METAHEURÍSTICAS

Práctica 2 - APC

Grupo 1

Mario Muñoz Mesa
mario43514@correo.ugr.es

20 de mayo de 2022

Índice

1. Descripción del problema.

Nos ocupamos del problema del Aprendizaje de Pesos en Características (APC). Plantea, en el contexto de un problema de clasificación, elegir o ajustar un vector de pesos $\mathbf{w} = (w_1, \ldots, w_d) \in [0,1]^d$ asociado a las características en base a un criterio de mejora del clasificador. Hemos denotado con d al número de características.

En nuestro caso trabajaremos con clasificador tipo 1-NN, y el criterio será maximizar:

$$F(\mathbf{w}) := \alpha \cdot \text{tasa\_clas}(\mathbf{w}) + (1-\alpha) \cdot \text{tasa\_red}(\mathbf{w})$$

donde

$$\text{tasa-clas} := 100\frac{\text{nº instancias bien clasificadas en training}}{\text{nº instancias en training}}, \qquad \text{tasa-red} := 100\frac{\text{nº valores } w_i < 0.2}{\text{nº características}}$$

y $alpha = 0.5$ que pondera la importancia entre el acierto y la reducción de características para el nuevo mejor clasificador que se pretende encontrar.

De lo que nos ocupamos por tanto es de obtener $\arg\max_{\mathbf{w}\in[0,1]^d} F(\mathbf{w})$ para un clasificador 1-NN que utilizará la distancia ponderada

$$d_{\mathbf{w}}(u,v) = \sqrt{\sum_{i=1}^{d} w_i(u_i - v_i)^2}, \quad u, v \in \mathbb{R}^d$$

para clasificar. Queremos que aumente el acierto y reduzca el número de características, cualquier característica con peso menor estricto a 0.2 se descarta.

2. Descripción de la aplicación de los algoritmos empleados.

2.1. Asociados al preprocesado y representación. P1.

Partimos de un conjunto de entrenamiento T formado por instancias de una muestra. Cada instancia se ha representado mediante una estructura de datos `SampleElement` que recoge las valores de las características en un vector `vector<double>` y la clase como una cadena de caracteres.

De esta forma, cada dataset se lee mediante la función `read_arff` y lo almacenamos como un vector de elementos muestrales `vector<SampleElement>`.

Para la normalización utilizamos min-max normalization, se ha implementado en `normalization`.

Las particiones para la validación cruzada $k-fold$ se realizan respetando la proporción de clases en cada partición, para ello cada elemento de una clase se va asignando a una partición de forma cíclica.

La solución será un vector de pesos con valores entre 0 y 1, cada componente indica el peso de una característica, se representa como un `vector<double>`

2.2. Asociados al preprocesado y representación. P2.

Para los algoritmos genéticos se ha decidido representar cada cromosoma como un estructura de datos `Chromosome` que contiene un `vector<double>`, que representa los genes del cromosoma, junto con el el valor de la función objetivo asociado a ese cromosoma.

Además se han definido los comparadores mayor, $>$, y menor, $<$. Un cromosoma es mayor a otro si su función objetivo es mayor, y menor si la función objetivo es menor.

Para representar una población de cromosomas hacemos uso de una simple estructura de vector, `vector<Chromosome>`.

2.3. Funciones generales. P1.

Nuestro clasificador utiliza la distancia euclídea, para manejarla se han implementado dos funciones: `euclidean_distance2` y `euclidean_distance2_w`.

Algorithm 1: euclidean_distance2

Input: vector de reales a, vector de reales b
Output: la distancia euclídea al cuadrado entre a y b
begin
  dist $\leftarrow 0$
  for $i = 0$ *to* $a.size() - 1$ **do**
    dist $\leftarrow$ dist $+ (a[i] - b[i])^2$
  end
  return dist
end

Algorithm 2: euclidean_distance2_w

Input: vector de reales a, vector de reales b, vector de pesos $weights$
Output: la distancia euclídea ponderada y al cuadrado entre a y b
begin
  dist $\leftarrow 0$
  for $i = 0$ ***to*** $a.size() - 1$ **do**
    `// se descartan las características con peso <2`
    if $weights[i] \geq 0.2$ **then**
      dist $\leftarrow$ dist $+ (a[i] - b[i])^2$
    end
  end
  return dist
end

La función `euclidean_distance2` se utiliza exclusivamente para el cálculo de amigo y enemigo más cercano en el algoritmo Relief. Con `euclidean_distance2_w` calculamos la distancia ponderada descartando pesos menores a 0.2, se utiliza en 1-NN y Búsqueda Local.

En ambos casos calculamos la distancia euclídea al cuadrado por motivos de eficiencia y porque lo que nos interesa es comparar distancias, y como la raíz cuadrada es una función creciente, si $a > b \Rightarrow \sqrt{a} > \sqrt{b}$, no hay inconveniente en trabajar con la distancia al cuadrado.

El algoritmo 1-NN, haciendo uso de distancia ponderada, se ha implementado en `one_NN`.

Algorithm 3: one_NN

Input: elemento muestral sam_el a clasificar, conjunto de elementos muestrales sobre el que buscar el más cercano $sam\_elements$, vector de pesos $weights$
Output: la clase o etiqueta del elemento más cercano, min_l
begin
  $min\_dist \leftarrow \infty$
  for e ***in*** $sam\_elements$ **do**
    $d \leftarrow$ euclidean_distance2_w(e.features, sam_el.features, weights)
    if $d < min\_dist$ **then**
      $min\_l \leftarrow$ e.label
      $min\_dist \leftarrow d$
    end
  end
  return min_l
end

La versión con leave-one-out, que se utiliza en Búsqueda Local, simplemente consiste en dejar un elemento fuera.

Algorithm 4: one_NN_lo

Input: elemento muestral *sam_el* a clasificar, conjunto de elementos muestrales sobre el que buscar el más cercano *sam_elements*, vector de pesos *weights*, posición de elemento a dejar fuera *leave_out*

Output: la clase o etiqueta del elemento más cercano, *min_l*

```
begin
    min_dist ← ∞
    for i = 0 to sam_elements.size() − 1 do
        if i ≠ leave_out then
            d ← euclidean_distance2_w(sam_elements[i].features, sam_el.features, weights)
            if d < min_dist then
                min_l ← sam_elements[i].label
                min_dist ← d
            end
        end
    end
    return min_l
end
```

2.4. Funciones generales. P2.

El operador de selección utiliza un torneo binario; escoge dos cromosomas de la población aleatoriamente y devuelve el que de mayor valor en la función objetivo.

