Selección de Atributos en una Base de Datos de Facturación Eléctrica aplicando Programación Cóncava

Manuel Mejía-Lavalle, Juan Frausto Solís[1], Francisco Javier García[2]
Instituto de Investigaciones Eléctricas
Reforma 113, 62490 Cuernavaca, Morelos, México
[1] ITESM – Cuernavaca
Reforma 182-A, 62589 Temixco, Morelos, México
[2]Universidad de Granada, España
mlavalle@iie.org.mx,juan.frausto@itesm.mx,fjgc@decsai.ugr.es

Resumen. La selección de atributos es una actividad crucial del pre-procesamiento de datos cuando se desea realizar descubrimiento de conocimiento en bases de datos. Su principal objetivo es eliminar atributos irrelevantes y/o redundantes para obtener problemas computacionalmente tratables, sin afectar la calidad de la solución. Diversas técnicas han sido propuestas principalmente bajo dos enfoques: *wrapper* y *ranking*. En este artículo se evalúa un novedoso enfoque propuesto por Bradley [1] que utiliza programación cóncava buscando minimizar el error de clasificación y el número de atributos necesarios para realizar la tarea. A diferencia de Bradley, nosotros evaluamos esta técnica usando una base de datos de facturación eléctrica en México, tres veces mayor que las empleadas anteriormente. Además comparamos los resultados obtenidos contra lo que arrojan las técnicas tradicionales evaluando: reducción de atributos, tiempo de procesamiento, tamaño del conocimiento descubierto y calidad de la solución. De esta experimentación resulta que lo propuesto por Bradley posee características promisorias susceptibles de mejorar.

1 Introducción

La minería de datos se aplica principalmente cuando se han almacenado grandes cantidades de datos históricos con la expectativa de explotarlos, buscando el conocimiento implícito en esta información, o dicho en otras palabras, se busca determinar tendencias o patrones de comportamiento que permitan mejorar los procedimientos actuales, ya sean éstos de mercadotecnia, producción, operación, mantenimiento o cobranza, entre otros. Para lograr aprovechar esta enorme cantidad de información se han desarrollado algoritmos y herramientas especializadas en el descubrimiento automático del conocimiento oculto en dicha información; el proceso de la extracción no trivial de información importante que está implícita en los datos se conoce como descubrimiento de conocimiento (Knowledge Discovery in Databases ó KDD), en donde la etapa de minería de datos juega un papel central en dicho proceso.

Sin embargo se ha visto que cuando las bases de datos a minar llegan a ser muy grandes, los algoritmos mineros son muy lentos, requiriéndose demasiado tiempo para procesar la información, por lo que el problema se vuelve intratable. Una forma de atacar este problema es reducir los datos antes de aplicar la minería [2]. En particular la

selección de atributos aplicada como una etapa de pre-procesamiento a la minería ha resultado útil, pues busca eliminar los atributos irrelevantes y/o redundantes que hacen que las herramientas de minería se vuelvan ineficientes, pero sin afectar la calidad de la clasificación que realiza el algoritmo minero; es más, en ocasiones el porcentaje de instancias correctamente clasificadas llega a ser más alto al emplear la selección de atributos, pues los datos a minar quedan libres de ruido o datos que provocan que la herramienta minera genere más nodos de los necesarios [3].

La selección de atributos se ha realizado aplicando métodos *wrapper* y filtro. Los métodos *wrapper*, aunque son eficaces para eliminar atributos irrelevantes y redundantes, son muy lentos pues aplican numerosas veces el algoritmo minero, variando en cada ejecución el número de atributos, siguiendo algún criterio de búsqueda y paro [4]. Los métodos tipo filtro emplean algún tipo de medida de ganancia de información, distancia o consistencia, entre el atributo y la clase, siendo éstos mucho más eficientes que los *wrapper* [5]; sin embargo, debido a que miden la importancia de cada atributo en forma aislada, no pueden detectar si existen atributos redundantes, y tampoco son capaces de determinar si la combinación de dos o más atributos, aparentemente irrelevantes en forma aislada, se pueden transformar en relevantes [6].

En este artículo se evalúa el empleo de técnicas de optimización para selección de atributos, como una alternativa a los métodos que tradicionalmente se han propuesto. En particular se evalúa un interesante y novedoso enfoque propuesto por Bradley y Mangasarian [1] el cual utiliza programación cóncava buscando minimizar el error de clasificación y minimizando al mismo tiempo el número de atributos necesarios para realizar la tarea. El enfoque de Bradley parecería contar con la eficiencia de los métodos filtro y con la eficacia de los métodos *wrapper,* según los resultados reportados por este autor, de ahí la importancia de la presente evaluación.

Sin embargo, a diferencia de [1], nosotros evaluamos esta técnica usando una base de datos de facturación eléctrica en México, y que es del mundo real, buscando la detección de usos ilícitos de la energía; esta base de datos tiene, al menos, tres y media veces más instancias que las que emplearon en [1] para experimentación, lo cual hace este problema más difícil de atacar. En efecto, actualmente la Gerencia Comercial (GC) de la Comisión Federal de Electricidad (CFE) enfrenta el problema de detectar de manera más certera a aquellos usuarios que incurren en el uso ilícito de la energía, con el fin de reducir las pérdidas por este concepto. A la fecha se cuenta con mucha información histórica en el Sistema Comercial (SICOM); esta base de datos tiene varios años en operación y por tanto una gran cantidad de datos acumulados.

