Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Matemática Computacional
Recibido: 28/05/2018 | Aceptado: 11/09/2018

# Evaluación del algoritmo AR-NSGEP en colecciones de datos desbalanceadas

# *Evaluation of the AR-NSGEP algorithm in unbalanced datasets*

**Alain Guerrero Enamorado[1*], Carlos Morell[2], Sebastián Ventura[3]**

[1]Universidad de las Ciencias Informáticas (UCI), Cuba
[2]Universidad Central "Marta Abreu"de las Villas (UCLV), Cuba
[3]Universidad de Córdoba (UCO), España

*Autor para correspondencia: alaing@uci.cu

**Resumen**
Uno de los grandes problemas que tiene la minería de datos es la existencia del desbalance. Este fenómeno puede afectar gravemente la efectividad de los sistemas de clasificación. Este trabajo persigue como objetivo fundamental obtener información empírica del desempeño del algoritmo AR-NSGEP en colecciones de datos no-balanceados. Se evalúa dicho algoritmo en colecciones de datos con diferentes niveles de desbalance. Se utilizaron colecciones con razones de desbalance entre 1,5 y 40. Durante la etapa de evaluación se utilizaron técnicas de validación cruzada y pruebas estadísticas no-paramétricas para consolidar los resultados obtenidos. La evaluación se realizó con tres métricas muy utilizadas para medir el desempeño en Sistemas Clasificadores con Aprendizaje. Los resultados obtenidos muestran la competitividad del algoritmo AR-NSGEP en colecciones de datos no-balanceados.

**Palabras claves:** Sistemas Clasificadores con Aprendizaje, Desbalance, Clasificación

**Abstract**
*One of the biggest problems with data mining is the existence of imbalance. This phenomenon can seriously affect the effectiveness of classification systems. The main objective of this work is to obtain empirical information of the performance of the AR-NSGEP algorithm in unbalanced datasets. This algorithm is evaluated in datasets with different levels of imbalance. Were used datasets with an unbalance rate between* 1,5 *and* 40*. During the evaluation stage, cross-validation techniques and non-parametric statistical tests were used to consolidate the results obtained. The evaluation was carried out with three metrics widely used to measure the performance in Learning Classifier Systems. The obtained results show the competitiveness of the AR-NSGEP algorithm in unbalanced data collections.*

**Keywords:** *Learning Classifier Systems, Unbalance, Classification*

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

# 1 Introducción

Las Tecnologías de la Información y las Comunicaciones juegan un papel protagónico en la digitalización de grandes volúmenes de datos. La extracción automática de información comprensible y útil desde esta avalancha de datos es de gran importancia para muchos dominios de aplicación. Una de las maneras que han sido desarrolladas para minar estos datos son las técnicas de Aprendizaje Automático (AA). El AA (Gollapudi, 2016) lo define como un mecanismo para la búsqueda de patrones que le permita aprender y hacer predicciones a las máquinas, lo cual implica que sean capaces de mejorar su desempeño futuro a partir de la experiencia. Se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. Así, los algoritmos de Aprendizaje Automático pueden utilizarse para construir modelos de clasificación a partir de un proceso de extracción de patrones desde ejemplos previamente etiquetados. Esto permite que los modelos obtenidos puedan predecir ejemplos nuevos.

Existe una familia de algoritmos de Aprendizaje Automático que utilizan las reglas como forma de Representar el Conocimiento, la cual se combina con un Método de Solución de Problemas (usualmente Algoritmos Genéticos) para obtener clasificadores interpretables. A este tipo de tecnologías se les llama Sistemas Clasificadores con Aprendizaje (SCA) (Urbanowicz and Browne, 2017).

A pesar de todo este esfuerzo por el desarrollo de algoritmos que sean capaces de procesar eficazmente los datos, existen problemas inherentes a estos que pueden afectar el desempeño de los SCA. Uno de los más comunes cuando se trata de datos reales es el desbalance. En (Sanatkar and Haratizadeh, 2015) definen las colecciones de datos no-balanceados como aquellas que tienen una distribución de instancias donde una de las clases es mayoritaria respecto al resto. En presencia de este fenómeno un algoritmo de clasificación puede que no pierda mucho en exactitud predictiva (ACC) incluso si ignora por completo las instancias de la clase minoritaria. Esto puede ser un problema serio, puesto que precisamente la clase minoritaria puede ser la de mayor interés para un problema dado. Algunos ejemplos de problemas de este tipo pueden ser:

- Detección de productos defectuosos en una línea de ensamblaje.
- Detección de enfermedades en una población aparentemente sana. Si la enfermedad que se pretende detectar es rara el problema puede tener un elevado nivel de desbalance.
- Detección de fraudes bancarios (tarjetas de crédito, transacciones).

