

## Práctica 2: Búsquedas con trayectorias múltiples

Antonio Álvarez Caballero 15457968-J  
5º Doble Grado Ingeniería Informática y Matemáticas  
Grupo de prácticas del Viernes 17:30-19:30  
[analca3@correo.ugr.es](mailto:analca3@correo.ugr.es)

22 de abril de 2016

### Índice

## 1. Descripción del problema

El problema a resolver es el problema de *Selección de características*. En el ámbito de la *Ciencia de Datos*, la cantidad de datos a evaluar para obtener buenos resultados es excesivamente grande. Esto nos lleva a la siguiente cuestión: ¿Son todos ellos realmente importantes? ¿Podemos establecer dependencias para eliminar los que no nos aportan información relevante? La respuesta es que sí: en muchas ocasiones, no todos los datos son importantes, o no lo son demasiado. Por ello, se intentará filtrar las características relevantes de un conjunto de datos.

La selección de características tiene varias ventajas: se reduce la complejidad del problema, disminuyendo el tiempo de ejecución. También se aumenta la capacidad de generalización puesto que tenemos menos variables que tener en cuenta, además de conseguir resultados más simples y fáciles de entender e interpretar.

Para conseguir este propósito se deben usar técnicas probabilísticas, ya que es un problema *NP-hard*. Una técnica exhaustiva sería totalmente inviable para cualquier caso de búsqueda medianamente grande. Usaremos metaheurísticas para resolver este problema, aunque también podríamos intentar resolverlo utilizando estadísticos (correlación entre características, medidas de separabilidad o basadas en teoría de información o consistencia, etc).

## 2. Descripción de aplicación de los algoritmos al problema

Los elementos comunes de los algoritmos son:

- Representación de las soluciones: Se representan las soluciones como vectores 1-dimensionales binarios (los llamaremos *bits* para poder hacer uso de términos como *darle la vuelta a un bit*):

$$s = (x_1, x_2, \dots, x_{n-1}, x_n); x_i \in \{True, False\} \forall i \in \{1, 2, \dots, n\}$$

- Función objetivo: La función a maximizar es la tasa de clasificación de los datos de entrada:

$$tasa\_clas = 100 \cdot \frac{instancias\ bien\ clasificadas}{instancias\ totales}$$

- Generación de vecino: La función generadora de vecinos es bien simple. Se toma una solución y se le da la vuelta a uno de sus bits, el cual se escoge aleatoriamente.

```
Tomar una solución
indice = generarAleatorio(0, numero_caracteristicas)
caracteristicas[indice] = not caracteristicas[indice]
```

## 3. Descripción de la estructura del método de búsqueda

Veamos los esquemas de cada algoritmo en pseudocódigo:

- Búsqueda Multiarranque Básica:

```
mejor_solucion = [False, False, ..., False]
mejor_tasa = 0
numero_busquedas = 25
```

```
Desde 1 hasta numero_busquedas
```

```

caracteristicas_seleccionadas, tasa = LS(datos_entrenamiento,
etiquetas_entrenamiento, clasificador)

Si tasa > mejor_tasa:
    mejor_tasa = tasa
    mejor_solucion = caracteristicas_seleccionadas

return mejor_solucion, mejor_tasa

```

- GRASP:

```

mejor_solucion = [False,False,...,False]
mejor_tasa = 0
numero_busquedas = 25

Desde 1 hasta numero_busquedas
    caracteristicas_seleccionadas, tasa = SFSrandom(datos_entrenamiento,
etiquetas_entrenamiento, clasificador)
    caracteristicas_seleccionadas, tasa = LS(datos_entrenamiento,
etiquetas_entrenamiento, clasificador, caracteristicas_seleccionadas)

    Si tasa > mejor_tasa:
        mejor_tasa = tasa
        mejor_solucion = caracteristicas_seleccionadas

return mejor_solucion, mejor_tasa

```

- Iterated Local Search:

```

solucion_inicial = solucionAleatoria()
mejor_solucion = solucion_inicial
num_searchs = 25
mejor_tasa = 0

caracteristicas_seleccionadas, _ = LS(datos_entrenamiento,
etiquetas_entrenamiento, clasificador, solucion_inicial)

Desde 1 hasta numero_busquedas-1
    nuevas_caracteristicas, nueva_tasa = LS(datos_entrenamiento,
etiquetas_entrenamiento, clasificador,
mutacion(caracteristicas_seleccionadas))

    Si nueva_tasa > mejor_tasa:
        mejor_tasa = nueva_tasa
        mejor_solucion = nuevas_caracteristicas

return mejor_solucion, mejor_tasa