Buscando hacer factible minar esta gran base de datos, en forma eficiente y eficaz, en el presente artículo se presenta la evaluación y comparación realizada de diferentes métodos tipo filtro, *wrapper* y el enfoque de Bradley. La evaluación que realizamos toma en cuenta no sólo la calidad de la clasificación y el tiempo de procesamiento logrados por la aplicación previa de cada método seleccionador de atributos, sino también considera el tamaño del conocimiento descubierto, el cual, entre más pequeño, resulta más fácil de interpretar y que, para el caso que nos ocupa, es de especial interés; también considera el costo-beneficio obtenido al aplicar cada método. El artículo se organiza de la siguiente

manera: en la sección 2 se presenta una reseña de los trabajos más citados y recientes relacionados con la selección de atributos; en la 3 se describe el dominio, o base de datos, sobre el que se experimentó; la sección 4 detalla la experimentación realizada, concluyéndose en la sección 5.

Aunque nuestro trabajo se enfoca a datos del SICOM, las lecciones aprendidas al trabajar en un caso del mundo real pueden ser de utilidad para atacar problemas similares.

2 Trabajo Previo

El surgimiento de las grandes Bases de Datos (Very Large Databases ó VLDB) conlleva nuevos retos que los algoritmos mineros de los 90´s son incapaces de atacar de manera eficiente. Es por eso que se requieren nuevos algoritmos mineros especializados en VLDB. En particular estos nuevos algoritmos requieren de la reducción de los datos, lo cual ha dado origen a la sub disciplina de Reducción de Datos (Data Reduction o Dimensional Reduction). La Reducción de Datos busca eliminar variables, atributos e instancias que no aporten información (o no aporten mucha información) al proceso de KDD, o agrupar los valores que una variable puede tomar (discretizar). Estos métodos, generalmente, se aplican antes de realizar la minería propiamente. Aunque en los años 90´s las actividades previas a la Minería eran mínimas, y se dejaba casi todo el trabajo de descubrimiento al algoritmo minero, cada vez más se ha visto la necesidad de ejecutar una serie de pasos previos a la Minería, que en general se denominan el Pre-procesamiento de Datos. Este pre-procesamiento permite al algoritmo minero realizar su trabajo de manera más eficiente (más rápido) y eficaz (mejor calidad de respuesta).

Buscando poder tratar las Bases de Datos muy grandes (que son las que poseen decenas de atributos y cientos de miles de instancias), han surgido dos grandes grupos de métodos: los llamados *wrappers*, y los filtro.

Los métodos tipo *wrapper* tienen como característica principal emplear el mismo algoritmo de inducción tanto para realizar la selección de variables relevantes como para realizar el proceso inductivo de clasificación, o proceso de Minería. El algoritmo minero se ejecuta múltiples veces variando el número de atributos presentes en cada corrida: para problemas con 100 atributos, el total de estados y corridas posibles llegaría a $1.26 \text{ X } 10^{30}$, lo que destaca que usar un método exhaustivo está fuera de consideración, excepto para Bases de Datos con muy pocos atributos.

Los métodos tipo filtro usan algoritmos que son independientes al algoritmo minero y se ejecutan como un paso anterior a la Minería en sí. Dentro de los métodos tipo filtro se encuentran aquellos algoritmos para selección de variables relevantes (denominados generalmente como *feature selection*) y los algoritmos de muestreo de instancias (conocidos como algoritmos *sampling* o *subsampling*).

Existe una gran variedad de métodos tipo filtro para selección de atributos, en donde el algoritmo ID3 [7] (y sus extensiones) es considerado por algunos autores como una de las primeras formas propuestas de filtrar, sin embargo ID3 más bien es un algoritmo Minero. Dentro de los métodos filtro pioneros y más citados están FOCUS [8], que realiza una búsqueda exhaustiva de todos los posibles subconjuntos de atributos, lo cual lo hace sólo

apropiado para problemas con pocos atributos, y RELIEF [9] que tiene el inconveniente de no poder detectar atributos redundantes.

Dentro de los métodos filtro para seleccionar atributos destacan los denominados *ranking,* pues ordenan los atributos basándose en su relevancia con respecto al atributo de clase o foco de atención. Por ejemplo, Koller [10] usa una distancia métrica denominada *cross-entropy* o *KL-distance*, que compara dos distribuciones de probabilidad e indica el error, o distancia, entre ellas, y logra reducciones de atributos de alrededor del 50%, manteniendo la calidad de clasificación y pudiendo reducir tiempos de procesamiento (por ejemplo, de 15 horas para el esquema *wrapper*, a 15 minutos para el algoritmo propuesto). El resultado final es "sub óptimo" pues asume independencia entre los atributos, lo que no siempre es válido. Piramuthu [6] evalúa 10 diferentes medidas de distancia atributo-clase, empleando *Sequential Forward Search* (SFS) que agrega los mejores atributos a un subconjunto de atributos, de tal suerte que al final se tiene el mejor subconjunto de atributos, más que simplemente un grupo formado por cada uno de los mejores atributos. No se comparan resultados con y sin la selección de atributos, por lo que no se puede concluir sobre la efectividad de cada medida y aunque SFS logra reducir el espacio de búsqueda, aún se tienen que realizar múltiples corridas variando el subconjunto de atributos, por lo que resulta computacionalmente costoso.

Otro esquema tipo filtro denominado SAPPP [5] es capaz de manejar atributos continuos y discretos y consiste en seleccionar subgrupos con incrementos de instancias y usarlas para alimentar a un algoritmo constructor de un árbol de decisión y obtener así secuencias de atributos relevantes: si existen atributos adicionales con respecto al árbol anterior, se verifica si no usarlos afectaría la calidad de clasificación: si no afecta, se descartan (se considera que son irrelevantes) y se detiene el proceso. Se obtuvieron reducciones en tiempo de 30%, logrando mantener la calidad de clasificación. Aún se debe resolver la determinación del número de instancias de inicio y la selección de la secuencia de incrementos.