Para evitar este problema, es necesario hacer un tratamiento a nivel de algoritmo o a nivel de datos. Ejemplo del primer caso, es el trabajo de (Bria et˜al., 2014) donde se presenta un enfoque que utiliza una cascada de

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

clasificadores, donde cada nodo es entrenado por un algoritmo basado en ranking en lugar de utilizar el error de clasificación. El segundo caso se basa en re-balancear las clases disminuyendo la clase mayoritaria o aumentando la minoritaria. En los trabajos (Xue and Hall, 2015; Japkowicz and Stephen, 2002) se realiza un análisis más profundo de este tema. En general se utilizan varios esquemas: sobre-muestreo, sub-muestreo y muestreo aleatorio. En los trabajos (Chawla et˜al., 2002; Wang et˜al., 2015) se han utilizado estas técnicas combinadas con algoritmos evolutivos. Otra forma de abordar el problema es utilizando los Algoritmos Evolutivos Multi-Objetivo (AEMO), de manera que se utiliza un objetivo por cada clase, maximizando al mismo tiempo la exactitud predictiva para la clase minoritaria y para la clase mayoritaria. Este enfoque puede utilizar los enfoques clásicos NSGA-II (Deb et˜al., 2002) o SPEA2 (Zitzler et˜al., 2001).

Con este trabajo se persigue como objetivo fundamental realizar una evaluación del algoritmo AR-NSGEP (Guerrero-Enamorado et˜al., 2018) frente a diferentes niveles de desbalance, para esto se utilizaron colecciones de datos del repositorio UCI (Dua and Karra Taniskidou, 2017). La Sección 2 describe las características fundamentales del algoritmo sometido a evaluación. Posteriormente en la Sección 3 se muestra la metodología seguida para evaluar el algoritmo y se exponen posteriormente los resultados obtenidos. Finalmente, la Sección 4 expone las conclusiones fundamentales del trabajo.

## 2 Materiales y métodos

En esta sección se explica el principio de funcionamiento del algoritmo utilizado en la evaluación del algoritmo, AR-NSGEP, así como los elementos de diseño fundamentales. Mayor nivel de detalles se pueden encontrar en el trabajo donde se introdujo (Guerrero-Enamorado et˜al., 2018). Este algoritmo está basado en el trabajo de (Deb and Jain, 2014). Básicamente tiene dos fases, la primera es para el descubrimiento de reglas y sigue el esquema del algoritmo NSGA-III (Deb and Jain, 2014) pero utilizando el punto utópico (Deb, 2014) como punto de referencia. En la segunda fase del algoritmo dichas reglas son ordenadas en forma de lista de decisión.

La primera fase utiliza un ciclo evolutivo clásico (ver figura 1), se parte de la generación y evaluación de la población inicial, posteriormente se aplican los operadores genéticos de selección, mutación y cruzamiento.

A continuación se evalúa la población obtenida y se da paso a un proceso de reemplazo elitista, que utiliza el esquema del NSGA-III. En este proceso se ordenan las soluciones en frentes de no-dominancia, las soluciones muy cercanas (a menos de una distancia epsilon) entre sí, dentro de un mismo frente son agrupadas dejando solo una al azar. Dentro de un mismo frente de no-dominancia son preferibles las soluciones que tienen una distancia euclidiana menor hasta el punto de referencia utópico de los objetivos. Finalmente se realiza una competición para generar un vector-solución de individuos no redundantes. En cada iteración del ciclo evolutivo se repite el proceso anterior y a su vez esto se realiza para cada clase de la colección de datos.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu


Figura 1: Ciclo evolutivo o fase de descubrimiento de reglas.

La segunda fase inicia al terminar las iteraciones de la primera fase por cada clase. Se crea entonces un clasificador vacío. Se ordenan las reglas obtenidas en la primera fase, se toma la mejor regla y se adiciona al clasificador, se eliminan de la colección de datos las instancias que esta regla cubre, se reordenan las reglas pero ahora en la colección de datos reducida, se extrae nuevamente la mejor regla y se adiciona al clasificador. Este proceso se repite hasta que no queden reglas en la lista de la primera fase o instancias en la colección de datos. El clasificador así obtenido es el que se utiliza para evaluar el algoritmo.

## 2.1 Esquema de codificación de los individuos

En el diseño del algoritmo AR-NSGEP se utilizó la Programación de Expresiones Genéticas (PEG) propuesta por Ferreira (2006). Con esta, se estableció la codificación de los individuos de la población. Esta tecnología aprovecha la eficiencia de los Algoritmos Genéticos (AG) para evolucionar y la potencia de los árboles de la Programación Genética (PG) para construir reglas en forma de Funciones Discriminantes (FD). En este trabajo los individuos son representados genotípicamente en forma de cadenas de caracteres y fenotípicamente en forma de árboles de expresión que se traducen en una función discriminante.

La tecnología PEG separa el genotipo en dos partes: la “cabeza” y la “cola”. La cabeza puede generarse con funciones, atributos predictores o constantes, su tamaño es uno de los parámetros del algoritmo. Una vez definida la cabeza, el tamaño de la cola queda determinado puesto que está diseñada para garantizar que existan suficientes elementos terminales en caso de que la cabeza esté compuesta solo de funciones. Esto garantiza la completitud de los árboles de expresión. La cola se genera solamente con elementos terminales, es

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

decir, atributos predictores o constantes. En los casos donde la cabeza no tenga solo funciones, la información presente en el genotipo (cabeza + cola) no se expresa del todo a nivel fenotípico, pero permanece latente hasta que una condición externa (cruzamiento o mutación) permita su activación. Siguiendo estas restricciones se genera la población inicial de manera aleatoria.

Por otro lado, el algoritmo AR-NSGEP solo utiliza funciones aritméticas básicas (suma, resta, multiplicación y división) en las FD. Se utiliza un enfoque Michigan (Holland and Reitman, 1977) por lo cual cada individuo codifica una sola regla como una FD. El clasificador final se crea como una base de reglas que es ordenada en la segunda fase del algoritmo como se explicó en la sección 2.

