Obtención de reglas de clasificación usando SOM+PSO

Augusto Villa Monte[1], Franco Ronchetti[2], Laura Lanzarini [3]
III-LIDI (Instituto de Investigación en Informática LIDI)
Facultad de Informática. Universidad Nacional de La Plata
La Plata, Buenos Aires, Argentina
{avillamonte, fronchetti, laural}@lidi.info.unlp.edu.ar

Marcela Jeréz[4],
Facultad de Ingeniería. Universidad Nacional de Tierra del Fuego.
Ushuaia, Tierra del Fuego, Argentina
mjerez@untdf.edu.ar

Abstract. Los mapas auto-organizativos (SOM – Self organizing maps) son un tipo de red neuronal ampliamente conocido por su capacidad para preservar la topología de los datos de entrada permitiendo mapear un espacio n-dimensional en otro de dos dimensiones. Luego de entrenar la red, las neuronas competitivas que la forman agrupan la información disponible facilitando de esta forma la identificación de similitudes; esto ha motivado su estudio como base para la obtención de reglas de asociación y clasificación. Este artículo presenta un nuevo método capaz de obtener reglas de clasificación que operan tanto sobre atributos numéricos como nominales, denominado *SOM+PSO*, que combina una red SOM con una metaheurística basada en cúmulo de partículas (variante de PSO - Particle swarm optimization). El método propuesto ha sido comparado con PART y medido sobre 19 bases del repositorio UCI con resultados satisfactorios.

Keywords: Reglas de clasificación, mapas auto-organizativos, inteligencia de cúmulo, minería de datos

1. Introducción

La Minería de Datos engloba un conjunto de técnicas capaces de modelizar la información disponible. Es una de las etapas más importantes del proceso de extracción de conocimiento y se caracteriza por obtener información útil y novedosa sin disponer de una hipótesis previa. Entre las técnicas preferidas por quienes tienen que tomar decisiones se encuentran las reglas de asociación.
Una regla de asociación es una expresión de la forma *SI condic1 ENTONCES condic2* donde ambas condiciones son conjunciones de proposiciones de la forma

[1] Becario CIN. Auxiliar docente. Fac.de Informática. UNLP
[2] Becario de postgrado UNLP Tipo A. Auxiliar docente. Fac.de Informática. UNLP
[3] Prof.Titular. Fac. De Informática. UNLP
[4] Docente UNTDF - Ushuaia

(atributo=valor) y cuya única restricción es que los atributos que intervienen en el antecedente de la regla no formen parte del consecuente. Cuando el conjunto de reglas de asociación presenta en el consecuente el mismo atributo se dice que se trata de un conjunto de reglas de clasificación [1] [2].
Este artículo presenta un nuevo método de obtención de reglas de clasificación que combina una red neuronal con una técnica de optimización. El énfasis está puesto en alcanzar una buena cobertura utilizando un número reducido de reglas.
La sección 2 describe brevemente algunos trabajos relacionados. Las secciones 3 y 4 describen la red neuronal y la metaheurística utilizadas, respectivamente. La sección 5 detalla el método propuesto. La sección 6 expone los resultados obtenidos y la sección 7 resume las conclusiones y describe algunas líneas de trabajo futuras.

2. Trabajos relacionados

Existen diversos métodos de construcción de reglas. Cuando se trata de obtener reglas de asociación puede utilizarse el método a priori [3] o algunas de sus variantes. En este caso se trata de identificar los conjuntos de atributos más frecuentes para luego combinarlos para obtener las reglas. Existen variantes del método a priori generalmente orientadas a reducir el tiempo de cálculo.
Si se trabaja con reglas de clasificación, la literatura contiene distintos métodos de construcción basados en árboles como el C4.5 [4] o en árboles recortados como el método PART [5]. En cualquiera de los casos, lo fundamental es obtener un conjunto de reglas que cubra los ejemplos cumpliendo con una cota de error preestablecida. Los métodos de construcción de reglas a partir de árboles son partitivos y se basan en distintas métricas de los atributos a fin de estimar su capacidad de cobertura.
Este artículo presenta un enfoque diferente basado en cúmulos de partículas (PSO - Particle Swarm Optimization) para determinar las reglas. Si bien existen métodos de obtención de reglas utilizando PSO [6], cuando se opera sobre atributos nominales es preciso contar con suficientes ejemplos como para cubrir todas las zonas del espacio de búsqueda y esto no siempre es factible. El resultado es una pobre inicialización de la población lo que da lugar a la convergencia prematura. Como forma de resolver este problema y a la vez reducir el tiempo de obtención, el estado inicial se obtiene de una red neuronal competitiva SOM (Self Organizing Map). Existen en la literatura métodos que utilizan PSO como forma de determinar la cantidad óptima de neuronas competitivas a utilizar en la red, como por ejemplo [7]. Esta no es la propuesta de este trabajo ya que la red SOM utilizada, si bien es dimensionada a priori, podría ser reemplazada por una red competitiva dinámica como la definida en [8].

