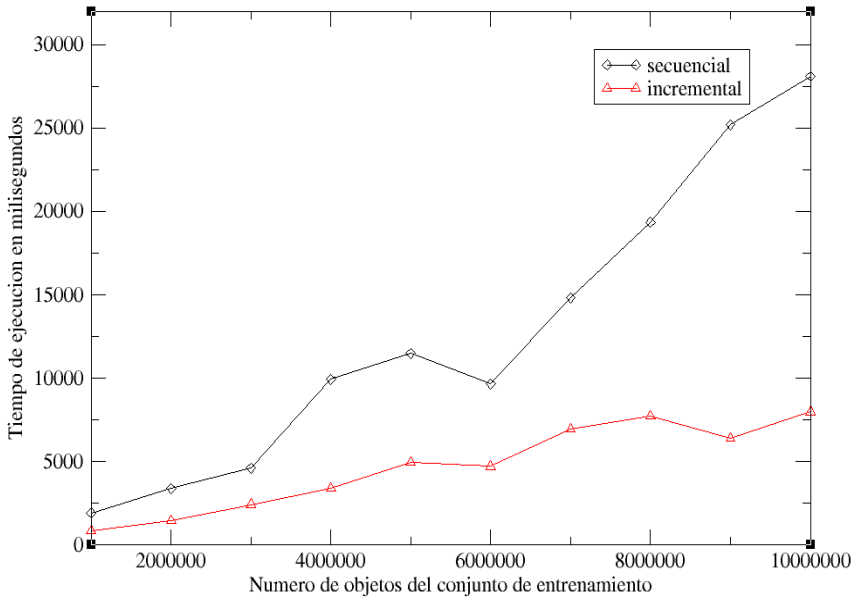


Figura 1. Tiempo de ejecución de los algoritmos *k-MSN* e *inc-k-MSN*, con $k=5$, procesando desde 1000 hasta 100000 objetos de la muestra de entrenamiento



Autor de corr).sanchez@uaslp.mx, tel. +52(444)-

Figura 2. Tiempo de ejecución de los algoritmos *k-MSN* e *inc-k-MSN*, con $k=5$, procesando desde 1000000 hasta 10000000 objetos de la muestra de entrenamiento.

Estos conjuntos fueron tomados como prueba, ya que inicialmente se usaron algunas bases de datos reales de [2]. Sin embargo, el tiempo de ejecución utilizado en estos datos de entrenamiento no fue significante para realizar una comparación que muestre el desempeño del algoritmo. Por ejemplo, usando una base de datos denominada MAGIC gamma telescope data (telescopy DB), compuesta por 19,018 objetos y 10 atributos. El tiempo de ejecución del algoritmo incremental para clasificar un nuevo objeto fue menor de 90 mili-segundos.

Las últimas pruebas fueron realizadas en un servidor HP ProLiant DL380, con procesador Intel Xeon X5550 a 3.0 Gigahertz, Quad-Core, con 32 Gb de memoria RAM, bajo el sistema operativo Windows 7 de 64 bits.

Discusión

En la tabla 2 para el ejemplo mostrado en particular, cuando la muestra de aprendizaje contiene 2000, 3000 y 4000 objetos, se puede observar que el algoritmo *fast k-MSN* [22] tuvo mejores tiempos de ejecución que el algoritmo *inc-k-MSN*. Sin embargo, conforme va aumentando el número de objetos en la muestra de entrenamiento, el algoritmo *inc-k-MSN* obtiene mejores tiempos de ejecución que el algoritmo *fast k-MSN*. Es importante señalar, que el algoritmo *fast k-MSN* no tiene la característica de

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

poder añadir nuevos objetos en la muestra de entrenamiento sin la necesidad de procesar nuevamente la muestra completa para generar el árbol que utiliza para su funcionamiento. Si este caso ocurriera, el algoritmo *inc-k-MSN* al pedir como entrada los k vecinos más similares obtenidos hasta el último objeto procesado de la muestra de entrenamiento, el tiempo de ejecución requerido para procesar los nuevos objetos agregados sería insignificante en comparación de los tiempos de ejecución requeridos para procesar toda la muestra de aprendizaje. En las figuras 1 y 2, puede observarse el mejor desempeño que tuvo el algoritmo *inc-k-MSN* en relación al algoritmo *k-MSN*. *inc-k-MSN* mantiene el crecimiento en su tiempo de ejecución de manera estable, inclusive cuando el número de objetos del conjunto de entrenamiento crece de manera significativa.

