Practica 5:

Búsquedas Híbridas para el Problema de la Selección de Características

Ahisahar Pretel Rodríguez 75937643S
Correo: approdriguez@correo.ugr.es
Grupo Practicas 2/Jueves 5:30/7:30
2014/2015

Índice

1. Descripción del problema:

Estamos ante un problema de clasificación, con aprendizaje supervisado. Disponemos de una muestra de objetos ya clasificados en función de sus valores en una serie de atributos y cada objeto pertenece a una clase.

El objetivo a realizar es obtener un sistema que permita clasificar automáticamente dichos objetos con el mayor porcentaje de éxito posible.

Al ser un problema supervisado, conocemos el numero de clases existentes en el problema y la clase concreta a la que pertenece cada objeto del conjunto de muestras.

Para llevar a cabo esta tarea necesitamos realizar dos acciones: Aprendizaje y Validación.
Por lo que tenemos que dividir el conjunto de ejemplos en dos conjuntos:

-” Train” o entrenamiento: utilizado para aprender el clasificador.
-”Test” o prueba: utilizado para validar el clasificador, es decir, para probarlo y calcular el porcentaje de clasificación sobre los ejemplos del conjunto.

Para mayor seguridad, se crean varios conjuntos tanto de test como de training, para ello utilizaremos la técnica de validación cruzada:

- Crearemos 5 conjuntos distintos de los datos al 50%.
-Aprenderemos con el conjunto de train y validaremos con el de test.
-De cada partición obtendremos dos valores de porcentaje.
-La calidad del método vendrá dada por la media de los 10 porcentajes de clasificación.

2.Descripción de la aplicación de los algoritmos

El objetivo de esta práctica es estudiar el funcionamiento de los Algoritmos Meméticos.

Para representar las soluciones usaremos un struct que representará a la cadena de características, 0 si no coge la característica y 1 si la coge, asimismo contendra dos varibles, tasa_class y tasa_red

Para llevar a cabo dicha tarea emplearemos los siguientes algoritmos:

- La función objetivo: Será el porcentaje de acierto del clasificador 3-NN generado a partir de la selección de características codificada en la solución actual. El objetivo será maximizar esta función.

El algoritmo 3NN es una generalización del algoritmo knn, que es uno de los clasificadores mas utilizados por su simplicidad, su esquema es el siguiente:

```
KNN(inicio,tamaño,k,solucion)

clase:=0
vmax:=0
vecinos[k]
distanciaVecinos[k],distancia:=0
i:=0
j:=0
solucion:=0

desde p:=0 hasta k:
        vecinos[p]:= i
        distanciaVecino[p] = distancia(tabla[i] , tabla[ejemplo], tamaño)
        sivmax<distanciaVecino[p]:
                vmax = distanciaVecino[p]
                pos_max = p
        p++, i++

desde p:=k hasta tamaño:
        dist := distanciaVecino[p] = distancia(tabla[i] , tabla[ejemplo], tamaño)
        sivmax>dist:
        vecinos[pos_max]:=p
        distanciaVecino[pos_max] := dist
        pos_max := pos_max(distanciaVecino, k)
        vmax := distanciaVecino[pos_max]

desde i:=0 hasta k
        clase := clase + clases_tabla[vecinos[i]]

si clase > k*0.5
```

```
        solucion := 1

return solucion
```

- Algoritmo de búsqueda local/método de mejora: Se considerará la búsqueda local (BL) que sigue el enfoque del primer mejor vecino

```
LocalSearch (ini,int tam ,int k,  aux, iteracion)

   mejora:=false
   ind := Randint(), cont := 0
   humbral := coste(ini, tam, k, aux)

   desde i=0 hasta i<n_caracteristicas && i<200 && !mejora

      aux.s[i]:=!aux.s[i];
      aux.tasa_clas := coste(ini, tam, k, aux);
      iteracion++

     si aux.tasa_clas > humbral
         aux.tasa_red = calcularTasaRed()
         mejora := true

         i++

      return aux;
```

Se detendrá la ejecución del algoritmo bien cuando no se encuentre mejora en todo el entorno o bien cuando se hayan evaluado 200 vecinos distintos en la ejecución.

- Algoritmo genético AGG: cuya descripción y componentes verémos a continuación.