3. SOM (Self-Organizing Maps)

La red neuronal SOM (Self Organizing Maps) fue definida por Kohonen en 1982 [9]. Su aplicación principal es el agrupamiento de la información disponible y se caracteriza por su capacidad para preservar la topología de los datos de entrada.
Puede ser representada como una estructura de dos capas: la capa de entrada cuya función es sólo permitir el ingreso de la información a la red y la capa competitiva que es la encargada de realizar el agrupamiento. Las neuronas que forman esta

segunda capa se encuentran conectadas y poseen la capacidad de identificar la cantidad de "saltos" o conexiones que la separan de cada una de las restantes dentro de este nivel. Cada neurona competitiva lleva asociado un vector de pesos o centroide representado por los valores de los arcos que llegan a ella desde la capa de entrada.
De esta forma, la red SOM maneja dos estructuras de información: una referida a los centroides asociados a las neuronas competitivas y otra encargada de determinar la proximidad entre neuronas. Esto, a diferencia de un método del estilo "winner-take-all" como el método k-medias, brinda información adicional con respecto a los agrupamientos ya que neuronas cercanas dentro de la arquitectura representarán agrupamientos similares en el espacio de los datos de entrada.
Inicialmente los pesos de la red, almacenados en una matriz que se denominará W, son aleatorios y se adaptan con las sucesivas presentaciones de los datos de entrada. Por tratarse de una estructura competitiva, cada vector de entrada se considera representado por (o asociado con) la neurona competitiva que posea el vector de pesos más parecido según una medida de similitud dada. El valor final de W se obtiene mediante un proceso iterativo que se repite hasta que los vectores de pesos no presenten modificaciones significativas o lo que es lo mismo, hasta que cada vector de entrada sea representado por la misma neurona competitiva que en la iteración anterior. En cada iteración, para cada vector de entrada se determina la neurona que lo representa. A esta neurona se le llama neurona ganadora ya que es la que gana la competencia por la representación del vector (es la más parecida hasta el momento). Luego se actualiza el vector de peso de dicha neurona y de su vecindad según la ecuación (1)

$$w_{ij} = w_{ij} + \alpha\,(x_i - w_{ij}) \qquad i = 1..n \tag{1}$$

siendo *n* la dimensión del espacio de entrada, *j* la neurona competitiva cuyo vector se desea actualizar y α un valor entre 0 y 1 que representa la velocidad de aprendizaje.
La ecuación (1) tiene variantes que pueden consultarse en [10].
El concepto de vecindad es utilizado para permitir que la red se adapte adecuadamente. Esto implica que neuronas competitivas vecinas representen patrones de entrada similares. Por tal motivo, durante el proceso de entrenamiento (obtención de los valores de W) se comienza con una vecindad amplia para luego ir reduciéndola a lo largo de las iteraciones.