Algorithm 5: selection

Input: Población de cromosomas *population*

Output: El cromosoma elegido por torneo binario

```
begin
    pop_size_1 ← population.size() − 1
    c1 ← population[random(0, pop_size_1)]
    c1 ← population[random(0, pop_size_1)]
    if c1.obj > c2.obj then
        return c1
    end
    else
        return c2
    end
end
```

El operador de cruce blx toma dos cromosomas y genera dos descendientes. Cada gen de cada cromosoma descendiente es generado aleatoriamente en intervalo $[c_{min} - I \cdot \alpha, c_{max} + I \cdot \alpha]$ donde c_{min} y c_{max} son el mínimo y máximo entre los dos genes correspondientes de los cromosomas iniciales, $I = c_{max} - c_{min}$ y $\alpha = 0.3$. Los valores resultantes en cada gen se restringen a $[0, 1]$.

Algorithm 6: blx_cross

Input: Dos cromosomas $p1$ y $p2$
Output: Los dos cromosomas descendientes $d1$ y $d2$
begin
  $pgenes\_size \leftarrow p1.genes.size()$
  for $i = 0$ ***to*** $pgenes\_size$ **do**
    $c_{min} \leftarrow \text{mín}\{p1.genes[i], p2.genes[i]\}$
    $c_{max} \leftarrow \text{máx}\{p1.genes[i], p2.genes[i]\}$
    $I\alpha \leftarrow (c_{max} - c_{min}) * 0.3$
    $d1.genes[i] \leftarrow random(c_{min} - I\alpha, c_{max} + I\alpha)$
    $d2.genes[i] \leftarrow random(c_{min} - I\alpha, c_{max} + I\alpha)$
    `// restringimos a intervalo [0,1]`
    $restrict\_to\_0\_1(d1.genes[i])$
    $restrict\_to\_0\_1(d2.genes[i])$
  end
  `// para evitar evaluaciones de f.obj redundantes antes de la mutación`
  $d1.obj \leftarrow -1$
  $d2.obj \leftarrow -1$
  return $d1$, $d2$
end

El operador de cruce aritmético toma dos cromosomas y genera también dos descendientes. En este caso para cada descendiente en vez de tomar $\alpha = 0.5$ se toma un α aleatorio entre 0 y 1 (concretamente entre 0.01 y 0.99 en la implementación), esto se hace para evitar la generación de descendientes idénticos.

Algorithm 7: arith_cross

Input: Dos cromosomas $p1$ y $p2$
Output: Los dos cromosomas descendientes $d1$ y $d2$
begin
  $gen\_size \leftarrow p1.genes.size()$
  $\alpha \leftarrow random(0.01, 0.99)$
  for $i = 0$ ***to*** gen_size **do**
    $d1.genes[i] \leftarrow \alpha * p1.genes[i] + (1 - \alpha) * p2.genes[i]$
  end
  $\alpha \leftarrow random(0.01, 0.99)$
  for $i = 0$ ***to*** gen_size **do**
    $d2.genes[i] \leftarrow \alpha * p1.genes[i] + (1 - \alpha) * p2.genes[i]$
  end
  `// para evitar evaluaciones de f.obj redundantes antes de la mutación`
  $d1.obj \leftarrow -1$
  $d2.obj \leftarrow -1$
  return $d1$, $d2$
end

El operador de mutación coincide con el de generación de vecinos en la BL, introduce una perturbación con una normal de media 0 y desviación típica 0.3.

Algorithm 8: mutation

Input: vector de pesos $weights$, índice a mutar i, distribución normal n_dist
begin
  $weights[i] \leftarrow weights[i] + e \in \mathcal{N}_{n\_dist}(0, 0.3)$
  if $weigths[i] > 1$ **then**
    $weights[i] \leftarrow 1$
  end
  else
    if $weights[i] < 0$ **then**
      $weights[i] \leftarrow 0$
    end
  end
end

Para realizar un número de mutaciones sobre una población de cromosomas se define la función `mutate_pop`, ésta controla que no se repitan las mutaciones sobre un gen de un cromosoma.

Algorithm 9: mutate_pop

Input: vector de cromosomas $population$, número de mutaciones a realizar num_muts
begin
  $pop\_size\_1 \leftarrow population.size() - 1$
  $genes\_size\_1 \leftarrow num\_of\_genes - 1$ `// equivale a num_feats-1 en implementación`
  $mutated \leftarrow \{\}$
  $muts \leftarrow 0$
  while $muts < num\_muts$ **do**
    $chr\_i \leftarrow random(0, pop\_size\_1)$
    $gen\_i \leftarrow random(0, genes\_size\_1)$
    if $\{chr\_i,\ gen\_i\} \notin mutated$ **then**
      $mutation(population[chr\_i].genes,\ gen\_i,\ \mathcal{N}(0, 0.3))$
      $muts \leftarrow muts + 1$
      $mutated \leftarrow mutated \cup \{chr\_i,\ gen\_i\}$
      $population[chr\_i].obj \leftarrow -1$
    end
  end
end

Para evaluar la función objetivo asociada a los genes de un cromosoma y asignarlo al cromosoma se ha creado la función `obj_to_chromosome`:

Algorithm 10: obj_to_chromosome

Input: cromosoma c, vector de elementos muestrales $training$
begin
  for $i = 0$ ***to*** $training.size()$ **do**
    $class\_labels.push\_back(\ one\_NN\_lo(training[i], training, c.genes, i)\ )$
  end
  $c.obj \leftarrow obj\_function(class\_rate(class\_labels, training),\ red\_rate(c.genes))$
end

2.5. Función objetivo.

Para el cálculo de las tasas se han implementado las funciones `class_rate` y `red_rate` que simplemente implementan la propia de definición de tasa-class y tasa-red.

Algorithm 11: class_rate

Input: vector con clases o etiquetas de elementos clasificados *class_labels*, elementos de test *test*
Output: el porcentaje de acierto
begin
  $n \leftarrow 0$
  $class\_labels\_s \leftarrow class\_labels.size()$
  for $i = 0$ ***to*** $class\_labels\_s - 1$ **do**
    if $class\_labels\_s[i] == test[i].label$ **then**
      $n \leftarrow n + 1$
    end
  end
  return $100\frac{n}{class\_labels\_n}$
end

Algorithm 12: red_rate

Input: vector de pesos *weights*
Output: el porcentaje de reducción
begin
  $feats\_reduced \leftarrow 0$
  for w ***in*** $weights$ **do**
    if $w < 0.2$ **then**
      $feats\_reduced \leftarrow feats\_reduced + 1$
    end
  end
  return $100\frac{feats\_reduced}{initial\_number\_of\_features}$
end

La función objetivo, teniendo en cuenta que trabajamos con $\alpha = 0.5$, nos quedaría:

Algorithm 13: obj_function

Input: tasa de clasificación *class_rate*, tasa de reducción *red_rate*
Output: el valor de la función objetivo o fitness
begin
  return $\alpha \cdot class\_rate + (1 - \alpha) \cdot red\_rate$
end

3. Descripción de los algoritmos.

3.1. Algoritmos genéticos.

Los algoritmos genéticos generacionales y estacionarios que se han implementado siguen un cierto esquema:

1. Generar una población inicial de 30 cromosomas con valores de distribución uniforme $\mathcal{U}(0, 1)$ para los genes.