Molina [11] trató de caracterizar 10 diferentes métodos para seleccionar atributos midiendo la habilidad de cada uno de los métodos para manejar atributos redundantes, atributos irrelevantes y número de instancias. Al final no se pudieron obtener diferencias significativas, observándose que en general los diferentes métodos dependían de los datos con que eran alimentados. Stoppiglia [12] propone introducir una variable aleatoria adicional a la Base de Datos de tal suerte que, a la hora de realizar el *ranking* de las variables, todas aquellas variables que hallan obtenido calificaciones inferiores a la variable *random,* podrán considerarse como irrelevantes. Este criterio representa una forma alterna a realizar una prueba estadística de Fisher (Fisher´s test). Los resultados muestran que el método logra una buena selección de atributos, comparable a otras técnicas. Este método es atractivo por su sencillez, aunque falta ser sometido a más experimentos para probar su eficacia: por ejemplo, pudiera ser que la variable *random* en la mayor parte de los casos sólo logre eliminar/ discriminar muy pocos atributos (o ninguno), lo cual le restaría poder como método de selección de atributos.

Otras propuestas de selección de atributos exploran el uso de redes neuronales, lógica difusa, algoritmos genéticos, y *Support Vector Machines* [3], pero son costosos

computacionalmente. En general se observa que los métodos que han sido propuestos: a) se prueban en bases de datos pequeñas, académicas y/o simuladas; b) el resultado varía con el dominio; c) a mayor calidad del resultado, mayor costo computacional; d) dependen de un adecuado *tuning*; e) no evalúan el tamaño del conocimiento extraído, que es clave para comprender el fenómeno que subyace en los datos; f) no consideran el costo-beneficio obtenido; y g) caen en esquemas *wrapper* o *ranking*.

En este sentido, el esquema propuesto por Bradley [1] resulta atractivo pues, además de obtener excelentes reducciones en el número de atributos, lo logra en poco tiempo y con buena calidad de solución. Este esquema se describe, evalúa y compara en la sección 4.

3 El Problema de la Detección del Uso Ilícito de la Energía

La función principal de la GC de la CFE es distribuir entre los usuarios consumidores la energía eléctrica que se produce en las diferentes plantas generadoras de México. Relacionado con la distribución, la GC enfrenta diferentes problemas que le impiden recuperar, en ingresos, el 100% de la energía que reciben para venta. Se calcula que existe en promedio un 21% de pérdidas en la distribución de la energía. Estas pérdidas se deben principalmente a dos tipos de problemas: técnicos y de control administrativo. Las pérdidas de energía por problemas técnicos están calculadas en un 10% y se necesitaría una gran inversión en nuevas tecnologías de distribución para poder reducir este porcentaje. En cuanto al otro 11% de las pérdidas, por problemas de control administrativo, se pueden clasificar en tres categorías de anomalías: a) errores en la facturación, b) errores en la medición, y c) uso ilícito de la energía. El porcentaje de impacto de las dos primeras es mínimo y el mayor problema se tiene con el uso ilícito de la energía, es decir, personas que se roban la energía y por lo tanto no la pagan.

La GC ha enfrentado este problema aplicando diferentes mecanismos (como incrementar la frecuencia de lecturas a medidores de usuarios sospechosos o instalar equipo para lecturas en automático) y ha logrado reducir el porcentaje de pérdidas por usos ilícitos, lo cual representa la recuperación de varios miles de millones de pesos mexicanos. Dado que el problema persiste, es importante enfrentarlo con otras tecnologías, por lo que se propuso aplicar la minería de datos, mediante la obtención de patrones de comportamiento de ilícitos, como alternativa de solución que no requiere una gran inversión y que ha probado su efectividad en casos similares, como la detección de fraudes en tarjetas de crédito.

La información a minar se encuentra en el SICOM, que es un sistema desarrollado en lenguaje Cobol y que contiene alrededor una veintena de tablas con información de contrataciones, facturación y cobranza de usuarios a nivel nacional. Este sistema no fue diseñado con fines de detección de usuarios ilícitos, sin embargo contiene un campo denominado *Tipo-adeudo* en el cual se registra, entre otros, si el adeudo es por incurrir en usos ilícitos. Tomando como centro la tabla que tiene esta información, se concatenó con tres tablas más, obteniéndose una mina con 2,770 registros y con atributos como: *Registro permanente de usuario (RPU), Año, Mes, Tipo-adeudo, Dígito, kWh, Energía, Cve-facturación, Total, Status, Giro, Tarifa, Nombre, Carga-Instalada, Carga-Contratada*, y otros que en total suman 21 atributos, donde el atributo *Tipo-adeudo* es el que indica un

uso ilícito, y es nuestra columna de clase o foco de atención. Por motivos de confidencialidad no se reporta aquí el significado específico de cada uno de los atributos en cuestión. Con esta mina se procedió a realizar diversos experimentos para evaluar el esquema propuesto por Bradley [1] y compararlo contra diferentes métodos *ranking* y *wrapper*; estos experimentos se describen a continuación.

4 Evaluación de la Programación Cóncava

4.1 Detalles de implementación

Mangasarian y Bradley han trabajado extensamente en la búsqueda y aplicación de métodos de optimización a problemas de Minería de Datos [13]. Entre estos trabajos destaca su esquema para clasificar y seleccionar atributos, el cual obtuvo excelentes resultados al ser comparado contra un método basado en *Support Vector Machines* (SVM). La evaluación realizada por nosotros se fundamenta en el modelo presentado en [1] y otras publicaciones de los mismos autores [14] que soportan tal modelo.

La idea esencial del modelo de Bradley, denominado *Feature Seletion via concaVe minimization* (FSV), se compone de dos partes: la clasificación de instancias y la selección de atributos.