4. PSO aplicado a la obtención de reglas

La optimización mediante cúmulo de partículas o PSO (Particle Swarm Optimization) es una metaheurística poblacional propuesta por Kennedy y Eberhart [11] donde cada individuo de la población, denominado partícula, representa una posible solución del problema y se adapta siguiendo tres factores: su conocimiento sobre el entorno (su valor de aptitud), su conocimiento histórico o experiencias anteriores (su memoria) y el conocimiento histórico o experiencias anteriores de los individuos situados en su vecindario (su conocimiento social).
PSO fue definido originalmente para trabajar sobre espacios continuos pero si se trata de operar sobre un espacio discreto es preciso tener en cuenta algunas consideraciones. Por tal motivo, Kennedy y Eberhart definieron en [12] una versión binaria del método PSO. Uno de los problemas centrales que presenta este último

método es su dificultad para cambiar de 0 a 1 y de 1 a 0 una vez que se ha estabilizado. Esto ha dado lugar a distintas versiones de PSO binario que buscan mejorar su capacidad exploratoria. En particular, en este trabajo se utiliza la variante definida por Lanzarini et al. en [13].

La obtención de reglas de clasificación utilizando PSO, capaces de operar sobre atributos nominales y numéricos, requiere de una combinación de los métodos citados anteriormente ya que es preciso decir cuáles serán los atributos que formarán parte del antecedente (discreto) y cuál es el valor o rango de valores que podrán tomar (continuo). En este artículo se ha decidido adoptar el enfoque Michigan [14] es decir que cada partícula de la población representa una única regla. Por simplicidad, el consecuente de la regla no será codificado en el individuo y por lo tanto, todas las partículas buscarán obtener reglas de una misma clase decidida a priori. A fin de no perder diversidad, se aplicará el método propuesto de manera iterativa hasta lograr la cobertura total buscada obteniendo una única regla en cada iteración: el mejor individuo de la población.

Para desplazarse en un espacio n-dimensional, cada partícula p_i está compuesta por

- $indivBinario_i = (indBin_{i1}, indBin_{i2}, \ldots, indBin_{in})$ almacena la posición actual de la partícula.
- $veloc1_i = (v1_{i1}, v1_{i2}, \ldots, v1_{in})$ y $veloc2_i = (v2_{i1}, v2_{i2}, \ldots, v2_{in})$ se combinan para determinar la dirección en la cual se moverá la partícula.
- $pBestBinario_i = (pBin_{i1}, pBin_{i2}, \ldots, pBin_{in})$ almacena la mejor solución encontrada por la partícula hasta el momento.
- $fitness_i$ es el valor de aptitud del individuo.
- $fitness\_pBest_i$ es el valor de aptitud de la mejor solución local encontrada (vector $pBestBinario_i$)
- $indivReal_i = (indR_{i1}, indR_{i2}, \ldots, indR_{in})$ sólo se utiliza para los atributos numéricos y contiene los límites actuales de los intervalos.
- $velocReal_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ indica la dirección de cambio de $indivReal_i$.
- $pBestReal_i = (pReal_{i1}, pReal_{i2}, \ldots, pReal_{in})$ almacena la mejor solución encontrada por la partícula para los límites de los intervalos.

Cada vez que la i-ésima partícula se mueve se modifica su posición actual y los intervalos correspondientes a los atributos numéricos siguiendo los correspondientes vectores de velocidad, de la siguiente forma:

Parte binaria

$$v1_{ij}(t+1)=w_{bin}.v1_{ij}(t)+\varphi_1.rand_1.(2*pBin_{ij}-1)+\varphi_2.rand_2.(2*localBest_{ij}-1) \quad (2)$$

donde, w_{bin} representa el factor de inercia, $rand_1$ y $rand_2$ son valores aleatorios con distribución uniforme en [0,1] y φ_1 y φ_2 son valores constantes de indican la importancia que se desea darle a las respectivas soluciones halladas previamente. Los valores $pBin_{ij}$ y $localBest_{ij}$ corresponden al j-ésimo dígito de los vectores binarios $pBestBinario_i$ y $localBest_i$ respectivamente. En el método propuesto, cada partícula tendrá en cuenta la posición de su vecino más cercano con un valor de aptitud superior al suyo; por lo tanto el valor de $localBest_i$ corresponde al vector de

indivBinario de la partícula más cercana a p_i con un valor de *fitness* superior a $fitness_i$ usando distancia euclídea.
Nótese que, a diferencia del PSO Binario de [12], el desplazamiento del vector $veloc1_i$ en las direcciones correspondientes a la mejor solución encontrada por la partícula y al mejor local no dependen de la posición actual de dicha partícula, como se indica en [13]. Luego, cada elemento del vector velocidad $veloc1_i$ es controlado según (3)

$$v1_{ij}(t) = \begin{cases} delta1_j & si\ v1_{ij}(t) > delta1_j \\ -delta1_j & si\ v1_{ij}(t) \geq -delta1_j \\ v1_{ij}(t) & en\ caso\ contrario \end{cases} \quad (3)$$

donde $$delta1_j = \frac{\lim ite1_{\sup erior_j} - \lim ite1_{\inf erior_j}}{2} \quad (4)$$

