Ciencias Matemáticas, Vol. 29, No. 2, Pag. 87-92, 2015
Recibido 03-2015

Análisis de un algoritmo multiclasificador incremental con diferentes clasificadores bases
Analysis of incremental ensemble algorithm with different basic classifiers

Alberto Verdecia Cabrera[1]*, David La O Naranjo[2], Ramón Osmany Ramírez Tasé[1], Agustín Alejandro Ortiz Díaz[1], Isvani Frías Blanco[1]

Resumen Los algoritmos de clasificación que se adaptan a los cambios de conceptos en la minería de flujos de datos son actualmente muy importantes para muchas aplicaciones como: bioinformática, medicina, educación, economía y finanzas, industria y medio ambiente. Por otro lado, los algoritmos multiclasificadores se han mostrado particularmente eficientes en el trabajo sobre espacio de datos grandes y complejos. El siguiente trabajo tiene como propósito analizar el comportamiento de un nuevo algoritmo multiclasificador incremental, que se adapta a los cambios de conceptos, utilizando diferentes clasificadores bases no incrementales para procesar datos sintéticos discretos generados bajo el concepto LED. Este algoritmo, presentado por primera vez en el congreso Informática 2013, está basado en la familia MultiCIDIM desarrollada por investigadores de la Universidad de Málaga. Se utiliza el entorno de trabajo MOA (Massive Online Analysis) para implementar el algoritmo, generar los datos sintéticos y realizar los experimentos.

Abstract Classification algorithms that adapt to changing concepts in data streams mining are currently very important for many applications such as bioinformatics, medicine, education, economics and finance, industry and environment. Furthermore, incremental ensemble algorithms have proven particularly efficient at work on large and complex data spaces. The following paper aims to analyze the behavior of a a new incremental ensemble algorithm, which adapts to changes in concepts, using different not incremental base classifiers to process discrete synthetic data generated under the LED concept. This algorithm, first introduced at the Informática 2013 Congress, is based on the MultiCIDIM family developed by researchers at the University of Málaga. The MOA (Massive Online Analysis) work environment is used to implement the algorithm, to generate synthetic data and to conduct experiments.

Palabras Clave
Aprendizaje incremental, flujos de datos, cambio de concepto, clasificadores múltiples

[1] *Universidad de Granma, Cuba, averdeciac@udg.co.cu, rtase@udg.co.cu, aortizd@udg.co.cu, ifriasb@grm.uci.cu*
[2] *IMNi, Niquero, Cuba, dlnaranjo@niquero.capgrm.co.cu*
***Autor para Correspondencia**

1. Introducción

En las últimas décadas, el almacenamiento, organización y recuperación de la información se ha automatizado gracias a los sistemas de bases de datos, pero la ubicuidad de la información en formato electrónico ha empezado a ser patente a finales de siglo XX con la irrupción de Internet. Como resultado de este tremendo crecimiento, los datos en bruto, se han convertido en una vasta fuente de información. Gran parte de esta información es histórica, es decir, representa transacciones o situaciones que se han producido. Aparte de su función de ”memoria”, la información histórica es útil para explicar el pasado, entender el presente y predecir la información futura. La mayoría de las decisiones de empresas, organizaciones e instituciones se basan en información sobre experiencias pasadas extraídas de fuentes muy diversas por lo que el verdadero valor de los datos radica en la posibilidad de extraer de ellos información útil para la toma de decisiones o la exploración y comprensión de los fenómenos que le dieron lugar [1]. Debido a tal crecimiento y variedad de la información se hace necesario e importante el análisis de datos en ramas como: Bioinformática, medicina, educación, economía y finanzas, industria, medio ambiente, entre otras. Más importante aún es además del conocimiento que puede inferirse y la capacidad de poder usarlo, tener un conjunto de reglas que a partir de los antecedentes, comportamiento y otras características de los datos nos permitan predecir su comportamiento futuro [2].

Para el análisis de grandes cantidades de datos normalmente se usan técnicas de Minería de Datos (Data Mining) que no es más que la búsqueda de patrones e importantes regularidades en bases de datos de gran volumen [3]. La clasificación es una de las tareas más importante dentro de la Minería de Datos. En ella, cada instancia (o registro de la base de datos) pertenece a una clase, la cual se indica mediante el valor de un atributo que llamamos la clase de la instancia. Este atributo puede tomar diferentes valores discretos, cada uno de los cuales corresponde a una clase. El resto de los atributos de la instancia (los relevantes a la clase) se utilizan para predecir la clase. El objetivo es predecir el atributo clase en nuevos registros en los que se desconoce éste.