El consecuente de cada regla representa la clase que predice y se utilizaron algunos de los operadores genéticos que Ferreira (2006) definió para PEG. A continuación los más importantes:

- Operador de selección: no realiza ningún cambio a nivel genotípico, sin embargo contribuye a generar variación en la frecuencia relativa de diferentes genotipos en una población (*genetic drift*), debido a que brinda la posibilidad de desaparición de genes particulares a medida que los individuos mueren o no se reproducen. La selección se realiza utilizando un torneo binario que tiene en cuenta los frentes de no-dominancia y la distancia euclidiana pesada hasta el punto de referencia utópico. Estos elementos permiten crear un sesgo que provoca que los mejores individuos tengan mayor probabilidad de generar descendencia.
- Operador de mutación simple: puede aplicarse con una probabilidad definida como parámetro, se cambia en cualquier lugar del genotipo un elemento por otro cualquiera.
- Mutación de transposición con inserción de secuencia (TIS), la probabilidad de ocurrencia se define como parámetro. Se selecciona aleatoriamente una secuencia de elementos a transponer, esta se escoge aleatoriamente en tamaño y lugar de inicio. Posteriormente esta secuencia es insertada en una posición aleatoria de la cabeza exceptuando el inicio para evitar que se formen individuos de un solo elemento.
- Mutación de transposición con inserción de secuencia en la raíz (TISR), este operador selecciona aleatoriamente una secuencia de elementos a transponer, pero con la restricción de que inicie con una función. Esta secuencia es insertada en el inicio de la cabeza del genotipo.
- Recombinación por un punto, se escoge aleatoriamente un mismo punto de corte en los progenitores, se intercambian entonces las partes resultantes para generar dos nuevos individuos. Por ejemplo, los individuos Aa y Bb generan la descendencia Ab y aB.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

- Recombinación por dos puntos, se seleccionan dos puntos de corte en cada uno de los progenitores, estos puntos determinan una secuencia de genes en cada uno de ellos. Las secuencias de los progenitores son intercambiadas para generar dos nuevos individuos. Por ejemplo, AaA y BbB generarían la descendencia AbA y BaB.

### 2.2 Clasificación con Funciones Discriminantes

En el algoritmo AR-NSGEP las reglas codifican en su antecedente una FD. Las FD son expresiones matemáticas en las cuales diferentes tipos de operadores y funciones se aplican a los atributos de una instancia devolviendo un valor numérico (Espejo et˜al., 2010). Este valor numérico se puede interpretar como una salida de clasificación binaria a partir de definir un umbral (normalmente cero). El clasificador consistiría en una lista de FD, donde cada función tiene asociada una clase de salida. Para el caso de clasificación binaria, quedaría como en la ecuación 1, donde $X$ es el vector de atributos de entrada. En este caso la función $f(X)$ divide el espacio característico solamente en dos regiones. Para el caso de problemas multi-clase se utiliza el enfoque uno-contra-todos ($OVA$) donde un problema con $n$-clases es transformado en $n$ problemas binarios. Este enfoque utiliza cada clase como positiva mientras el resto de las clases son negativas.

$$if \quad (f(X) > 0) \quad then \quad X \in Class_1 \quad else \quad X \in Class_2 \tag{1}$$

### 2.3 Objetivos que se utilizan para la aptitud

El algoritmo AR-NSGEP utiliza cuatro objetivos para guiar el proceso de búsqueda multi-objetivo, tres de ellos en la dirección de la exactitud y el cuarto en la comprensibilidad de las reglas. Cada uno de estos objetivos se representan en las ecuaciones 2, 3, 4 y 5 respectivamente.

$$Sensibilidad = \frac{vp}{(vp + fn)} \tag{2}$$

$$Especificidad = \frac{vn}{(vn + fp)} \tag{3}$$

$$Precisión = \frac{vp}{(vp + fp)} \tag{4}$$

$$Simplicidad = \frac{TamañoMáximo - 0{,}2(TamañoFenotipo) - 0{,}8}{(TamañoMáximo - 1)} \tag{5}$$

En las ecuaciones anteriores $vp$, $vn$, $fp$ y $fn$ representan los verdaderos positivos, verdaderos negativos, falsos

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

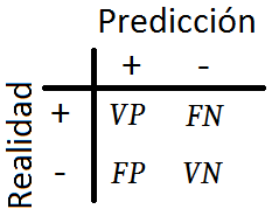


Figura 2: Matriz de confusión de una regla.

positivos y falsos negativos respectivamente de la matriz de confusión (ver figura 2) que se obtiene al evaluar un individuo en el conjunto de entrenamiento. La ecuación 5 es una métrica de la simplicidad de un individuo. El término $TamañoMáximo$ representa el tamaño más grande que puede tomar un individuo. La variable $TamañoFenotipo$, es la longitud del fenotipo del individuo ya expresado. En la sección 2.1 se planteó como puede existir una parte del genotipo que no se expresa en el fenotipo.

