UNIVERSIDAD DE GRANADA
E.T.S.I. INFORMÁTICA Y TELECOMUNICACIÓN

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Metaheurísticas

http://sci2s.ugr.es/graduateCourses/Metaheuristicas
https://decsai.ugr.es

Guión de Prácticas

Práctica 1.b:
Técnicas de Búsqueda Local y Algoritmos Greedy para el Problema del Aprendizaje de Pesos en Características

Curso 2021-22

Tercer Curso del Grado en Ingeniería Informática

Práctica 1.b

Técnicas de Búsqueda Local y Algoritmos Greedy para el Problema del Aprendizaje de Pesos en Características

1. Objetivos

El objetivo de esta práctica es estudiar el funcionamiento de las *Técnicas de Búsqueda Local y de los Algoritmos Greedy* en la resolución del problema del aprendizaje de pesos en características (APC) descrito en las transparencias del Seminario 2. Para ello, se requerirá que el estudiante adapte los siguientes algoritmos a dicho problema:

- Algoritmo *greedy* RELIEF.
- Algoritmo de Búsqueda Local (BL).

El estudiante deberá comparar los resultados obtenidos por el clasificador 1-NN original que considera todas las características igualmente ponderadas (sin pesos) con el clasificador 1-NN obtenido empleando los pesos aprendidos por el método *greedy* RELIEF e, igualmente, con los pesos aprendidos por el algoritmo de BL, en una serie de casos del problema.

La práctica se evalúa sobre un total de **2 puntos**, distribuidos de la siguiente forma:

- BL (**1.25 puntos**).
- *Greedy* RELIEF (**0.75 puntos**).

La fecha límite de entrega será el **Domingo 10 de Abril de 2019** antes de las 23:59 horas. La entrega de la práctica se realizará por internet a través del espacio de la asignatura en PRADO.

2. Trabajo a Realizar

El estudiante podrá desarrollar los algoritmos de la práctica siguiendo la modalidad que desee: trabajando con cualquiera de los *frameworks* de metaheurísticas

estudiados en el Seminario 1, implementándolos a partir del código C proporcionado en la web de la asignatura o considerando cualquier código disponible en Internet.

Los métodos desarrollados serán ejecutados sobre una serie de casos del problema. Se realizará un estudio comparativo de los resultados obtenidos y se analizará el comportamiento de cada algoritmo en base a dichos resultados. **Este análisis influirá decisivamente en la calificación final de la práctica**.

En las secciones siguientes se describen los aspectos relacionados con cada algoritmo a desarrollar y las tablas de resultados a obtener.

3. Problema y Casos Considerados

3.1. Introducción al Problema del Aprendizaje de Pesos en Características

El problema del APC consiste en optimizar el rendimiento de un clasificador basado en vecinos más cercanos a partir de la inclusión de pesos asociados a las características del problema que modifican su valor en el momento de calcular las distancias entre ejemplos. En nuestro caso, el clasificador considerado será el 1-NN (*k*-NN, *k* vecinos más cercanos, con *k*=1 vecino). La variante del problema del APC que afrontaremos busca optimizar tanto la precisión como la complejidad del clasificador. Así, se puede formular como:

$$tasa - clas = 100 \cdot \frac{nº\ instancias\ bien\ clasificadas\ en\ T}{nº\ instancias\ en\ T}$$

Maximizar $F(W) = \alpha \cdot tasa\_clas(W) + (1-\alpha) \cdot tasa\_red(W)$

sujeto a que:

- $w_i = [0, 1],\ 1 \leq i \leq n$

donde:

- $W = (w_1, \ldots, w_n)$ es una solución al problema que consiste en un vector de números reales $w_1 \in [0,1]$ de tamaño n que define el peso que pondera o filtra a cada una de las características f_i.
- 1-NN es el clasificador *k*-NN con *k*=1 vecinos generado a partir del conjunto de datos inicial utilizando los pesos en W que se asocian a las n características.
- T es el conjunto de datos sobre el que se evalúa el clasificador, ya sea el conjunto de entrenamiento (usando la técnica de validación *leave-one-out*) o el de prueba.