Un flujo de datos es una secuencia muy grande, potencialmente infinita, de pares (x,y) que se van adquiriendo a lo largo del tiempo, donde x representa los atributos de los datos de entrada e y su clase correspondiente, a estos pares suele llamárselos instancias o experiencias. Estas experiencias proceden de entornos dinámicos y no se tienen previamente almacenadas, lo que obliga a procesarlas secuencialmente, de forma incremental.

Uno de los problemas fundamentales del aprendizaje incremental se debe a que la función objetivo puede depender del contexto, el cual no es recogido mediante los atributos de los datos de entrada. Como consecuencia, cambios ocurridos en el contexto pueden inducir variaciones en la función objetivo, dando lugar a lo que se conoce como cambio de concepto (concept drift) [4].

Es difícil en la actualidad encontrar un clasificador eficiente para resolver cada situación planteada. Se ha comprobado en investigaciones anteriores que ciertas partes del espacio de datos son mejor modeladas por un método de clasificación y otra parte del espacio, por un clasificador diferente. Esto ha originado que utilizar una combinación de clasificadores (Multiclasificación) sea una buena alternativa. Varios métodos para creación de combinaciones de clasificadores ya han sido propuestos, aunque, no existe una clara forma de saber cuál método de ensamble es mejor que otro, o cuando emplear un determinado método o cuando otro diferente.

El objetivo de este trabajo es analizar el comportamiento de un nuevo algoritmo multiclasificador incremental, que se adapta a los cambios de conceptos, utilizando diferentes clasificadores bases no incrementales para procesar datos sintéticos discretos generados bajo el concepto LED[1].

Este algoritmo, fue desarrollado en la Universidad de Granma con la asesoría de un grupo de trabajo de la Universidad de Málaga y el apoyo de investigadores de otras universidades cubanas. El mismo fue publicado y sometido a debate en La Segunda Conferencia Internacional de Ciencia de la Computación e Informática del Congreso Informática 2013 desarrollado en Cuba [5].

[1] Light Emitting Diode, diodo emisor de luz

1.1 Antecedentes. Trabajos relacionados

Los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC [6] están basados en modificaciones hechas al conocido algoritmo SEA (Streaming Ensemble Algorithm) [7], como son: utilizar como clasificador base a CIDIM, cambiar la forma de entrada y salida de los clasificadores e introducir el control por filtros correctores, en la última versión.

Los artículos de Kolter y Maloof [8] y [9] destacan algunas deficiencias que pueden presentar los algoritmos basados en el algoritmo SEA para responder y adaptarse eficientemente a los cambios de conceptos:

- Los clasificadores, miembros del multiclasificador, dejan de aprender una vez que son creados.
- El método de remplazar clasificadores no ponderados suele no converger muy rápido a la hora de adaptarse a un nuevo concepto.
- El algoritmo trabaja dividendo los datos en bloques por lo que no se puede adaptar al aprendizaje incremental ya que cuando una nueva instancia llega éste no puede tomar acción inmediata.

Resumiendo, SEA y los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC, por ser modificaciones a éste, carecen de la habilidad de adaptarse a cambios rápidos y abruptos y bajan mucho su precisión de predicción global cuando se enfrentan a cadenas de datos con pocas instancias en el tiempo.

Debido a estas deficiencias se propuso un nuevo algoritmo, basado en MultiCIDIM-DS con las siguientes características:

- Utiliza votación ponderada. Da la posibilidad de conocer y ajustar, a través de los pesos de cada clasificador base, la confianza que se tiene en estos en cada instante de tiempo.
- Los pesos de los clasificadores bases son penalizados si hay fallos y son bonificados si hay acierto. Esto permite que cada clasificador pueda permanecer más tiempo dentro del multiclasificador si su comportamiento es estable, favoreciendo el tratamiento de conceptos recurrentes.
- Utiliza un umbral para eliminar de forma rápida los clasificadores bases ineficientes, favoreciendo la adaptación a los cambios de conceptos [5].