Es decir que, el vector velocidad $veloc1_i$ se calcula según (2) y se controla según (3). Su valor se utiliza para actualizar el valor del vector velocidad $veloc2_i$, como se indica en (5).

$$v2_{ij}(t+1) = v2_{ij}(t) + v1_{ij}(t+1) \quad (5)$$

El vector $veloc2_i$ también se controla de manera similar al vector $veloc1_i$ cambiando $limite1_{superior\ j}$ y $limite1_{inferior\ j}$ por $limite2_{superior\ j}$ y $limite2_{inferior\ j}$ respectivamente. Esto dará lugar a $delta2_j$ que será utilizado como en (3) para acotar los valores de $veloc2_i$. Luego se le aplica la función sigmoide (6) y se calcula la nueva posición de la partícula según (7).

$$sig(x) = 1/1 + e^{-x} \quad (6)$$

$$indivBinario_{ij}(t+1) = \begin{cases} 1 & si\ rand_{ij} < sig(v2_{ij}(t+1)) \\ 0 & si\ no \end{cases} \quad (7)$$

donde $rand_{ij}$ es un número aleatorio con distribución uniforme en [0,1].

Parte continua

$$indR_{ij}(t+1) = indR_{ij}(t) + v_{ij}(t+1) \quad (8)$$

$$v_{ij}(t+1) = w_{Real}.v_{ij}(t) + \varphi_3.rand_3.(pReal_{ij} - indR_{ij}(t)) + \varphi_4.rand_4.(lBestReal_{ij} - indR_{ij}(t)) \quad (9)$$

donde nuevamente, w_{Real} representa el factor de inercia, $rand_3$ y $rand_4$ son valores aleatorios con distribución uniforme en [0,1] y φ_3 y φ_4 son valores constantes de indican la importancia que se desea darle a las respectivas soluciones halladas previamente. En este caso, *lBestReal* corresponde al vector *indivReal* de la partícula más cercana a p_i con un valor de *fitness* superior a $fitness_i$ usando distancia euclídea. Esta es la misma partícula de la que se tomó el vector *indivBinario* para realizar el ajuste de $veloc1_i$ en (2).
Los valores asignados a w_{Real} [15], φ_1, φ_2, φ_3 y φ_4 son importantes para garantizar la convergencia del algoritmo. Puede encontrar información más detallada referida a la selección de estos valores en [11] y [16].

Además, es importante remarcar que la incorporación de la función sigmoide (6) cambia radicalmente la manera de utilizar el vector velocidad para actualizar la posición de la partícula. En PSO continuo, el vector velocidad toma valores mayores al inicio para facilitar la exploración del espacio de soluciones y al final se reduce para permitir que la partícula se estabilice. En PSO binario ocurre precisamente todo lo contrario. Los valores extremos, al ser mapeados por la función sigmoide, producen valores de probabilidad similares, cercanos a 0 o a 1, reduciendo la chance de cambio en los valores de la partícula. Por otro lado, valores del vector velocidad cercanos a cero incrementan la probabilidad de cambio. Es importante considerar que si la velocidad de una partícula es el vector nulo, cada uno de los dígitos binarios que determina su posición tiene probabilidad 0.5 de cambiar a 1. Es la situación más aleatoria que puede ocurrir.