```

Para el GRASP se ha tenido que implementar una versión aleatorizada del SFS, la cual mostramos aquí:

```

caracteristicas_seleccionadas = [False,False,...,False]
mejor_tasa_temporal = 0
peor_tasa_temporal = 0
mejor_caracteristica = 0
mejor_tasa = 0
alpha = 0.3

Mientras mejor_caracteristica no sea None
    mejor_caracteristica = None

    caracteristicas_disponibles = Índices de características de
        caracteristicas_seleccionadas que están a False
    tasas = [0,0,...,0]
    caracteristicas_restringidas = ListaVacía

    ### Enumeración devuelve las características con su índice en el vector
    Para idx,data_idx en enumeración(caracteristicas_disponibles)

        caracteristicas_seleccionadas[data_idx] = True
        tasas[idx] = clasificador.tasarSolucion(caracteristicas_seleccionadas)
        caracteristicas_seleccionadas[data_idx] = False

        Si tasas[idx] > mejor_tasa_temporal
            mejor_tasa_temporal = tasas[idx]
        Si no tasas[idx] < peor_tasa_temporal
            peor_tasa_temporal = tasas[idx]

    Para idx,data_idx en enumeración(caracteristicas_disponibles)
        Si tasas[idx] > umbral
            caracteristicas_restringidas.añadir(data_idx)

    caracteristica_aleatoria = aleatorio de caracteristicas_restringidas

    caracteristicas_seleccionadas[caracteristica_aleatoria] = True
    tasa = clasificador.tasarSolucion(caracteristicas_seleccionadas)

    Si tasa > mejor_tasa
        mejor_tasa = tasa
        mejor_caracteristica = caracteristica_aleatoria
    En otro caso
        caracteristicas_seleccionadas[caracteristica_aleatoria] = False

return caracteristicas_seleccionadas, mejor_tasa

```

Para el ILS el operador de mutación es darle la vuelta al 10 % de los bits de la máscara. Se ha implementado así:

```

cambios = EnteroPorArriba(0.1 * longitud(caracteristicas))
mascara = repetir(True, cambios)
intactos = repetir(False, longitud(caracteristicas) - cambios)
mascara_completa = concatenar((mascara,intactos))

```

```

Shuffle(mascara_completa)
caracteristicas_mutadas = np.logical_xor(caracteristicas,mascara_completa)
return caracteristicas_mutadas

```

## 4. Descripción del algoritmo de comparación

El algoritmo de comparación es un algoritmo greedy: el *Sequential Forward Selection(SFS)*. La idea es muy simple: se parte del conjunto vacío de características (todos los bits a 0) y se recorren todas las características, evaluando la función de coste. La característica que más mejora ofrezca, se coje. Y se vuelve a empezar. Así hasta que ninguna de las características mejore el coste.

```

caracteristicas = (1,2,...,n)
caracteristicas_seleccionadas = (0,0,...,0,0)
fin = falso
mejor_caracteristica = 0

Mientras mejor_caracteristica != -1
    mejor_tasa = 0
    mejor_caracteristica = -1
    Para cada característica
        tasa = coste(característica)
        Si tasa > mejor_tasa
            mejor_tasa = tasa
            mejor_caracteristica = característica
    Si mejor_caracteristica != -1
        caracteristicas_seleccionadas.añadir(mejor_caracteristica)

```

## 5. Desarrollo de la práctica

En primer lugar, comentar que las bases de datos han sido modificadas en su estructura (que no en sus datos) para que sean homogéneas. Así, se han puesto todas las clases como numéricas (en Wdbc no lo estaban) y se han colocado en la última columna.

La práctica se ha desarrollado usando el language de programación *Python*, ya que su velocidad de desarrollo es bastante alta. Para intentar lidiar con la lentitud que puede suponer usar un lenguaje interpretado, utilizaremos las librerías *NumPy*, *SciPy* y *Scikit-Learn*, que tienen módulos implementados en C (sobre todo *NumPy*) y agilizan bastante los cálculos y el manejo de vectores grandes.

Usaremos alguna funcionalidad directa de estas bibliotecas:

- *NumPy*: Generación de números aleatorios y operaciones rápidas sobre vectores.
- *SciPy*: Lectura de ficheros ARFF de WEKA.
- *Scikit-Learn*: Particionamiento de los datos, tanto las particiones estratificadas de la validación cruzada 5x2 como las de *Leave One Out* para calcular la función de coste. También se ha tomado un clasificador KNN, ya que está implementado usando estructuras de datos complejas como *Ball Tree* y lo hace muy rápido.