2. Metodología computacional

2.1 Entorno de trabajo utilizado

El algoritmo multiclasificador fue programado en el lenguaje Java, sobre el entono de desarrollo Netbeans IDE 7.0.1. Fue implementado en compatibilidad y bajo las exigencias de MOA (Massive Online Analysis) [10] uno de los más completos entornos de trabajos para minería de datos que está relacionado con el proyecto WEKA (Waikato Environment for Knowledge Analysis) [11]. MOA incluye una colección de

algoritmos para el aprendizaje automático (Clasificación, regresión y agrupamiento) para trabajar sobre flujos de datos en los que están presentes cambios de concepto. Además incluye herramientas para evaluar los algoritmos y generar datos de forma artificial, incluyendo la posibilidad de simular cambios de conceptos.

2.2 Algoritmo general del multiclasificador incremental

Trabaja sobre flujos de datos y se adapta a los cambios de conceptos.

$n \in N^*$: Número de experiencias (potencialmente infinito).
$m \in N^*$: Número de clasificadores bases.

$DS = \left\{ \vec{x}, y \right\}_n^1$: DS: Flujo de datos de entrada. Estructurados en $\vec{x}$: Vector de atributos; y: Clase.

$E = \{cb, w\}_m^1$: E: Multiclasificador. Compuesto por cb: Clasificador base; w: peso asociado al clasificador.

Λ, λ: Valores de las clases predichas: global y local.

θ: Umbral utilizado para eliminar clasificadores bases del multiclasificador.

ne: número de experiencias de un bloque.

Bloqexp: vector de bloques de experiencias.

Algorithm 1 Algoritmo general

```
1:  Construir el primer bloque de experiencias, Bloqexp;
2:  Crear el multiclasificador con el mínimo número de clasi-
    ficadores, E;
3:  Bloqexp ← 0;
4:  for i = ne + 1 to n do
5:     Bloqexp.add(DS_i);
6:     if i mod ne = 0 then
7:        NewClassifier.build(Bloqexp);
8:        Agregar nuevo clasificador a E;
9:        Actualizar E;
10:       for j = 1 to m do
11:          if E_j.w < θ then
12:             E_j.delete;
13:          end if
14:       end for
15:       Bloqexp ← 0;
16:    end if
17:    ∧ = classify(E, x_i)
18:    if ∧ = y_i then
19:       Actualizar E;
20:       for j = 1 to m do
21:          if E_j.w < θ then
22:             E_j.delete;
23:          end if
24:       end for
25:    end if
26: end for
```

2.3 Características de los clasificadores bases utilizados

CIDIM (Control de Inducción por División Muestral). Este algoritmo fue propuesto por Ramos, Morales y Villalba [12], [13], investigador de la Universidad de Málaga.

Sus características principales son:

- Atributos categóricos.
- Elevada precisión, genera árboles de tamaño reducido.
- División del conjunto de entrenamiento, la agrupación de valores consecutivos y un control local para detener la inducción.
- Una hoja se expande sólo si se produce una mejora en la precisión calculada sobre el subconjunto de control (CLS).
- Termina cuando no hay más atributos para expandir.

2.3.1 Algoritmo J48

La versión utilizada de este algoritmo es la implementada en el entorna de trabajo WEKA [11] del conocido algoritmo C4.5, desarrollado por Quinlan en 1993 [14], como una extensión (mejora) del algoritmo ID3 que el mismo desarrollo en 1986 [15].

Las características principales del algoritmo C4.5 son:

- Forma parte de la familia de los TDIDT (Top Down Induction Decision Trees).
- Permite trabajar con valores continuos para los atributos.
- Los árboles son poco frondosos.
- Utiliza el método ”divide y vencerás”para generar el árbol de decisión inicial.
- Es Recursivo.

2.3.2 Naive Bayes

Naive Bayes es un algoritmo de clasificación muy conocido por su simplicidad y bajo costo computacional. Para la experimentación, en este trabajo, se utiliza la versión no incremental de este algoritmo implementada en WEKA [11].

Las Redes Bayesianas (BN) representan las dependencias que existen entre los atributos a través de una distribución de probabilidad condicional en un grafo dirigido no cíclico [16].

El clasificador Naive Bayes (NB) es un caso especial de una red bayesiana, en el que se asume que los atributos son condicionalmente independientes dado un atributo clase [17]. Este clasificador es muy eficiente en ciertos dominios, debido a que es muy robusto ante atributos irrelevantes.