### 2.4 Proceso adaptativo de AR-NSGEP

Shen et˜al. (2008) plantea que en los problemas multi-objetivo no todos los objetivos son igualmente importantes y en distintos momentos del proceso evolutivo la prioridad de estos puede cambiar. Para enfrentar estas problemáticas AR-NSGEP incorpora una estrategia adaptativa que modifica la importancia relativa entre los objetivos, de manera que toda la población recibe una presión selectiva en la dirección del mejor individuo de la iteración anterior. Para esto se modifican los pesos de la distancia euclidiana pesada que utiliza el operador de selección (ver sección 2.1) utilizando meta-información del movimiento del mejor individuo en un espacio ROC (ver figura 3).

De esta forma cuando el mejor individuo se mueve en la dirección A, de una generación a otra significa una buena dirección en los objetivos sensibilidad y especificidad al mismo tiempo, por lo cual se resta importancia a ambos objetivos para reducir la velocidad de exploración del algoritmo y así este se concentre más en esa vecindad. En cambio, si el mejor individuo se movió en la dirección C, significa que hubo un empeoramiento en ambos objetivos al mismo tiempo, por lo cual se aumenta la velocidad de exploración del algoritmo para permitir que se escape de esa zona. Los casos intermedios B y D se manejan variando la velocidad en uno u otro objetivo según esta misma lógica. Finalmente, a las ecuaciones de ajuste de los pesos que permiten modificar los niveles de importancia de cada objetivo se aplica un amortiguamiento para lograr que los cambios se realicen de manera menos brusca (ver figura 4).

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

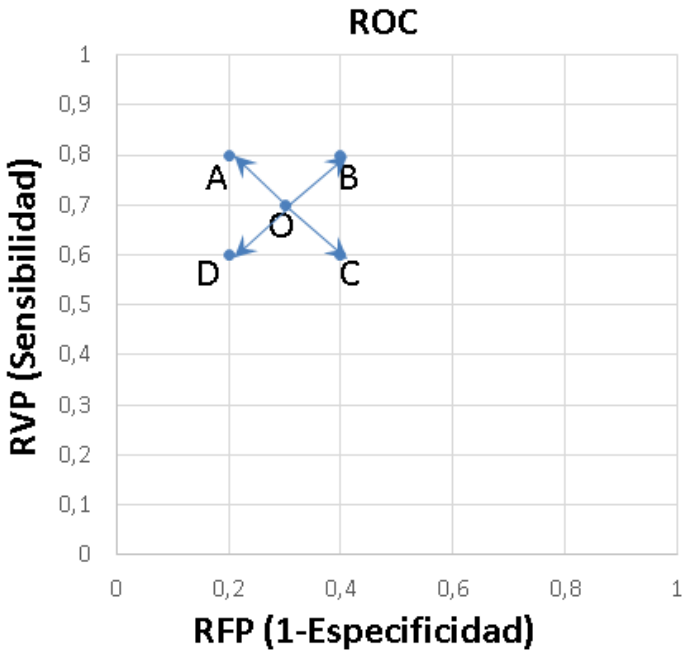


Figura 3: Análisis ROC del movimiento de O.

## 3 Resultados y discusión

En esta sección mostramos la metodología seguida para realizar la evaluación del algoritmo AR-NSGEP. Se utilizaron técnicas de validación cruzada en 10 partes, además de utilizar en cada partición cinco semillas aleatorias distintas, generando un total de 50 experimentos por cada pareja algoritmo base de datos. La evaluación se realiza en 15 colecciones de datos desbalanceadas frente a seis algoritmos que generan clasificadores interpretables (UCS, GASSIST, HIDER, SLAVE, LOGIT-BOOST y CORE).

### 3.1 Bases de datos desbalanceadas

Se utilizaron las siguientes colecciones de datos desbalanceadas tomadas del repositorio UCI (Dua and Karra Taniskidou, 2017): glass1, ecoli-0_vs_1, iris0, glass0, glass-0-1-2-3_vs_4-5-6, ecoli1, ecoli2, glass6, ecoli3, ecoli4, glass-0-1-6_vs_5, glass2, glass4, glass5 y ecoli-0-1-3-7_vs_2-6. Como se puede ver en la tabla 1 las primeras nueve tienen una razón de desbalance menor que 9 y las últimas 6 mayor que 9.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

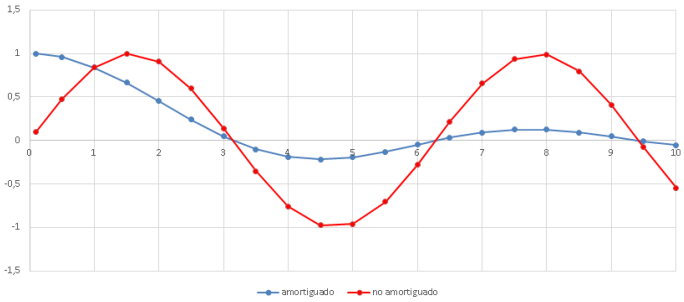


Figura 4: Efecto del factor de amortiguamiento.

### 3.2 Herramientas utilizadas

Para la experimentación se utilizó el marco de trabajo KEEL (Triguero et˜al., 2017), reutilizando las implementaciones que esta herramienta trae de los algoritmos UCS, GASSIST, HIDER, SLAVE, LOGIT-BOOST y CORE. En cada unos de ellos se utilizaron las configuraciones óptimas que sus desarrolladores recomiendan. Dichas configuraciones pueden encontrarse condensadas en el trabajo de Fernández et˜al. (2009). Se utilizó un procesador Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz con GNU/Linux Ubuntu 16.04 LTS y 8GiB de memoria RAM. Para el análisis estadístico se utilizaron pruebas no-paramétricas. En particular se utilizó la prueba de Friedman, combinada con las pruebas *post hoc* de Finner y Li. Según Derrac et˜al. (2011), estas últimas logran los p-valores más pequeños en las comparaciones.