- $\alpha \in [0,1]$ pondera la importancia entre el acierto y la reducción de la solución encontrada.

3.2. Conjuntos de Datos Considerados

El estudiante trabajará con los **3 conjuntos de datos** siguientes (obtenidos de la web http://www.ics.uci.edu/~mlearn/MLRepository.html, procesados en algún caso y disponibles en el espacio de PRADO y en la web de la asignatura):

1. **Ionosphere**: Conjunto de datos de radar que fueron recogidos por un sistema en *Goose Bay*, Labrador. 352 ejemplos con 34 características que deben ser clasificados en 2 clases.
2. **Parkinsons**: Conjunto de datos orientado a distinguir entre la presencia y la ausencia de la enfermedad de Parkinson en una serie de pacientes a partir de medidas biomédicas de la voz. 195 ejemplos con 22 características que deben ser clasificados en 2 clases.
3. **Spectf-heart:** Conjunto de datos de detección de enfermedades cardiacas a partir de imágenes médicas de tomografía computerizada (SPECT) del corazón de pacientes. 267 ejemplos con 44 características que deben ser clasificados en 2 clases.

El formato de los ficheros es el ARFF de WEKA. Puede consultarse en las transparencias del Seminario 2 y en http://www.cs.waikato.ac.nz/ml/weka/.

3.3. Determinación de la Calidad de un Algoritmo

El modo habitual de determinar la calidad de un algoritmo de resolución aproximada de problemas de optimización es ejecutarlo sobre un conjunto determinado de instancias y comparar los resultados obtenidos con los mejores valores conocidos para dichas instancias (en caso de existir).

Además, los algoritmos pueden tener diversos parámetros o pueden emplear diversas estrategias. Para determinar qué valor es el más adecuado para un parámetro o saber la estrategia más efectiva los algoritmos también se comparan entre sí.

La comparación de los algoritmos se lleva a cabo fundamentalmente usando dos criterios, la *calidad* de las soluciones obtenidas y el *tiempo de ejecución* empleado para conseguirlas. Además, es posible que los algoritmos no se comporten de la misma forma si se ejecutan sobre un conjunto de instancias u otro.

Por otro lado, a diferencia de los algoritmos determinísticos, los algoritmos probabilísticos se caracterizan por la toma de decisiones aleatorias a lo largo de su ejecución. Este hecho implica que un mismo algoritmo probabilístico aplicado al mismo caso de un problema pueda comportarse de forma diferente y por tanto proporcionar resultados distintos en cada ejecución.

Cuando se analiza el comportamiento de una metaheurística probabilística en un caso de un problema, se desearía que el resultado obtenido no estuviera sesgado por una secuencia aleatoria concreta que pueda influir positiva o negativamente en las

decisiones tomadas durante su ejecución. Por tanto, resulta necesario efectuar varias ejecuciones con distintas secuencias probabilísticas y calcular el resultado medio y la desviación típica de todas las ejecuciones para representar con mayor fidelidad su comportamiento.

Dada la influencia de la aleatoriedad en el proceso, es recomendable disponer de un generador de secuencia pseudoaleatoria de buena calidad con el que, dado un valor semilla de inicialización, se obtengan números en una secuencia lo suficientemente grande (es decir, que no se repitan los números en un margen razonable) como para considerarse aleatoria. En el espacio de PRADO y la web de la asignatura se puede encontrar una implementación en lenguaje C de un generador aleatorio de buena calidad (*random.h*).

Como norma general, el proceso a seguir consiste en realizar un número de ejecuciones diferentes de cada algoritmo probabilístico considerado para cada caso del problema. Es necesario asegurarse de que se realizan diferentes secuencias aleatorias en dichas ejecuciones. Así, el valor de la semilla que determina la inicialización de cada secuencia deberá ser distinto en cada ejecución y estas semillas deben mantenerse en los distintos algoritmos (es decir, la semilla para la primera ejecución de todos los algoritmos debe ser la misma, la de la segunda también debe ser la misma y distinta de la anterior, etc.). Para mostrar los resultados obtenidos con cada algoritmo en el que se hayan realizado varias ejecuciones, se deben construir tablas que recojan los valores correspondientes a estadísticos como el **mejor** y **peor** resultado para cada caso del problema, así como la **media** y la **desviación típica** de todas las ejecuciones. También se pueden emplear descripciones más representativas como los boxplots, que proporcionan información de todas las ejecuciones realizadas mostrando mínimo, máximo, mediana y primer y tercer cuartil de forma gráfica, Finalmente, se construirán unas tablas globales con los resultados agregados que mostrarán la calidad del algoritmo en la resolución del problema desde un punto de vista general.

