Universidad De Granada

Grado ingeniería informática (2017 – 2018)

METAHEURÍSTICAS

Práctica 1: APC (1NN, RELIEF y Búsqueda Local)

Trabajo realizado por: Antonio Miguel Morillo Chica – 77379031Y - Grupo 3

Índice

1. Descripción del problema ..... pág. 3

2. Algoritmos empleados ..... pág. 4

   2.1. Descripción esquema de representación soluciones..... pág. 4

   2.2. Descripción de la función objetivo. ..... pág. 4

   2.3. Operadores comunes ..... pág. 5

3. Descripción de los algoritmos específicos ..... pág. 7

   3.1. Algoritmo RELIEF ..... pág. 7

   3.2. Algoritmo de Búsqueda Local ..... pág. 8

4. Descipción del Algoritmo de Comparación ..... pág. 9

5. Desarrollo de la práctica ..... pág. 10

   5.1. Manual de la práctica ..... pág. 10

6. Experimentos y analisis de los resultados ..... pág. 11

   6.1. Descripción de los problemas y valores parámetros ..... pág. 11

   6.2. Resultados obtenidos ..... pág. 12

   6.3. Comentario sobre los resultados ..... pág. 13

1. Descripción del problema.

El problema expuesto para la realización de la práctica es el "**Problema del Aprendizaje de Pesos en Carácteriticas**". Este problema consiste en optimizar el rendimiento del clasificador a partir de una serie de pesos asociados a unos determinados ejemplos. Disponemos de una muestra de objetos clasificados $w_{0,} w_{1,} \ldots, w_n$ representados en función de una serie de atributos en un vector de características: $(x_1(w_i), \ldots, x_n(w_i))$ cada objeto pertenecerá a una clase asociada:

$w_k = (x_1(w_k), \ldots, x_n(w_k)) \rightarrow C_{ik}$

El objetivo será obtener un sistema que pueda clasificar todos los ejemplos de forma automática infiriendo esta clasificación através de las características. Para ello deberemos será entrenarnos sobre un conjunto de datos para aprender y posteriormente usaremos un conjunto de datos de prueba para validar el aprendizaje relizado.

El clasificador hace uso de la distancia, que se define como la similitud que tendrán dos ejemplos con relación a las características de los mismos. La menor distancia entre un ejemplo que se quiere clasificar y todos los de la muestra se usará para determinar la clasificación de este nuevo.

En la práctica generaremos muchos conjuntos de pesos através de diferentes técnicas para obtener el vector de pesos que mejor clasifique los ejemplos. Las distintas formas en las que obtendremos estos vectores serán:

- De forma aleatoria (aunque no lo usemos para ver los resultados)
- Todos los pesos a 1.
- Algoritmo Relief.
- Algoritmo aleatorio con Búsqueda local.

2. Aplicación de los algoritmos.

A continuación describiremos la descripción del esquema de representación de soluciones, la descripción de la función objetivo y la de los operadores comunes usados.

2.1. Descripción esquema de representación soluciones.

La solución la denotaremos con W que será un vector de pesos asociados, donde cada posición será el peso de la caracteristica asociada y que habrá tantas como atributos tenga el ejemplo: $W=(w_1, w_2, \ldots, w_{N-1}, w_N)$ dónde $w_i \in [0,1], N \in \mathbb{N}$

w1	w2	...	w_{n-1}	w_n

Si el valor asociado a la posición i es cero quiere decir que la característica no se usará en el cálculo, en cambio si es uno quiere decir que se usará completamente en el cálculo de la distancia. Los valores intermedios del intervalor indican el grado de importacia de cada característica.