2. Seleccionar los cromosomas a los que se aplicará un cruce, ya sean 2 en el caso de AGE o igual a la población inicial en el caso de AGG.

3. Cruzar los individuos ya sea con cruce blx o cruce aritmético en base a una probabilidad de cruce.

4. Mutar ciertos cromosomas en base a una probabilidad de mutación.

5. Reemplazar la población antigua por la obtenida, ya sea conservando el de mejor función objetivo de la población antigua como en el caso de AGG, o sustituyendo en la población antigua los peores que tengan peor función objetivo que alguno de la nueva población generada, se sustituye por el que lo supera; este es el caso de AGE.

6. Volver al paso 2 hasta una condición de parada.

   La condición de parada consiste en no realizar más de 15000 evaluaciones de la función objetivo. La operación de selección, mutación y los tipos de cruces son los descritos en la sección 2.4.

Nota: en la implementación/código correspondiente a los pseudocódigos mostrados a continuación se incluyó una opción de cruce adicional para SBX2, ésta fue añadida a posteriori para el experimento extra.

3.1.1. Algoritmo genético generacional (AGG).

Para los algoritmos genéticos generacionales la probabilidad de cruce es 0.7 y la probabilidad de mutación 0.1 por cromosoma.

Para evitar la excesiva generación de aleatorios se estima previamente el número de cruces y mutaciones esperados.

- El número de cruces esperado es $\approx \frac{0.7 \cdot 30}{2} \approx 11$.
- El número de mutaciones esperado es $\frac{0.1}{\text{número genes}}$ por gen, se realizarán por tanto $\frac{0.1}{\text{número genes}} \cdot 30 \cdot \text{número genes} = 3$ mutaciones.

Para conservar el elitismo la mejor solución de la población anterior, s_1, reemplaza a la peor de la nueva población si s_1 es mejor que la mejor solución de la nueva población.

Algorithm 14: agg

Input: vector de elementos muestrales *training*, entero que indica el tipo de cruce a utilizar *cross_type*

Output: los genes del mejor cromosoma encontrado

begin

$pop\_size \leftarrow 30$

$exp\_num\_crosses,\ exp\_num\_muts \leftarrow round(0.7 * pop\_size/2),\ round(0.1 * pop\_size)$

$chr\_to\_cross \leftarrow 2 * exp\_num\_crosses$

$evals \leftarrow 0$

$cross\_type \leftarrow [blx\_cross,\ arith\_cross][cross\_type]$

`// 1. se genera población inicial`

for $i = 0$ ***to*** pop_size **do**

  for $j = 0$ ***to*** num_feat **do**

    $population[i].genes[j] \leftarrow random(0, 1)$

  end

  $obj\_to\_chromosome(population[i], training)$

  $evals \leftarrow evals + 1$

end

$best\_parent \leftarrow \arg\max\{c.obj\ :\ c \in population\}$ `// el de mayor f.obj`

while $evals < 15000$ **do**

  `// 2, 3. se seleccionan población que se va a cruzar y se cruza`

  for $i = 0$ ***to*** chr_to_cross ***step*** $= 2$ **do**

    $sel1 \leftarrow selection(population)$

    $sel2 \leftarrow selection(population)$

    $inter\_population[i],\ inter\_population[i + 1] \leftarrow cross(sel1, sel2)$

  end

  `// 2. se selecciona población restante que no se cruza`

  for $i = chr\_to\_cross$ ***to*** pop_size **do**

    $inter\_population[i] \leftarrow selection(population)$

  end

  `// 4. se realizan exp_num_muts mutaciones de genes de cromosomas de forma aleatoria`

  $mutate\_pop(inter\_population,\ exp\_num\_muts)$

  `// 5. se reemplaza la población`

  $population \leftarrow inter\_population$

  for c ***in*** $population$ **do**

    if $c.obj == -1$ **then**

      $obj\_to\_chromosome(c,\ training)$

      $evals \leftarrow evals + 1$

    end

  end

  `// 5. se mantiene el mejor`

  $best\_descendent \leftarrow \arg\max\{c.obj\ :\ c \in population\}$

  if $best\_parent > best\_descendent$ **then**

    $\arg\min\{c.obj\ :\ c \in population\} \leftarrow best\_parent$

  end

  else

    $best\_parent \leftarrow best\_descendent$

  end

end

return $(\arg\max\{c.obj\ :\ c \in population\}).genes$

end

3.1.2. Algoritmo genético estacional (AGE).

Para los algoritmos genéticos estacionales la probabilidad de cruce es 1, se cruzan los dos cromosomas que se seleccionan.

La probabilidad de mutación es 0.1 por cromosoma al igual que en el caso generacional, en este caso, al partir de solo 2 cromosomas tras la selección, el número esperado de mutaciones es menor a 1 por lo que sí generaremos el aleatorio en cada caso.

En este caso las dos soluciones generadas compiten con las anteriores sustituyendo a las peores en caso de mejorarlas.

Algorithm 15: age

Input: vector de elementos muestrales *training*, entero que indica el tipo de cruce a utilizar *cross_type*
Output: los genes del mejor cromosoma encontrado
begin
  $pop\_size \leftarrow 30$
  $evals \leftarrow 0$
  $cross\_type \leftarrow [blx\_cross,\ arith\_cross][cross\_type]$
  `// 1. se genera población inicial`
  for $i = 0$ ***to*** pop_size **do**
    for $j = 0$ ***to*** num_feat **do**
      $population[i].genes[j] \leftarrow random(0, 1)$
    end
    $obj\_to\_chromosome(population[i], training)$
    $evals \leftarrow evals + 1$
  end
  while $evals < 15000$ **do**
    `// 2, 3. se seleccionan población que se va a cruzar y se cruza`
    $sel1 \leftarrow selection(population)$
    $sel2 \leftarrow selection(population)$
    $inter\_population[0],\ inter\_population[1] \leftarrow cross(sel1, sel2)$
    `// 4. se realizan mutaciones con probabilidad de mutación de 0.1 por cromosoma`
    for $i = 0$ ***to*** *2* **do**
      if $random(0, 1) \leq 0.1$ **then**
        $mutation(inter\_population[i].genes,\ random(0,\ num\_feats - 1),\ \mathcal{N}(0, 0.3))$
        $inter\_population[i].obj \leftarrow -1$
      end
    end
    `// se evalúa f.obj en población intermedia`
    for c ***in*** $inter\_population$ **do**
      if $c.obj == -1$ **then**
        $obj\_to\_chromosome(c,\ training)$
        $evals \leftarrow evals + 1$
      end
    end
    `// 5. sustituir peores elementos de la población por los de pob. intermedia si son mejores`
    for c ***in*** $inter\_population$ **do**
      if $\arg\min\{c.obj\ :\ c \in population\} < c$ **then**
        $\arg\min\{c.obj\ :\ c \in population\} \leftarrow c$
      end
    end
  end
  return $(\arg\max\{c.obj\ :\ c \in population\}).genes$
end

3.2. Algoritmos meméticos.

Los algoritmos meméticos consisten en aplicar un algoritmo genético realizando BL de baja intensidad sobre el total o un sobjunto de la población cada 10 generaciones en nuestro caso.