### 3.3 Resultados

En la Tabla 2 se muestran los resultados obtenidos para la métrica de exactitud predictiva (ACC). La última fila muestra el rankin promedio de cada algoritmo entre todas las colecciones. Se puede notar como el algoritmo UCS logra el primer lugar 3,37 en el rankin promedio para esta métrica. Sin embargo, la mayoría de los algoritmos están muy cercanos en este ranking.

Aplicando la prueba de Friedman se obtiene un p-valor = 2.2782E-1. Por lo cual no se logra rechazar la hipótesis nula de que todos los algoritmos se desempeñan de igual manera. Como era de esperar esta métrica no es útil para evaluar colecciones de datos desbalanceadas.

En la Tabla 3 se muestran los resultados obtenidos para la métrica AUC. La última fila muestra el ranking promedio de cada algoritmo entre todas las colecciones de datos. Se puede notar como el algoritmo AR-NSGEP logra el primer lugar 2,17 en el rankin promedio para esta métrica.

Aplicando la prueba de Friedman se obtiene un p-valor = 1.0000E-5. En la Tabla 5 se muestran los resultados

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

Tabla 1: Imbalanced data sets

| Data set | Imbalanced Ratio |
|---|---|
| glass1 | 1.82 |
| ecoli-0_vs_1 | 1.86 |
| iris0 | 2.00 |
| glass0 | 2.06 |
| glass-0-1-2-3_vs_4-5-6 | 3.19 |
| ecoli1 | 3.36 |
| ecoli2 | 5.46 |
| glass6 | 6.38 |
| ecoli3 | 8.19 |
| glass2 | 10.39 |
| ecoli4 | 13.84 |
| glass4 | 15.47 |
| glass-0-1-6_vs_5 | 19.44 |
| glass5 | 22.81 |
| ecoli-0-1-3-7_vs_2-6 | 39.15 |

Tabla 2: Resultados de los algoritmos en la métrica ACC

| Colección | CORE | GAS-SIST | LOGIT BOOST | HIDER | SLAVE | UCS | AR-NSGEP |
|---|---|---|---|---|---|---|---|
| glass1 | 67.20 | 75.42 | 75.79 | 69.91 | 65.42 | 75.51 | 73.17 |
| ecoli-0_vs_1 | 98.45 | 98.45 | 95.45 | 96.45 | 98.18 | 97.73 | 97.36 |
| iris0 | 99.87 | 99.47 | 100.00 | 100.00 | 98.67 | 99.60 | 99.73 |
| glass0 | 72.90 | 81.59 | 84.30 | 75.61 | 71.96 | 81.78 | 75.04 |
| glass-0-1-2-3_vs_4-5-6 | 89.72 | 92.90 | 89.72 | 90.56 | 92.99 | 92.43 | 91.95 |
| ecoli1 | 87.02 | 88.51 | 91.37 | 85.95 | 89.27 | 90.12 | 89.92 |
| ecoli2 | 89.52 | 93.93 | 95.83 | 88.69 | 91.31 | 93.69 | 93.40 |
| glass6 | 95.05 | 94.49 | 85.51 | 94.02 | 96.17 | 96.92 | 96.59 |
| ecoli3 | 90.77 | 92.14 | 91.55 | 91.61 | 91.60 | 91.31 | 92.14 |
| glass2 | 91.59 | 91.40 | 89.91 | 91.50 | 91.96 | 89.35 | 89.63 |
| ecoli4 | 96.61 | 97.32 | 97.56 | 95.83 | 96.42 | 98.10 | 97.97 |
| glass4 | 94.30 | 97.29 | 94.49 | 93.64 | 94.77 | 95.14 | 95.52 |
| glass-0-1-6_vs_5 | 98.26 | 95.00 | 96.30 | 95.11 | 94.57 | 96.30 | 97.29 |
| glass5 | 98.41 | 95.51 | 98.22 | 94.30 | 95.70 | 97.38 | 97.57 |
| ecoli-0-1-3-7_vs_2-6 | 98.65 | 98.51 | 97.15 | 98.08 | 98.79 | 98.36 | 97.95 |
| Rankin | 4.20 | 3.53 | 3.77 | 5.23 | 4.33 | **3.37** | 3.57 |