2.2. Descripción de la función objetivo.

El objetivo en este problema es maximizar la función objetivo que tiene dos criterios, **precisión** y **simplicidad**, por un lado, la tasa de clasificación que mide el porcentaje de ojetos bien clasificados, es decir, la precisión, siguiendo la siguiente formula: $Tasa-Clasificación = 100 \cdot \frac{n^o\, bien\, clasificados}{n^o\, ejemplos}$

```
bienClasificados = 0
for i in numEjemplosClasificados
   if etiquetas[i] == NN1(datos[i])  then
      bienClasificados++
return bienclasificados/numClasificados
```

Por otro lado la tasa de reducción que mide el porcentaje de características descartadas, aquellas cuyo peso esté cercano a cero en W, con respecto a "n". Consideraremos un umbral de 0.1 en w_i para considerar que se descarta una

característica: $Tasa-Deducción = 100 \cdot \frac{n^o.\, valores\, w_i < 0.2}{n^o\, características}$

```
NumDescartados = 0
for i in W
   if W[i] < 0.2 them
      numDescartados++;
return numDescartados/W.size()
```

La función objetivo es la combinación de ambos criterios donde ambos tendrán para nosotros la misma importancia por tanto lo que buscamos es maximizar el accierto a la vez que consideramos el mínimo númeor de características posibles.

$$F(W) = \alpha \cdot tasa-clasificación(W) + (1-\alpha) \cdot tasa-reducción(W)$$

2.3. Operadores comunes.

Los siguientes operadores han sido usados para los diferentes procedimientos de los algoritmos o para la preparación de los valores en las características.

a) **Generación de soluciones iniciales aleatorias:** Usado para el calcular una posible valoración inicial para los pesos.

```
W = vector(num_caracteristicas)
for in W
   W[i] = randFloat(0,1)
return W
```

Nota: De igual forma este operador ha sido implementado con la inicializacióon de los pesos a 1 para el algoritmo 1NN.

b) **Mutación del vector de Pesos**: Usado en el algoritmo de Búsqueda Local para la mutación uno a uno del vector de características.

```
mutar_vector(vector<float> w)
   w = pesos_actual
   posMutar = randomInt(1,num_caracteristicas)
   w[posMutar] = w[posMutar] + DistribucionNormal(media,desviacion)
   w[posMutar] = truncar(w[posMutar])
return w;
```

c) **Distancia Euclidia**: entre dos ejemplos sería:

```
float distasnciaEuclidia(e1,e2){
   w = solucion_actual;
   resultado = 0
   for i in num_caracteristicas {
      resta = e1[i] - e2[i]
      resultado += w[i] * resta * resta
   }
   resultado = sqrt(resultado)
   return resultado
}
```

- Es importante tener en cuenta que las características estén normalizados para que al calcular las distancias no se prioricen unos sobre otros.

3. Descripción de los algoritmos específicos.

A continuación mostraré y describiré los dos algoritmos usados para esta práctica a excepción del 1NN que lo explicaremos en el siguiente apartado, en la descrpción del algoritmo de compraración.

3.1. Algoritmo RELIEF.

El algoritmo RELIEF se utiliza para el calculo del vector de pesos asociado a las características. El algoritmo comieza con el vector de pesos w inicializado a 0. Por todos los ejemplos calcula su amigo y su enemigo, el amigo es un ejempllo cercano de su misma clase, el enemigo es un ejemplo de clase distinta más cercano.

Una vez calculado su amigo y enemigo recorre todos los pesos y los modifica sumandole la diferencia entre el peso amigo y enemigo más cercano. Tras esto se normalizan los valores de los pesos.

```
vector<float> RELIEF(){
   matriz = ejemplos_y_características
   w = vector(matriz.getNumAtributos(),0)
   for i in num_ejemplos {
      pos_enemigo = buscarEnemigo(matriz[i])
      pos_amigo = buscarAmigo(matriz[i])
      for j in num_caracteristicas {
         enemigo = matriz[i][j] - matriz[pos_enemigo][j]
         amigo = matriz[i][j]  matriz[pos_amigo][j]
         w[j] += enemy - friend
      }
   }
   w = normalizar(w)
   return w
}
```

- Los datos se normalizan por si aparecen valores que se salen de nuestro rango.
- Matriz contiene por filas los ejemplos y cada columna es su vector de características.