La búsqueda local de baja intensidad implementada equivale a una adaptación de la desarrollada

en la práctica 1.b, donde ahora se indica el cromosoma con el que comenzar la búsqueda, el máximo número de vecinos a evaluar será $2 \cdot n$ con n el tamaño de cromosoma (su número de genes), y el número de evaluaciones comienza a partir del dado como argumento por referencia.

Algorithm 16: li_ls

Input: conjunto de elementos muestrales *training*, cromosoma c, evaluaciones de f.obj ya realizadas *obj_eval_count*

begin

$n \leftarrow c.genes.size()$ `// equivale a number_of_features`

`// inicialización de índices de genes`

for $i = 0$ ***to*** n **do**

$comp\_indexes.push\_back()$

end

$best\_obj \leftarrow c.obj$

$gen\_neighbours \leftarrow 0$

$mod\_pos \leftarrow 0$

while $gen\_neighbours < 2 * n$ ***and*** $obj\_eval\_count < 15000$ **do**

`// aleatorizamos componentes a mutar si se han recorrido todas o se mejoró f.obj`

if new_best_obj ***or*** $mod\_pos \,\%number\_of\_features == 0$ **then**

$new\_best\_obj \leftarrow false$

$shuffle(comp\_indexes)$

$mod\_pos \leftarrow 0$

end

`// Se toma componente a mutar y se muta`

$comp\_to\_mut \leftarrow comp\_indexes[mod\_pos \,\%n]$

$muted\_c \leftarrow c$

$mutation(muted\_c.genes,\ comp\_to\_mut,\ \mathcal{N}(0, 0.3))$

$gen\_neighbours \leftarrow gen\_neighbours + 1$

`// se asigna f. obj a cromosoma con los pesos mutados (con leave one out)`

$obj\_to\_chromosome(muted\_c,\ training)$

`// si se ha mejorado, actualizamos mejor objetivo, cromosoma y vecinos generados`

if $muted\_c.obj > best\_obj$ **then**

$c \leftarrow muted\_c$

$best\_obj \leftarrow obj$

$new\_best\_obj \leftarrow true$

end

$obj\_eval\_count \leftarrow obj\_eval\_count + 1$

$mod\_pos \leftarrow mod\_pos + 1$

end

end

Los algoritmos meméticos implementados se basan en AGG-BLX que fue el algoritmo generacional que mejor resultados proporcionó.

La forma de aplicar la búsqueda local de baja intensidad varía en las 3 versiones implementadas. Para no repetir pseudocódigo se indica solo la parte de pseudocódigo que difiere de AGG-BLX.

Nota: en la implementación/código correspondiente a los pseudocódigos mostrados a continuación se incluyó un argumento adicional para indicar cruce BLX o SBX2, éste fue añadido a posteriori para el experimento extra.

En AM-(10,1.0) se aplica la búsqueda local a todos los cromosomas de la población.

Algorithm 17: am_1010

```
Input: vector de elementos muestrales training
Output: los genes del mejor cromosoma encontrado
begin
    pop_size ← 10
    ...
    cross_type ← blx_cross
    ...
    while evals < 15000 do
        ...
        if generation %10 == 0 then
            for c in population do
                li_ls(training, c, evals)
            end
        end
    end
    ...
end
```

En AM-(10,0.1) se aplica la búsqueda local a cada cromosoma con probabilidad 0.1. Esto es, se aplica sobre 1 cromosoma.

Algorithm 18: am_1001

```
Input: vector de elementos muestrales training
Output: los genes del mejor cromosoma encontrado
begin
    pop_size ← 10
    ...
    cross_type ← blx_cross
    ...
    while evals < 15000 do
        ...
        if generation %10 == 0 then
            li_ls(training, population[random(0, pop_size − 1), evals)
        end
    end
    ...
end
```

En AM-(10,0.1 mej) el cromosoma sobre el que aplica la búsqueda local ya no es aleatorio, se aplica sobre el mejor de la población.

Algorithm 19: am_1001_mej

Input: vector de elementos muestrales *training*
Output: los genes del mejor cromosoma encontrado
begin
  $pop\_size \leftarrow 10$
  ...
  $cross\_type \leftarrow blx\_cross$
  ...
  while $evals < 15000$ **do**
    ...
    if $generation\,\%10 == 0$ **then**
      $li\_ls(training,\ \arg\max\{c.obj\ :\ c \in population\},\ evals)$
    end
  end
  ...
end

4. Procedimiento considerado en el desarrollo.

Para el desarrollo de la práctica no se ha utilizado ningún framework, se han implementado las funciones necesarias en C++. Todo el código se encuentra en `p2.cpp` que hace uso de `random.hpp`.

Para la ejecución basta indicar un argumento con el archivo a ejecutar, en este caso se utiliza la semilla por defecto 1234, que es la misma que se utilizó para los resultados de esta memoria.

También se incluye un fichero `p2_all_datasets.sh` que ejecuta, con la semilla por defecto, `p2` sobre los 3 datasets.

5. Experimentos y análisis de resultados.

Los 3 datasets con los que se ha trabajado son:

- **Ionosphere**: Conjunto de datos de radar que fueron recogidos por un sistema *Goose Bay*, Labrador. Consta de 352 instancias con 34 características y 2 clases.
- **Parkinsons**: Conjunto de datos orientado a distinguir entre la presencia y ausencia de la enfermedad de Parkinson. Consta de 195 ejemplos con 22 características y 2 clases.
- **Spectf-Heart**: Conjunto de datos de detección de enfermedades cardiacas a partir de imágenes médicas de tomografía computerizada del corazón de pacientes. Consta de 267 ejemplos con 44 características y 2 clases.

