Práctica 2 Metaheurísticas.
Búsquedas multiarranque para el problema de la selección de características

Jacinto Carrasco Castillo
N.I.F. 32056356-Z
jacintocc@correo.ugr.es

11 de abril de 2016

Curso 2015-2016
Problema de Selección de Características.
Grupo de prácticas: Viernes 17:30-19:30
Quinto curso del Doble Grado en Ingeniería Informática y Matemáticas.

Algoritmos considerados:

1. Búsqueda Multiarranque Básica
2. GRASP
3. ILS

Índice

1. Descripción del problema

El problema que nos ocupa es un problema de clasificación. Partimos de una muestra de los objetos que queremos clasificar y su clasificación, es decir, la clase a la que pertenece y pretendemos, en base a esta muestra, poder clasificar nuevas instancias que nos lleguen. La clasificación se realizará en base a una serie de características, que nos permitan determinar si un individuo pertenece a un grupo u otro. Por tanto, tendremos individuos de una población Ω representados como un vector de características: $\omega \in \Omega; \omega = (x_1(\omega), \dots x_n(\omega))$, donde ω es un individuo de la población y $x_i, i = 1, \dots n$ son las n características sobre las que se tiene información. Buscamos $f : \Omega \longrightarrow C = \{C_1, \dots, C_M\}$, donde $C = \{C_1, \dots, C_M\}$ es el conjunto de clases a las que podemos asignar los objetos.
El problema de clasificación está relacionado con la separabilidad de las clases en el sentido de que existirá la función f anteriormente mencionada siempre que las clases sean separables, es decir, siempre que un individuo con unas mismas características pertenzcan a una misma clase. Sin embargo, si se da que dos individuos $\omega_1, \omega_2 \in \Omega$, $(x_1(\omega_1), \dots x_n(\omega_1)) = (x_1(\omega_2), \dots x_n(\omega_2))$ y sin embargo $f(\omega_1) \neq f(\omega_2)$, no podrá existir f. En todo caso, querríamos obtener la mayor tasa de acierto posible.
Por tanto, queremos, en base a unos datos, hallar la mejor f posible. De esto trata el aprendizaje clasificado: Se conocen instancias de los datos y las clases a las que pertenecen. Usaremos como técnica de aprendizaje supervisado la técnica estadística conocida como k vecinos más cercanos. Se trata de buscar los k vecinos más cercanos y asignar al objeto la clase que predomine de entre los vecinos. En caso de empate, se seleccionará la clase con más votos más cercana.
Pero no nos quedamos en el problema de clasificación, sino que buscamos reducir el número de características. Con esto pretendemos seleccionar las características que nos den un mejor resultado (por ser las más influyentes a la hora de decidir la categoría). Usaremos los datos de entrenamiento haciendo pruebas mediante diferentes metaheurísticas hasta obtener la mejor selección que seamos capaces de encontrar.
El interés en realizar la selección de características reside en que se aumentará la eficiencia, al requerir menos tiempo para construir el clasificador, y que se mejoran los resultados al descartar las características menos influyentes y que sólo aportan ruido. Esto hace también que se reduzcan los costes de mantenimiento y se aumente la interpretabilidad de los datos.
Las funciones de evaluación pueden estar basadas en la consistencia, en la Teoría de la Información, en la distancia o en el rendimiento de clasificadores. Nosotros usaremos el rendimiento promedio de un clasificador $3 - NN$.

2. Descripción de la aplicación de los algoritmos

2.1. Representación de soluciones

Para este problema tenemos varias formas posibles de representar las soluciones:

- Representación binaria: Cada solución está representada por un vector binario de longitud igual al número de características, donde las posiciones seleccionadas tendrán un 1 o `True` y las no seleccionadas un 0 o `False`. Esta opción, que será la que tomaremos, sólo es recomendable si no tenemos restricciones sobre el número de características seleccionadas.

- Representación entera: Cada solución es un vector de tamaño fijo $m \leq n$ con las características seleccionadas. Esta representación sí es adecuada si tenemos restricciones sobre el número de características tomadas ya que no podemos hacerlo con más de m.

- Representación de orden: Cada solución es una permutación de n elementos, ordenados según la importancia de cada característica. Aquí también se maneja el cumplimiento de restricciones pues una vez encontrada la solución, tomaremos sólo las primeras m características.