En cuanto a la clasificación, el modelo de programación matemática busca minimizar el promedio de las distancias de las instancias incorrectamente clasificadas por la separación de una recta, plano o hiperplano.

En cuanto a la selección de atributos, se busca minimizar el número de atributos requeridos para realizar una buena separación de clases, bajo el supuesto de que es posible obtener un hiperplano, en menos dimensiones que los datos originales, tal que la clasificación siga siendo aceptable.

Estas ideas, extensamente expuestas, demostradas, experimentadas y discutidas en [1] [13][14] fueron implementadas por nosotros, específicamente para ser usadas y ejecutadas con la herramienta de optimización denominada General Algebraic Modeling System (GAMS) [15] (a diferencia de Bradley que empleó CPLEX en sus experimentos). La Tabla 1 muestra (en forma editada) las instrucciones específicas en GAMS del modelo FSV, donde *TABLE A(I,J)* contiene las *I* instancias de la clase "no-ilícito" y *TABLE B(L,J)* las *L* instancias de la clase "ilícito", para las *J* características o atributos. *Y* y *Z* son las distancias o magnitudes de las instancias, clase *A* y *B* respectivamente, incorrectamente separadas por el plano formado por los coeficientes *W* y el término independiente *G,* en tanto que *P* representa el peso que se da al término "clasificación" y al término "selección de atributos"; *ALFA* controla la forma de la función exponencial que selecciona atributos (mientras más grande, más se asemeja a una función escalón). Finalmente *F* es el valor de la función objetivo que se pretende minimizar. Esta implementación en GAMS, denominada ahora FSV-GAMS, la validamos con pocos datos controlados, donde de antemano conocíamos la respuesta correcta. Una vez que la implementación funcionó para todos los casos controlados, procedimos a experimentar con los datos procedentes del SICOM, para clasificar instancias de "ilícitos" y "no-ilícitos".

4.2 Detalles de experimentación

El objetivo de nuestra experimentación consistió en observar el comportamiento de FSV-GAMS en cuanto a calidad de solución y tiempo de respuesta en un problema del mundo

Tabla 1. Modelo FSV implementado en GAMS (FSV-GAMS)

```
SET

  J  / 1*24/
  I  / 1*2200/
  L  / 1*570/
;


TABLE
A(I,J)

          1   2  3          4           5     6          7     8     9      10     11
    1  1998   8  2     479640      271560     0     298753     1    71     836    836
    2  1998   9  9     475090   273349.68     0     300723     1    71     836    836
.....
 2200  2000   1  7          0      999.06     0       1099     9    71     690    690
;

TABLE
B(L,J)
          1    2     3        4          5     6        7    8     9      10      11
    1  1999    1     6    64350   25895.29     6    28528    2    71   657.3     395
    2  1999    1     6   102000   39982.36     6    44023    2    71   402.2     242
.....
  570  2000    1     6     8560     4875.6     0     5363    1    71     116     116
;

VARIABLES
W(J), G , Y(I), Z(L), V(J), F;

POSITIVE VARIABLES
Y, Z;

SCALAR  M /2200/, K /570/, P /0.01/, ALFA / 0.1 /;

EQUATIONS
OBJ, RA(I), RB(L), RV1(J), RV2(J);
OBJ..  F =E=  (1-P) * (( SUM(I, Y(I))/ M ) + ( SUM(L, Z(L)) / K ))
           +  P * ( SUM(J, (1.0 - EXP (-1.0 * ALFA * V(J)) )));
RA(I)..    SUM(J, -1.0* A(I,J) * W(J)) + G + 1.0  =L= Y(I);
RB(L)..    SUM(J, B(L,J) * W(J)) - G + 1.0  =L= Z(L);
RV1(J)..    -1.0* V(J)   =L=  W(J) ;
RV2(J)..          V(J)   =G=  W(J) ;

MODEL RECTA /ALL/;
SOLVE RECTA USING NLP MINIMIZING F;
```

real (antes descrito en la sección 3) y desde un punto de vista imparcial. Para ello diseñamos 25 experimentos en los que variamos los valores de *P* y *ALFA:* aunque Bradley obtuvo buenos resultados con valores de 0.05 y 5 respectivamente, él mismo advierte que pueden variar con el dominio.

Adicionalmente, nosotros agregamos tres variables *random* para observar si FSV-GAMS era capaz de eliminar tales atributos: esta idea se inspira en lo propuesto en [12] que emplea una variable *random* como umbral de decisión en selección de atributos bajo el enfoque *ranking* y que separa a los atributos relevantes de los irrelevantes; nosotros empleamos tres, en vez de una sola variable *random*, para evitar algún sesgo en la generación de pseudo-*randoms*. También realizamos experimentos variando el número de instancias del problema, para observar el tiempo de respuesta de FSV-GAMS. Todos los experimentos se hicieron en una computadora personal con procesador Pentium 4. En la siguiente sub sección se muestran los resultados obtenidos.

4.3 Resultados de la experimentación

Los resultados obtenidos al variar los parámetros *ALFA* y *P* se muestran en la Tabla 2, donde se reporta el número de atributos seleccionados (#Ats.); el valor de F, es decir, el valor de la función objetivo a minimizar (Val.Fun.Obj.); el tiempo de procesamiento en segundos (Tiempo(seg)); y el número de iteraciones que se requirieron para llegar a la solución final (Iters.). En la Tabla 2 se observa que:

- El número de atributos seleccionados varía desde 1 hasta 16, lo que indica que *ALFA* y *P* afectan notablemente a FSV-GAMS en este sentido.
- El valor de la función objetivo no se vio seriamente afectada por *ALFA* y *P,* así como el número de iteraciones y en consecuencia el tiempo de procesamiento, los cuales se mantuvieron en niveles aceptables, la mayoría de las veces.
- El número de iteraciones obtenidas contrasta con lo reportado por Bradley, que menciona que requiró de 5 a 7 iteraciones, aunque hay que considerar que las bases de datos que usó Bradley fueron, en el mejor de los casos, tres y media veces más pequeñas.
- Sólo para *ALFA* =1 el número de atributos seleccionados se redujo conforme se aumentó el valor de *P* según lo esperado*;* para los demás valores de *ALFA* se obtuvo un comportamiento más bien irregular conforme se aumentó *P* (ver sección 5).