Esta elección se ha hecho para poder preocuparme sólo y exclusivamente de la implementación de las metaheurísticas.

Los requisitos para ejecutar mis prácticas son *Python3* (importante que sea la 3), *NumPy*, *SciPy* y *Scikit-Learn*. En mi plataforma (Archlinux) están disponibles desde su gestor de paquetes.

Una vez instalados los paquetes, sólo hay que ejecutar la práctica diciéndole al programa los algoritmos que queremos ejecutar. La semilla aleatoria está fijada dentro del código como 12345678 para no inducir a errores. Veamos algunos ejemplos de llamadas a la práctica. Primero notamos que los algoritmos disponibles son:

- -SFS: Ejecuta el algoritmo greedy SFS.
- -LS: Ejecuta la Local Search.
- -SA: Ejecuta el Simulated Annealing.
- -TS: Ejecuta la Tabu Search.
- -TSext: Ejecuta la Tabu Search extendida.
- -BMB: Ejecuta la Búsqueda Multiarranque Básica.
- -GRASP: Ejecuta el GRASP.
- -ILS: Ejecuta la Iterated Local Search

```
$ python featureSelection.py -TS
```

Se ejecutará la Tabu Search. Pero no sólo se limita el programa a un algoritmo. Si le pasamos varios, los ejecutará en serie uno detrás de otro. Esto ha cambiado desde la práctica anterior por la entrada de CUDA, que hay que iniciarlo debidamente y no es tan sencillo de ejecutar cosas en paralelo.

```
$ python featureSelection.py -BMB -GRASP -ILS
```

Se ejecutarán en paralelo BMB, GRASP e ILS en serie.

Una vez ejecutado, irán saliendo por pantalla mensajes de este tipo, que proporcionan datos en tiempo real del estado de la ejecución:

```
INFO:__main__:W - TS - Time elapsed: 2265.526112794876.  
Score: 98.2394337654. Score out: 95.0877192982 Selected features: 15
```

Este mensaje nos dice todo lo necesario: W es la base de datos (Wdbc), TS el algoritmo, el tiempo transcurrido para esta iteración (recordemos que hay 10), el score de entrenamiento, el score de validación y las características seleccionadas.

## 6. Experimentos

Como se ha comentado antes, la semilla está fija a 12345678 para no tener problemas de aleatoriedad. El número de evaluaciones máxima de todos los algoritmos es de 15000. Por lo demás, todos los demás parámetros propios de cada algoritmo están tal y como se explica en el guión ( $\mu = 0,3$ , los valores de vecinos máximos, soluciones máximas aceptadas, etc).

# KNN

	Wdbc				Movement_Libras				Arrhythmia			
	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T
Partición 1-1	96.12676	96.84211	0.0	0.0	66.66667	65.0	0.0	0.0	62.5	66.49485	0.0	0.0
Partición 1-2	96.49123	96.83099	0.0	0.0	65.55556	80.55556	0.0	0.0	61.85567	62.5	0.0	0.0
Partición 2-1	96.12676	96.49123	0.0	0.0	68.88889	74.44444	0.0	0.0	64.0625	64.43299	0.0	0.0
Partición 2-2	95.78947	96.12676	0.0	0.0	75.55556	68.88889	0.0	0.0	61.34021	64.58333	0.0	0.0
Partición 3-1	96.47887	95.4386	0.0	0.0	75.55556	71.66667	0.0	0.0	63.02083	63.91753	0.0	0.0
Partición 3-2	96.84211	96.83099	0.0	0.0	68.88889	65.55556	0.0	0.0	62.37113	64.58333	0.0	0.0
Partición 4-1	97.53521	96.14035	0.0	0.0	66.66667	66.11111	0.0	0.0	65.625	64.43299	0.0	0.0
Partición 4-2	93.33333	97.88732	0.0	0.0	72.77778	72.77778	0.0	0.0	60.30928	64.58333	0.0	0.0
Partición 5-1	96.47887	96.84211	0.0	0.0	75.0	71.11111	0.0	0.0	65.625	65.97938	0.0	0.0
Partición 5-2	97.89474	95.42254	0.0	0.0	70.0	70.0	0.0	0.0	61.85567	63.54167	0.0	0.0
Media	96.30974	96.48530	0.0	0.0	70.55556	70.61111	0.0	0.0	62.85653	64.50494	0.0	0.0