En este trabajo, los valores de *limite1* y *limite2* son iguales para todas las dimensiones; estos son [0,1] y [0,6] respectivamente. Por lo tanto, los valores de los vectores velocidad *veloc1* y *veloc2* fueron limitados a los rangos [-0.5, 0.5] y [-3,3] respectivamente. Es decir que pueden obtenerse probabilidades en el intervalo [0.0474, 0.9526]. Los valores para φ_1, φ_2, φ_3 y φ_4 fueron establecidos en 0.25, 0.25, 0.5 y 0.25 respectivamente. Los valores de w_{bin} y w_{Real} fueron establecidos entre 1.25 y 0.25 de manera lineal y proporcional a la cantidad de iteraciones realizadas, en forma ascendente para w_{bin} y en forma descendente para w_{Real}.

El fitness de cada partícula se calcula en base al soporte, la confianza de la regla y la cantidad de atributos utilizados en el antecedente de la siguiente forma:

$$Fitness = penalizacion * (soporte/L) * confianza - factor * NumAtribs/MaxAtribs \quad (10)$$

donde *soporte* y *confianza* son las métricas correspondientes a la regla que representa la partícula, *L* es la cantidad de ejemplos a considerar, *factor* es una constante que cuantifica la importancia que se le da a la longitud del antecedente, *NumAtribs* es la cantidad de comparaciones que intervienen en el antecedente de la regla (una por atributo) y *MaxAtribs* es la máxima cantidad de comparaciones que podrían presentarse.

La *penalización* es un valor entre 0.1 y 1 que reduce el producto de soporte y confianza si es necesario. Se calcula de la siguiente forma:

```
Function Penalizar(VALOR, UMBRAL1, UMBRAL2, MIN,MAX)
  if (VALOR < UMBRAL1),
       rta = MIN;
  elseif (VALOR < UMBRAL2)
       rta = MIN+(MAX-MIN)*[(VALOR-UMBRAL1)/(UMBRAL2-UMBRAL1)+UMBRAL1];
  else rta = 1;
  end
  return(rta)
end


SOP_MIN1 = max(2, 5% de los ejemplos de la clase);
SOP_MIN2 = max(4, 7.5% de los ejemplos de la clase);
Pena_SOP = Penalizar(soporte, SOP_MIN1, SOP_MIN2,0.1,0.75);

CONF_MIN1 = límite de confianza inferior (ej: 0.4);
CONF_MIN2 = límite de confianza superior (ej: 0.7);
Pena_CONF = Penalizar(confianza, CONF_MIN1, CONF_MIN2,0.1,0.7);
```

Penalizacion = min(Pena_SOP, Pena_CONF)

5. SOM+PSO. Método de obtención de reglas propuesto

Las reglas se obtienen a través de un proceso iterativo que analiza los ejemplos no cubiertos de cada clase comenzando por las más numerosas. Cada vez que se obtiene una regla, los ejemplos cubiertos correctamente por dicha regla son retirados del conjunto de datos de entrada. El proceso continúa hasta lograr cubrir todos los ejemplos o hasta que la cantidad de ejemplos no cubiertos de cada clase se encuentre por debajo del respectivo soporte mínimo establecido o hasta que se hayan realizado la máxima cantidad de intentos por obtener una regla, lo que ocurra primero. Es importante tener en cuenta es que, dado que los ejemplos son retirados del conjunto de datos de entrada a medida que son cubiertos por las reglas, las mismas constituyen una lista de clasificación. Es decir que, para clasificar un ejemplo nuevo, las reglas deben ser aplicadas en el orden en que fueron obtenidas y el ejemplo será clasificado con la clase correspondiente al consecuente de la primera regla cuyo antecedente se verifique para el ejemplo en cuestión.

Antes de comenzar con el proceso iterativo de obtención de reglas, el método inicia con el entrenamiento no supervisado de una red SOM utilizando el conjunto completo de ejemplos que se espera cubrir. La red se entrena de manera no supervisada siguiendo el algoritmo descripto en la sección 2 y su función es identificar las zonas más prometedoras del espacio de búsqueda.

Dado que la red SOM sólo opera con datos numéricos, los atributos nominales son representados en forma binaria. Además, antes de iniciar el entrenamiento, cada dimensión correspondiente a un atributo numérico es escalada linealmente en [0,1]. La medida de similitud utilizada es distancia euclídea. Una vez finalizado el entrenamiento, cada centroide contendrá aproximadamente el promedio de los ejemplos que representa.