En cuanto al detalle de los atributos seleccionados, esto se muestra en la Tabla 3, donde se observa que, en el 60% de los experimentos, FSV-GAMS eliminó exitosamente a las tres variables *random* (variables núms. 22 a 24).

A continuación se procedió a usar los atributos seleccionados por FSV-GAMS para alimentar al algoritmo de inducción de árboles de decisión *J4.8* implementado en la herramienta *Weka* [16] y que es la última versión de *C4.5,* el cual a su vez es uno de los algoritmos de inducción más conocidos y utilizados en la minería de datos [7].

Los resultados obtenidos se muestran en la Tabla 4, donde el tamaño del árbol generado se reporta como el número de nodos del árbol (TmArb); el porcentaje de instancias correctamente clasificadas, usando 10-fold cross validation (AccTest); el tiempo requerido para generar el árbol (Tiempo(seg)); el costo-beneficio obtenido (C/Ben)

asumiendo un beneficio de +97.5 cuando se predice correctamente un "ilícito" y un costo o pérdida de igual magnitud en caso contrario, y un beneficio de 2.5 cuando se predice correctamente un "no-ilícito" (por lo que hay un ahorro al no tener que efectuar la inspección correspondiente) y un costo o pérdida de igual magnitud en caso contrario. La última columna (C/Ben 1000) es el costo-beneficio obtenido normalizado con respecto al costo-beneficio que se obtiene cuando se alimentan los 21 atributos a *J4.8,* es decir, cuando no se hace ninguna selección de atributos .

Tabla 2. Resultados para diversos valores de *P* y *ALFA*

ALFA	P	# Ats.	Val.Fun.Obj.	Tiempo (seg)	Iters.
0.1	0.01	15	0.6868	274.26	4521
0.1	0.05	16	0.6628	265.29	5039
0.1	0.20	11	0.5620	273.19	5205
0.1	0.50	12	0.3763	216.58	4268
0.1	0.90	8	0.1005	192.59	4209
1	0.01	15	0.6876	462.95	8354
1	0.05	11	0.6729	269.52	5057
1	0.20	8	0.6100	238.81	4182
1	0.50	7	0.4983	225.13	4015
1	0.90	5	0.1733	269.84	4821
5	0.01	12	0.6965	288.51	5561
5	0.05	3	0.6975	186.31	2560
5	0.20	7	0.9156	228.47	4363
5	0.50	1	0.7513	191.49	2556
5	0.90	8	0.1997	252.35	4483
20	0.01	12	0.7768	160.29	3554
20	0.05	8	0.7110	193.26	3013
20	0.20	1	0.7530	175.63	2626
20	0.50	5	1.3454	187.59	3671
20	0.90	6	1.8683	242.78	4348
100	0.01	9	0.7647	275.78	10000
100	0.05	4	0.7503	172.17	2559
100	0.20	8	2.1533	186.90	3683
100	0.50	5	0.8467	224.59	5046
100	0.90	8	7.2600	164.61	2935
Promedios		8.20	1.0208	232.75	4425

En la Tabla 4 se observa que:

- El tamaño de árbol más grande es de 55 nodos, en tanto que el más pequeño sólo tiene 1 nodo; sin embargo ninguno de estos dos obtiene la mejor calidad de solución.

- El porcentaje de instancias correctamente clasificadas varió de 80 a 97%, es decir, en todos los experimentos se logró un aceptable porcentaje de predicción.
- El tiempo de procesamiento no superó los 3 segundos.
- Cuatro experimentos lograron un costo-beneficio muy similar al que se obtiene usando todos los atributos.

Tabla 3. Detalle de los atributos seleccionados por FSV-GAMS

ALFA	P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TOT
0.1	0.01	1	1	1	1	1	1		1		1	1		1			1	1					1	1	1	15
0.1	0.05	1	1	1	1	1	1		1					1		1	1		1	1	1	1		1	1	16
0.1	0.20	1			1	1	1							1		1	1		1	1				1	1	11
0.1	0.50	1	1	1	1	1	1						1			1	1			1				1	1	12
0.1	0.90				1		1	1								1	1		1	1					1	8
1	0.01	1	1	1	1		1		1			1	1	1		1	1		1			1		1	1	15
1	0.05	1	1		1	1	1							1		1	1			1				1	1	11
1	0.20		1		1	1	1						1			1	1			1						8
1	0.50				1		1	1								1	1		1	1						7
1	0.90				1	1					1				1				1							5
5	0.01	1	1		1	1	1							1		1	1		1	1				1	1	12
5	0.05		1				1										1									3
5	0.20				1		1				1		1			1	1		1							7
5	0.50						1																			1
5	0.90				1	1					1	1				1	1		1	1						8
20	0.01		1	1	1	1	1				1		1				1	1	1					1	1	12
20	0.05				1		1	1								1	1		1	1					1	8
20	0.20						1																			1
20	0.50				1		1											1	1	1						5
20	0.90				1	1	1									1	1		1							6
100	0.01				1	1	1				1	1					1	1	1		1					9
100	0.05					1	1				1										1					4
100	0.20				1	1	1				1						1	1	1		1					8
100	0.50				1		1									1	1		1							5
100	0.90				1	1	1				1						1	1	1		1					8
	Total	7	9	5	21	15	23	3	3	0	9	4	5	6	1	14	20	6	18	10	5	2	1	8	10	205
	No.Atributo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	