Conclusiones

En este trabajo se presentó un algoritmo de clasificación incremental para grandes volúmenes de información, denominado *inc-k-MSN*. En las pruebas realizadas en este trabajo, se utilizaron distancias para la realización de los experimentos. Sin embargo, el algoritmo propuesto fue desarrollado para procesar datos que pueden ser de naturaleza cualitativa o cuantitativa (datos mezclados), permitiendo utilizar funciones de semejanza que no necesariamente son distancias. El algoritmo propuesto permite manejar conjuntos de datos de entrenamiento sin mantenerlos en memoria principal, además de permitir agregar nuevos objetos en el conjunto de entrenamiento, sin tener que realizar nuevamente el cálculo del conjunto de entrenamiento modificado. Esto es factible de realizar porque por cada objeto procesado, se actualizan los k vecinos más similares hasta el paso *t*. El algoritmo propuesto, fue concebido para ser eficiente ante la adición de nuevos objetos en el conjunto de datos, el cual puede ser grande (millones de objetos) e inclusive, puede procesarse sin almacenarlo completo en la memoria principal.

En este trabajo, no se muestra la aplicación del algoritmo propuesto a ningún problema real de clasificación en particular. Sin embargo, existen problemas de clasificación reales, para los cuales es

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

adecuado el uso del algoritmo propuesto, debido a que van incrementando el número de objetos de la muestra original de aprendizaje. Algunos problemas de este tipo son: a) la detección y clasificación de casos de epidemias en la población, ya que puede aumentar la muestra original de pacientes contagiados con diferentes variantes del virus; b) la detección de fraudes realizados por pagos con tarjetas bancarias, donde se van incrementando e identificando las maneras de efectuar los fraudes mencionados, entre otros.

Agradecimientos

Este trabajo fue apoyado por PROMEP México, bajo la modalidad de NPTC convocatoria 2012, no. oficio del proyecto: PROMEP/103-5/11/3671

Referencias

1. A. Faragó, T. Linder, G. Lugosi. “Fast nearest-neighbor search in dissimilarity spaces”. IEEE Transactions in Pattern Analysis and Machine Intelligence. Vol. 9. pp. 957–962. 1993.
2. A. Frank, A. Asuncion. "UCI Machine Learning Repository" [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. 1998
3. C. Bohm C. Krebs. “The k-nearest neighbor join: turbo charging the kdd process”. Knowledge Information Systems. Vol. 6. pp. 728–749. 2004.
4. C. Chien-Hsing, K. Bo-Han, C. Fu. “The generalized condensed nearest neighbor rule as a data reduction method”. Proc. of the 18th International Conference on Pattern Recognition. vol. 02. Hong Kong, China. pp. 556–559. 2006 .
5. C. Xia, H. Lu, BC. Ooi, J. Hu, “Gorder: an efficient method for knn join processing”. Proc. of the 30th international conference on very large data bases. Toronto, Canada. pp. 756–767. 2004 .
6. C. Yong-Sheng, H. Yi-Ping, F. Chiou-Shann. “Fast and versatile algorithm for nearest neighbor search based on lower bound tree”. Pattern Recognition Letters. Vol. 2. pp. 360–375. 2007.