Para obtener cada una de las reglas se determina, en primer lugar, cual es la clase correspondiente al consecuente. Buscando obtener reglas con soporte alto, el método propuesto comenzará a analizar primero las clases que posean un mayor número de ejemplos no cubiertos. El soporte mínimo que debe cumplir una regla es proporcional a la cantidad de ejemplos no cubiertos de la clase al momento en que fue obtenida. Es decir, que el soporte mínimo requerido para cada clase disminuye a lo largo de las iteraciones, a medida que los ejemplos de la clase correspondiente se van cubriendo. De esta forma, es de esperar que las primeras reglas posean mayor soporte que las últimas.

Una vez seleccionada la clase, queda determinado el consecuente de la regla. Para obtener el antecedente se optimizará, utilizando el algoritmo descripto en la sección 3, una población de partículas inicializada con la información de todos los centroides capaces de representar un número mínimo de ejemplos de la clase seleccionada y sus vecinos inmediatos. La información del centroide se utiliza para determinar el vector *veloc2* descripto en la sección 3. Si se trata de un atributo nominal, la información del centroide se escala linealmente al intervalo [$limite2_{inferior\ j}$, $limite2_{superior\ j}$] pero si se trata de un atributo numérico el valor a escalar es $(1\text{-}1.5*desviacion_j)$ siendo $desviacion_j$ la j-ésima dimensión de la desviación de los ejemplos representados por el centroide. En ambos casos se pretende operar con un valor entre 0 y 1 que mida el

grado de participación del atributo (si es numérico) o del valor del atributo (si es nominal) en la construcción del antecedente de la regla. Cuando se trata de atributos nominales es claro que el promedio indica la proporción de elementos representados por el centroide que coinciden en el mismo valor pero cuando son numéricos, esta proporción no se encuentra presente en el centroide sino en la desviación de los ejemplos (siempre considerando una dimensión específica). Si la desviación en cierta dimensión es cero, todos los ejemplos coindicen en el valor del centroide pero si es demasiado amplia, debería entenderse que no es representativo del grupo y por lo tanto no sería conveniente incluirlo en el antecedente de la regla. Utilizando $(1\text{-}1.5*desviacion_j)$ si la desviación es alta, el valor de la velocidad *veloc2*, argumento de la función sigmoide, será menor y se reducirá la probabilidad de que el atributo sea utilizado. En todos los casos la velocidad *veloc1* se inicializa en forma aletoria en $[limite1_{inferior\ j}\ ,\ limite1_{superior\ j}]$. La figura 1 muestra el pseudocódigo del método propuesto.

Entrenar la red SOM utilizando todos los ejemplos de entrenamiento.
Calcular el soporte mínimo para cada clase.
Mientras (no se alcance el criterio de terminación)
    Elegir la clase con mayor nro.de ejemplos no cubiertos
    Construir una población reducida de individuos a partir de los centroides
    Evolucionar la población utilizando PSO según lo visto en la sección 4.3
    Obtener la mejor regla de la población
    Si (la regla cumple con el soporte y la confianza pedidos) ***entonces***
        Agregar la regla al conjunto de reglas
        Considerar como cubiertos los ejemplos correctamente clasificados por la regla anterior.
        Recalcular el soporte mínimo para esta clase.
    Fin Si
Fin mientras

Fig.1: Pseudocódigo del método propuesto

6. Resultados obtenidos

En esta sección se compara la performance del método propuesto con el método PART definido por Frank y Witten en [5] en la obtención de las reglas de clasificación para un conocido conjunto de bases de datos del repositorio UCI [17].
Para ello, se realizaron 10 corridas independientes de cada método y se utilizó una red SOM de 30 neuronas organizadas en 6 filas y 5 columnas con 4 vecinos por neurona.
El método PART fue ejecutado con un factor de confianza de 0.3 para el podado del árbol y con el resto de sus parámetros con sus valores por defecto.
La Tabla 1 resume los resultados obtenidos al aplicar ambos métodos. En cada caso se ha considerado no sólo la precisión de la cobertura del conjunto de reglas sino también la claridad del modelo obtenido; esto último se refleja en la cantidad promedio de reglas obtenidas y en la cantidad promedio de términos utilizados para formar el antecedente. Se ha realizado en cada caso un test de Student de diferencia de medias de dos colas con un nivel de significación de 0.05 donde la hipótesis nula

implica que las medias son iguales. Cuando la diferencia es significativa, según el nivel indicado, se ha marcado en la tabla la mejor opción en negrita.

