Fundamentos de Aprendizaje Automático 2016/2017

PRÁCTICA Nº 3

1. Objetivo

El objetivo de la práctica es determinar un conjunto de reglas de clasificación mediante algoritmos genéticos. En líneas generales, los algoritmos genéticos siguen dos enfoques respecto a la forma de codificar las reglas dentro de una población de individuos:

- Enfoque "*Cromosoma = Regla*". En esta aproximación cada individuo codifica una sola regla. A este enfoque corresponden dos propuestas:
  - En la propuesta que se denomina *Michigan*, la solución final será la población final o un subconjunto de la misma.
  - En la propuesta conocida como *Aprendizaje de Reglas Iterativo*, la solución del algoritmo genético es el mejor individuo, pero la solución global estará formada por los mejores individuos de una serie de ejecuciones sucesivas.
- Enfoque "*Cromosoma = Base de Reglas*". En esta aproximación, que se conoce con el nombre de *Pittsburgh*, cada individuo representa un conjunto de reglas.

Para llevar a cabo la implementación del algoritmo genético orientado a la tarea de clasificación, se seguirá en este caso la aproximación de *Pittsburgh*, es decir, cada cromosoma representará un conjunto de reglas completo.

2. Tareas

En esta práctica se puede seguir la planificación temporal que cada uno considere más apropiada. No obstante, debe tenerse en cuenta que la implementación de un algoritmo genético supone realizar fundamentalmente las siguientes tareas:

1. Determinar el esquema de representación más adecuado para codificar cada una de las soluciones.
2. Determinar el conjunto de operadores genéticos y su operativa concreta.
3. Definir la función de adaptación o función de fitness.

2.1 Esquema de representación

Para tareas de clasificación, la forma natural de expresar conceptos es mediante un conjunto de reglas. Cada regla tendría una estructura como la siguiente:

IF <condición$_1$> AND <condición$_2$> …. AND <condición$_n$> THEN <conclusión>

Donde cada <condición$_i$> en el lado izquierdo puede incluir también el operador OR para combinar condiciones disyuntivas. Por su parte, el lado derecho <conclusión> representa el concepto que se quiere clasificar a partir de los ejemplos que coincidan con la parte izquierda de la regla. Cada condición se puede representar mediante una cadena binaria en la que un 1 representa que el valor del atributo o característica está presente y un 0 representa la ausencia del valor del atributo o característica. Teniendo además en cuenta que en el conjunto de entrada puede haber N atributos o características, se puede representar cada regla como una cadena binaria de longitud fija, cuya longitud depende del tipo del atributo (nominal, ordinal, etc.). De esta forma, si una característica o atributo tiene *k* valores, necesitaremos *k* bits, uno por cada valor, para representar la característica. Por ejemplo, si un atributo del conjunto de entrada representa los días de diario (*weekdays*), podría estar especificado en el algoritmo genético como el patrón 0111110, que se referiría a cualquier día excepto el domingo y el sábado. Tomando el conjunto completo de atributos, si existieran 3 atributos, con 7, 4 y 2 valores posibles respectivamente, la representación de una regla como una cadena binaria arbitraria de longitud fija sería la siguiente:

atr_1	atr_2	atr_3
0110010	*1001*	*01*

El significado de esta regla correspondería a:

IF (atr_1=$valor_2$ OR atr_1=$valor_3$ OR atr_1= $valor_6$) AND
(atr_2=$valor_1$ OR atr_2=$valor_4$) AND
(atr_3=$valor_2$) THEN <conclusión>

El lado derecho, <conclusión>, especificaría la clase o concepto al que pertenece el ejemplo y se puede representar de la misma forma, es decir, mediante una cadena binaria. **Para facilitar la implementación, en esta práctica se considerará que los datos a clasificar contienen únicamente valores nominales[1] y dos clases, es decir, la conclusión vendrá representada por un único bit.**

Para terminar con el esquema de representación solo falta comentar que, al seguir la aproximación de *Pittsburgh*, cada individuo de la población corresponde a una base de reglas completa, por lo que cada individuo se representará mediante una cadena binaria de longitud variable que incluirá un conjunto no ordenado de reglas de longitud fija como las descritas anteriormente. El número de reglas de cada individuo no está restringido en principio, siendo éste un parámetro bastante relevante que deberá analizarse con detalle.