3.2. Algoritmo de Búsqueda Local.

El algoritmo de busqueda local transforma una solución inicial en otra atavés de una serie de mutaciones. El algoritmo necesita una serie de datos de entrada, numero de evaluaciones y vecinos, el primero porque el tamaño del entorno es infinito y necesitamos un límite de evaluaciones, el segundo, vecinos, se usa para saber cuantas mutaciones mínimas por característica se tiene que hacer si no se encontrase una solución mejor a la actual.

Por cada mutación se llama a la función de evaluación para saber si es mejor que la solución optima actual, en tal caso se actualizan los datos y se cambia la solución optima por la nueva. En caso de que ninguna mutación surja efecto, el vector devuelto debería ser el inicial, ninguna modificación se lleva acabo.

```
vector<float> BL(num_evals, vecinnos, desviacion){
   num_vecinos = 0 ; mejor = 0 ; evals = 0
   w = solucion_actual
   while (evals < num_evals && num_vecinos < vecinos*num_caracteristicas) {
      w_nuevo = w;
      w_nuevo = mutar(w_nuevo)
      if evaluar(w_nuevo) > mejor {
            mejor = evaluar(w_nuevo)
            w = w_nuevo
            num_vecionos = 0
      }
      evals++ ;  vecinos++
   }
   return w_nuevo
}
```

- El la práctica el numero de evaluaciones es de 1500.
- Sin embargo en número de vecinos es 20 que, en nuestro datos:
  - S-Heart: 45 caracterçisticas * 20 = 900 mutaciones como mínimo.
  - Parkinson: 23 características * 20 = 460 mutaciones como mínimo.
  - Ozone: 73 características * 20 = 1460 mutaciones como mínimo

4. Descipción del Algoritmo de Comparación.

El algoritmo usado para la comparación es el algoritmo KNN en nuestro caso es el 1NN ya que solo usaremos el elemento más cercano y no los k más cercanos. Para determinar la cercania usaremos la distancia euclídia ya que los valores de las variables no son nominales, en tal caso usariamos hamilton por ejmplo.

```
pair<string, float> 1NN( ejemplos_y_caracteristicas, clasificacion,
ejemplo_actual) {
   etiqueta_resultado = v.etiquetas[0]
   distancia_minima = distanciaEucludia(v.etiqueta[0], ejemplo)
   for i in num_ejemplos{
      aux = distanciaEuclidia(matriz[i], ejemplo)
      if aux < distancia_minima them {
         distancia_minima = aux
         etiqueta_resultado = v_etiquetas[i]
      }
   }
   return <etiqueta_resultado, distancia_minima>
}
```

El algoritmo 1NN se utilida para clacular la clase de un ejemplo que en principio se desconocía. Devuelve la etiqueta de la clase comparando con todos los ejemplos de la muestra. Se usa Leave-One-Out para no compararlo consigo mismo ya que sino la distancia con el mismo sería 0.

Al principio guardamos la distancia y la etiqueta del primer ejemplo. Recorremos el vector completo calculando la distancia con cada uno, si es menor actualizamos los valores distancia_mínima y la etiqueta_resultado hasta revisar todos los ejemplos.

Finalmente se devolverá la etiqueta de distancia mínima con respecto al ejemplo y su ditancia que será usada para otros algoritmos.

5. Desarrollo de la práctica.

La práctica ha sido desarrollada en C++ sin ningún uso de libreerías externas salvo las aportadas y las típicas de la STL. Sin embargo al compiezdo del desarrollo se iba a usar la librería Arff encontrada en github, una serie de clases que sirven para lectura de estos archivos. Finalmente fue descartado porque daban errores con los archivos propuestos y la lectura de los mismos se realizó a mano. El código de evaluación de los algoritmos ha sido paralelizado gracias a OpenMP visto y usado en asignaturas como Estructura de los Computadores. Además el código se compila con optimización O3.

La practica está compuesta por tres clases principales y dos ficheros con funciones "amigas" para el uso de los randoms y la medida de tiempos en las ejecuciones.

- **Datos**: Se usa para la lectura de los datos de entrada así como el procedimiento para la división entre los datos de entrenamiento y los datos de prueba de forma aleatoria en dos mitades iguales. Además contendrá los vectores de las etiquetas, de entrenamiento y prueba y el vector de pesos asociado a las características.
- **Clasificador**: Contiene los algortirmos de comparación y RELIEF para el calculo de los pesos. Además será donde he impleentado el operador de la distancia euclídia y la función de evaluación.
- **Búsqueda Local**: Contine el clasificado rque se ha usado ya que este encapsula un objeto datos que contiene el vector a mutar, además contiene las operaciones de mutación para el vector.
- **Main**: Será el encargado de medir los tiempos y mostrar los resultados de la tasa de clasificación, de red y el resultado de los tiempo.

5.1. Manual de la práctica.

La práctica contiene un makefile así que unicamente hay que ejecutar la orden make por terminal para compilar el proyecto. Para la ejecución:

```
$ ./bin/<ejecutable>  datos/<fichero>.arff  <semilla>
```

6. Experimentos y analisis de los resultados.

6.1. Descripción de los problemas y valores parámetros.

Los problemas a los que nos hemos enfrentado son tres de variables numéricas donde deberemos de clasificar los ejemplos.

1. **Ozone**: es una base de datos para la detección del nivel de ozono según las mediciones realizadas a lo largo del tiempo. Tiene 320 ejemplos y consta de 73 atribustos y dos clases, dia normal o dia de alta concentración de ozono. **Deberemos de averiguar si los nuevos ejemplos será días con alto o baja concentración de ozono.** La ejecución para este problema es la siguiente:

```
$ ./bin/p1    datos/ozone.arff       62
```

2. **Parkinsons**: es una base de datos que contiene datos que se utilizan para 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. Tiene 195 ejemplos de 23 atributos y de dos clases, tiene parkinson o no tiene. **Deberemos de averiguar si los nuevos ejemplos serán personas con o sin parkinsons.** La ejecución para este problema es la siguiente:

```
$ ./bin/p1    datos/parkinsons.arff      94
```

3. **Spectf-heart**: es una base de datos que contiene atributos calculados a partir de imágenes médicas de tomografía computerizada (SPECT) del corazón de pacientes humanos. La tarea consiste en determinar si la fisiología del corazón analizado es correcta o no. Consta de 267 ejemplos de 45 atributos enteros y dos clases, paciente sano o patología. **Deberemos de averiguar si los nuevos ejemplos serán personas con o sin patologías**. La ejecución para este problema es la siguiente:

```
$ ./bin/p1    datos/spectf-heart.arff      55
```

6.2 Resultados optenidos.

La tabla de los resultados se ajustan a los pedidos según la tabla aportada en la página web de la asignatura.

- Particiones 1NN:

	Ozone				Parkinsons				Spectf-heart			
	%_clas	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T
Partición 1	79,08	0,00	39,54	0.011311	93,31	0,00	46,66	0.010653	73,63	0,00	36,82	0.007461
Partición 2	78,46	0,00	39,23	0.005467	89,21	0,00	44,60	0.001457	77,37	0,00	38,68	0.011937
Partición 3	76,78	0,00	38,39	0.005534	89,57	0,00	44,78	0.001464	76,89	0,00	38,45	0.005462
Partición 4	77,20	0,00	38,60	0.005951	89,35	0,00	44,67	0.004116	76,15	0,00	38,07	0.005519
Partición 5	76,19	0,00	38,10	0.006265	88,81	0,00	44,40	0.002236	75,36	0,00	37,68	0.007353
Media	77,54	0,00	38,77	#¡DIV/0!	90,05	0,00	45,02	#¡DIV/0!	75,88	0,00	37,94	#¡DIV/0!

- Particiones RELIEF:

	Ozone				Parkinsons				Spectf-heart			
	%_clas	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T
Partición 1	77,04	13,52	45,28	0.006319	96,94	0,00	48,47	0.00196	77,87	12,07	44,97	0.010819
Partición 2	77,04	12,58	44,81	0.005418	93,37	0,00	46,68	0.001404	73,94	13,18	43,56	0.011211
Partición 3	75,48	12,01	43,75	0.005711	93,86	0,00	46,93	0.001402	76,30	12,14	44,22	0.005383
Partición 4	75,01	11,65	43,33	0.005808	91,27	0,00	45,64	0.001405	74,18	11,98	43,08	0.00537
Partición 5	72,65	12,40	42,53	0.006803	91,59	0,00	45,79	0.001414	73,80	11,75	42,78	0.005345
Media	75,45	12,43	43,94	#¡DIV/0!	93,41	0,00	46,70	#¡DIV/0!	75,22	12,22	43,72	#¡DIV/0!

- Particiones BL:

	Ozone				Parkinsons				Spectf-heart			
	%_clas	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T
Partición 1	83,65	13,52	48,58	8,66	97,96	0,00	48,98	0,67	86,21	13,22	49,71	6,01
Partición 2	83,18	12,11	47,64	18,49	95,92	0,00	47,96	0,91	87,67	12,32	50,00	8,76
Partición 3	83,19	10,76	46,98	13,09	96,94	0,00	48,47	0,73	89,87	12,62	51,24	6,68
Partición 4	83,75	10,40	47,07	25,15	94,09	0,00	47,05	0,73	88,09	13,85	50,97	9,32
Partición 5	82,09	11,09	46,59	21,91	94,36	0,00	47,18	0,93	88,93	13,99	51,46	8,35
Media	83,17	11,58	47,37	17,46	95,85	0,00	47,93	0,79	88,15	13,20	50,68	7,82

- Datos globales:

	Ozone				Parkinsons				Spectf-heart			
	%_clas	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T	*%_clas*	*%red*	*Agr.*	T
1-NN	77,54	0,00	38,77116	0	90,05	0,00	45,02428	0	75,88	0,00	37,93962	0,00
RELIEF	75,45	12,43	43,93901	0	93,41	0,00	46,7026	0	75,22	12,22	43,72091	0,00
BL	83,17	11,58	47,37	17,46	95,85	0,00	47,93	0,79	88,15	13,20	50,68	7,82

6.3. Comentario sobre los resultados.

El primer apunte es sobre los tiempos de los algoritmos 1NN y RELIEF, ambos son muy rápidos por lo que el tiempo es práctica 0. En las primeras tablas se reflejan estos resultados pero la media es erronea debido a que si camio la coma por punto se redondea a los dos últimos decimales y solo pondría un 0. Esto es normal ya que la cantidad de operaciones que se realizan en cada algoritmo es muy baja a diferencia de la búsqueda local en la que debido a que se han de generar los vecinos en relación por cada carácteristica y por cada mutación del vector se evalua el tiempo crece.

La tendencia de los resultados es la alza, es evidente que el algoritmo BL obtiene mejores resultados que RELIEF ya que en esencia lo que hace es mejorar los la valoración de pesos para las caracteristicas. Como podemos ver RELIEF debe de descartar alguna característica más que la BL hace que aparezca por eso la tasa de reducción aumenta ligeramente.

La tasa de reducción para los datos del parkinson es realmente extraña, al parecer todos los algortimos consideran que todas las características son importantes (por encima del valor 0,2) cosa que me parece un tanto extraña aunque puede ser error de programación o por los tipos de datos pero esto no he podido indagar en ello lo suficiente por falta de tiempo.

Conlusión: Como lo que buscamos es máximizar la "Agregación" me siento satisfecho pues, en mis resultados se refleja una tendencia a que cada algoritmo lo mejora, se hace mayor, aún así, sorprende que la tasa de clasificación para el ozone con 1NN es mayor que para el RELIEF pero con respecto a las tasas hay una mejora mayor por parte de RELIEF ya que descarta características a diferencia de 1NN.