Tabla 1 : Resultados obtenidos al aplicar los métodos SMO+PSO y PART a un conjunto de bases del repositorio UCI. Se ha medido la precisión, la longitud promedio del antecedente de cada regla y la cantidad de reglas utilizadas en cada caso. Se indica en negrita la mejor solución utilizando un test Student con nivel de significación 0.05.

Data set	Precision		Long.Promedio Antecedente		Tamaño Conj.Reglas	
	SOM+PSO	PART	SOM+PSO	PART	SOM+PSO	PART
balance_scale	0,719±0,013	**0,809±0,03**	**1,919±0,114**	3,091±0,059	**6,980±0,518**	38,360±1,249
breast_cancer	**0,712±0.020**	0,660±0,022	**1,166±0,036**	2,009±0,060	**5,060±0,267**	19,140±1,815
breast_w	0,946±0,011	**0,957±0,004**	**1,903±0,198**	2,087±0,127	**2,860±0,212**	10,140±0,568
credit_a	**0,858±0,016**	0,736±0,025	**1,271±0,128**	2,438±0,079	**3,390±0,179**	33,460±2,664
credit_g	0,703±0,012	0,699±0,012	**1,058±0,024**	2,561±0,090	**2,330±0,134**	61,980±1,629
diabetes	**0,745±0,013**	0,734±0,011	**1,193±0,105**	1,945±0,110	**2,540±0,280**	7,590±0,631
heart_c	0,725±0,016	**0,765±0,022**	**1,579±0,061**	2,507±0,092	**3,200±0,183**	19,050±0,580
heart_statlog	0,729±0,023	**0,768±0,013**	**1,645±0,120**	2,888±0,082	**4,330±0,411**	17,630±0,650
iris	0,920±0,032	0,939±0,014	1,145±0,050	**1,025±0,047**	**3,610±0,120**	4,130±0,231
mushroom	0,938±0,007	**0,999±0,002**	1,342±0,122	**1,252±0,021**	**3,530±0,125**	11,260±0,276
promoters	0,647±0,043	0,664±0,033	1,102±0,018	**1,027±0,046**	**7,010±0,233**	7,420±0,483
soybean	**0,859±0,014**	0,555±0,086	5,918±0,253	**2,732±0,051**	**24,810±0,360**	31,640±0,568
kr_vs_kp	0,934±0,008	**0,991±0,001**	**2,357±0,052**	3,119±0,089	**3,000±0,000**	22,210±0,545
zoo	**0,920±0,021**	0,188±0,059	1,533±0,111	1,474±0,009	**6,980±0,132**	7,650±0,071
splice	**0,812±0,013**	0,717±0,017	**1,492±0,046**	2,649±0,032	**10,130±0,283**	102,690±1,854
vinos	0,877±0,025	0,888±0,016	3,073±0,426	**1,503±0,048**	**4,810±0,401**	5,440±0,327
drugte	0,849±0,010	0,873±0,057	**1,733±0,076**	2,069±0,038	**7,330±0,236**	15,410±0,378
drugY	**0,836±0,012**	0,671±0,084	**1,712±0,136**	1,864±0,049	**6,520±0,140**	11,850±0,201
adult_data	0,802±0,004	0,802±0,009	**2,235±0,246**	4,676±0,050	**2,170±0,149**	975,850±10,218

Como puede observase, en los 6 casos en los que la precisión de PART es superior a la del método propuesto, la cobertura obtenida por SOM+PSO es muy buena si se tiene en cuenta el reducido número de reglas que utiliza. Por ejemplo, para la base 'balance_scale', SOM+PSO tiene una precisión inferior a PART en aprox. un 8% pero utiliza un quinto de la cantidad de reglas con menos consultas en cada antecedente. Algo parecido ocurre en los otros 5 casos donde PART posee una precisión superior a SOM+PSO. Esto se debe al énfasis puesto en la simplificación del modelo. En los 13 casos restantes o no hay diferencias significativas en la precisión alcanza o SOM+PSO es mejor. Independientemente de la precisión, la longitud promedio del conjunto de reglas siempre ha sido menor en el método propuesto.