El tamaño de la población del AGG será de 10 cromosomas. Las probabilidades de cruce y mutación serán 0,7 y 0,001 (por gen) en ambos casos. El criterio de parada consistirá en realizar 15000 evaluaciones de la función objetivo, incluidas por supuesto las de la BL.

3. Estructuras de los algoritmos

Para implentar el algoritmo memético hemos tenido previamente que implementar el algoritmo genético AGG, pero previamente tenemos que definir sus operadores:

- Mutación. Permite que un gen del cromosoma varíe, es decir seleccione una característica o la des-seleccione. Esto permite añadir diversidad a la población.

```
mutacion( s, gen)

    s.s[gen]:=!s.s[gen]            //Operador flip
```

- Seleccion de padres, selecciona de la población dos padres a reproducirse

```
seleccionPadres(padre1, padre2){

    si padre1.tasa_clas > padre2.tasa_clas
        return sol_copy(padre1)
    si no
        return sol_copy(padre2)
```

- Cruce, selecciona dos puntos al azar que determinan 3 subcadenas las cuales determinan que parte heredan la descendencia de cada padre

```
cruce( padre1, padre2, hijo1,  hijo2){

    rand1 := Randint(0, n_caracteristicas-2)
    rand2 := Randint(rand1, n_caracteristicas-1)

    desde int i:=0 hasta i<n_caracteristicas;

        si i<rand1
            hijo1.s[i] := padre1.s[i]
            hijo2.s[i] := padre2.s[i]
        si no
            si i>=rand1 y i<rand2
                hijo1.s[i] := padre2.s[i]
                hijo2.s[i] := padre1.s[i]
            si no

                hijo1.s[i] := padre1.s[i]
                hijo2.s[i] := padre2.s[i]
```

```
        i++
```

Una vez unidos el AGG con la Busqueda Local, obtendremos el siguiente algoritmo memético:

```
MemeticoA ( inicio, tam, k, n_interaciones){


    n_cruces := 0.7 * (10 * 0.5)
    n_padres := n_cruces * 2
    n_mutaciones := (10 * n_caracteristicas)
    p_mejor:=0
    p_peor:=0
    seleccionado := false
    peor := 100.0
    contador := 0


    poblacion[10];
   padres[10];
    mejor;
   mejor.tasa_clas = 0.0;

   //Inicializamos la poblacion actual

   desde i := 0 hasta i < 10

      poblacion[i] := generarSolRandom()
      poblacion[i].tasa_clas := coste(inicio, tam, k, poblacion[i])

         i++


    iteracion:=10;

   mientras iteracion<n_interaciones



      //Seleccion del mejor individuo de la poblacion

   desde  i := 0 hasta i < 10

      si poblacion[i].tasa_clas>mejor.tasa_clas
         mejor := sol_copy(poblacion[i])
         p_mejor := i
      i++
```

```
  //seleccion de los padres

desde  i = 0 hasta i < 10

  r1 := Randint(0,9)
  r2 := Randint(0,9)

   padres[i] := seleccionPadres(poblacion[r1],poblacion[r2])

    si r1==p_mejor || r2==p_mejor
        seleccionado := true
 i++




//Cruce de padres

 desde  i = 0 hasta i < n_padres

    cruce(padres[i],padres[i+1],poblacion[i],poblacion[i+1])
    poblacion[i].tasa_clas := coste(inicio,tam,k,poblacion[i])
    poblacion[i+1].tasa_clas := coste(inicio,tam,k,poblacion[i+1])
    iteracion+:=2;
     i+:=2


 //Rellenamos la poblacion vacia con los padres

desde i = n_padres hasta i < 10

   poblacion[i] := padres[i]


  //Realizamos las mutaciones

desde i = 0 hasta i < 10

   desde j = 0 hasta j < n_caracteristicas-1

      //Probabilidad de mutacion de cada gen de 0.001 es decir 1/1000
      int r := Randint(0,1000);
      if(r==1)
        mutacion(poblacion[i],j)
   i++


//Introducimos al mejor individuo anterior por el peor
```

```
    si !seleccionado
        desde i = 0 hasta i < 10

          si poblacion[i].tasa_clas < peor
             peor := poblacion[i].tasa_clas;
             p_peor := i;
          i++

          poblacion[p_peor] := mejor;

//APLICAMOS la BL cuando lleva 10 generaciones

      si contador%10==0

          desde i = 0 hasta i < 10

              poblacion[i] := LocalSearch(inicio,tam ,k, poblacion[i],iteracion);
          i++




   s := sol_copy(mejor);
   return s;
```