Se ha de mencionar que en las dos últimas representaciones el espacio de soluciones es mayor que el espacio de búsqueda, justificado en la representación de orden porque da más información (podríamos tomar soluciones de longitud variable), pero que en la representación entera sólo es razonable asumir si tenemos una restricción de longitud fija. Además, otra ventaja de la representación binaria es la facilidad para aplicarle operadores (de vecindario, en posteriores prácticas de cruce...) manteniendo la consistencia.

2.2. Función objetivo

La función objetivo será el porcentaje de acierto en el conjunto de test para el clasificador $3 - NN$ obtenido usando las distancias de los individuos ω en las dimensiones representadas por las características seleccionadas en el vector solución para el conjunto de entrenamiento. El objetivo será maximizar esta función. A la hora de buscar esta solución sólo estaremos manejando los datos de entrenamiento, luego aquí la función objetivo será la media de tasa de acierto para cada uno de los datos de entrenamiento con respecto a todos los demás, por lo que tenemos que usar la técnica de *Leave-One-Out*. Esta técnica consiste en quitar del conjunto de datos cada uno de los elementos, comprobar el acierto o no para este dato en concreto, y devolverlo al conjunto de datos. Así evitamos que los resultados estén sesgados.

```
targetFunction(data_train, categories_train, data_test,
                                    categories_test, solution):
```

```
BEGIN
    num_items ← length(data_test)
    sum_score ← 0

    data_train' ← {col_i from data_train if solution_i is True}
    classifier ← Make3NNClasifier(data_train', categories_train)

    data_test' ← {col_i from data_test if solution_i is True}

    FOR item IN data_test'
        predicted_class ← classifier(item)
        IF predicted_class == categories_test_item THEN
            sum_score ← sum_score + 1
    END

    RETURN sum_score / num_items * 100
END
```

La función de evaluación incluida en el código no la he realizado yo sino que he utilizado el paquete `sklearn` de `Python` por razones de eficiencia. A lo largo de la práctica (en los métodos de búsqueda) haré referencia a una función $getValue(data, categories, solution)$ que representa la función de evaluación con la técnica del *Leave-One-Out* explicada anteriormente.

2.3. Operadores comunes

Entenderemos como vecindario de una solución a los vectores que sólo difieren en una posición. Por tanto, el operador para movernos a una solución vecina consistirá en cambiar una posición determinada:

```
flip(solution, positon):
BEGIN
        neighbour ← copy(solution)
        actual_value ← solution_position
        neighbour_position ← NOT actual_value
        RETURN neighbour
END
```

3. Estructura del método de búsqueda

3.1. Búsqueda multiarranque básica

3.2. GRASP

3.3. ILS

4. Algoritmo de comparación

Como algoritmo de comparación tenemos el algoritmo *greedy* SFS. Partiendo de un vector con ninguna característica seleccionada, exploramos por el entorno y nos quedamos con el vecino que genera una mejor tasa de acierto. Repetimos este proceso hasta que ningún vecino aporta una mejora a la solución obtenida.

```
greedySFS(data, categories) BEGIN
    solution ← {0,...,0: size = num_features}
    current_value ← getValue(data, categories, solution)

    REPEAT
        neighbours ← {flip(solution,i): i in characteristics}

        best_value ← máx_neighbours getValue(data, categories, ·)

        IF best_value > current_value THEN
            solution ← argmax_neighbours getValue(data, categories, ·)

    WHILE(best_value > current_value)

    RETURN solution
END
```

5. Procedimiento para desarrollar la práctica

El código de la práctica está realizado en `Python 3.5.1`. Comencé por el algoritmo del KNN y el algoritmo *greedy* SFS. Sin embargo, viendo los tiempos (la iteración en la base de datos de Arrhythmia duraba en torno a 10 minutos) me decanté por usar el módulo de `Python sklearn` para realizar la normalización de los datos, la validación cruzada y el algoritmo KNN, permitiendo entrenar el clasificador con los datos y obtener un *score* al comprobar si concuerda la clase predicha con la original.

Los paquetes utilizados son `sklearn`, como se ha mencionado, `scipy` para leer de una manera sencilla la base de datos, `numpy` para el manejo de vectores y matrices y tratar que sea algo más eficiente en lugar de las listas de `Python` y `ctype` para importar el generador de números aleatorios en `C` disponible en la página web de la asignatura. La semilla con la que he realizado las ejecuciones es 3141592, insertada tanto en el generador en `C` como por el generador de número de aleatorios de `numpy`. He usado los dos porque pretendía usar el primero, que es con el que se realizan las particiones, pero al llegar a los métodos que usan los generadores de números pseudoaleatorios en su funcionamiento me di cuenta de que tendría que repetir el código de importación del módulo en `C` para cada método, por lo que opté por usar en los métodos el `random` de `numpy`. Mientras he ido realizando la práctica he ido creando funciones para permitir un más cómodo manejo del programa intentando que el código fuese entendible.