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

Tabla 3: Resultados de los algoritmos en la métrica AUC

| Colección | CORE | GAS-SIST | LOGIT BOOST | HIDER | SLAVE | UCS | AR-NSGEP |
|---|---|---|---|---|---|---|---|
| glass1 | 56,18 | 72,19 | 72,48 | 66,68 | 55,99 | 71,32 | 69,04 |
| ecoli-0_vs_1 | 97,91 | 97,97 | 94,35 | 95,41 | 97,40 | 97,05 | 96,95 |
| iris0 | 99,80 | 99,20 | 100,00 | 100,00 | 98,00 | 99,50 | 99,80 |
| glass0 | 72,01 | 79,20 | 82,02 | 70,06 | 61,33 | 79,78 | 74,28 |
| glass-0-1-2-3_vs_4-5-6 | 86,38 | 89,54 | 88,13 | 86,39 | 88,39 | 89,37 | 91,22 |
| ecoli1 | 84,10 | 84,52 | 88,38 | 77,11 | 77,51 | 87,11 | 85,61 |
| ecoli2 | 77,31 | 88,55 | 92,67 | 65,35 | 73,65 | 84,49 | 89,51 |
| glass6 | 86,67 | 87,80 | 84,64 | 89,27 | 89,35 | 92,11 | 92,99 |
| ecoli3 | 66,57 | 75,42 | 77,36 | 63,00 | 65,02 | 76,97 | 81,01 |
| glass2 | 49,75 | 49,64 | 52,06 | 49,70 | 49,95 | 54,44 | 52,72 |
| ecoli4 | 78,99 | 85,93 | 87,46 | 70,15 | 70,00 | 88,68 | 90,49 |
| glass4 | 62,43 | 87,05 | 82,68 | 59,20 | 59,80 | 72,23 | 82,80 |
| glass-0-1-6_vs_5 | 92,76 | 68,91 | 81,19 | 60,54 | 53,93 | 80,14 | 94,80 |
| glass5 | 90,67 | 60,48 | 87,39 | 56,66 | 49,95 | 77,39 | 93,98 |
| ecoli-0-1-3-7_vs_2-6 | 79,82 | 83,92 | 77,66 | 82,31 | 85,46 | 85,24 | 88,35 |
| Rankin | 4.63 | 3.73 | 3.23 | 5.70 | 5.47 | 3.07 | **2.17** |

Tabla 4: Resultados de los algoritmos en la métrica NR

| Colección | CORE | GAS-SIST | LOGIT BOOST | HIDER | SLAVE | UCS | AR-NSGEP |
|---|---|---|---|---|---|---|---|
| glass1 | 2.98 | 5.70 | 50.00 | 15.26 | 6.86 | 6075.66 | 10.64 |
| ecoli-0_vs_1 | 2.00 | 4.00 | 50.00 | 2.14 | 2.02 | 5171.80 | 3.92 |
| iris0 | 2.00 | 4.00 | 50.00 | 2.00 | 2.00 | 2552.22 | 2.00 |
| glass0 | 6.18 | 4.70 | 50.00 | 15.68 | 5.40 | 5682.18 | 11.18 |
| glass-0-1-2-3_vs_4-5-6 | 4.12 | 4.32 | 50.00 | 9.74 | 3.24 | 5867.44 | 6.42 |
| ecoli1 | 4.00 | 4.64 | 50.00 | 3.76 | 3.52 | 5656.18 | 8.62 |
| ecoli2 | 5.14 | 4.12 | 50.00 | 3.02 | 2.30 | 5273.98 | 7.50 |
| glass6 | 5.00 | 4.18 | 50.00 | 6.76 | 3.72 | 5645.74 | 4.92 |
| ecoli3 | 5.32 | 4.66 | 50.00 | 2.12 | 3.62 | 5490.24 | 7.94 |
| glass2 | 1.98 | 4.66 | 50.00 | 6.20 | 2.90 | 5139.84 | 10.10 |
| ecoli4 | 2.48 | 4.02 | 50.00 | 1.62 | 2.06 | 5147.96 | 5.46 |
| glass4 | 2.40 | 3.98 | 50.00 | 4.72 | 3.14 | 5655.32 | 6.24 |
| glass-0-1-6_vs_5 | 2.04 | 3.88 | 50.00 | 3.48 | 3.04 | 4014.06 | 3.20 |
| glass5 | 2.08 | 3.90 | 50.00 | 3.54 | 2.56 | 4023.86 | 3.68 |
| ecoli-0-1-3-7_vs_2-6 | 4.20 | 3.72 | 50.00 | 1.00 | 2.00 | 3599.40 | 4.74 |
| Rankin | 2.37 | 3.40 | 6.00 | 3.17 | **1.83** | 7.00 | 4.23 |

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

Tabla 5: Pruebas estadísticas para la métrica AUC

| i | algoritmo | $p_{sinajuste}$ | $p_{Rom}$ | $p_{Finner}$ | $p_{Li}$ |
|---|---|---|---|---|---|
| 1 | HIDER | 0.000007 | 0.000043 | 0.000045 | 0.000010 |
| 2 | SLAVE | 0.000029 | 0.000136 | 0.000086 | 0.000038 |
| 3 | CORE | 0.001766 | 0.006734 | 0.003528 | 0.002361 |
| 4 | GASSIST | 0.047021 | 0.141063 | 0.069696 | 0.059285 |
| 5 | LogitBoost | 0.176296 | 0.253887 | 0.207635 | 0.191126 |
| 6 | UCS | 0.253887 | 0.253887 | 0.253887 | 0.253887 |
| | AR-NSGEP | Control | | | |

de varias pruebas *post hoc* para el AUC donde se nota que los algoritmos HIDER, SLAVE y CORE fueron mejorados significativamente por el algoritmo de control AR-NSGEP. En cambio los algoritmos GASSIST, LogitBoost y UCS no pudieron ser mejorados. Esto constata el efecto que tiene en esta métrica el proceso adaptativo de AR-NSGEP ya que a pesar de no tener en cuenta el desbalance de manera explicita, el algoritmo logra sobreponerse a este.