En el pseudocódigo tenemos la descripcion del primer AM. Los dos restantes consistirán en cambiar cuando aplicamos la BL. Resultando 3 algoritmos meméticos:

1. AM-(10,1.0): Cada 10 generaciones, se aplica la BL sobre todos los cromosomas de la población.
2. AM-(10,0.1): Cada 10 generaciones, se aplica la BL sobre un subconjunto de cromosomas de la población seleccionado aleatoriamente con probabilidad p LS igual a 0.1 para cada cromosoma.
3. AM-(10,0.1mej): Cada 10 generaciones, aplicar la BL sobre los 0.1·N mejores cromosomas de la población actual (N es el tamaño de ésta).

Greedy

Para valorar los resultados de los algoritmos meméticos, los compararemos con el obtenido por el algoritmo greedy, que genera soluciones formadas por las características mas prometedoras. Selecciona primero la característica que mas nos explica de todas, después se evalúa cual de las siguientes restantes proporciona mayor tasa de clasificación, y la que mas tenga se añade a la solución.

4. Explicación del desarrollo de la práctica

Para el desarrollo de esta práctica he reutilizado código para cargar las bases de datos y funciones valoración, e implementado los algoritmos meméticos desde cero. Todo el desarrollo ha sido realizado en Sublime Text, compilando con g++ desde Ubuntu 14.10.

Para ejecutar la práctica es necesario mantener la estructura de las tablas guardadas en los directorios tal y como aparecen en este zip, simplemente será necesario abrir un terminal y ejecutar las siguientes instrucciones:

g++ -c /home/ahisa/Escritorio/MH/memeticos.cpp -o /home/ahisa/Escritorio/MH/memeticos.o
g++ -o /home/ahisa/Escritorio/MH/memeticos /home/ahisa/Escritorio/MH/memeticos.o

Para cambiar las opciones de que algoritmos probar, es necesario abrir el fichero memeticos.cpp, y en la función main comentar/descomentar la solucion según dicho criterio, ej:

// sol = MemeticoC(ini_entrena,num_elementos,3,15000);
// sol = MemeticoC(ini_test,num_elementos,3,15000);

5. Experimentos y análisis de los resultados:

knn

	ozono							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	94	565216	98	611115	0	277315	100	0
Particion 1-2	100	0	97	222221	0	358954	97	826088
Particion 2-1	96	739128	98	611115	0	243816	97	826088
Particion 2-2	97	826088	98	611115	0	241827	96	739128
Particion 3-1	94	565216	98	611115	0	240981	100	0
Particion 3-2	100	0	97	222221	0	357717	97	826088
Particion 4-1	100	0	95	833336	0	467686	98	913040
Particion 4-2	92	391304	98	611115	0	237711	100	0
Particion 5-1	94	565216	98	611115	0	239836	100	0
Particion 5-2	100	0	97	222221	0	361865	97	826088
tasa_red media:	97	916672						
tasa_clas media	97	65224						

	sonar							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	100	0	98	333336	0	261511	100	0
Particion 1-2	100	0	98	333336	0	222295	100	0
Particion 2-1	100	0	98	333336	0	227112	100	0
Particion 2-2	100	0	98	333336	0	223723	100	0
Particion 3-1	100	0	98	333336	0	222565	100	0
Particion 3-2	100	0	98	333336	0	220435	100	0
Particion 4-1	100	0	98	333336	0	221281	100	0
Particion 4-2	100	0	98	333336	0	221391	100	0
Particion 5-1	100	0	98	333336	0	221479	100	0
Particion 5-2	100	0	98	333336	0	221810	100	0
tasa_red media:	98	333328						
tasa_clas media	100	0						

	spam							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	93	913040	94	736839	1	836730	93	478264
Particion 1-2	76	521736	91	228073	2	691984	96	521744
Partición 2-1	90	0	92	982452	2	244638	94	782608
Particion 2-2	92	608696	92	982452	2	300207	94	782608
Particion 3-1	93	43480	91	228073	2	675075	94	782608
Particion 3-2	90	0	94	736839	1	797311	95	652176
Particion 4-1	94	782608	91	228073	2	681271	93	913040
Partición 4-2	60	869564	92	982452	2	238215	96	956520
Particion 5-1	94	782608	94	736839	1	846869	93	478264
Particion 5-2	93	43480	92	982452	2	244137	96	86952
tasa_red media:	92	982445						
tasa_clas media	87	956520						

AM1

AM-(10,1.0): Cada 10 generaciones, se aplica la BL sobre todos los cromosomas de la población.

	spam							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	93	913040	57	894737	##	400747	94	347832
Particion 1-2	92	608696	35	87719	##	768013	92	608696
Particion 2-1	87	826088	52	631580	##	959403	95	217392
Particion 2-2	95	652176	50	877193	##	29276	95	652176
Particion 3-1	96	86952	50	877193	##	837375	93	43480
Particion 3-2	93	43480	50	877193	##	991910	93	43480
Particion 4-1	90	434784	49	122807	##	433379	95	217392
Particion 4-2	93	913040	50	877193	##	235033	93	913040
Particion 5-1	91	304352	49	122807	##	342525	94	782608
Particion 5-2	95	217392	57	894737	##	190620	95	217392
tasa_red media:	50	526314						
tasa_clas media	93	0						
	ozone							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	93	478264	51	388889	30	711502	100	0
Particion 1-2	100	0	47	222221	31	890017	98	913040
Particion 2-1	96	739128	50	0	31	874942	97	826088
Particion 2-2	97	826088	56	944443	32	122480	96	739128
Particion 3-1	94	565216	44	444443	32	156279	100	0
Particion 3-2	100	0	51	388889	32	343881	98	913040
Particion 4-1	100	0	56	944443	33	305218	98	913040
Particion 4-2	94	565216	44	444443	32	19264	100	0
Particion 5-1	94	565216	43	55557	32	2536	100	0
Particion 5-2	100	0	47	222221	31	511731	98	913040
tasa_red media:	49	305553						
tasa_clas media	97	173912						
	sonar							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	100	0	50	0	33	618744	100	0
Particion 1-2	100	0	41	666668	33	634306	100	0
Particion 2-1	100	0	43	333332	37	710536	100	0
Particion 2-2	100	0	48	333332	34	337440	100	0
Particion 3-1	100	0	55	0	33	547979	100	0
Particion 3-2	100	0	61	666668	33	366386	100	0
Particion 4-1	100	0	51	666668	33	555837	100	0
Particion 4-2	100	0	51	666668	33	623811	100	0
Particion 5-1	100	0	50	0	33	465405	100	0
Particion 5-2	100	0	53	333332	33	327572	100	0
tasa_red media:	50	666664						
tasa_clas media	100	0						

AM2

2. AM-(10,0.1): Cada 10 generaciones, se aplica la BL sobre un subconjunto de cromosomas de la población seleccionado aleatoriamente con probabilidad p LS igual a 0.1 para cada cromosoma.

	spambase							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	91	304352	57	894737	##	639448	94	347832
Particion 1-2	94	347832	54	385963	##	613927	94	347832
Particion 2-1	88	695648	57	894737	##	632065	95	652176
Particion 2-2	94	782608	54	385963	##	865599	94	782608
Particion 3-1	94	347832	49	122807	##	108913	93	478264
Particion 3-2	93	478264	43	859650	##	393616	93	478264
Particion 4-1	90	434784	45	614037	##	712994	95	217392
Particion 4-2	95	652176	56	140350	##	379172	95	652176
Particion 5-1	90	869560	49	122807	##	465023	95	217392
Particion 5-2	95	217392	49	122807	##	88952	95	217392
tasa_red media:	51	754387						
tasa_clas media	92	913048						
	sonar							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	100	0	50	0	34	11362	100	0
Particion 1-2	100	0	48	333332	33	409269	100	0
Particion 2-1	100	0	50	0	33	445058	100	0
Particion 2-2	100	0	55	0	33	568355	100	0
Particion 3-1	100	0	36	666668	33	690054	100	0
Particion 3-2	100	0	43	333332	33	340847	100	0
Particion 4-1	100	0	45	0	32	824577	100	0
Particion 4-2	100	0	46	666668	33	784171	100	0
Particion 5-1	100	0	50	0	33	171732	100	0
Particion 5-2	100	0	51	666668	33	106203	100	0
tasa_red media:	47	666664						
tasa_clas media	100	0						
	ozone							
	%_clas	,	%_red	,	T	Tasa Entrenamiento		
Particion 1-1	93	478264	51	388889	31	42912	100	0
Particion 1-2	100	0	43	55557	31	577439	98	913040
Particion 2-1	96	739128	52	777779	31	810696	97	826088
Particion 2-2	96	739128	48	611111	30	693345	96	739128
Particion 3-1	94	565216	41	666668	30	324395	100	0
Particion 3-2	100	0	51	388889	30	179813	98	913040
Particion 4-1	100	0	50	0	30	342364	98	913040
Particion 4-2	92	391304	47	222221	31	108926	100	0
Particion 5-1	95	652176	62	500000	30	235490	100	0
Particion 5-2	100	0	51	388889	31	548789	97	826088
tasa_red media:	50	4						
tasa_clas media	96	956520						

