Práctica 5.b

Algoritmos meméticos

Selección de características

Algoritmos considerados: AM1, AM2, AM3

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

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

Nuestro algoritmo memético utiliza el esquema del algoritmo genético AGG de la práctica 3.

Los parámetros son los indicados en el documento de prácticas.

Vamos a ver un esquema general del algoritmo memético, y luego las funciones de reemplazamiento del modelo AGG y el método de mejora. Se usan las funciones explicadas en la sección anterior para cruce, mutación y selección.

3.1. Algoritmo Memético

- 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.
  - Si la población debe ser mejorada —en este caso, el número de la generación es múltiplo de 10— se mejora con la búsqueda local.
- Devuelve la mejor solución de la generación actual y el valor de su función objetivo.

3.2. Reemplazamiento 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.

3.3. Método de mejora

Nuestro método de mejora consta de un algoritmo de búsqueda local primero el mejor, limitado a una sola iteración. Aquí tenemos el pseudocódigo del mismo:

- Elige los índices de los genes a mejorar. Si la selección es con elitismo, coge a los n mejores genes. Si no, toma n genes distintos aleatorios.
- Para cada gen:
    - Para cada vecino generado aleatoriamente:
        - Evalúa el vecino.
        - Aumenta el número de evaluaciones.
        - Si el vecino es mejor que nuestro gen, aplica la mejora al gen y pasa al siguiente gen a mejorar.
- Devuelve el número de evaluaciones.

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. Los nombres de las Heurísticas son AM1, AM2 y AM3, cada uno con los parámetros indicados en la práctica.

Cuadro 1: Resumen

	Wdbc			Libras			Arrhythmia		
	% train	% red	tiempo	% train	% red	tiempo	% train	% red	tiempo
3-NN	96.44	0.0	0.02	68.89	0.0	0.04	63.36	0.0	0.12
SFS	96.86	44.66	0.44	71.83	47.88	2.35	67.97	50.53	51.6
AGG	96.97	49.33	126.26	72.55	50.22	229.14	66.11	50.25	977.53
AM-1	98.13	50.33	133.39	77.11	47.88	212.9	73.21	50.28	997.518
AM-2	98.17	49.99	125.83	77.44	51.55	210.87	74.303	48.95	917.45
AM-3	98.06	49.00	137.78	78.10	51.66	200.10	73.31	50.57	956.98

6. Análisis de resultados

Los resultados se encuentran al final del documento.

Hemos dejado los resultados de la práctica 3, en concreto los relativos al modelo AGG, por ser la base de nuestro algoritmo memético. Podemos observar que los 3 algoritmos meméticos que hemos implementado supera, ampliamente, el resultado del AGG. Esto nos muestra la gran utilidad que tiene la mejora de los genes mediante búsqueda local.

Viendo los resultados, parece que el mejor de los 3 algoritmos es AM-2, pues gana en 2 de las 3 bases de datos. Este algoritmo realizaba la mejora a un sólo gen de la población, de manera aleatoria, cada 10 generaciones. Vamos a compararlo con los otros dos algoritmos para intentar comprender su mejor funcionamiento.

Con respecto a AM-1, que mejoraba todos los genes, la respuesta podría ser un mejor equilibrio entre exploración y exploración. Realizar tantas evaluaciones en la mejora de las soluciones hace que perdamos bastante en el número de generaciones que conseguimos explorar. Por tanto, la exploración de AM-1 es menor, por lo que AM-2 puede encontrar zonas mejores que explotar, y con el paso de las generaciones, conseguir buscar el mínimo.

Con respecto a AM-3, que mejoraba también solo un gen, pero siempre el mejor, la razón parece más simple, pero está oculta en el funcionamiento del mismo. Mejorar el mejor, parece una buena idea, conseguimos mejorar la que, si paráramos en ese mismo instante, sería la mejor solución. Pero si ya es máxima, todas las mejoras —y las evaluaciones asociadas a la misma—

se vuelven inútiles, haciendo que nuestro algoritmo en general utilice menos evaluaciones para la búsqueda al eliminar todas las evaluaciones perdidas en intentos de mejora imposibles. Por otro lado, esa mejora constante de la mejor solución hace que si una solución es muy prometedora, AM-3 va a obtener el máximo de la zona seguro. Esto puede hacer que AM-3 mejore a AM-2 en la base de datos Libras.

En resumen, AM-2 parece el mejor algoritmo, pero estos 3 experimentos nos indican lo importante que es el ajuste de la mejora en un algoritmo memético. Factores como el número de genes a mejorar, la profundidad de la mejora o el número de generaciones entre mejora y mejora, entre otros, afecta al rendimiento del algoritmo y además, esto es algo que varía según el problema. Por tanto, es una labor muy importante el ajustar bien todos estos parámetros para conseguir un buen resultado con esta técnica que como hemos podido ver, mejora el funcionamiento de los algoritmos genéticos.