[1] Si se quisiera experimentar con conjuntos de datos que contengan atributos continuos, una opción es discretizar estos atributos en intervalos. Para ello, se puede utilizar la función "cut" del paquete pandas de Python: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.cut.html

2.2 Operadores genéticos

Gracias al esquema de representación anterior, los operadores genéticos que pueden aplicarse tienen pocas restricciones. Estos operadores serán el cruce y la mutación. En cuanto al cruce se puede producir en cualquier parte con la única restricción de que debe mantener la semántica de las reglas, es decir, si en un padre se elige el punto de cruce en el límite de una regla, el punto de cruce del otro padre debería estar en el mismo sitio. Estas limitaciones tienen como objetivo mantener la semántica en los hijos producidos. Por su parte la mutación no está restringida y se implementa simplemente como una variación a nivel de bit. No obstante, se puede realizar cualquier otra implementación de estos operadores si se considera necesario o se mejoran los resultados.

2.3 Función de adaptación o función de fitness

La función que permite evaluar los individuos es una de las cuestiones más importantes a resolver en el diseño de un algoritmo genético. Para el propósito de clasificación, en esta práctica vamos a centrarnos exclusivamente en el rendimiento obtenido al clasificar. Según este criterio, el *fitness* de un individuo (que corresponderá a una base de datos de reglas) se calcula probando la base de datos de reglas sobre el conjunto de datos de entrenamiento. De forma numérica:

Fitness del individuo i = % de aciertos sobre los datos de entrenamiento

3. Detalles adicionales

Para mantener la filosofía de las prácticas anteriores, el algoritmo de clasificación genético debe implementar un aprendizaje en ***modo batch***, que significa dividir el conjunto de datos en *train* y *test*. Durante el entrenamiento se evoluciona la población de individuos para obtener el mejor, es decir, la base de datos de reglas que define el clasificador, utilizando los datos de *train*. Posteriormente, en la clasificación se prueba este conjunto de reglas para clasificar los datos de *test*, calculando la tasa de acierto/error de la misma manera que en las prácticas anteriores. De esta forma el algoritmo genético se ejecuta bajo el mismo esquema que los demás clasificadores.

A la hora de clasificar, se prueba el dato de test con las reglas evolucionadas. Si alguna regla del conjunto puede dispararse se asocia la condición de la regla como clase estimada. En caso de que se disparen varias reglas que den lugar a diferentes condiciones, habrá que decidir la estrategia a tomar y comentarla en el apartado 1 del notebook a entregar. En caso de que ninguna de las condiciones de las reglas se dispare, puede devolverse una clase por defecto o devolver fallo. No obstante, la devolución de fallo no deberá implicar la no ejecución del algoritmo genético, si no que deberá tratarse de forma apropiada en el código.

Los parámetros que deben tenerse en cuenta a la hora de implementar el algoritmo son los siguientes:

- Probabilidad de cruce: 60%
- Probabilidad de mutación: 0.1%
- Proporción de elitismo: 5%
- Selección proporcional al fitness
- Tamaño de la población: Probar con 10, 100, 200 y 500 individuos
- Condición de terminación: Probar con 100, 500 y 1000 épocas o generaciones

4. Conjuntos de datos

El algoritmo genético se aplicará a la clasificación de los siguientes conjuntos de datos:

- Conjunto **Tic-Tac-Toe** que ya se utilizó en la práctica 1 (http://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame).
- Conjunto **example2.data** de la práctica 1. Problema de clasificación de dos clases y con dos atributos nominales.
- Conjunto **ejemplo5.data**. Problema de clasificación de dos clases y con dos atributos nominales. Disponible en Moodle junto con el enunciado de la práctica.
- Conjunto **ejemplo6.data**. Problema de clasificación de dos clases y un atributo nominal. Disponible en Moodle junto con el enunciado de la práctica.
- Conjunto **titanic.data** con información sobre los pasajeros del barco. Este conjunto de datos se ha utilizado para tratar de predecir la supervivencia de un pasajero en base a otra serie de variables como edad, sexo, o la clase del billete. Ver por ejemplo: https://www.kaggle.com/c/titanic. Este *dataset* está disponible en Moodle junto con el enunciado de la práctica y contiene las siguientes variables:
  - **pclass**: Clase del pasajero. (1 = primera clase; 2 = segunda clase; 3 = tercera clase)
  - **sex**: Género del pasajero
  - **age(5 years)**: Edad del pasajero discretizada en intervalos de 5 años.
  - **survived(class)**: Supervivencia (0 = No; 1 = Sí). Esta es la clase a predecir.

5. Lenguaje y Diseño

El lenguaje a utilizar para el desarrollo de la práctica será Python. Utilizar la estructura de clases y métodos propuesta en la práctica 1 para la implementación de la clase *AlgoritmoGenetico* con los métodos entrenamiento y clasifica. **Es importante mostrar mensajes aclarativos durante la salida del programa.**

NOTA SOBRE LA EJECUCIÓN: Los algoritmos evolutivos suelen ser costosos computacionalmente, por lo que los tiempos de ejecución de esta práctica pueden ser

más largos de lo habitual, especialmente cuando se trabaje con poblaciones grandes durante un número elevado de generaciones.

6. Fecha de entrega y entregables

Semana del 5 al 9 de Diciembre de 2016. La entrega debe realizarse antes del comienzo de la clase de prácticas correspondiente. Se deberá entregar un fichero comprimido .zip con nombre **`FAAP3_<grupo>_<pareja>.zip`** (ejemplo `FAAP3_1461_1.zip`) y el siguiente contenido:

1. **Ipython Notebook (.ipynb)** con las instrucciones necesarias para realizar la clasificación de los conjuntos descritos en el apartado 4 y el correspondiente análisis de resultados. El Notebook debe estructurarse para contener los siguientes apartados:

Apartado 1	Breve descripción de algunos detalles de la implementación. Solo es necesario especificar los siguientes aspectos: a) Generación de la población inicial con especial indicación del número de reglas por individuo consideradas b) Mecanismo de cruce implementado c) Mecanismo de mutación implementado d) Mecanismo de clasificación implementado
Apartado 2	Resultados de la clasificación para los siguientes casos: a) Tamaño de población = 10 ; Generaciones = 100 b) Tamaño de población = 10 ; Generaciones = 500 c) Tamaño de población = 200 ; Generaciones = 100 d) Tamaño de población = 200 ; Generaciones = 500 e) Tamaño de población = 500 ; Generaciones = 100 f) Tamaño de población = 500 ; Generaciones = 500 Además de los errores de clasificación, deberá incluirse en el notebook el código para mostrar el conjunto de reglas correspondiente al mejor individuo y los diccionarios asociados al conjunto de datos.
Apartado 3	Análisis de resultados: importancia del número de reglas, tamaño de la población, generaciones, tasas de cruce y mutación. En aquellos casos donde sea posible, se deberá de dar una interpretación al conjunto de reglas obtenido.
Apartado 4	Solo para la fase de entrenamiento, evolución en forma de gráfica: a) Del fitness del mejor individuo de la población b) Del fitness medio de la población

2. **Ipython Notebook exportado como html.**

3. Código Python (**ficheros .py**) necesario para la correcta ejecución del Notebook

4. **Ficheros de datos** en la ruta adecuada para que el Notebook ejecute (rutas relativas).

7. Referencias

1. W. Spears, K. De Jong. Using Genetic Algorithms for Supervised Concept Learning (Artículo disponible en moodle como material complementario para la práctica)