3. AM-(10,0.1mej): Cada 10 generaciones, aplicar la BL sobre los 0.1·N mejores cromosomas de la población actual (N es el tamaño de ésta).

spambase

	%_clas	,	%_red	,		T	Tasa Entrenamiento		
Particion 1-1	92	608696	57	894737	##		795081	93	478264
Particion 1-2	90	869560	40	350876	##		973055	95	217392
Particion 2-1	90	0	49	122807	##		68553	94	347832
Particion 2-2	91	304352	49	122807	##		167615	95	652176
Particion 3-1	96	521744	45	614037	##		568037	93	43480
Particion 3-2	89	130440	56	140350	##		774119	96	956520
Particion 4-1	90	869560	47	368420	##		261226	94	782608
Particion 4-2	91	739128	50	877193	##		202249	95	652176
Particion 5-1	90	869560	49	122807	##		648205	96	86952
Particion 5-2	92	173912	57	894737	##		310745	94	782608
tasa_red media:	50	350876							
tasa_clas media	91	608696							

ozone

	%_clas	,	%_red	,		T	Tasa Entrenamiento		
Particion 1-1	93	478264	51	388889		30	691068	100	0
Particion 1-2	100	0	47	222221		30	621478	98	913040
Particion 2-1	96	739128	47	222221		31	323065	97	826088
Particion 2-2	97	826088	52	777779		30	538117	96	739128
Particion 3-1	94	565216	48	611111		31	90833	100	0
Particion 3-2	100	0	54	166668		31	155860	98	913040
Particion 4-1	100	0	56	944443		31	724633	98	913040
Particion 4-2	93	478264	59	722221		30	883412	100	0
Particion 5-1	93	478264	47	222221		31	77899	100	0
Particion 5-2	100	0	50	0		30	336400	96	739128
tasa_red media:	51	527779							
tasa_clas media	96	956528							

sonar

	%_clas	,	%_red	,		T	Tasa Entrenamiento		
Particion 1-1	100	0	50	0		33	943562	100	0
Particion 1-2	100	0	56	666668		33	811109	100	0
Particion 2-1	100	0	58	333332		33	915145	100	0
Particion 2-2	100	0	41	666668		33	414965	100	0
Particion 3-1	100	0	48	333332		33	204863	100	0
Particion 3-2	100	0	41	666668		32	977071	100	0
Particion 4-1	100	0	56	666668		33	417386	100	0
Particion 4-2	100	0	55	0		32	925702	100	0
Particion 5-1	100	0	51	666668		33	477171	100	0
Particion 5-2	100	0	55	0		33	328357	100	0
tasa_red media:	51	500000							
tasa_clas media	100	0							

Como podemos ver todos los algoritmos nos dan tasas de clase entorno al 90% o superior, esto evidentemente refleja algún problema y no es real, probablemente en el knn. Aparte de dicho problema podemos ver como el primer algoritmo memetico converge más rápidamente, gracias a aplicar la BL a toda la población, mientras que el último algoritmo solo aplica la BL a un individuo, haciendo que converja mucha mas lentamente.

Además podemos ver como aun el knn dando resultados elevadamente falsos, la base de datos spam es la que alcanza peores resultados, esto puede ser debido a que las muestras en dicha base de datos están mas dispersas, por lo que obtiene peores resultados.