2.3.3 Algoritmo NBTree

La versión del algoritmo NBTree utilizada en este trabajo es la implementada en el entorno de trabajo WEKA [11].

NBTree es un algoritmo creado como una solución híbrida, debido a que la eficiencia del algoritmo Naive Bayes no alcanza el mismo rendimiento de los árboles de decisión cuando el set de datos es muy grande. El algoritmo NBTree intenta utilizar las ventajas de los clasificadores, árbol de decisión para la segmentación y Naive Bayes para la acumulación de la evidencia. En este algoritmo, el árbol de decisión segmenta los datos, y cada segmento de datos, representado por una hoja, es descrito por un clasificador NaiveBayes [18].

2.4 Características del conjunto de datos utilizado

El objetivo es tratar de predecir el dígito mostrado por una pantalla-LED de siete segmentos, donde cada atributo tiene un **x** (10 % es el más utilizado) por ciento posible de cambio. El generador utilizado produce para los experimentos 24 atributos binarios, de los cuales 17 son irrelevantes. El cambio de concepto es simulado a través del intercambio de atributos relevantes. Los datos utilizados en los experimentos son generados utilizando las funcionalidades del entorno de trabajo MOA [10].

3. Resultados y discusión

3.1 Experimento 1. Análisis sobre un mismo concepto

En el primer experimento se utilizan 10 000 experiencias generadas bajo el concepto LED. No existen cambios de conceptos en los datos. Se utilizan tres conjuntos de datos, los tres bajo el concepto LED, pero variando el nivel de ruido en 5 %, 10 % y 15 %. Se prueba el multiclasificador con los cuatro clasificadores bases, uno a la vez, sobre el mismo conjunto de datos. Para cada conjunto de datos y para cada clasificador base se realizan 10 pruebas de las cuales se toma el promedio del los por cientos de las experiencias correctamente clasificadas.

En el análisis de los datos, del primer experimento, puede apreciarse que los mejores por cientos de clasificación se obtienen con los clasificadores bases NBTree (LED al 5 % y al 15 % de ruido) y Naive Bayes (LED al 10 % de ruido).

3.2 Experimento 2. Análisis agregando un punto de cambio de concepto en los datos

En el segundo experimento también se utilizan 10 000 experiencias generadas bajo el concepto LED. Igualmente se utilizan tres conjuntos de datos LED variando el nivel de ruido en 5 %, 10 % y 15 %. Como diferencia, para este experimento, en los tres conjuntos de datos se inserta un punto de cambio de concepto a partir de la experiencia 5000. Se utiliza la misma base LED pero variando los atributos relevantes.

Nuevamente, se prueba el multiclasificador con los cuatro clasificadores bases, uno a la vez, sobre el mismo conjunto de datos. Para cada conjunto de datos y para cada clasificador base se realizan 10 pruebas de las cuales se toma el promedio del los por cientos de experiencias correctamente clasificadas.

En el análisis de los datos, del segundo experimento, puede apreciarse que los mejores por cientos de clasificación se obtienen con los clasificadores bases CIDIM (LED al 10 % y al 15 % de ruido) y NBTree (LED al 5 % de ruido).

Para ilustrar el por qué del cambio en los resultados, se incluyen a continuación los gráficos para el segundo experimento.

Se debe de resaltar, que en la figura 1, los resultados con los clasificadores bases Naive Bayes y NBTree fueron idénticos por lo que en la figura uno solapa al otro.

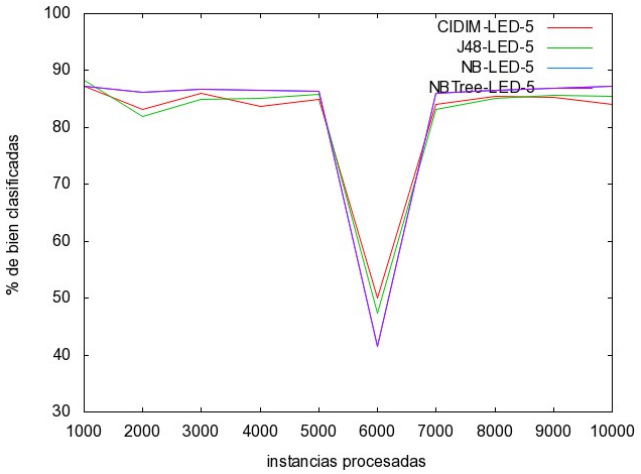

Figura 1. LED con 5 % de ruido. Punto de cambio de concepto a partir de la experiencia 5000.

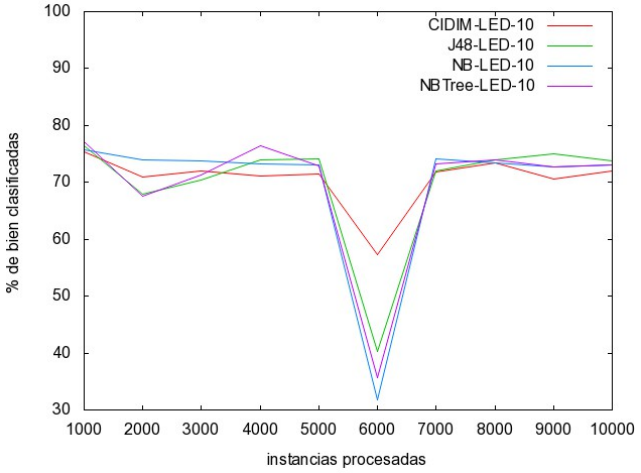

Figura 2. LED con 10 % de ruido. Punto de cambio de concepto a partir de la experiencia 5000.

Se puede ver claramente, en los tres gráficos, que los resultados de CIDIM alrededor del punto de cambio son mucho mejores que el resto de los clasificadores. Es decir, la caída del por ciento de acierto a partir del punto crítico es mucho menor. Por lo que podría afirmarse que CIDIM se comporta, en el experimento, mucho mejor frente al cambio de concepto que el resto de los clasificadores bases.

Tabla 1. Promedios de los por ciento de experiencias correctamente clasificadas. Base LED sin cambio de concepto.

Bases / Clasificador	CIDIM	NB	J48	NBTree
LED-5	85,79 %	87,12 %	85,97 %	87,14 %
LED-10	72,74 %	73,66 %	73,13 %	73,29 %
LED-15	61,37 %	62,26 %	61,26 %	62,3 %

Tabla 2. Promedios de los por ciento de experiencias correctamente clasificadas. Concepto LED con cambio de concepto a partir de la experiencia 5000.

Bases / Clasificador	CIDIM	NB	J48	NBTree
LED-5	81,4 %	82,14 %	81,31 %	82,14 %
LED-10	70,6 %	69,49 %	69,8 %	69,41 %
LED-15	60,77 %	58,37 %	57,62 %	59,06 %

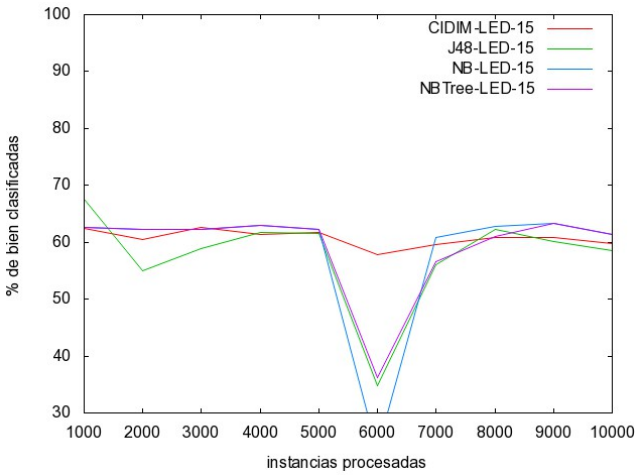

Figura 3. LED con 15 % de ruido. Punto de cambio de concepto a partir de la experiencia 5000.

En la tabla 3 se ilustran los promedios de tiempo que demora el algoritmo, con cada clasificador base, en procesar los datos. Se debe de resaltar que con el clasificador base NBTree el tiempo de procesamiento de los datos es mucho mayor que con el restos de los clasificadores bases. Esto podría traer problemas con bases de datos mucho más grandes (en el orden de los millones).

4. Conclusiones

En general, el algoritmo multiclasificador obtuvo buenos resultados, en ambos experimentos, con cada uno de los clasificadores bases seleccionados, frente a cada uno de los conjuntos de datos estudiados.