Donde el número de atributo corresponde a la siguiente descripción nmemotécnica (para respetar confidencialidad): 1.anio 2.mes 3.digito 4.kwh 5.enrgia 6.factra 7.total 8.status 9.tarifa 10.cgInst 11.cgCont 12.clMcC 13.kwEen 14.toMcl 15.kwMen 16.toMen 17.toMkw 18.kwMcl 19.toMcC 20.cCEto 21.clEen 22.Random1 23.Random2 24.Random3

Tabla 4. Resultados de *J4.8* al usar los atributos seleccionados por FSV-GAMS

ALFA	P	TmArb	AccTest	Tiempo (seg)	C/Ben	C/Ben1000
0.1	0.01	39	97.36	2.34	54630	996.8
0.1	0.05	39	97.36	2.52	54820	1000.3
0.1	0.2	15	95.37	1.66	52645	960.6
0.1	0.5	25	95.7	1.78	47560	867.8
0.1	0.9	3	90.07	0.88	27780	506.9
1	0.01	39	97.47	2.27	54835	1000.5
1	0.05	21	95.52	1.87	48105	877.7
1	0.2	9	90.28	1.09	28570	521.3
1	0.5	3	90.14	0.83	27790	507.1
1	0.9	1	79.38	0.76	-50080	-913.8
5	0.01	21	95.52	1.9	48105	877.7
5	0.05	3	90.54	0.27	28035	511.5
5	0.2	19	90.68	0.79	25585	466.8
5	0.5	3	90.18	0.04	27985	510.6
5	0.9	7	79.35	1.07	-49895	-910.4
20	0.01	55	90.18	1.33	24565	448.2
20	0.05	3	90.07	0.64	27780	506.9
20	0.2	3	90.18	0.04	27985	510.6
20	0.5	3	90.25	0.28	27995	510.8
20	0.9	3	90.14	0.45	27790	507.1
100	0.01	15	90.46	1.3	23655	431.6
100	0.05	3	89.96	0.75	24345	444.2
100	0.2	19	90.28	1.15	23440	427.7
100	0.5	3	90.14	0.53	27790	507.1
100	0.9	19	90.28	0.97	23440	427.7
	Promedio	14.92	91.07	1.10	27410.2	500.1

Finalmente se ejecutaron diversos experimentos variando el número de instancias de la base de datos (usando *ALFA=1* y *P=0.01)*, para observar cómo se comportaba el tiempo de procesamiento de FSV-GAMS. La Figura 1 muestra los resultados obtenidos, donde se aprecia que existe un crecimiento exponencial del tiempo con relación al número de instancias a considerar. Este resultado es muy interesante pues implica que FSV-GAMS puede resultar muy costoso computacionalmente cuando el número de instancias, de una base de datos dada, sea muy grande.

4.4 Comparación contra métodos *Wrapper* y *Ranking*

Para tener una idea de la bondad de FSV-GAMS, en esta sección compararemos los resultados mostrados en la sub sección 4.3 contra lo que arrojan los métodos tradicionales, más conocidos y usados, *wrapper* y *ranking*.

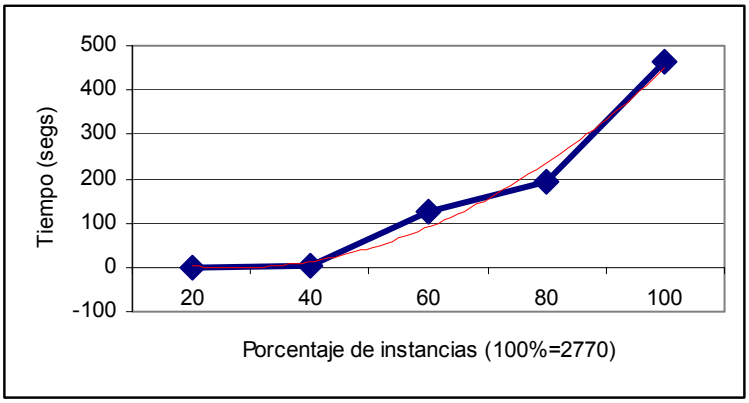

Figura 1. Tiempo de procesamiento de FSV-GAMS vs núm. de instancias

En primera instancia consideraremos el uso de un *wrapper exhaustivo*. En este caso podemos calcular que, si el tiempo promedio requerido para obtener un árbol con *J4.8* es de 1.1 segundos (ver Tabla 4) y si multiplicamos esto por el número de combinaciones posibles de atributos, es decir 2^{24} , entonces obtendremos que se requerirían 7.12 meses para concluir tal proceso, lo cual resulta prohibitivo. Aunque existen variantes menos costosas de *wrapper,* éstas no garantizan obtener el óptimo subconjunto de atributos, y aunque son más rápidas, típicamente requieren desde algunas horas hasta varios días para satisfacer su criterio de paro.

Por otro lado, la aplicación de métodos filtro-*ranking* en la selección de atributos de grandes bases de datos es adecuada debido a su bajo costo computacional. En los experimentos realizados usando las herramientas *Weka* [16] y *Elvira* [17], el tiempo de procesamiento resulta despreciable (menos de un segundo). Nosotros aplicamos diferentes métodos *ranking* a la base de datos de facturación eléctrica; los resultados se muestran en la Tabla 5. Una explicación detallada de estos métodos se puede encontrar en [6].

En la Tabla 5 se observa que, aunque algunos métodos *ranking* coinciden en posicionar algunos atributos arriba o debajo de la tabla, en general cada método arroja distintos ordenamientos de los atributos, inclusive de las tres variables *random*.