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

7. C. Yu, B. Cui, S. Wang, J. Su, “Efficient index-based knn join processing for high-dimensional data”. Inf. Softw. Technol. Vol. 4. pp. 332–344. 2007.
8. C. Yu, R. Zhang, Y. Huang, H. Xiong, “High-dimensional kNN joins with incremental updates”. Geoinformatica. Nº. 14. pp. 55–82. 2010.
9. H. Chen, B. Yang, G. Wang, J. Liu, X. Xu, S. Wang, D. Liu. “A novel bankruptcy prediction model based on an adaptive fuzzy k-nearest neighbor method”. Knowledge-Based Systems. Vol. 24. pp. 1348-1359. 2011
10. H. Latifi, F. Fassnacht, B. Koch. “Forest structure modeling with combined airborne hyperspectral and LiDAR data”. Remote Sensing of Environment. Vol. 121. Pp.10-25. 2012.
11. I. Sone, R. Olsen, A. Sivertsen, G. Eilertsen, K. Heia. “Classification of fresh Atlantic salmon (Salmo salar L.) fillets stored under different atmospheres by hyperspectral imaging”. Journal of Food Engineering. Vol. 109. Pp. 482-489. 2012.
12. J. Breidenbach, E. Nasset, V. Lien, T. Gobakken, S. Solberg. “Prediction of species specific forest inventory attributes using a nonparametric semi-individual tree crown approach based on fused airborne laser scanning and multispectral data”. Remote Sensing of Environment. vol. 114. no. 4. pp. 911–924. 2010.
13. J. Friedman, F. Baskett, L. Shustek, “An algorithm for finding nearest neighbors”. IEEE Transactions on Computers. vol. C-24. issue 10. pp. 1000–1006, 1975 .
14. J. Ruiz, M. Abidi. “Logical combinatorial pattern recognition: A review”. Ed. Transworld Research Network. Kerala, India. pp. 133-176. 2002 .
15. J. Ruiz. “Pattern recognition with mixed and incomplete data”. Pattern Recognition and Image Analysis. Vol. 18. pp. 563-576. 2008.
16. K. Figueroa, E. Chávez, G. Navarro, R. Paredes. “On the last cost for proximity searching in metric

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

spaces". Ed. LNCS 4007. Menorca Island, Spain. pp. 279–290. 2006 .

17. M. Adler, B. Heeringa. "Search Space Reductions for Nearest-Neighbor Queries". Lecture Notes in Computer Science, vol. 4978. Springer. pp. 554–567. 2008 .
18. P. Packalen, M. Maltamo. "The k-MSN method for the prediction of species-specific stand attributes using airborne laser scanning and aerial photographs". Remote Sensing of Environment. Vol. 109. Issue 3.pp. 328-341. 2007
19. R. McRoberts, S. Magnussen, E. Tomppo, G. Chirici. "Parametric, bootstrap, and jackknife variance estimators for the k-Nearest Neighbors technique with illustrations using forest inventory and satellite image data". Remote Sensing of Environment. vol. 115. no 2. pp. 3165–3174. 2011.
20. S. Berchtold, D. Keim, H. Kriegel, T. Seidl, "Indexing the solution space: a new technique for nearest neighbor search in high dimensional space". IEEE Transactions on Knowledge Data Engineering. Vol. 1. pp. 45–57. 2000.
21. S. Hernández, J. Carrasco, J. Martínez. "Fast k Most Similar Neighbor Classifier for Mixed Data Based on Approximating and Eliminating". Ed. LNAI 5012, Springer. Osaka, Japan. pp. 697–704. 2008 .
22. S. Hernández, J. Martínez, A. Carrasco."Fast k most similar neighbor classifier for mixed data (tree k-MSN)". Pattern Recognition. vol. 43. issue 3. pp. 873-886. 2010 .
23. T. Cover, P. Hart, "Nearest neighbor pattern classification". Transactions on Information Theory. vol. 13, no. 1. pp 21–27.. 1967 .
24. U. Escobar, G. Sánchez. "Algoritmo de votación incremental INC-ALVOT para clasificación supervisada". Revista Facultad de Ingeniería, Universidad de Antioquia. Nº. 50. pp. 195-204. 2009.
25. V. Ramasubramanian, K. Paliwal, "Fast nearest-neighbor search based on approximation-elimination search". Pattern Recognition, vol. 33 issue. 9 pp. 1497–1510. 2000 .

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336

26. X. Tian, Z. Su, E. Chen, Z. Li, C. Van der Tol, J. Guo, Q. He. “Estimation of forest above-ground biomass using multi-parameter remote sensing data over a cold and arid area”. Int. Journal of Applied Earth Observation and Geoinformation. vol. 14. issue 1. pp. 160–168. 2012 .

Autor de correspondencia: Guillermo Sánchez-Díaz, correo electrónico: guillermo.sanchez@uaslp.mx, tel. +52(444)-8262330 ext. 6010; fax ext. 2336