En la Tabla 4 se muestran los resultados obtenidos para la métrica NR. La última fila muestra el ranking promedio de cada algoritmo entre todas las colecciones de datos. Se puede notar como ahora es el algoritmo SLAVE el que logra el primer lugar 1,83 en el rankin promedio para esta métrica. Se aplicó entonces el método de multi-comparación por pares de Bergmann y Hommel. Para ejecutar esta prueba estadística se utilizó la librería "scmamp" de R (Calvo and Santafé, 2015). Con esta se generó el gráfico de la figura 5 donde el algoritmo SLAVE está resaltado con el primer lugar del ranking. Las líneas conectan aquellos algoritmos que no tienen diferencia significativa entre ellos.

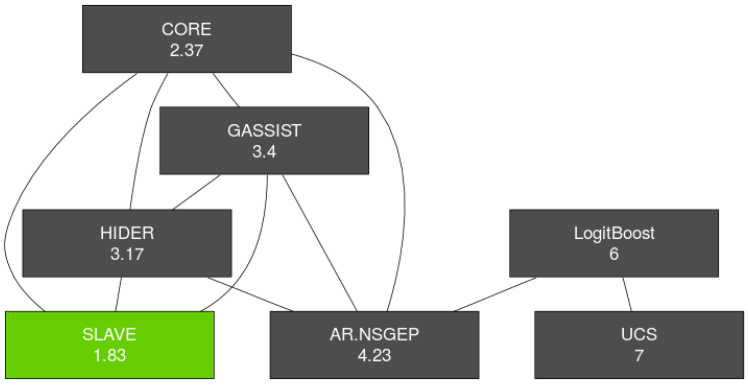


Figura 5: Prueba de Bergmann y Hommel para la métrica NR

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

Finalmente, se creó la figura 6 que combina en un único gráfico el desempeño de los algoritmos en las tres métricas evaluadas (ACC, AUC y NR), el área sombreada abarca la zona donde se encuentran los algoritmos que se desempeñan de manera equivalente respecto a las métricas ACC y AUC al mismo tiempo. En dicha zona se encuentran los mejores algoritmos: GASSIST, LOGITBOOST, UCS y AR-NSGEP, este último muy cercano al punto de mejor desempeño. Las líneas conectan a los algoritmos que no tienen diferencia significativa entre ellos según la prueba de Bergmann y Hommel (ver figura 5). Teniendo en cuenta al mismo tiempo las tres métricas se puede decir que los algoritmos GASSIST, LOGITBOOST y AR-NSGEP se desempeñan de manera equivalente.