# SFS

	Wdbc				Movement_Libras				Arrhythmia			
	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T
Partición 1-1	97.53521	92.2807	83.33333	0.15793	72.22222	66.66667	93.33333	0.50295	80.20833	70.61856	96.76259	2.41429
Partición 1-2	96.84211	94.01408	86.66667	0.12852	77.22222	66.66667	87.77778	0.91927	73.19588	67.1875	97.48201	1.84876
Partición 2-1	95.42254	91.22807	83.33333	0.15797	84.44444	68.88889	85.55556	1.0978	79.16667	69.58763	97.48201	1.83412
Partición 2-2	97.54386	92.60563	86.66667	0.12911	77.77778	61.66667	93.33333	0.49637	74.2268	64.58333	97.48201	1.88374
Partición 3-1	96.12676	92.98246	90.0	0.09982	83.33333	71.66667	87.77778	0.92519	76.5625	68.5567	98.20144	1.30598
Partición 3-2	97.54386	96.47887	86.66667	0.12927	72.22222	60.0	93.33333	0.50101	71.64948	66.14583	98.20144	1.33008
Partición 4-1	98.23944	96.49123	86.66667	0.12883	70.55556	62.22222	91.11111	0.66223	76.04167	69.58763	97.84173	1.55438
Partición 4-2	94.73684	94.3662	90.0	0.10095	80.55556	65.55556	90.0	0.74934	82.98969	73.95833	97.1223	2.16781
Partición 5-1	94.71831	91.92982	90.0	0.09997	78.88889	66.11111	92.22222	0.57865	77.08333	68.04124	97.48201	1.85395
Partición 5-2	98.94737	93.66197	76.66667	0.21516	76.66667	66.66667	90.0	0.74815	82.98969	71.35417	96.40288	2.75023
Media	96.76563	93.60390	86.00000	0.13475	77.38889	65.61111	90.44444	0.71810	77.41140	68.96209	97.44604	1.89433

# BMB

	Wdbc				Movement_Libras				Arrhythmia			
	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T
Partición 1-1	98.23944	96.84211	26.66667	1.08139	73.33333	70.0	47.77778	4.66891	70.83333	65.97938	44.96403	76.8779
Partición 1-2	97.89474	96.83099	46.66667	1.10791	72.77778	85.55556	58.88889	4.28663	69.58763	61.45833	47.84173	58.7874
Partición 2-1	97.53521	97.19298	36.66667	1.11772	80.55556	64.44444	57.77778	4.13646	70.3125	63.91753	49.64029	72.29524
Partición 2-2	97.89474	96.47887	43.33333	1.15078	74.44444	75.0	47.77778	4.33228	70.61856	66.14583	46.76259	64.78203
Partición 3-1	98.23944	97.19298	20.0	1.05315	83.33333	67.22222	45.55556	4.48178	73.4375	60.82474	51.07914	73.30478
Partición 3-2	97.19298	97.53521	36.66667	1.1128	75.55556	76.66667	46.66667	4.16955	69.58763	64.0625	46.40288	50.91528
Partición 4-1	97.88732	95.4386	30.0	0.99947	77.22222	73.33333	45.55556	4.35166	72.39583	59.27835	50.0	63.49354
Partición 4-2	98.24561	97.1831	46.66667	1.119	76.11111	71.11111	50.0	4.18398	72.16495	61.97917	43.16547	57.8558
Partición 5-1	98.23944	95.08772	43.33333	1.06416	75.55556	76.11111	43.33333	4.42512	69.79167	64.94845	49.28058	71.22764
Partición 5-2	97.89474	95.77465	30.0	1.01079	77.77778	72.77778	36.66667	4.52043	71.64948	66.66667	48.56115	62.83519
Media	97.92637	96.55572	36.00000	1.08172	76.66667	73.22222	48.00000	4.35568	71.03791	63.52609	47.76979	65.23748