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 2: 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 3: 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 4: AM-1

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	97.89	95.44	53.33	134.82	75.0	66.67	50.0	234.99	73.44	64.95	51.08	994.81
P 1-2	98.25	95.77	53.33	131.03	74.44	80.56	44.44	210.45	72.16	65.62	47.48	1024.17
P 2-1	98.24	93.68	60.0	117.73	78.89	68.89	42.22	240.83	73.96	63.4	46.04	1024.52
P 2-2	97.89	94.01	30.0	143.7	79.44	73.89	53.33	188.87	75.26	62.5	52.52	831.44
P 3-1	97.89	96.84	43.33	190.93	83.89	74.44	47.78	204.23	73.44	59.28	54.68	1164.88
P 3-2	97.89	94.01	40.0	99.82	75.56	76.11	38.89	224.16	72.16	62.5	51.8	976.58
P 4-1	97.18	93.68	46.67	112.98	75.0	73.33	54.44	213.97	73.44	60.31	47.84	964.65
P 4-2	99.65	92.25	60.0	123.46	75.56	73.33	45.56	204.52	71.65	65.1	52.16	940.34
P 5-1	98.24	94.39	50.0	135.74	74.44	75.56	57.78	184.04	71.88	63.92	52.52	1110.32
P 5-2	98.25	93.66	66.67	143.78	78.89	76.11	44.44	222.94	74.74	63.02	46.76	943.47
Medias	98.13	94.37	50.33	133.39	77.11	73.88	47.88	212.9	73.21	63.06	50.28	997.518

Cuadro 5: AM-2

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	98.24	95.44	53.33	142.62	75.0	66.67	50.0	236.19	74.48	64.95	51.08	943.15
P 1-2	98.25	97.18	43.33	95.28	76.11	81.67	51.11	191.51	71.65	65.1	48.2	836.15
P 2-1	98.24	93.68	60.0	113.63	82.22	68.89	42.22	219.2	75.52	63.4	46.04	972.64
P 2-2	97.89	94.37	43.33	125.32	79.44	73.33	53.33	257.11	77.84	56.25	44.24	885.35
P 3-1	97.89	96.84	43.33	190.99	84.44	74.44	47.78	198.85	73.96	59.28	54.68	1045.36
P 3-2	98.6	96.13	56.67	124.98	77.22	73.33	54.44	215.67	73.2	62.5	50.0	833.47
P 4-1	97.18	93.68	46.67	97.96	72.78	73.33	54.44	186.44	74.48	60.31	47.84	932.43
P 4-2	98.95	92.96	60.0	118.62	75.0	73.33	55.56	198.53	71.65	67.71	48.92	869.1
P 5-1	97.89	94.39	50.0	125.17	74.44	75.56	57.78	176.31	73.96	63.92	52.52	968.56
P 5-2	98.6	95.07	43.33	123.77	77.78	75.56	48.89	228.96	76.29	59.38	46.04	888.37
Medias	98.17	94.97	49.99	125.83	77.44	73.61	51.55	210.87	74.303	62.28	48.95	917.45

Cuadro 6: AM-3

	Wdbc				Libras				Arrhythmia			
	% train	% test	% red	tiempo	% train	% test	% red	tiempo	% train	% test	% red	tiempo
P 1-1	97.89	95.44	53.33	135.71	75.56	66.67	50.0	217.95	75.52	64.95	51.08	975.7
P 1-2	97.89	94.72	50.0	137.64	78.33	81.11	63.33	177.13	72.68	60.42	50.0	808.97
P 2-1	98.24	93.68	60.0	117.2	80.0	68.89	42.22	224.4	73.44	63.4	46.04	1029.67
P 2-2	98.6	94.72	40.0	113.45	79.44	72.78	46.67	203.47	75.26	64.58	50.36	937.19
P 3-1	97.89	96.84	43.33	194.07	84.44	74.44	47.78	203.15	72.92	59.28	54.68	1093.29
P 3-2	97.89	95.07	30.0	180.36	76.67	73.33	51.11	206.5	72.16	62.5	51.8	887.7
P 4-1	97.18	93.68	46.67	100.74	74.44	73.33	54.44	187.67	72.4	60.31	47.84	1030.21
P 4-2	98.95	94.72	60.0	146.58	79.44	72.78	47.78	176.01	73.2	63.02	49.64	872.89
P 5-1	97.89	94.39	50.0	120.79	74.44	75.56	57.78	177.34	71.88	63.92	52.52	1046.08
P 5-2	98.25	95.42	56.67	131.34	78.33	74.44	55.56	227.41	73.71	64.58	51.8	888.12
Medias	98.06	94.86	49.00	137.78	78.10	73.33	51.66	200.10	73.31	62.69	50.57	956.98