Con los algoritmos NBtree y Naive Bayes, utilizados como clasificadores bases, se obtuvieron muy buenos resultados (en cuanto al por ciento de experiencias correctamente clasificadas) en el primer experimento, es decir, frente a conjuntos de datos donde no existen cambios de conceptos. Sin embargo los tiempos de ejecución del NBTree fueron mucho más altos que los tiempos del resto de los clasificadores bases en ambos experimentos.

Por otro lado, los resultados (en cuanto al por ciento de experiencias correctamente clasificadas) del clasificador base CIDIM mejoraron mucho en el segundo experimento; esto se debió a su baja caída de precisión en los alrededores del punto de cambio, las cuales no pudieron mantener el resto de los clasificadores bases.

Es importante resaltar, que todos estos resultados fueron obtenidos frente a datos generados sintéticamente bajo el concepto LED, el cual es un concepto complejo de clasificar aunque todos sus atributos sean discretos.

Referencias

[1] Ruiz, R. Heurísticas de selección de atributos para datos de gran dimensionalidad. Departamento de Lenguajes y Sistemas Informáticos. Sevilla, Universidad de Sevilla, 2006.

[2] Caballero, Y. Aplicación de la Teoría de los Conjuntos Aproximados en el Preprocesamiento de los Conjuntos de Entrenamiento para Algoritmos de Aprendizaje Automatizado. Tesis de doctorado, Universidad Central ”Marta Abreu”de la Villas, 2007.

[3] Michalsky, R. and G. Tecuci. Machine Learning: A Multistrategy Approach. EE.UU, Morgan Kauffinan, 1994.

[4] Wang, H.; Wei F.; Philip Y.; Jiawei H. Mining Concept-Drifting Data Streams Using Ensemble Classifiers. In 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, DC, 2003.

[5] Ortíz, A. Propuesta de algortimo multiclasificador para minar flujos de datos y adaptarse a cambios de concepto. Segunda Conferencia Internacional de Ciencia de la Computación e Informática. Congreso Informática 2013.

Tabla 3. Promedio del tiempo (en segundos), que demora en procesar los datos con cada clasificador base.

Bases / Clasificador	CIDIM	NB	J48	NBTree
Tiempo (experimento 1)	2	3	1	49
Tiempo (experimento 2)	3	4	2	55

[6] Del Campo, J. Nuevos Enfoques en el Aprendizaje Incremental. Tesis de doctorado. Universidad de Málaga, España, 2007.

[7] Street, W. and K. YongSeog. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. In 7th International Conference on Knowledge Discovery and Data Mining. New York City, NY. 2001.

[8] Kolter, J. y M. Maloof. Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. in 3rd International IEEE Conference on Data Mining. Melbourne, FL, 2003.

[9] Yue, S. Mining Concept Drifts from Data Streams Based on Multiclassifiers. In Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology. Beijing, China. 2007.

[10] Bifet A. and R. Gavalda. Learning from time-changing data with adaptive windowing. SIAM International Conference on Data Mining, MOA. 2007.

[11] Hall, M.; E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. SIGKDD Explorations Newsletter, Weka. ISSN 1931-0145. 2009.

[12] Ramos-Jiménez G., Morales-Bueno R. y A. Villalba-Soria. CIDIM. Control of Induction by Sample Division Methods. En Proceedings of the International Conference on Artificial Intelligence (ICAI-2000), 2000.

[13] Ramos-Jiménez G. Nuevos Desarrollos en Aprendizaje Inductivo. Tesis Doctoral, Universidad de Málaga, España. 2001.

[14] Quinlan, J. R. C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[15] Quinlan, J. R. Induction of decision trees. Machine Learning 1: 81-106, 1986.

[16] Sahami, M. Learning Limited Dependence Bayesian Classifiers. 2th International Conference on Knowledge Discovery in Databases (KDD96), Menlo Park, CA, AAAI Press. 1996.

[17] Singh, M. and G. M. Provan. Efficient Learning of Selective Bayesian Network Classifier. International Conference on Machine Learning. Philadelphia, PA., Computer and Information Science Department, University of Pennsylvania. 1995.

[18] Kohavi, R. Scaling up the accuracy of naive-Bayes classifiers: a decision tree hybrid. Procedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD96), Portland, OR., AAAI Press, 1996.