5.1. Ejecución del programa

La salida de cada ejecución (10 iteraciones de un algoritmo con una base de datos) se puede elegir entre mostrar por pantalla o redirigir a un archivo `.csv` para manejarlo posteriormente, por ejemplo para incluir la tabla en LaTeX.
Los parámetros que acepta el programa son:

- Base de datos: Será una letra `W,L,A` que representa cada una de las bases de datos a utilizar. Este parámetro es el único obligatorio.
- Algoritmo utilizado: Por defecto es el KNN. Para introducir uno distinto, se usa `-a` seguido de una letra entre `K,G,L,S,T,E` que se corresponden con KNN, *greedy* SFS, *local search*, *simulated annealing*, *tabu search* y *extended tabu search*, respectivamente.
- Semilla. Para incluir una semilla, se añade `-seed` seguido del número que usaremos como semilla. Por defecto es 3141592.
- Salida por pantalla o a fichero. Se utiliza con el parámetro opcional `-write` para escribir en un fichero en una carpeta llamada `Resultados`. El nombre del fichero será la primera letra de la base de datos utilizada seguida por las iniciales del algoritmo. Incluye también la media para cada columna en la última fila.
- `-h` o `--help` para mostrar la ayuda y cómo se introducen los parámetros.

Por tanto, la ejecución del programa se hará de la siguiente manera:

```
python Practica1.py base_de_datos [-a algoritmo -seed semilla -write T/F ]
```

Si por ejemplo queremos lanzar la base de datos de `WDBC` con la búsqueda local, semilla 123456 y que los resultados se muestren por pantalla, escribimos

```
python Practica1.py W -a L -seed 123456
```

Si optamos por la base de datos `Arrhythmia` con la búsqueda tabú extendida y guardar el resultado en un fichero:

```
python Practica1.py A -a E -write True
```

Para mostrar la introducción de parámetros:

```
python Practica1.py --help
```

6. Experimentos y análisis de resultados

6.1. Descripción de los casos

Los casos del problema planteados son tres, cada uno de ellos asociado a una base de datos:

- WDBC: Base de datos con los atributos estimados a partir de una imagen de una aspiración de una masa en la mama. Tiene 569 ejemplos, 30 atributos y debemos clasificar cada individuo en dos valores.
- Movement Libras: Base de datos con la representación de los movimientos de la mano en el lenguaje de signos LIBRAS. Tiene 360 ejemplos y consta de 91 atributos.
- Arrhythmia: Contiene datos de pacientes durante la presencia y ausencia de arritmia cardíaca. Tiene 386 ejemplos y 278 atributos para categorizar en 5 clases.

Hay que mencionar que tanto la ejecución de la búsqueda tabú como la búsqueda tabú extendida se han realizado no con 15000 comprobaciones como indica el guión de la práctica sino con 5000 debido a que lleva demasiado tiempo ejecutarlas al no tener otra condición de parada que realizar estas ejecuciones. En la medida de los tiempos se ha incluido la calificación del conjunto de test, con lo que se añade un poco de tiempo extra en cada iteración, que no es representativo con respecto a lo que se tarda en obtener la solución. Esto nos da una idea sobre la influencia del tamaño de los datos sobre cada evaluación de la solución actual y la influencia que esto tiene en el tiempo para obtener la solución, pues no sólo influye el tamaño del espacio de búsqueda.

6.2. Resultados

6.2.1. KNN

	WDBC				Movement Libras				Arrhythmia			
	%Clas. in	%Clas. out	red	T	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T
Particion 1-1	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528
Particion 1-2	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587
Particion 2-1	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544
Particion 2-2	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755
Particion 3-1	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462
Particion 3-2	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653
Particion 4-1	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652
Particion 4-2	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604
Particion 5-1	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589
Particion 5-2	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559
Media	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009