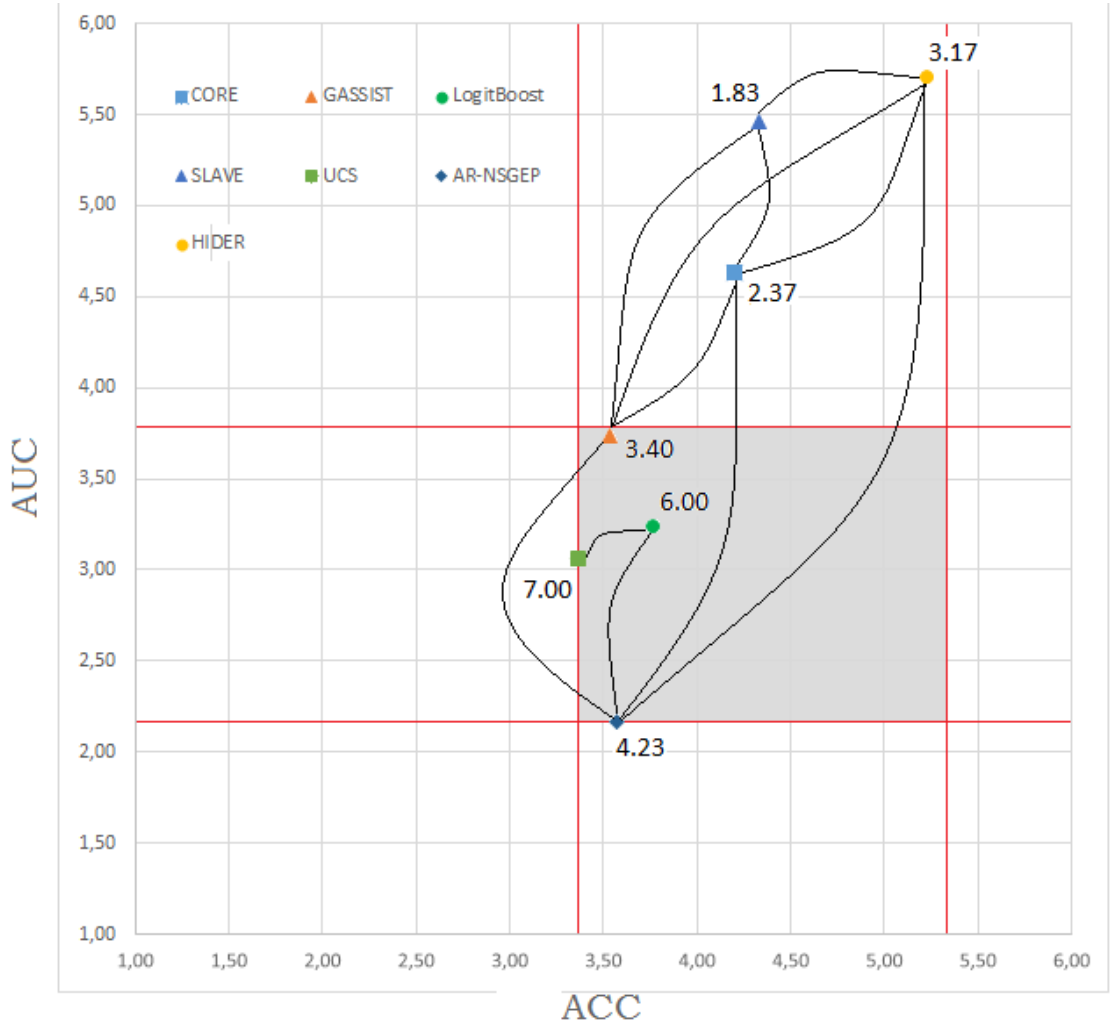


Figura 6: Gráfico del comportamiento de los algoritmos con el aumento del desbalance para la métrica ACC

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 42-57
http://rcci.uci.cu

## 4 Conclusiones

Tras la evaluación experimental se pudo constatar la competitividad del algoritmo AR-NSGEP en presencia del desbalance. Los experimentos mostraron que AR-NSGEP logra quedar entre los primeros lugares del rankin teniendo en cuenta al mismo tiempo las tres métricas (ACC, AUC y NR) utilizadas para evaluar los algoritmos. Para el caso de la métrica AUC, AR-NSGEP quedó en primer lugar del ranking mejorando significativamente a varios de los algoritmos utilizados en la comparación. Junto con AR-NSGEP, los algoritmos GASSIST y LOGITBOOST se desempeñaron de manera equivalente teniendo en cuenta las tres métricas al mismo tiempo. A pesar de estos resultados, no se puede dejar de subrayar que todos los algoritmos que se utilizaron se vieron afectados por este indeseable fenómeno. La métrica ACC no permitió establecer diferencias entre los algoritmos. Esto permitió reafirmar la inefectividad de la métrica ACC para evaluar algoritmos cuando se está en presencia del desbalance entre las clases.

## Referencias

Bria, A., Karssemeijer, N., and Tortorella, F. (2014). Learning from unbalanced data: A cascade-based approach for detecting clustered microcalcifications. *Medical Image Analysis*, 18:241–252.

Calvo, B. and Santafé, G. (2015). scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. *The R Journal*, 8(1):248–256.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Deb, K. (2014). Multi-objective Optimization. In Burke, E. K. and Graham, K., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 15, pages 403–446. Springer, New York, second edition.

Deb, K. and Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577 – 601.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on evolutionary computation*, 6(2):182–197.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Dua, D. and Karra Taniskidou, E. (2017). {UCI} Machine Learning Repository.

Espejo, P. G., Ventura, S., and Herrera, F. (2010). A Survey on the Application of Genetic Programming to Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(2):121–144.

Fernández, A., García, S., Bernadó-Mansilla, E., and Herrera, F. (2009). Genetics-Based Machine Learning for Rule Induction: Taxonomy , Experimental Study and State of the Art.

Ferreira, C. (2006). *Gene Expression Programming Mathematical Modeling by an Artificial Intelligence*. Springer-Verlag, Berlin Heidelberg, second edition.

Gollapudi, S. (2016). *Practical Machine Learning. Tackle the real-world complexities of modern machine learning with innovative and cutting-edge techniques*. Birmingham, UK.

Guerrero-Enamorado, A., Morell, C., and Ventura, S. (2018). A gene expression programming algorithm for discovering classification rules in the multi-objective space. *International Journal of Computational Intelligence Systems*, 11(1):540 – 559.

Holland, J. H. and Reitman, J. O. (1977). Cognitive Systems Based on Adaptive Algorithms. *SIGART Bulletin*, (63):49–49.

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449.

Sanatkar, H. and Haratizadeh, S. (2015). An XCS-Based Algorithm for Classifying Imbalanced Datasets. *International Journal of Intelligent Systems*, 4(6):101–105.

Shen, X., Guo, Y., Chen, Q., and Hu, W. (2008). A multi-objective optimization evolutionary algorithm incorporating preference information based on fuzzy logic. *Computational Optimization and Applications*, 46(1):159–188.

Triguero, I., González, S., Moyano, J. M., García, S., Alcalá-Fdez, J., Luengo, J., Fernández, A., Del Jeús, M. J., Sánchez, L., and Herrera, F. (2017). KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining. *International Journal of Computational Intelligence Systems*, 10:1238–1249.

Urbanowicz, R. J. and Browne, W. N. (2017). Introduction to Learning Classifier Systems. In Weiss, G. and Tuyls, K., editors, *Springer Briefs in Intelligent Systems*, page 135. Springer Berlin Heidelberg.

Wang, S., Minku, L. L., and Yao, X. (2015). Resampling-Based Ensemble Methods for Online Class Imbalance Learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368.

Xue, J.-h. and Hall, P. (2015). Why Does Rebalancing Class-Unbalanced Data Improve AUC for Linear Discriminant Analysis? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(5):1109–1112.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tik report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland.