Herramientas de la Inteligencia Artificial

Práctica final PG: La Programación Genética como método de clasificación

En esta práctica usaremos la PG como método de clasificación. Cada grupo trabajará en al menos un dominio UCI, abordándolo desde dos enfoques distintos. El objetivo es comparar ambos enfoques y tratar por todos los medios de conseguir tasas de clasificación comparables a las de la literatura.

Problemas de Clasificación

Los problemas de clasificación suelen consistir en una base de datos compuesta por un número N de ejemplos o instancias que están escritos por un número P de atributos y que pertenecen a una clase. El problema es tratar de aprender la forma de distinguir los ejemplos de las distintas clases. Por ejemplo, un banco intenta reducir sus riesgos al conceder prestamos. Para ello usa una base de datos de los clientes que han tenido prestamos hasta la fecha. De cada uno de ellos tiene una serie de atributos: edad, estado civil, salario, número de hijos y valor de sus posesiones inmuebles; y cada uno de ellos pertenece a una de dos clases posibles: ha devuelto el préstamo (clase 0) o se ha convertido en moroso (clase 1). Al banco le interesaría mucho hallar alguna manera de predecir si un cliente va a pertenecer a la clase 0 o a la 1 a partir de los atributos antes descritos. De esta manera optimizará enormemente su sistema de concesión de créditos. Esto es un problema de clasificación.

La base de datos UCI (UCI Machine Learning Repository) es un repositorio público de problemas de clasificación. En UCI podemos encontrar una gran cantidad de problemas reales y sintéticos que sirven para probar cómo de buenos son los distintos métodos de clasificación. Además, la gran ventaja de UCI es que cientos de grupos de investigación de todo el mundo utilizan sus conjuntos de datos, de manera que es muy sencillo comprar tu método con los mejores resultados que se han encontrado hasta el momento.

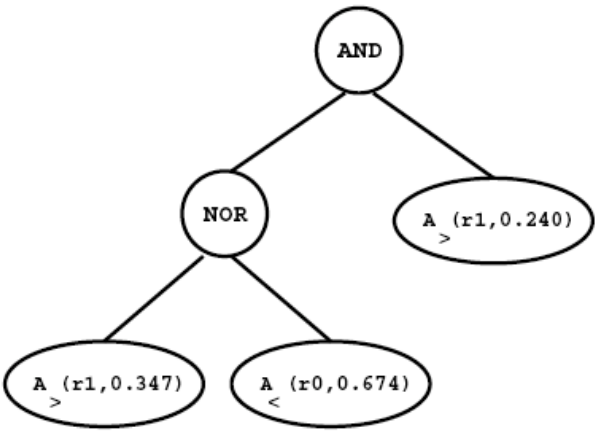
La PG para clasificación

Hay numerosas publicaciones que utilizan la PG en problemas de clasificación (podeis buscar en Google para meteros en ambiente). El método más básico consiste en utilizar un terminal por cada atributo del problema, mezclarlos entre si usando funciones aritméticas simples (+, -, *, /, etc.) y evaluar el árbol de PG para cada ejemplo del dataset. Si el resultado es mayor que un valor determinado, el ejemplo se clasifica como clase 1, si no, como clase 0. El fitness se calcula midiendo el error de clasificación: si tengo 1000 ejemplos y clasifico bien 756, entonces tengo un error de clasificación del 24.4% (o un acierto del 75.6%).

Este enfoque tiene una serie de problemas muy obvios: Primero, supone que todos los atributos del problema son numéricos (por ejemplo, en el problema del banco, sería muy difícil multiplicar por el estado civil); Segundo, supone que el problema sólo tiene dos clases (y puede tener muchas más); Y tercero, plantea un grave problema de coherencia al operar entre si variables que no tienen nada que ver: en el ejemplo del banco, por ejemplo, ¿qué sentido tendría restarle el salario bruto anual a la edad del cliente? Son variables que no tienen nada que ver y no se deberían de mezclar.

Representación atómica

Una alternativa es utilizar representación atómica: los atributos del problema se convierten a variables booleanas en los terminales del árbol, en lugar de esperar hasta ejecutar el árbol completo, y en el resto del árbol se opera sólo con variables booleanas. Un ejemplo:



El terminal más a la derecha, por ejemplo, devuelve true si el atributo 1 (aqui simbolizado por r1) es mayor que 0.347, y false si no. En el resto del árbol se opera con los resultados booleanos y finalmente el árbol devuelve devuelve un booleano que dice a qué clase pertenece el ejemplo (true = clase 0, false = clase 1).

Esta figura sólo es un esquema. En realidad no nos interesa programar terminales como los de la figura porque tendríamos que programar cientos y el espacio de búsqueda sería inabarcable. Nosotros en su lugar aprovecharemos que ProGen proporciona PG tipada: usaremos terminales de tipo numérico (double o int, según el problema), después los combinaremos mediante funciones de comparación (>, <, ==, max, min, etc.) que toman dos números y devuelven un boolean. Por encima usaremos funciones booleanas que combinan los resultados de las funciones de comparación.

Métodos de evaluación por pesos (SAW)

Otra forma contrastada de mejorar el comportamiento de la PG en problemas de clasificación es utilizar un sistema de pesos, en el que cada ejemplo contribuye de manera diferente al cálculo del fitness. Antes cada ejemplo bien clasificado era un acierto,

y cada ejemplo mal clasificado, un fallo. Con el sistema de pesos, lo que hacemos es asignarle más importancia a los ejemplos que son más difíciles de clasificar. La tarea de asignación de pesos es muy compleja y requiere un conocimiento muy profundo del problema, por eso, nosotros usaremos el sistema SAW (Stepwise Adaptation of Weights). El fitness se calculará ahora de esta forma:

$$f(x) = \sum_{r \in D} w_r \cdot error(x, r)$$

Donde Wr es el peso del ejemplo r, y error(x, r) una función muy sencilla que devuelve 1 si el ejemplo se ha clasificado bien y 0 si no es así.

Todos los pesos Wr se inicializarán a 1. Después de un número de generaciones, estos pesos serán actualizados añadiéndoles un factor de corrección deltaW a todos aquellos ejemplos que hayan sido mal clasificados por el mejor individuo encontrado hasta ese momento. El algoritmo sería algo así:

On-line weight update mechanism
set initial weights (thus fitness function f)
while not EA is finished **do**
for the next T_p fitness evaluations **do**
let the EA run with this f
end for
redefine weights in f and recalculate fitness of individuals
end while

Objetivos de la práctica y metodología

Cada grupo de prácticas escogerá o se le asignará una base de datos del repositorio UCI. Además, se le asignará una de las dos mejoras propuestas (SAW o representación atómica). El grupo debe implementar el problema de clasificación en ProGen siguiendo el esquema más básico que le sea posible (cada atributo un terminal, funciones aritméticas básicas, fitness basado en el error de clasificación, etc.) y además, tendrá que modificar ProGen para incluir la mejora que le haya sido asignada. El grupo deberá probar ambas opciones y compararlas mediante una serie de experimentos y un proceso de tuneado de parámetros. En la memoria de la práctica describirá los experimentos y discutirá si es mejor usar el método simple o el mejorado a la luz de sus resultados.

Por otro lado, el grupo debe tener siempre en mente cuál es el mejor resultado que se ha conseguido en ese problema (podéis conseguir esa información en la web de UCI, en la descripción de los dominios), y tratar por todos los medios de acercarse lo más posible a ese resultado. Todos los grupos que consigan resultados que se asemejen a los

oficiales de UCI tienen asegurada una muy buena nota (siempre que se haya conseguido a base de trabajo y no por suerte, y que se pueda demostrar que los datos están bien).

El día del examen de la asignatura, cada grupo tiene que entregar una memoria en la que se expliquen las decisiones tomadas durante el diseño del problema, qué experimentos se han llevado a cabo, qué resultados se han obtenido y qué conclusiones se extraen de dichos resultados.