6.2.2. SFS

	WDBC				Movement Libras				Arrhythmia			
	%Clas. in	%Clas. out	red	T	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T
Particion 1-1	78,6458	68,5567	98,5612	342,8862	78,6458	68,5567	98,5612	342,8862	78,6458	68,5567	98,5612	342,8862
Particion 1-2	81,9588	74,4792	97,482	562,2888	81,9588	74,4792	97,482	562,2888	81,9588	74,4792	97,482	562,2888
Particion 2-1	78,125	69,0722	98,2014	419,6417	78,125	69,0722	98,2014	419,6417	78,125	69,0722	98,2014	419,6417
Particion 2-2	77,3196	68,75	97,482	559,0299	77,3196	68,75	97,482	559,0299	77,3196	68,75	97,482	559,0299
Particion 3-1	82,2917	74,2268	96,7626	678,8885	82,2917	74,2268	96,7626	678,8885	82,2917	74,2268	96,7626	678,8885
Particion 3-2	81,9588	74,4792	97,482	553,0682	81,9588	74,4792	97,482	553,0682	81,9588	74,4792	97,482	553,0682
Particion 4-1	78,125	69,0722	97,482	535,9987	78,125	69,0722	97,482	535,9987	78,125	69,0722	97,482	535,9987
Particion 4-2	73,7113	67,1875	98,2014	417,0979	73,7113	67,1875	98,2014	417,0979	73,7113	67,1875	98,2014	417,0979
Particion 5-1	77,6042	69,0722	98,9209	268,0201	77,6042	69,0722	98,9209	268,0201	77,6042	69,0722	98,9209	268,0201
Particion 5-2	82,9897	77,0833	96,7626	736,5666	82,9897	77,0833	96,7626	736,5666	82,9897	77,0833	96,7626	736,5666
Media	79,273	71,1979	97,7338	507,3487	79,273	71,1979	97,7338	507,3487	79,273	71,1979	97,7338	507,3487

6.2.3. Búsqueda multiarranque básica

	WDBC				Movement Libras				Arrhythmia			
	%Clas. in	%Clas. out	red	T	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T
Particion 1-1	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528
Particion 1-2	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587
Particion 2-1	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544
Particion 2-2	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755
Particion 3-1	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462
Particion 3-2	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653
Particion 4-1	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652
Particion 4-2	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604
Particion 5-1	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589
Particion 5-2	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559
Media	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009

6.2.4. GRASP

	WDBC				Movement Libras				Arrhythmia			
	%Clas. in	%Clas. out	red	T	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T
Particion 1-1	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528
Particion 1-2	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587
Particion 2-1	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544
Particion 2-2	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755
Particion 3-1	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462
Particion 3-2	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653
Particion 4-1	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652
Particion 4-2	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604
Particion 5-1	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589
Particion 5-2	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559
Media	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009

6.2.5. ILS

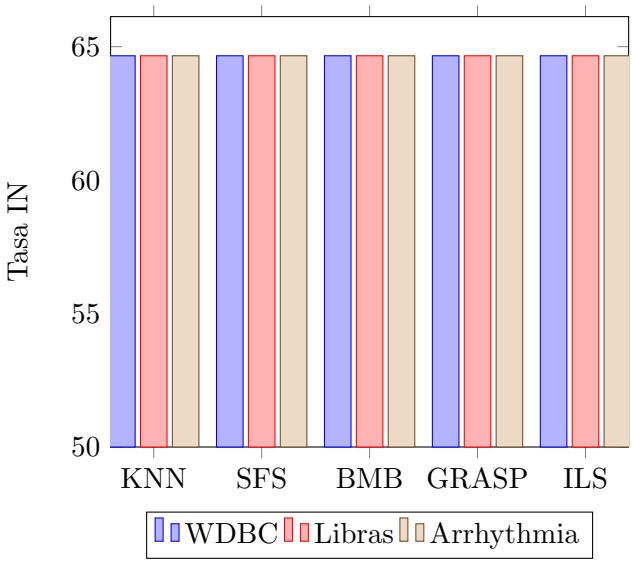
	WDBC				Movement Libras				Arrhythmia			
	%Clas. in	%Clas. out	red	T	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T
Particion 1-1	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528	64,5833	63,9175	0	0,4528
Particion 1-2	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587	62,8866	65,625	0	0,5587
Particion 2-1	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544	63,5417	65,4639	0	0,6544
Particion 2-2	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755	64,9485	65,1042	0	0,5755
Particion 3-1	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462	65,1042	65,9794	0	0,462
Particion 3-2	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653	67,0103	64,0625	0	0,4653
Particion 4-1	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652	66,1458	60,8247	0	0,4652
Particion 4-2	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604	65,9794	64,5833	0	0,4604
Particion 5-1	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589	60,4167	67,5258	0	0,4589
Particion 5-2	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559	65,9794	60,9375	0	0,4559
Media	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009	64,6596	64,4024	0	0,5009

6.2.6. Comparación

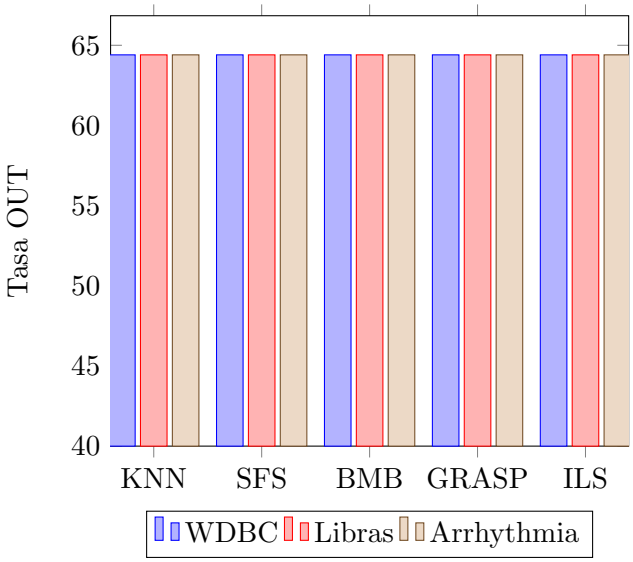
	WDBC				Movement Libras				Arrhythmia			
	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T	%Clas. in	%Clas. out	%red.	T
KNN	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009
SFS	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009
BMB	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009
GRASP	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009
ILS	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009	64.6596	64.4024	0.0000	0.5009

6.3. Análisis de los resultados

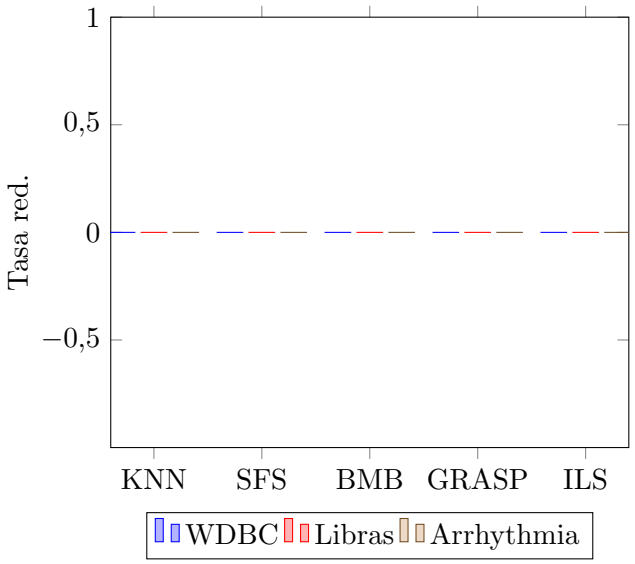
6.3.1. Tasa In



6.3.2. Tasa Out

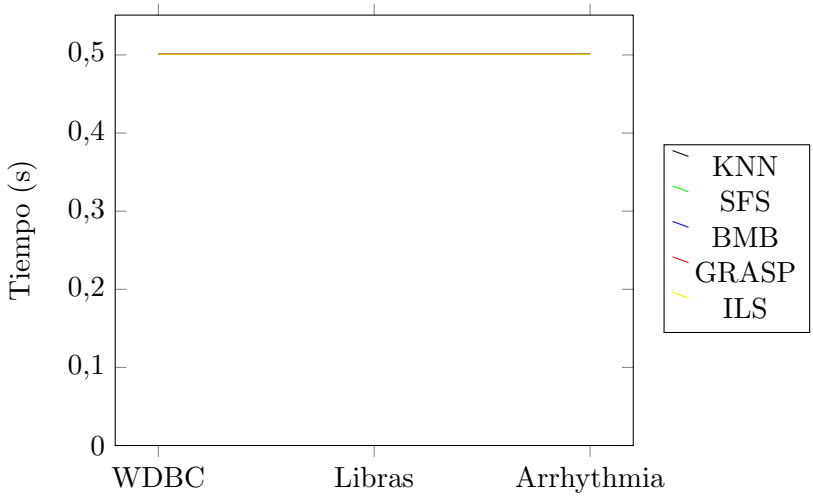


6.3.3. Tasa reducción



6.3.4. Tiempos

DB	0	1	2	3	4
TW	0.5009	0.5009	0.5009	0.5009	0.5009
TL	0.5009	0.5009	0.5009	0.5009	0.5009
TA	0.5009	0.5009	0.5009	0.5009	0.5009



7. Bibliografía

- Módulo en `scikit` para KNN
- Para realizar las tablas en LaTeX: Manual PGFPLOTSTABLE