Alternativamente, **en el caso de que se considere un número alto de casos del problema, se puede confiar en una única ejecución del algoritmo sobre cada caso del problema. Lo mismo ocurre en aquellos problemas de aprendizaje automático en los que se consideren varias ejecuciones del algoritmo sobre distintos subconjuntos del conjunto de datos original dentro del proceso de validación considerado. Esta condición se cumple en las prácticas que realizaremos con el APC**. Aun así, **será necesario inicializar la semilla del generador aleatorio para poder repetir el experimento** y obtener los mismos resultados si fuera necesario (en caso contrario, los resultados podrían variar en cada ejecución del mismo algoritmo sobre el mismo caso del problema).

En nuestro problema, consideraremos el método de validación ***5-fold cross validation*** (5fcv). Para ello, el conjunto de datos se divide en 5 particiones disjuntas al 20% *manteniendo la distribución de clases*. Aprenderemos un clasificador utilizando hasta un total del 80% de los datos disponibles (4 particiones de las 5) y validaremos con el 20% restante (la partición restante), teniendo por tanto 5 particiones posibles al 80-20%. Así, obtendremos hasta un total de cinco valores de porcentaje de clasificación en el conjunto de prueba, uno para cada partición que ha sido parte del conjunto de validación.

La medida de validación será la tasa de acierto del clasificador 1-NN sobre el conjunto de prueba. El resultado final será la media de los 5 valores obtenidos, uno para cada ejecución del algoritmo.

Para facilitar la comparación de algoritmos en las prácticas del APC se considerarán cuatro estadísticos distintos denominados *Tasa_clas*, *Tasa_red*, *Agregado* y *Tiempo*:

- *Tasa_clas* se calcula como la media de los porcentajes de acierto (las tasas de clasificación) obtenidos por cada método en cada partición del conjunto de datos (es el valor final resultante del 5fcv).

- *Tasa_red* corresponde al porcentaje de reducción obtenido en la selección del subconjunto de características respecto al total. El cálculo se realiza con la expresión mostrada en la sección anterior. El valor global se obtiene como la media de los porcentajes de reducción obtenidos por cada método en cada partición del conjunto de datos.

- *Agregado* corresponde al valor de la función objetivo que está optimizando el algoritmo; es decir, al valor a maximizar proveniente de la fórmula $F(W) = \alpha \cdot tasa\_clas(W) + (1-\alpha) \cdot tasa\_red(W)$. Igualmente, se indicará el valor medio de las 5 ejecuciones del 5fcv.

- *Tiempo* se calcula como la media del tiempo de ejecución empleado por el algoritmo para resolver cada caso del problema (cada conjunto de datos). Es decir, en nuestro caso es la media del tiempo empleado por las 5 ejecuciones.

Cuanto mayor es el valor de *Tasa_clas* para un algoritmo, mejor calidad tiene dicho algoritmo, porque obtiene clasificadores más precisos en media. Del mismo modo, cuanto mayor es el valor de *Tasa_red*, mejor calidad tiene también el algoritmo, porque obtiene clasificadores más simples en media. La elección entre un clasificador de mejor rendimiento o de mayor eficiencia depende de los requisitos de la aplicación concreta. Finalmente, si dos métodos obtienen soluciones con la misma calidad (tienen valores de *Tasa_clas* y *Tasa_red* similares), uno será mejor que el otro si emplea menos tiempo en media. En la web de la asignatura hay también disponible un código en C (*timer*) para un cálculo adecuado del tiempo de ejecución de los algoritmos metaheurísticos.

La hoja Excel *Tablas_APC_2020-21.xls*, disponible en la web de la asignatura, permite recopilar los estadísticos comentados para generar las tablas de resultados de la práctica.

4. Componentes del Algoritmo de Búsqueda Local

El algoritmo de BL tiene las siguientes componentes, varias de las cuales serán comunes a los algoritmos metaheurísticos de las siguientes prácticas:

- *Esquema de representación*: Se seguirá la representación real basada en un vector *W* de tamaño *n* con valores en *[0, 1]* que indican el peso asociado a cada característica y la capacidad para eliminarla si su peso es menor que 0.2, explicado en las transparencias del seminario.

- *Función objetivo*: Será la combinación con pesos de las medidas de precisión (tasa de acierto sobre el conjunto de entrenamiento) y la complejidad (la tasa de reducción de características con respecto al conjunto original) del clasificador 1-NN diseñado empleando el vector *W*. Para calcular la tasa de acierto será necesario emplear la técnica de validación *leave-one-out* explicada en las transparencias del Seminario 2. El valor de α considerado será α=0.5, dándole la misma importancia a ambos criterios. El objetivo será maximizar esta función.

- *Generación de la solución inicial*: La solución inicial se generará de forma aleatoria utilizando una distribución uniforme en *[0, 1]* en todos los casos.

- *Esquema de generación de vecinos*: Se empleará el movimiento de cambio por mutación normal *Mov(W,σ)* que altera el vector *W* sumándole otro vector *Z* generado a partir de una distribución normal de media 0 y varianza σ^2.

- *Criterio de aceptación*: Se considera una mejora cuando se aumenta el valor global de la función objetivo.

- *Exploración del vecindario*: En cada paso de la exploración se mutará una componente $i \in \{1,\ldots,n\}$ **distinta** sin repetición hasta que haya mejora o se hayan modificado todas las posiciones una vez sin conseguir una mejora. En ese momento, se comienza de nuevo la exploración sobre la nueva solución aceptada (si ha habido mejora) o sobre solución actual (en caso contrario).

Algoritmo

Como algoritmo de BL para el APC consideraremos el esquema del primer mejor, tal y como está descrito en las transparencias del Seminario 2.

Valores de los parámetros y ejecuciones

Se detendrá la ejecución del algoritmo de BL cuando no se encuentre mejora tras generar un máximo de 20·*n* vecinos (*n* es el número de características) o cuando se hayan realizado 15000 evaluaciones de la función objetivo, es decir, en cuanto se cumpla alguna de las dos condiciones. Además, se considerará un σ=0,3.

5. Algoritmo de Comparación: RELIEF

El algoritmo escogido para ser comparado con las metaheurísticas implementadas en esta práctica (la BL) y en las siguientes es el *greedy* RELIEF, que también deberá ser implementado por el estudiante. El objetivo de este algoritmo será generar un vector de pesos a partir de las distancias de cada ejemplo a su e*nemigo* más cercano y a su *amigo* más cercano. El *enemigo* más cercano a un ejemplo es aquel

ejemplo de diferente clase más cercano a él. El *amigo* más cercano a un ejemplo es otro ejemplo de su misma clase más cercano a él.

La intuición del algoritmo es doble: primero se basa en incrementar el peso a aquellas características que mejor separan a ejemplos que son enemigos entre sí; también reduce el valor del peso en aquellas características que separan ejemplos que son amigos entre sí. Este/a incremento/reducción es directamente proporcional a la distancia entre los ejemplos en cada característica. A continuación, se describe el algoritmo en los siguientes pasos:

- Se inicializa el vector de pesos *W* a cero.
- Para cada ejemplo del conjunto de entrenamiento, se identifica el enemigo y amigo más cercano. *W* se actualiza sumándole la distancia existente entre el ejemplo y su enemigo más cercano (considerando todas las características, suma componente a componente). Después, *W* se actualiza restándole la distancia existente entre el ejemplo y su amigo más cercano.
- Una vez hecho lo anterior para todos los ejemplos, *W* puede tener componentes que no estén en *[0, 1]*. Los valores negativos de *W* se truncarán a cero. El resto de valores se normalizarán dividiéndolo por el valor máximo encontrado en *W*.
- Nótese que el orden de actuación de los ejemplos no altera el resultado final.

Como la BL, el algoritmo RELIEF deberá ser ejecutado 5 veces para cada conjunto de datos de acuerdo a lo explicado en la sección anterior.

6. Tablas de Resultados a Obtener

Se diseñará una tabla para cada algoritmo (1-NN, RELIEF, BL) donde se recojan los resultados de la ejecución de dicho algoritmo en los conjuntos de datos considerados. Tendrá la misma estructura que la Tabla 5.1. En el caso de 1-NN, la tasa de reducción será siempre 0.

Tabla 5.1: Resultados obtenidos por el algoritmo X en el problema del APC

	Ionosphere				**Parkinsons**				**Spectf-heart**			
	%_clas	***%_red***	***Agr.***	**T**	***%_clas***	***%_red***	***Agr.***	**T**	***%_clas***	***%_red***	***Agr.***	**T**
Partición 1	x	x	x	x	x	x	x	x	x	x	x	x
Partición 2	x	x	x	x	x	x	x	x	x	x	x	x
Partición 3	x	x	x	x	x	x	x	x	x	x	x	x
Partición 4	x	x	x	x	x	x	x	x	x	x	x	x
Partición 5	x	x	x	x	x	x	x	x	x	x	x	x
Media	x	x	x	x	x	x	x	x	x	x	x	x

Finalmente, se construirá una tabla de resultados global que recoja los resultados medios de calidad y tiempo para todos los algoritmos considerados, tal como se muestra en la Tabla 5.2. Para rellenar esta tabla se hará uso de los resultados medios mostrados en las tablas parciales. Aunque en la tabla que sirve de ejemplo se han incluido todos los algoritmos considerados en esta práctica, naturalmente sólo se incluirán los que se hayan desarrollado.

Tabla 5.2: Resultados globales en el problema del APC

	Ionosphere				**Parkinsons**				**Spectf-heart**			
	%_clas	***%_red***	***Agr.***	**T**	***%_clas***	***%_red***	***Agr.***	**T**	***%_clas***	***%_red***	***Agr.***	**T**
1-NN	x	0	x	x	x	0	x	x	x	0	x	x
RELIEF	x	x	x	x	x	x	x	x	x	x	x	x
BL	x	x	x	x	x	x	x	x	x	x	x	x

A partir de los datos mostrados en estas tablas, el estudiante realizará un análisis de los resultados obtenidos, **que influirá significativamente en la calificación de la práctica**. En dicho análisis se deben comparar los distintos algoritmos en términos de las tasas de clasificación obtenidas (capacidad del algoritmo para obtener soluciones de calidad) y el tiempo requerido para obtener las soluciones (rapidez del algoritmo). En esta práctica se comparará el rendimiento de la metaheurística considerada, la BL, con respecto a los algoritmos de referencia, el 1-NN original y el RELIEF. En las siguientes prácticas se comparará también el rendimiento de las distintas metaheurísticas consideradas entre sí.

7. Documentación y Ficheros a Entregar

En general, la **documentación** de ésta y de cualquier otra práctica será un fichero pdf que deberá incluir, al menos, el siguiente contenido:

a) Portada con el número y título de la práctica, el curso académico, el nombre del problema escogido, los algoritmos considerados; el nombre, DNI y dirección e-mail del estudiante, y su grupo y horario de prácticas.

b) Índice del contenido de la documentación con la numeración de las páginas.

c) **Breve** descripción/formulación del problema (**máximo 1 página**). Podrá incluirse el mismo contenido repetido en todas las prácticas presentadas por el estudiante.

d) Breve descripción de la aplicación de los algoritmos empleados al problema (**máximo 4 páginas**): Todas las consideraciones comunes a los distintos algoritmos se describirán en este apartado, que será previo a la descripción de los algoritmos específicos. Incluirá por ejemplo la descripción del esquema de representación de soluciones y la descripción en pseudocódigo (no código) de la función objetivo y los operadores comunes.

e) Descripción en **pseudocódigo** de la **estructura del método de búsqueda** y de todas aquellas **operaciones relevantes** de cada algoritmo. Este contenido, específico a cada algoritmo se detallará en los correspondientes guiones de prácticas. El pseudocódigo **deberá forzosamente reflejar la implementación/ el desarrollo realizados** y no ser una descripción genérica extraída de las transparencias de clase o de cualquier otra fuente. La descripción de cada algoritmo no deberá ocupar más de **2 páginas**.

