Práctica 3.b

Genéticos

Selección de características

Algoritmos considerados: AGG,AGE

Luis Suárez Lloréns
DNI: 75570369-M
luissuarez@correo.ugr.es
5º Doble Grado Ingeniería Informática y Matemáticas
Grupo de Prácticas: 3

Índice

1. Descripción del problema **2**

2. Consideraciones generales **3**

3. Explicación de los algoritmos **5**
3.1. Genético general 5
3.2. AGE 6
3.3. AGG 6

4. Algoritmo de comparación **7**

5. Procedimiento **8**

6. Análisis de resultados **9**

7. Referencias **10**

1. Descripción del problema

Cuando se trata un problema de clasificación o de aprendizaje automático, nunca sabemos a priori los datos que nos serán útiles. Es más, añadir datos innecesarios puede incluso empeorar el rendimiento de nuestro clasificador.

El fin del problema de selección de características es tratar de tomar un conjunto de datos de calidad, que nos permita afrontar el posterior aprendizaje de una manera más rápida y con menos ruido en los datos.

Pese a no ser este un problema directamente de clasificación, vamos a necesitarla para valorar la calidad de una solución del problema. Por tanto, necesitamos un clasificador sencillo para esta tarea. Utilizaremos el clasificador k-nn — para ser más concreto, 3-nn —, y trataremos de encontrar las características con las que mejor clasifique un conjunto de prueba.

Entonces, usando el clasificador 3-nn, nuestro objetivo va a ser maximizar la función:

$$\frac{Instancias\, bien\, clasificadas}{Total\, de\, instancias}$$

2. Consideraciones generales

En esta sección veremos los componentes en común de los diferentes algoritmos.

- **Representación:** Array binario con la misma longitud que el número de datos.
- **Función objetivo:** Porcentaje de acierto del clasificador 3-nn. Para evaluarlo Tendríamos que hacer lo siguiente:
  - Tomar las columnas que nos indique la solución.
  - Entrenar el clasificador con los datos de entrenamiento y sus etiquetas.
  - Clasificar los datos de test y comprobar si coinciden con sus verdaderas etiquetas.

  Además, para poder ver lo bien que clasifica al propio conjunto de entrenamiento, realizamos "Leave One Out", que consiste en, para cada dato del conjunto de entrenamiento, quitarlo de los datos de entrenamiento, clasificarlo y ver si hemos acertado al clasificar o no.
- **Generación de soluciones aleatorias:** Se guardan en la máscara los resultados de muestrear una binomial que nos devuelve valores verdadero y falso con la misma probabilidad.
- **Selección:** Dado n el tamaño de la población objetivo. Se seleccionan $2n$ individuos de la población anterior aleatoriamente. Se comparan en orden — primero con segundo, tercero con cuarto— y devolvemos los individuos que sean mejores de cada comparación.
- **Operación de cruce de una pareja:** Si ambos padres tienen el mismo gen —verdadero o falso— sus hijos tienen ese gen. Si los padres tienen un gen distinto, aleatoriamente un hijo recibirá un valor y el otro hijo el otro.
- **Operación de cruce:** Dado p la probabilidad de cruce. Se tomarán los primeros $2n$ padres y se cruzan con el operador de cruce de parejas en orden –primero con segundo, tercero con cuarto,...—. n es el número de cruces, que se calcula como $n = \lfloor \frac{n_{padres} * p}{2} \rfloor$.

- **Operación de mutación:** Sea p la probabilidad de mutación y g el número de genes en total de la población a mutar. Entonces se deciden mutar $\lfloor p * g \rfloor$ genes. Generamos aleatoriamente que genes se van a mutar, y para cada uno de ellos, cambiamos su valor al opuesto.

3. Explicación de los algoritmos

Los algoritmos AGE y AGG se diferencian en los parámetros de probabilidad de cruce, tamaño de la población de hijos a generar y probabilidad de mutación, además de la función de reemplazamiento de la población.

Los parámetros son los indicados en el documento de prácticas.

Vamos a ver un esquema general del algoritmo genético, y luego las funciones de reemplazamiento de cada modelo concreto. Se usan las funciones explicadas en la sección anterior para cruce, mutación y selección.

3.1. Genético general

- Generar una población inicial aleatoria, evaluarla y ordenarla. Se considera como población actual.
- Mientras no se supere el límite de evaluaciones:
  - Seleccionar los padres de la nueva generación.
  - Cruzar los padres.
  - Mutar el resultado anterior. El resultado es la nueva generación.
  - Evaluamos y ordenamos la nueva generación.
  - Reemplazamos y ordenamos la generación actual usando la función de reemplazamiento.
- Devuelve la mejor solución de la generación actual y el valor de su función objetivo.

3.2. AGE

Función de reemplazamiento:

- Comparamos las dos peores soluciones de la generación actual y las dos soluciones de la nueva generación. Devolvemos en generación actual las $n - 2$ mejores soluciones de la generación actual y las 2 mejores soluciones de las comparadas antes.

3.3. AGG

Función de reemplazamiento:

- Miramos si la mejor solución de la nueva generación supera a la mejor solución de la generación actual. Si es así, la nueva generación pasará a ser la generación actual. Si no es así, cambiamos el peor valor de la nueva generación por el mejor de la generación actual, y el resultado es la nueva generación actual.

4. Algoritmo de comparación

El algoritmo de comparación es el algoritmo greedy SFS, que consiste en:

- Partimos de la solución completamente a 0.
- Hasta que no encontremos mejora, realizar:
  - Para cada bit que sea 0, ponerlo a uno y calcular la función objetivo.
  - Tomamos la mejor de todas, y si mejora a la solución que teníamos, hacemos permanente el cambio y seguimos iterando.

5. Procedimiento

Para la realización de las prácticas, he usado el lenguaje Python 3 y varios paquetes adicionales.

Usamos scikit para la creación de particiones y para normalizar los datos.

Para el uso del clasificador 3-nn, tanto para el cálculo del acierto del test como para Leave One Out, utilizamos una implementación en CUDA realizada por Alejandro García Montoro, pues la mejora de tiempo es sustancial con respecto al k-nn implementado en scikit, que sólo usa la CPU del ordenador.

Para la realización de los algoritmos, se utilizó Python 3 de manera directa, basandose en los códigos de la asignatura. Con el fin de poder empezar la ejecución del programa desde una partición intermedia, cada partición tiene una seed asociada en vez de usarse una única seed para todo el fichero. Las seeds son, por orden: $12345678, 90123456, 78901234, 456789012, 34567890$.

Para usar el programa, hay que ejecutar la orden python3 main.py BaseDatos Heurística Semilla. Si no se introduce semilla, se utilizan las usadas para obtener los resultados.

6. Análisis de resultados

Los resultados se encuentran al final del documento.

Hemos obtenido unos resultados algo sorprendentes. Por un lado, el genético generacional, pese a realizar un número muy grande de evaluaciones, se encuentra en los rangos de acierto de SFS, uno de las heurísticas de comparación. Esto nos indica que el genético no funciona muy bien en su estado actual. Más adelante se realizará una ampliación de estos modelos genéticos utilizando una búsqueda local para mejorar a la población, y realizando esto se espera un mejor comportamiento.

De todas maneras, vamos a analizar las diferencias entre AGG y AGE. Ambos tienen unos tiempos de ejecución parecidos, pues están limitados por el número de ejecuciones de la función de coste. Sin embargo, AGE tiene resultados bastante mejores. Esto puede deberse al comportamiento de AGE, que es más selectivo y tiene menos variabilidad de la población que AGG, haciendo que explore menos espacio en el espacio de soluciones, pero de una manera más exhaustiva. Por tanto podríamos esperar una gran mejora de AGG al ser utilizado en una metaheurística memética, y no tanta mejora en el AGE.

En resumen, las heurísticas genéticas básicas no funcionan del todo bien para el problema. De tener que utilizar una, seleccionaríamos AGE, pues obtiene unas soluciones de más calidad al explotar más los óptimos.

7. Referencias

Aparte de la documentación de la asignatura, he usado las páginas de referencia del software usado para desarrollar las prácticas:

- Python: `https://docs.python.org/3/`
- Numpy y Scipy: `http://docs.scipy.org/doc/`
- Scikit-learn: `http://scikit-learn.org/stable/documentation.html`
- K-nn CUDA: `https://github.com/agarciamontoro/metaheuristics`

Cuadro 1: KNN

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	96.13	96.14	0.0	0.02	66.67	70.0	0.0	0.04	62.5	65.98	0.0	0.14
P 1-2	96.84	95.77	0.0	0.02	65.56	85.56	0.0	0.04	61.86	61.46	0.0	0.12
P 2-1	96.83	95.79	0.0	0.02	75.0	69.44	0.0	0.04	64.58	63.4	0.0	0.13
P 2-2	95.44	96.13	0.0	0.02	71.67	75.56	0.0	0.04	65.46	63.02	0.0	0.12
P 3-1	97.18	96.49	0.0	0.02	75.0	74.44	0.0	0.04	61.98	61.86	0.0	0.13
P 3-2	97.54	94.72	0.0	0.02	68.89	75.0	0.0	0.04	64.43	65.1	0.0	0.12
P 4-1	95.42	97.54	0.0	0.02	65.56	71.67	0.0	0.04	64.06	63.92	0.0	0.13
P 4-2	97.54	95.42	0.0	0.02	68.33	73.33	0.0	0.04	60.82	64.06	0.0	0.12
P 5-1	95.42	95.79	0.0	0.02	62.78	72.78	0.0	0.04	62.5	65.98	0.0	0.13
P 5-2	96.14	96.83	0.0	0.02	69.44	76.67	0.0	0.04	65.46	60.42	0.0	0.12
Medias	96.44	96.06	0.0	0.02	68.89	74.44	0.0	0.04	63.36	63.52	0.0	0.12