Los resultados obtenidos son en 1-NN, BL y Relief fueron:

1-NN

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	87.3239	0	43.662	0.373	95	0	47.5	0.181	85.7143	0	42.8571	0.468
P2	64.2857	0	32.1429	0.325	25	0	12.5	0.073	72.8571	0	36.4286	0.407
P3	64.2857	0	32.1429	0.325	25.641	0	12.8205	0.072	72.8571	0	36.4286	0.397
P4	64.2857	0	32.1429	0.325	23.6842	0	11.8421	0.07	72.8571	0	36.4286	0.397
P5	64.2857	0	32.1429	0.381	23.6842	0	11.8421	0.071	73.5294	0	36.7647	0.388
Media	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114

Relief

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	88.7324	2.94118	45.8368	1.753	95	9.09091	52.0455	0.364	90	0	45	2.28
P2	84.2857	5.88235	45.084	1.757	97.5	4.54545	51.0227	0.366	87.1429	0	43.5714	2.222
P3	90	2.94118	46.4706	1.76	94.8718	4.54545	49.7086	0.371	80	0	40	2.201
P4	88.5714	2.94118	45.7563	1.761	92.1053	4.54545	48.3254	0.369	84.2857	0	42.1429	2.172
P5	87.1429	2.94118	45.042	1.766	100	0	50	0.374	77.9412	0	38.9706	2.237
Media	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224

Búsqueda Local (BL)

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	78.8732	85.2941	82.0837	3311.36	87.5	86.3636	86.9318	390.064	85.7143	88.6364	87.1753	6719.91
P2	92.8571	85.2941	89.0756	4673.61	82.5	90.9091	86.7045	408.803	85.7143	86.3636	86.039	7458.06
P3	95.7143	82.3529	89.0336	1986.46	87.1795	86.3636	86.7716	339.172	88.5714	88.6364	88.6039	5512.85
P4	88.5714	88.2353	88.4034	3914.9	86.8421	50	68.4211	265.049	85.7143	77.2727	81.4935	3448.21
P5	90	85.2941	87.6471	2388.89	97.3684	86.3636	91.866	581.204	80.8824	88.6364	84.7594	3612.81
Media	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368

Los resultados de AGG con cruce BLX y AC:

AGG-BLX

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	85.9155	88.2353	87.0754	25015.6	82.5	86.3636	84.4318	4286.68	82.8571	84.0909	83.474	24133.3
P2	87.1429	85.2941	86.2185	19740.3	82.5	90.9091	86.7045	4682.84	82.8571	81.8182	82.3377	25265.7
P3	88.5714	82.3529	85.4622	26666.9	94.8718	86.3636	90.6177	5068.52	85.7143	81.8182	83.7662	24147.6
P4	80	85.2941	82.6471	23259.7	89.4737	90.9091	90.1914	4688.53	91.4286	81.8182	86.6234	24087.2
P5	88.5714	91.1765	89.8739	20168.3	89.4737	86.3636	87.9187	4415.09	89.7059	86.3636	88.0348	29249.7
Media	86.04024	86.47058	86.25542	22970.16	87.76384	88.1818	87.97282	4628.332	86.5126	83.18182	84.84722	25376.7

AGG-CA

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	85.9155	85.2941	85.6048	21950.1	87.5	90.9091	89.2045	4317.95	90	72.7273	81.3636	24144.4
P2	90	70.5882	80.2941	19750.6	97.5	81.8182	89.6591	4257.85	84.2857	75	79.6429	24035.7
P3	85.7143	88.2353	86.9748	19780.7	87.1795	86.3636	86.7716	5172.6	87.1429	79.5455	83.3442	24472.7
P4	84.2857	76.4706	80.3782	19828.3	89.4737	77.2727	83.3732	4358.13	90	79.5455	84.7727	24173.7
P5	91.4286	85.2941	88.3613	23206.7	97.3684	90.9091	94.1388	5163.74	80.8824	79.5455	80.2139	24515.2
Media	87.46882	81.17646	84.32264	20903.28	91.80432	85.45454	88.62944	4654.054	86.4622	77.27276	81.86746	24268.34

Los resultados de AGE con cruce BLX y CA:

AGE-BLX

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	83.0986	91.1765	87.1375	27456.7	87.5	90.9091	89.2045	4358.16	87.1429	88.6364	87.8896	24550.7
P2	87.1429	85.2941	86.2185	20435	85	90.9091	87.9545	5555.47	87.1429	88.6364	87.8896	23992.7
P3	87.1429	85.2941	86.2185	19653.2	87.1795	81.8182	84.4988	4289.27	85.7143	90.9091	88.3117	24350.1
P4	87.1429	88.2353	87.6891	24475.5	89.4737	90.9091	90.1914	4835.86	84.2857	81.8182	83.0519	26047.9
P5	88.5714	91.1765	89.8739	26234.3	92.1053	86.3636	89.2344	4379.06	86.7647	77.2727	82.0187	23996.1
Media	86.61974	88.2353	87.4275	23650.94	88.2517	88.18182	88.21672	4683.564	86.2101	85.45456	85.8323	24587.5

AGE-CA

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	83.0986	82.3529	82.7258	19609	90	90.9091	90.4545	5143.69	82.8571	86.3636	84.6104	28003
P2	84.2857	85.2941	84.7899	19991	92.5	86.3636	89.4318	5880.09	87.1429	79.5455	83.3442	24065.9
P3	88.5714	79.4118	83.9916	19748.8	87.1795	90.9091	89.0443	5397.15	82.8571	75	78.9286	24004.3
P4	82.8571	88.2353	85.5462	21253.6	92.1053	90.9091	91.5072	5158.17	87.1429	86.3636	86.7532	24044.5
P5	84.2857	82.3529	83.3193	24610.3	86.8421	86.3636	86.6029	4998.48	82.3529	75	78.6765	24085.8
Media	84.6197	83.5294	84.07456	21042.54	89.72538	89.0909	89.40814	5315.516	84.47058	80.45454	82.46258	24840.7

Los resultados de algoritmos meméticos AM-(10,1.0), AM-(10,0.1) y AM-(10,0.1mej) (basados en AGG-BLX, BLX mejor que AC en 2 de los 3 datasets):

AM-(10,1.0)

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	90.1408	88.2353	89.1881	20077.6	92.5	81.8182	87.1591	4355.65	84.2857	90.9091	87.5974	24793.6
P2	87.1429	91.1765	89.1597	25105.4	82.5	90.9091	86.7045	4850.57	84.2857	93.1818	88.7338	28944.3
P3	87.1429	88.2353	87.6891	24701.4	89.7436	86.3636	88.0536	5280.74	82.8571	93.1818	88.0195	29400.3
P4	94.2857	88.2353	91.2605	25443	81.5789	90.9091	86.244	4388.85	90	90.9091	90.4545	28191.6
P5	87.1429	85.2941	86.2185	20579.1	86.8421	90.9091	88.8756	5422.68	82.3529	88.6364	85.4947	24595.9
Media	89.17104	88.2353	88.70318	23181.3	86.63292	88.18182	87.40736	4859.698	84.75628	91.36364	88.05998	27185.14

AM-(10,0.1)

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	85.9155	91.1765	88.546	20039.7	82.5	90.9091	86.7045	5595.39	82.8571	86.3636	84.6104	31948.6
P2	85.7143	91.1765	88.4454	26416.5	95	86.3636	90.6818	4342.22	84.2857	90.9091	87.5974	27851.7
P3	85.7143	91.1765	88.4454	25659.9	89.7436	81.8182	85.7809	4284.02	82.8571	88.6364	85.7468	24348.3
P4	80	91.1765	85.5882	23920.3	86.8421	86.3636	86.6029	5441.42	82.8571	79.5455	81.2013	24233.2
P5	85.7143	88.2353	86.9748	25404	86.8421	90.9091	88.8756	6258.87	91.1765	86.3636	88.7701	23697.4
Media	84.61168	90.58826	87.59996	24288.08	88.18556	87.27272	87.72914	5184.384	84.8067	86.36364	85.5852	26415.84

AM-(10,0.1mej)

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	83.0986	91.1765	87.1375	26851.6	85	90.9091	87.9545	5115.58	87.1429	86.3636	86.7532	27727
P2	80	91.1765	85.5882	27375	87.5	86.3636	86.9318	5949.21	85.7143	77.2727	81.4935	28248.2
P3	88.5714	91.1765	89.8739	24859.1	84.6154	86.3636	85.4895	4361.47	84.2857	90.9091	87.5974	24500.2
P4	78.5714	88.2353	83.4034	24726	92.1053	90.9091	91.5072	5269.1	91.4286	90.9091	91.1688	26596.5
P5	92.8571	88.2353	90.5462	19819.8	86.8421	86.3636	86.6029	5571.82	79.4118	86.3636	82.8877	29583.1
Media	84.6197	90.00002	87.30984	24726.3	87.21256	88.1818	87.69718	5253.436	85.59666	86.36362	85.98012	27331

Se muestra la tabla resumen:

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
1-NN	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114
Relief	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224
BL	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368
AGG-BLX	86.04024	86.47058	86.25542	22970.16	87.76384	88.1818	87.97282	4628.332	86.5126	83.18182	84.84722	25376.7
AGG-CA	87.46882	81.17646	84.32264	20903.28	91.80432	85.45454	88.62944	4654.054	86.4622	77.27276	81.86746	24268.34
AGE-BLX	86.61974	88.2353	87.4275	23650.94	88.2517	88.18182	88.21672	4683.564	86.2101	85.45456	85.8323	24587.5
AGE-CA	84.6197	83.5294	84.07456	21042.54	89.72538	89.0909	**89.40814**	5315.516	84.47058	80.45454	82.46258	24840.7
AM-(10,1.0)	89.17104	88.2353	**88.70318**	23181.3	86.63292	88.18182	87.40736	4859.698	84.75628	91.36364	**88.05998**	27185.14
AM-(10,0.1)	84.61168	90.58826	87.59996	24288.08	88.18556	87.27272	87.72914	5184.384	84.8067	86.36364	85.5852	26415.84
AM-(10,0.1mej)	84.6197	90.00002	87.30984	24726.3	87.21256	88.1818	87.69718	5253.436	85.59666	86.36362	85.98012	27331

En cuanto a las tasas, en primer lugar vemos gráficamente la tasa de clasificación:

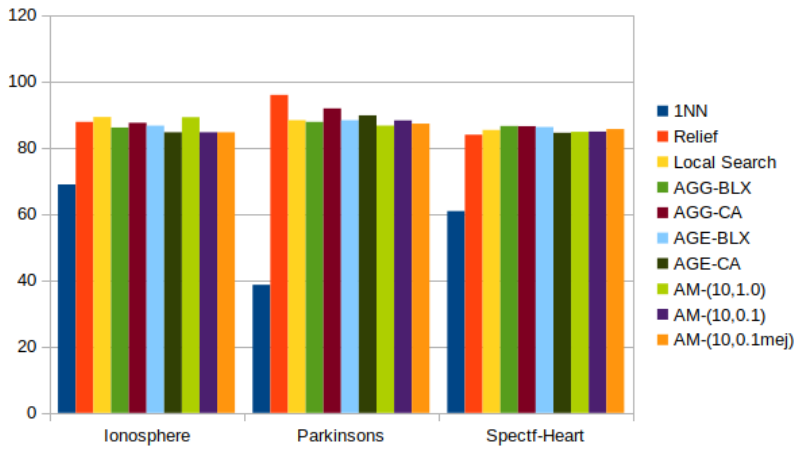

Figura 1: Tasa de clasificación.

Se observa que, a parte de 1NN, el resto de algoritmos proporcionan tasas de clasificación similares entre sí, dependiendo del dataset ganan unos u otros, no hay un patrón claro. Aunque Relief da buen resultado en un par de datasets tampoco es de mucho interés pues sabemos que no proporciona apenas tasa de reducción. Lo que sí podemos destacar es que tanto los algoritmos genéticos como meméticos han dado buenas tasas de clasificación, de hecho se han mejorado en Parkinsons y Spectf-Heart respecto a BL. En Ionosphere sigue ganando BL pero con una diferencia mínima a AM-(10,1.0). No se observa de forma consistente un mejor resultado de los algoritmos meméticos respecto a los genéticos, ni de los algoritmos generacionales respecto a los estacionarios, ni entre el cruce BLX y CA.

Vemos ahora las tasas de reducción:

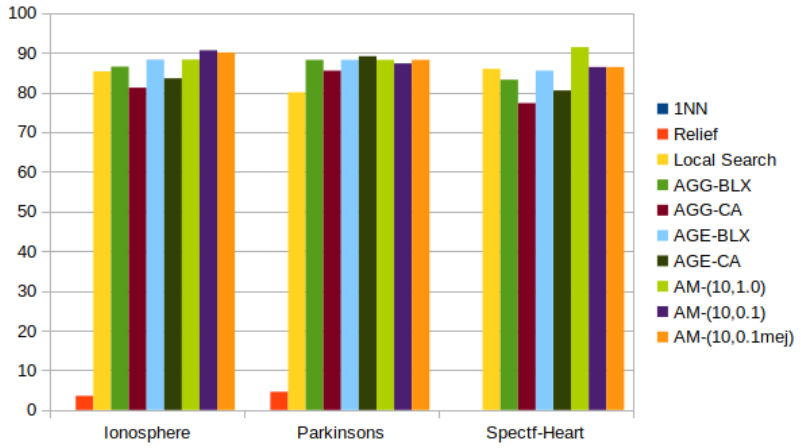

Figura 2: Tasa de reducción.

Sabemos que BL era el que mejor resultado proporcionaba hasta el momento. Vemos que se han mejorado los resultados en los 3 datasets. A excepción de Parkinsons, que el mejor resultado lo proporciona AGE-CA, en el resto de datasets el mejor resultado viene dado por un algoritmo memético. Además vemos que se consigue mejor tasa de reducción con cruce BLX que con CA tanto en Ionosphere como Spect-Heart.

Por último el valor agregado:

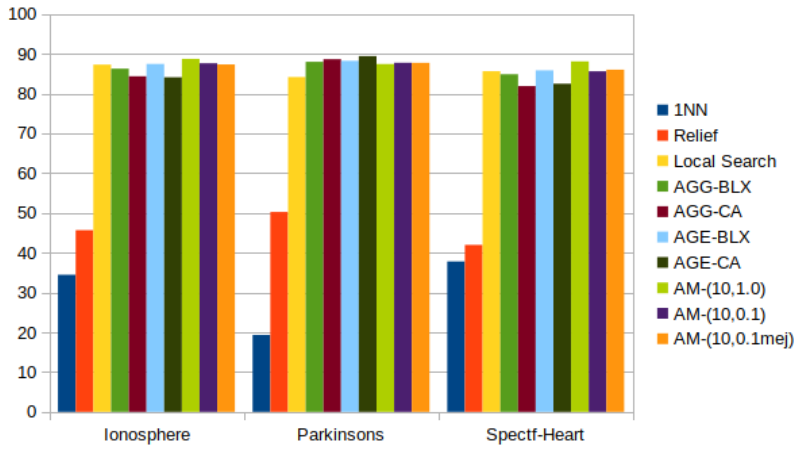

Figura 3: Valor de función objetivo o agregado.

De nuevo se han superado los resultados de BL en los 3 datasets. A excepción de Parkinsons, que es el datasets más pequeño, se observan ligeramente mejores resultados de los algoritmos meméticos

respecto a los genéticos. Dedicar evaluaciones de la función objetivo a la búsqueda local tras cada generación ha resultado efectivo. Tanto en Ionosphere como Spect-Heart se tiene el mejor resultado con AM-(10,1.0), siendo mejor aplicar BL sobre toda la población; aumentando la exploración, y siendo todos los chromosomas mejores en cada generación de cara a la selección, cruce y mutación. Aún así cabe recalcar que no se observan de forma consistente ganadores y perdedores entre los meméticos, no hay mucha diferencia entre los valores de función objetivo obtenidos y las diferencias pueden deberse a la casuística del dataset o la propia aleatoriedad introducida.

Se observa en Ionosphere y Spectf-Heart mejor resultado de cruce BLX respecto CA, aunque esto se invierte en Parkinsons, no se observa una muy clara superioridad de BLX respecto CA. Realmente, tal y como se ha implementado CA, tanto BLX como CA son cruces con alta capacidad de exploración.

Tanto los algoritmos genéticos generacionales como estacionarios proporcionan resultados similares. La alta presión selectiva del estacionario no ha supuesto una desventaja, de hecho AGE-BLX proporciona el mejor resultado de algoritmos generacionales tanto en Ionosphere como Spectf-Heart.

En cuanto a los tiempos de ejecución:

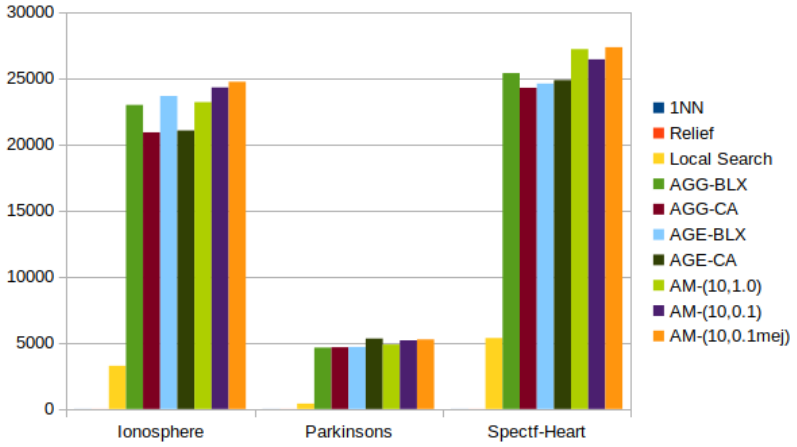

Figura 4: Tiempos de ejecución.

Anteriormente BL era el algoritmo con mayor tiempo de ejecución. Ahora observamos una gran diferencia respecto BL: los algoritmos generacionales y meméticos suponen más de 4 veces el tiempo de BL, se ha requerido mucho tiempo para mejorar, y no por mucho, al resultado de BL. Los algoritmos genéticos tienen tiempos de ejecución similares entre sí. Sí se observa mayor tiempo de ejecución de los algoritmos meméticos respecto a genéticos ya que, aunque realizan las mismas evaluaciones de función objetivo, tienen como "añadido" las operaciones de la búsqueda local.

5.1. Experimento extra:

Se ha decidido experimentar un nuevo operador de cruce, SBX (Simulated Binary Crossover), propuesto en 1995 por Deb y Agrawal. Este operador de cruce simula un cruce binario de un solo punto en variables reales. Para ello, dados dos padres $p_1(i)$ y $p_2(i)$, este cruce genera dos decendientes $d_1(i)$ y $d_2(i)$ dados por la siguiente expresión:

$$c_1(i) = 0.5[(1+\beta)p_1(i) + (1-\beta)p_2(i)]$$
$$c_2(i) = 0.5[(1-\beta)p_1(i) + (1+\beta)p_2(i)]$$

donde β es un factor de dispersión dado por

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta+1}} & u < 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta+1}} & u \geq 0.5 \end{cases}$$

donde $u \in \mathcal{U}(0,1)$, y $\eta \in \mathbb{R}^+$ es un parámetro dado.

Para la implementación se ha decidido tomar $\eta = 2$.
Referencia: https://content.wolfram.com/uploads/sites/13/2018/02/09-2-2.pdf

Mostramos el pseudocódigo:

Algorithm 20: sbx2_cross

Input: Dos cromosomas $p1$ y $p2$
Output: Los dos cromosomas descendientes $d1$ y $d2$
begin
  $gen\_size \leftarrow p1.genes.size()$
  $\eta \leftarrow 2$
  $u \leftarrow random(0,1)$
  if $u < 0.5$ **then**
    $b \leftarrow 2u^{\frac{1}{\eta+1}}$
  end
  else
    $b \leftarrow \frac{1}{2(1-u)}^{\frac{1}{\eta+1}}$
  end
  for $i = 0$ ***to*** gen_size **do**
    $d1.genes[i] \leftarrow 0.5 * ((1+b) * p1.genes[i] + (1-b) * p2.genes[i])$
    $d2.genes[i] \leftarrow 0.5 * ((1-b) * p1.genes[i] + (1+b) * p2.genes[i])$
  end
  `// para evitar evaluaciones de f.obj redundantes antes de la mutación`
  $d1.obj \leftarrow -1$
  $d2.obj \leftarrow -1$
  return $d1$, $d2$
end

Se muestran los resultados obtenidos para algoritmos genéticos y meméticos con cruce SBX2:

AGG-SBX2

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	83.0986	82.3529	82.7258	21359.2	95	77.2727	86.1364	4173.41	90	86.3636	88.1818	24516.1
P2	81.4286	85.2941	83.3613	19516.3	100	72.7273	86.3636	5158.73	87.1429	77.2727	82.2078	23955.1
P3	92.8571	88.2353	90.5462	19828	82.0513	81.8182	81.9347	4301.68	85.7143	81.8182	83.7662	24019.5
P4	87.1429	82.3529	84.7479	20047.1	94.7368	81.8182	88.2775	5413.85	88.5714	77.2727	82.9221	24072.4
P5	91.4286	91.1765	91.3025	23429	86.8421	90.9091	88.8756	5938.4	89.7059	75	82.3529	25851.1
Media	87.19116	85.88234	86.53674	20835.92	91.72604	80.9091	86.31756	4997.214	88.2269	79.54544	83.88616	24482.84

AGE-SBX2

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	90.1408	85.2941	87.7175	19900.6	92.5	72.7273	82.6136	5232.04	84.2857	70.4545	77.3701	24378
P2	85.7143	79.4118	82.563	21885	70	86.3636	78.1818	5331.24	90	70.4545	80.2273	25355.3
P3	92.8571	79.4118	86.1345	19431.4	89.7436	72.7273	81.2354	5167.61	87.1429	70.4545	78.7987	24750.8
P4	87.1429	85.2941	86.2185	19550.2	92.1053	77.2727	84.689	4325.87	92.8571	68.1818	80.5195	23929.5
P5	88.5714	82.3529	85.4622	19500.6	89.4737	81.8182	85.6459	5314.16	79.4118	75	77.2059	24308.7
Media	88.8853	82.35294	85.61914	20053.56	86.76452	78.18182	82.47314	5074.184	86.7395	70.90906	78.8243	24544.46

AM-(10,1.0)-SBX2

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	92.9577	91.1765	92.0671	22639.4	82.5	86.3636	84.4318	4229.03	88.5714	88.6364	88.6039	26515.3
P2	84.2857	91.1765	87.7311	20164.6	95	86.3636	90.6818	4280.44	81.4286	88.6364	85.0325	26422.7
P3	88.5714	91.1765	89.8739	23002.2	94.8718	90.9091	92.8904	5098	82.8571	88.6364	85.7468	28527.5
P4	94.2857	88.2353	91.2605	19438.2	81.5789	90.9091	86.244	4389.35	88.5714	86.3636	87.4675	24712.1
P5	85.7143	88.2353	86.9748	21609	94.7368	90.9091	92.823	5108.6	80.8824	79.5455	80.2139	24506.5
Media	89.16296	90.00002	89.58148	21370.68	89.7375	89.0909	89.4142	4621.084	84.46218	86.36366	85.41292	26136.82

AM-(10,0.1)-SBX2

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	90.1408	91.1765	90.6587	19940	72.5	90.9091	81.7045	4274.03	81.4286	86.3636	83.8961	24603.2
P2	87.1429	91.1765	89.1597	21074.3	95	90.9091	92.9545	5859.17	80	93.1818	86.5909	25159.7
P3	87.1429	91.1765	89.1597	21811.7	94.8718	90.9091	92.8904	5487.99	88.5714	88.6364	88.6039	24903.2
P4	87.1429	91.1765	89.1597	21474.9	92.1053	90.9091	91.5072	5938.07	81.4286	86.3636	83.8961	24152.5
P5	82.8571	91.1765	87.0168	23074	92.1053	90.9091	91.5072	5880.71	77.9412	81.8182	79.8797	24326.7
Media	86.88532	91.1765	89.03092	21474.98	89.31648	90.9091	90.11276	5487.994	81.87396	87.27272	84.57334	24629.06

AM-(10,0.1mej)-SBX2

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	87.3239	88.2353	87.7796	19999.1	87.5	81.8182	84.6591	4282	80	93.1818	86.5909	27844
P2	82.8571	91.1765	87.0168	20578.1	87.5	86.3636	86.9318	4264.69	78.5714	84.0909	81.3312	27425.5
P3	82.8571	88.2353	85.5462	22925.2	84.6154	86.3636	85.4895	4343.89	78.5714	79.5455	79.0584	27507.6
P4	92.8571	91.1765	92.0168	22993.9	92.1053	90.9091	91.5072	5917.92	87.1429	90.9091	89.026	27381.2
P5	81.4286	88.2353	84.8319	19575.8	78.9474	86.3636	82.6555	4868.58	86.7647	86.3636	86.5642	26928
Media	85.46476	89.41178	87.43826	21214.42	86.13362	86.36362	86.24862	4735.416	82.21008	86.81818	84.51414	27417.26

Se muestra la tabla resumen:

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
1-NN	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114
Relief	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224
BL	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368
AGG-BLX	86.04024	86.47058	86.25542	22970.16	87.76384	88.1818	87.97282	4628.332	86.5126	83.18182	84.84722	25376.7
AGG-CA	87.46882	81.17646	84.32264	20903.28	91.80432	85.45454	88.62944	4654.054	86.4622	77.27276	81.86746	24268.34
AGE-BLX	86.61974	88.2353	87.4275	23650.94	88.2517	88.18182	88.21672	4683.564	86.2101	85.45456	85.8323	24587.5
AGE-CA	84.6197	83.5294	84.07456	21042.54	89.72538	89.0909	**89.40814**	5315.516	84.47058	80.45454	82.46258	24840.7
AM-(10,1.0)	89.17104	88.2353	**88.70318**	23181.3	86.63292	88.18182	87.40736	4859.698	84.75628	91.36364	**88.05998**	27185.14
AM-(10,0.1)	84.61168	90.58826	87.59996	24288.08	88.18556	87.27272	87.72914	5184.384	84.8067	86.36364	85.5852	26415.84
AM-(10,0.1mej)	84.6197	90.00002	87.30984	24726.3	87.21256	88.1818	87.69718	5253.436	85.59666	86.36362	85.98012	27331
AGG-SBX2	87.19116	85.88234	86.53674	20835.92	91.72604	80.9091	86.31756	4997.214	88.2269	79.54544	83.88616	24482.84
AGE-SBX2	88.8853	82.35294	85.61914	20053.56	86.76452	78.18182	82.47314	5074.184	86.7395	70.90906	78.8243	24544.46
AM-(10,1.0)-SBX2	89.16296	90.00002	89.58148	21370.68	89.7375	89.0909	89.4142	4621.084	84.46218	86.36366	85.41292	26136.82
AM-(10,0.1)-SBX2	86.88532	91.1765	89.03092	21474.98	89.31648	90.9091	90.11276	5487.994	81.87396	87.27272	84.57334	24629.06
AM-(10,0.1mej)-SBX2	85.46476	89.41178	87.43826	21214.42	86.13362	86.36362	86.24862	4735.416	82.21008	86.81818	84.51414	27417.26

Se ha marcado en azul el mejor resultado con cruce SBX2.

Los resultados han sido buenos, aunque hay cierta influencia por el dataset; en Ionosphere y Parkinsons se ha obtenido nuevo máximo en función objetivo, en Spectf-Heart sin embargo no consigue mejorar los resultados.

Aunque haya marcado máximos no parece claro que sea mejor que el cruce BLX o AC, en los algoritmos genéticos no muestra ser superior a BLX o AC.

Desde luego sí que funciona y parece otro tipo de cruce que podría considerarse.