   Para esta primera práctica se incluirá la descripción en pseudocódigo del método de exploración del entorno, el operador de generación de vecino y la generación de soluciones aleatorias empleadas en el algoritmo de BL.

f) Descripción en **pseudocódigo** de los algoritmos de comparación.

g) Breve explicación del **procedimiento considerado para desarrollar la práctica**: implementación a partir del código proporcionado en prácticas o a partir de cualquier otro, o uso de un framework de metaheurísticas concreto. Inclusión de un pequeño **manual de usuario describiendo el proceso para que el profesor de prácticas pueda replicarlo**.

h) Experimentos y análisis de resultados:
   - Descripción de los casos del problema empleados y de los valores de los parámetros considerados en las ejecuciones de cada algoritmo (**incluyendo las semillas utilizadas**).
   - Resultados obtenidos según el formato especificado.
   - Análisis de resultados. El análisis deberá estar orientado a **justificar** (según el comportamiento de cada algoritmo) **los resultados** obtenidos en lugar de realizar una mera "lectura" de las tablas. Se valorará la inclusión de otros elementos de comparación tales como gráficas de convergencia, boxplots, análisis comparativo de las soluciones obtenidas, representación gráfica de las soluciones, etc.

i) Referencias bibliográficas u otro tipo de material distinto del proporcionado en la asignatura que se haya consultado para realizar la práctica (si se ha hecho).

Aunque lo esencial es el contenido, debe cuidarse la presentación y la redacción. **La documentación nunca deberá incluir listado total o parcial del código fuente**.

En lo referente al **desarrollo de la práctica**, se entregará una carpeta llamada **software** que contenga una versión ejecutable de los programas desarrollados, así como los ficheros de datos de los casos del problema y el código fuente implementado o los ficheros de configuración del framework empleado. El código fuente o los ficheros de configuración se organizarán en la estructura de directorios que sea necesaria y deberán colgar del directorio FUENTES en el raíz. Junto con el código fuente, hay que incluir los ficheros necesarios para construir los ejecutables según el entorno de desarrollo empleado (tales como *.prj, makefile, *.ide, etc.). La versión ejecutable de los programas y los ficheros de datos se incluirán en un subdirectorio del raíz de nombre BIN. En este mismo directorio se adjuntará un pequeño fichero de texto de nombre LEEME que contendrá breves reseñas sobre cada fichero incluido en el directorio. Es importante que los programas realizados puedan leer los valores de los parámetros de los algoritmos desde fichero, es decir, que no tengan que ser recompilados para cambiar éstos ante una nueva ejecución. Por ejemplo, la semilla que inicializa la secuencia pseudoaleatoria debería poder especificarse como un parámetro más.

El fichero pdf de la documentación y la carpeta software serán comprimidos en un fichero .zip etiquetado con los apellidos y nombre del estudiante (Ej. Pérez Pérez Manuel.zip). Este fichero será entregado por internet a través de la plataforma PRADO.

8. Método de Evaluación

Tanto en esta práctica como en las siguientes, se indicará la puntuación máxima que se puede obtener por cada algoritmo y su análisis. La inclusión de trabajo voluntario (desarrollo de variantes adicionales, experimentación con diferentes parámetros, prueba con otros operadores o versiones adicionales del algoritmo, análisis extendido, etc.) podrá incrementar la nota final por encima de la puntuación máxima definida inicialmente.

En caso de que el comportamiento del algoritmo en la versión implementada/ desarrollada no coincida con la descripción en pseudocódigo o no incorpore las componentes requeridas, se podría reducir hasta en un 50% la calificación del algoritmo correspondiente.