Cuadro 2: SFS

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	95.77	95.09	50.0	0.26	67.78	67.22	48.89	1.38	66.67	65.46	48.56	76.73
P 1-2	97.19	94.72	46.67	0.27	70.0	79.44	50.0	1.37	67.01	64.06	46.04	33.5
P 2-1	96.83	94.74	53.33	0.39	75.56	68.89	40.0	2.13	68.23	63.92	44.24	56.79
P 2-2	97.54	95.07	43.33	0.51	73.89	73.89	42.22	3.58	69.07	63.54	50.36	66.12
P 3-1	95.77	97.19	40.0	0.25	76.11	76.11	44.44	2.85	69.79	60.31	53.24	47.85
P 3-2	97.54	95.07	33.33	0.74	74.44	73.89	60.0	3.25	66.49	64.06	50.36	33.19
P 4-1	96.48	96.84	36.67	0.49	65.56	73.89	53.33	1.36	66.15	60.82	46.4	47.68
P 4-2	98.6	95.77	43.33	0.64	72.78	72.78	35.56	3.62	68.04	67.19	58.27	47.3
P 5-1	96.13	94.74	43.33	0.39	68.33	72.78	54.44	2.69	66.15	62.37	50.36	67.2
P 5-2	96.84	96.13	56.67	0.5	73.89	75.0	50.0	1.35	72.16	64.58	57.55	39.65
Medias	96.86	95.53	44.66	0.44	71.83	73.38	47.88	2.35	67.97	63.63	50.53	51.60

Cuadro 3: AGE

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	98.59	95.44	53.33	117.53	77.78	66.67	50.0	193.38	76.56	64.95	51.08	973.7
P 1-2	98.95	93.66	40.0	130.46	75.56	81.11	51.11	229.99	72.68	60.42	51.8	849.44
P 2-1	98.59	93.68	60.0	138.56	82.22	68.89	42.22	225.71	72.4	63.4	46.04	1096.86
P 2-2	98.6	96.13	56.67	125.87	81.11	73.89	43.33	209.65	74.23	61.46	50.36	872.29
P 3-1	97.89	96.84	43.33	159.41	82.22	74.44	47.78	269.8	75.52	59.28	54.68	937.9
P 3-2	98.6	96.13	56.67	138.8	77.78	75.56	43.33	260.64	75.26	61.46	48.2	834.29
P 4-1	96.83	93.68	46.67	116.65	76.67	73.33	54.44	189.66	74.48	60.31	47.84	1002.45
P 4-2	99.3	92.96	56.67	158.38	79.44	71.67	52.22	193.13	71.13	66.15	50.36	879.83
P 5-1	97.89	94.39	50.0	130.12	75.56	75.56	57.78	187.17	72.4	63.92	52.52	1088.45
P 5-2	98.6	97.89	60.0	108.95	79.44	74.44	43.33	225.09	76.29	62.5	47.48	888.86
Medias	98.38	95.08	52.33	132.47	78.77	73.77	48.55	218.42	74.09	62.38	50.03	942.40

Cuadro 4: AGG

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	97.54	95.44	53.33	128.93	70.56	66.67	50.0	225.29	66.15	64.95	51.08	1058.46
P 1-2	96.84	94.37	40.0	128.68	70.56	82.78	46.67	229.64	65.46	62.5	48.92	891.37
P 2-1	97.18	93.68	60.0	130.96	75.0	68.89	42.22	238.31	65.62	63.4	46.04	1065.89
P 2-2	97.19	96.13	40.0	128.88	73.33	75.0	54.44	219.1	65.98	62.5	46.4	904.13
P 3-1	96.48	96.84	43.33	130.23	77.22	74.44	47.78	239.9	64.58	59.28	54.68	1067.36
P 3-2	96.84	96.13	46.67	134.18	72.22	76.11	48.89	242.76	68.56	64.06	47.84	915.87
P 4-1	95.42	93.68	46.67	115.59	71.11	73.33	54.44	230.0	66.67	60.31	47.84	1029.91
P 4-2	97.89	91.9	56.67	123.28	72.78	73.33	50.0	227.29	64.95	61.98	54.68	903.67
P 5-1	96.83	94.39	50.0	117.73	68.33	75.56	57.78	211.71	65.62	63.92	52.52	1042.13
P 5-2	97.54	95.77	56.67	124.14	74.44	75.0	50.0	227.49	67.53	63.02	52.52	896.55
Medias	96.97	94.83	49.33	126.26	72.55	74.11	50.22	229.14	66.11	62.59	50.25	977.53