7. Conclusiones

Se ha presentado un nuevo método de obtención de reglas de clasificación que utiliza una variante de PSO binario cuya población de inicializa con la información de los centroides de una red SOM previamente entrenada permitiendo, a través de esta combinación, operar con atributos numéricos y nominales.

Las mediciones realizadas permiten afirmar que el método SOM+PSO obtiene un modelo más simple ya que en promedio utiliza aproximadamente el 20% de la

cantidad de reglas que genera PART, con antecedentes formados por pocas condiciones y una precisión aceptable.
Si bien no han sido incluídas en este artículo, las mediciones realizadas aplicando el método propuesto pero omitiendo la optimización que introduce PSO ha dado lugar a un conjunto de reglas de precisión similar pero con una cardinalidad ligeramente superior a PART. Esto demuestra que las pocas iteraciones que se realizan con PSO a la información de los centroides es fundamental para determinar un adecuado conjunto de reglas.
Actualmente, si bien en base a las pruebas realizadas no se ha evidenciado ninguna dependencia entre los resultados obtenidos y el tamaño inicial de la red SOM, se considera de interés repetir las mediciones utilizando una red SOM dinámica.

References

1. Witten Ian H., Frank Eibe, Hall Mark A.. Data Mining: Practical Machine Learning Tools and Techniques. 3rd Edition. Elsevier. 2011. ISBN 978-0-123-74856-0.
2. José Hernández Orallo, M. José Ramírez Quintana, Cèsar Ferri Ramírez. Introducción a la Minería de Datos. 1ra Edición. Pearson, 2004. ISBN 978-8-420-54091-7.
3. Rakesh Agrawal, Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.
4. Quinlan John Ross. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
5. Frank Eibe, Witten Ian H. Generating Accurate Rule Sets Without Global Optimization. Fifteenth International Conference on Machine Learning, 144-151, 1998.
6. Wang Ziqiang, Sun Xia y Zhang Dexian. A PSO-Based Classification Rule Mining Algorithm. Advanced Intelligent Computing Theories and Applications. LNCS 2007. Springer. Pag.377-384. Vol 4682. ISBN 978-3-540-74201-2.
7. Hung Chihli, Huang Lynn. Extracting Rules from Optimal Clusters of Self-Organizing Maps. 2nd International Conference on Computer Modeling and Simulation – Vol.1, pag. 382-386.2010.
8. Hasperué W, Lanzarini L. Dynamic Self-Organizing Maps. A new strategy to upgrade topology preservation. XXXI Conf. Latinoamericana de Informática, CLEI 2005
9. Kohonen Teuvo. Self-organized formation of topologically correct feature maps. Biological Cybernetics. Springer Berlin/ Heidelberg. Vol43, nro. 1. Pp.59-69. Ed. 1982.
10. Kohonen Teuvo. Self-organizing Maps. 3th Edition. Springer. 2001. ISBN 3-540-67921-9.
11. J. Kennedy and R. Eberhart. Particle swarm optimization. IEEE International Conference on Neural Networks, IV:1942–1948, 1995.
12. J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm Algorithm. Conference on Systems, Man, and Cybernetics, p. 4104-4109, 1997
13. Lanzarini L, López J., Maulini J., De Giusti A. A new Binary PSO with velocity control. Advances in Swarm Intelligence. LNCS 2011, Vol 6728/2011, 111-119. Springer 2011.
14. J. H. Holland. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. Machine Learning, an artificial intelligence approach: Volume II. Morgan Kaufmann, Los Alamos, CA, 1986.
15. Shi Y., Eberhart R. Parameter Selection in Particle Swarm Optimization. *7th* International *Conference on Evolutionary Programming*. pp. 591-600. Springer Verlag 1998.
16. J. Kennedy and R. Eberhart. Swarm Intelligence. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2001.
17. UC Irvine Machine Learning Repository. http://archive.ics.uci.edu/ml/