# GRASP

	Wdbc				Movement_Libras				Arrhythmia			
	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T
Partición 1-1	98.94366	96.84211	73.33333	3.96508	79.44444	70.0	77.77778	16.53877	83.85417	65.97938	90.28777	34.61972
Partición 1-2	98.24561	96.83099	63.33333	4.25855	80.0	85.55556	83.33333	18.53391	79.38144	61.45833	89.56835	35.33348
Partición 2-1	97.88732	96.14035	63.33333	3.48653	80.0	66.66667	83.33333	18.17763	84.89583	63.91753	92.80576	38.46195
Partición 2-2	97.89474	96.47887	63.33333	4.35416	78.33333	71.11111	84.44444	15.93595	80.41237	63.02083	90.28777	32.37455
Partición 3-1	97.88732	95.78947	60.0	3.85797	80.0	77.22222	82.22222	16.57017	82.8125	62.37113	93.88489	32.66044
Partición 3-2	98.94737	96.12676	66.66667	3.78628	80.55556	66.66667	84.44444	16.89488	82.98969	65.10417	94.60432	38.0697
Partición 4-1	98.59155	95.78947	53.33333	4.34525	81.66667	73.33333	80.0	17.6753	83.85417	59.79381	94.60432	32.38685
Partición 4-2	97.89474	95.77465	73.33333	3.71255	79.44444	72.22222	74.44444	18.31128	82.47423	64.58333	90.64748	39.51833
Partición 5-1	98.94366	95.4386	53.33333	4.27863	81.11111	72.22222	75.55556	18.13788	79.16667	63.91753	94.60432	28.64678
Partición 5-2	97.54386	96.83099	66.66667	3.95238	78.33333	80.0	77.77778	17.75007	83.50515	60.9375	93.88489	31.36983
Media	98.27798	96.20423	63.66667	3.99974	79.88889	73.50000	80.33333	17.45258	82.33462	63.10835	92.51799	34.34416

# ILS

	Wdbc				Movement_Libras				Arrhythmia			
	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T
Partición 1-1	98.23944	96.84211	33.33333	0.7476	72.77778	70.0	32.22222	3.77779	70.3125	65.97938	47.48201	65.18549
Partición 1-2	98.24561	96.83099	26.66667	0.74363	73.88889	85.55556	30.0	3.74002	72.16495	61.45833	46.04317	53.58842
Partición 2-1	98.59155	96.49123	26.66667	0.63564	74.44444	76.11111	33.33333	3.31625	71.35417	61.34021	44.2446	65.78307
Partición 2-2	97.89474	96.83099	30.0	0.89896	78.88889	72.77778	31.11111	3.50403	72.16495	64.0625	39.20863	51.64644
Partición 3-1	98.59155	96.14035	36.66667	0.80554	74.44444	74.44444	34.44444	3.43782	69.27083	63.91753	41.00719	67.36685
Partición 3-2	97.19298	95.42254	20.0	0.78028	78.88889	73.88889	35.55556	3.40206	70.61856	61.45833	36.69065	46.09727
Partición 4-1	97.53521	97.54386	36.66667	0.78197	79.44444	68.88889	32.22222	3.18754	72.91667	64.94845	45.68345	59.02578
Partición 4-2	98.24561	96.83099	30.0	0.70736	77.22222	72.77778	36.66667	4.11384	69.07216	61.97917	47.48201	45.23494
Partición 5-1	98.94366	95.4386	43.33333	1.01458	76.11111	71.11111	38.88889	3.96739	69.27083	64.43299	40.64748	66.28016
Partición 5-2	98.24561	95.42254	26.66667	0.70555	73.88889	76.66667	24.44444	3.05619	73.71134	61.45833	43.52518	53.40892
Media	98.17260	96.37942	31.00000	0.78211	76.00000	74.22222	32.88889	3.55029	71.08570	63.10352	43.20144	57.36173

## Media

	Wdbc				Movement Libras				Arrhythmia			
	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T	% clas in	% clas out	% red	T
KNN	96.30974	96.48530	0.0	0.0	70.55556	70.61111	0.0	0.0	62.85653	64.50494	0.0	0.0
SFS	96.76563	93.60390	86.00000	0.13475	77.38889	65.61111	90.44444	0.71810	77.41140	68.96209	97.44604	1.89433
BMB	97.92637	95.67606	36.00000	1.08608	76.66667	71.16667	48.00000	4.38071	71.03791	65.49077	47.76979	65.61655
GRASP	98.27798	95.25414	63.66667	4.05062	79.88889	70.72222	80.33333	17.57407	82.33462	71.49646	92.51799	34.47605
ILS	98.17260	95.60588	31.00000	0.80009	76.00000	70.38889	32.88889	3.61344	71.08570	64.09472	43.20144	57.19147

Meter conclusiones.

## 7. Referencias

Las referencias utilizadas han sido:

- *Scikit-Learn*: La propia [documentación<sup>1</sup>](http://scikit-learn.org/stable/modules/classes.html) de la biblioteca.
- *SciPy*: La propia [documentación<sup>2</sup>](http://docs.scipy.org/doc/scipy/reference/) de la biblioteca.

---

<sup>1</sup><http://scikit-learn.org/stable/modules/classes.html>

<sup>2</sup><http://docs.scipy.org/doc/scipy/reference/>