A continuación se aplicaron los atributos seleccionados por cada método *ranking* al clasificador *J4.8*. En todos los casos se emplearon los parámetros de *default* de *Weka* y siempre se seleccionaron los atributos del primer subconjunto que determina la aparición de la primera variable *random*. Los factores medidos son iguales a los de la Tabla 4. En la Tabla 6 se observa que la mayoría de los métodos logran una reducción del número de atributos superior a 0.50, reduciendo el tiempo de procesamiento en un orden de magnitud (excepto Relief). En cuanto al tamaño del conocimiento descubierto se observa que casi todos los métodos producen árboles más pequeños que el caso completo. Por otro lado, aunque aparentemente todos los métodos logran buenos porcentajes de clasificación

correcta, la columna de costo-beneficio destaca a aquellos métodos que mejor predicen el patrón de un uso ilícito de energía.

Tabla 5. Posición de atributos previa aplicación de diversas medidas *ranking*

Mutual Information	Euclidean distance	Matusita distance	Kullback-Leibler 1	Kullback-Leibler 2	Shannon entropy	Bhatta-charyya	Relief	OneR	Chi-Square
fctura	fctura	fctura	fctura	fctura	kwh	kwEen	anio	factra	factra
status	mes	kwEen	status	mes	enrgia	fctura	mes	status	status
kwEen	clMcC	kwMen	kwEen	status	total	kwMen	factra	anio	mes
cCEto	anio	**RAND3**	cCEto	cgInst	tarifa	**RAND3**	digito	tarifa	kwEen
RAND3	**RAND3**	status	**RAND3**	cgCont	cgInst	toMkw	**RAND3**	digito	kwMcl
kwMen	tarifa	cCEto	kwMen	cCEto	cgCont	toMcl	**RAND2**	mes	kwh
toMkw	digito	toMkw	toMkw	clMcC	kwEen	enrgia	**RAND1**	clMcC	toMcl
enrgia	status	enrgia	enrgia	anio	toMcl	cCEto	status	cgCont	toMcC
kwMcl	**RAND2**	toMcl	kwMcl	kwEen	kwMen	total	cgInst	cgInst	total
toMcl	clEen	total	toMcl	**RAND2**	toMen	**RAND2**	tarifa	**RAND1**	toMen
RAND2	cgInst	kwMcl	**RAND2**	**RAND3**	toMkw	kwMcl	cgCont	toMkw	enrgia
kwh	cgCont	**RAND2**	kwh	toMkw	kwMcl	toMcC	cCEto	**RAND2**	kwMen
total	**RAND1**	kwh	total	kwh	toMcC	**RAND1**	clEen	cCEto	toMkw
mes	cCEto	toMcC	mes	kwMen	cCEto	kwh	clMcC	kwh	cCEto
toMcC	toMcC	**RAND1**	toMcC	clEen	clEen	toMen	kwEen	toMcl	clEen
RAND1	kwMcl	toMen	**RAND1**	enrgia	status	mes	toMen	**RAND3**	cgInst
clEen	toMkw	mes	clEen	total	clMcC	status	total	total	cgCont
toMen	toMen	clEen	toMen	kwMcl	anio	clEen	toMcC	toMen	anio
cgInst	kwMen	cgInst	cgInst	**RAND1**	**RAND2**	cgInst	toMcl	kwEen	clMcC
cgCont	toMcl	cgCont	cgCont	tarifa	**RAND1**	cgCont	kwh	enrgia	tarifa
clMcC	kwEen	clMcC	clMcC	digito	**RAND3**	clMcC	kwMcl	kwMen	**RAND2**
anio	total	anio	anio	toMcC	digito	anio	kwMen	clEen	**RAND3**
tarifa	enrgia	tarifa	tarifa	toMcl	mes	tarifa	toMkw	kwMcl	**RAND1**
digito	kwh	digito	digito	toMen	fctura	digito	enrgia	toMcC	digito

Tabla 6. Resultados de métodos *ranking* induciendo árboles con *J4.8*

Método	#Ats.	TmArb	AccTest	Tiempo(seg)	C/Ben 1000
Caso completo	21	41	97.25	3.31	1000
Mutual Information	4	9	90.10	0.40	444
Euclidean distance	4	5	93.89	0.39	520
Matusita distance	3	3	90.21	0.26	507
Kullback-Leibler 1	4	9	90.10	0.37	444
Kullback-Leibler 2	9	33	97.50	0.46	1001
Shannon entropy	18	45	93.71	3.03	876
Bhattacharyya	3	3	90.21	0.30	507
Relief	4	5	93.89	0.38	520
OneR	9	23	95.95	0.49	892
Chi-Square	20	33	97.43	3.21	1004

5 Conclusiones y Trabajo Futuro

En base a la experimentación realizada se observa que:

- FSV se puede implementar de manera sencilla usando alguna herramienta de optimización; nosotros empleamos GAMS, pero en principio su implementación puede resultar poco complicada usando cualquier otra herramienta de optimización.
- FSV sólo maneja atributos continuos, sólo es capaz de clasificar dos clases y no es capaz de generar conocimiento en forma explícita (p.e. árboles de decisión o reglas de clasificación).
- FSV-GAMS es capaz de seleccionar atributos relevantes (sin afectar la calidad de predicción) aunque el número de éstos varía con los parámetros *ALFA* y *P,* por lo que es necesario realizar varios experimentos (nosotros hicimos 25) para hallar la correcta combinación de estos parámetros.
- El tamaño del conocimiento obtenido, atendiendo los atributos seleccionados por FSV-GAMS, es comparable a lo que se obtiene aplicando métodos *ranking.*
- En cuanto al costo-beneficio alcanzado con FSV-GAMS, se observa que logra resultados similares o ligeramente mejores que si se usaran todos los atributos para realizar la inducción, aunque esto no es de manera consistente, sino que varía con *ALFA* y *P.*
- Para nuestro caso de estudio (detección de ilícitos de energía eléctrica), el tiempo de procesamiento de FSV-GAMS fue razonable (no más de 5 minutos) comparado contra el esquema *wrapper* que típicamente requiere horas o días de ejecución; sin embargo los resultados sugieren que para bases de datos con mayor número de instancias, el tiempo de ejecución de FSV-GAMS crecerá exponencialmente.
- Comparando los resultados de FSV-GAMS contra diversos métodos *ranking*, se observa que en algunos casos la calidad de la solución es similar, aunque los métodos *ranking* tienen la ventaja de poseer complejidad lineal con respecto al número de atributos; sin embargo, FSV-GAMS posee la ventaja de hallar subconjuntos de atributos en los que queda inmerso la posible inter-dependencia entre variables, que es algo con lo que no cuentan los métodos *ranking.*

Dado lo anterior, surgen varias líneas de investigación que se podrían trabajar en el futuro con relación a FSV-GAMS, con el fin de mejorarlo. Por ejemplo:

- Diseñar una formulación tal que incluya el encontrar los valores óptimos para los parámetros *ALFA* y *P,* evitando así tener que realizar múltiples experimentos para hallar los mejores valores de tales parámetros.
- En vez de encontrar un plano de separación de las clases en forma rígida (crisp), re-formular FSV en términos de una clasificación difusa (fuzzy) tal que el grado de penalización de las instancias mal clasificadas sea gradual con respecto a su lejanía-cercanía con respecto al plano difuso de separación de clases.
- Como se comentó en la sección 4.3, el número de atributos seleccionados no respondió al parámetro *P* en la forma esperada. Esto podría deberse a la aparición en la solución del hiperplano de coeficientes muy pequeños (p.e. del orden de E-20) los

cuales no son descartados por FSV-GAMS. Se podría idear algún esquema de "poda" tal que elimine estos coeficientes pequeños y observar cómo afecta a la solución global.

- También sería importante re-formular FSV-GAMS de tal forma que no requiera realizar miles de iteraciones para llegar a la solución. Esto se podría lograr aplicando métodos de punto interior o linearizando la formulación (sustituir la expresión exponencial de la formulación por un equivalente lineal).
- FSV en su formulación actual no considera el aspecto de costo-beneficio en la clasificación de las clases, el cual es un aspecto importantísimo en aplicaciones del mundo real; entonces se podría trabajar en una formulación de FSV que tome en cuenta este aspecto.
- En la formulación de FSV de Bradley se incluyen dos planos paralelos y distantes en una unidad de magnitud al plano solución. En este sentido se podrían realizar diversos experimentos variando la distancia de estos dos planos para ver el efecto en la solución global.
- Finalmente sería importante re-formular FSV para que admita clasificar más de dos clases, manejar atributos mixtos (continuos y discretos) y generar conocimiento en forma explícita que sea "entendible" por el experto humano del dominio, lo cual convertiría a FSV en un algoritmo más cercano a lo requerido por la minería de datos.

6 Referencias

1. Bradley, P., Mangasarian, O., Feature selection via concave minimization and support vector machines. In J.Shavlik, editor, Machine learning ICML, Sn Fco., California, 1998, pp. 82-90.
2. Pyle, D., Data preparation for data mining, Morgan Kaufmann, Sn Fco, California,1999.
3. Guyon, I., Elisseeff, A., An introduction to variable and feature selection, Journal of machine learning research, 3, 2003, pp. 1157-1182.
4. Kohavi, R., John, G., Wrappers for feature subset selection, Artificial Intelligence Journal, Special issue on relevance, 1997, pp. 273-324.
5. Leite, R., Brazdil, P., Decisión tree-based attribute selection via subsampling, Workshop de minería de datos y aprendizaje, VIII Iberamia, Sevilla, Spain, Nov, 2002, pp. 77-83.
6. Piramuthu, S., Evaluating feature selection methods for learning in data mining applications, Proc. 31st annual Hawaii Int. conf. on system sciences, 1998, pp. 294-301.
7. Quinlan, J., Unknown attribute values in ID3, Int. conf. Machine learning, 1989, pp. 164-168.
8. Almuallim, H., Dietterich, T., Learning with many irrelevant features, Ninth nat. conf. on AI, MIT Press, 1991, pp. 547-552.
9. Kira, K., Rendell, L., The feature selection problem: traditional methods and a new algorithm, Tenth nat. conf. on AI, MIT Press, 1992, pp. 129-134.
10. Koller, D., Sahami, M., Toward optimal feature selection, Int. conf. on machine learning, 1996, pp. 284-292.
11. Molina, L., Belanche, L., Nebot, A., Feature selection algorithms, a survey and experimental evaluation, IEEE Int. conf. on data mining, Maebashi City Japan, 2002, pp. 306-313.
12. Stoppiglia, H., Dreyfus, G., et.al., Ranking a random feature for variable and feature selection, Journal of machine learning research, 3, 2003, pp. 1399-1414.

13. Bradley, P., Fayyad, U., Mangasarian, O., Mathematical programming for data mining: formulations and challenges, Informs Journal on Computing 11, 1999, pp. 217-238.
14. Mangasarian, O., Arbitrary-norm separating plane, Technical report 97-07, Computer science dept., Univ. Wisconsin Madison, 1997.
15. General Algebraic Modeling System, GAMS IDE, Rev. 135, 2003.
16. www. cs.waikato.ac.nz/ml/weka, 2004.
17. www. ia.uned.es/~elvira/, 2004.