Predicción de módulos defectuosos como un problema de optimización multiobjetivo

M. Martínez-Ballesteros[1], J.C. Riquelme[1], R. Ruiz[2] and D. Rodríguez[3]

[1] Universidad de Sevilla
{maria,riquelme}@us.es
[2] Universidad Pablo de Olavide, Sevilla
robertoruiz@upo.es
[3] Universidad de Alcalá
daniel.rodriguezg@uah.es

Resumen La dificultad de aplicar técnicas de análisis de datos al problema de la calidad del software radica principalmente en dos razones: la ausencia de datos generalistas y de herramientas específicas. En este trabajo exponemos los primeros pasos de una iniciativa para paliar estos inconvenientes. Con respecto al primero, hemos trabajado con dos conjuntos de datos públicos que han sido tratados de forma conjunta para poder lograr modelos más generales. Para el segundo propósito se ha aplicado un algoritmo multiobjetivo que mediante reglas cuantitativas establezca cuáles son los límites empíricos de los atributos que miden la complejidad a partir de los cuales la probabilidad de error aumenta significativamente e incluso la posibilidad de medir ese aumento.

Keywords. Reglas, Predicción defectos software, Multiobjetivo

1. Introducción

La calidad del software permanece como un asunto importante dentro de la comunidad de la ingeniería del software. Existen muchas definiciones de calidad del software, pero en este contexto nos referimos a la confiabilidad del software, que normalmente se define como la probabilidad de que un sistema esté libre de fallos por un periodo de tiempo especificado en un entorno determinado. Una forma de mejorar la confiabilidad del software y de dirigir el esfuerzo de las pruebas es mediante métricas estáticas capaces de predecir los módulos defectuosos.

Existe una gran variedad de técnicas de predicción de defectos que utilizan métodos estadísticos, y más recientemente, técnicas de minería de datos, quizás impulsado por el hecho de hacer públicos conjuntos de datos de proyectos software reales. Un denominador común en este tipo de conjuntos de datos es la gran diferencia entre los módulos defectuosos y los que no presentan defectos, y la consiguiente dificultad de aplicar muchas de las técnicas de minería de datos existentes. Por ejemplo, si el número de módulos sin defectos sobrepasa con creces a los defectuosos en un 90 %, un algoritmo que siempre prediga que un módulo no es defectuoso obtendría una exactitud muy alta. Como resultado, muchos algoritmos de minería obtendrían modelos sesgados que no tendrían en consideración a la clase minoritaria. Para solventar este problema en este trabajo, se ha aplicado un algoritmo evolutivo multiobjetivo denominado MOQAR

cuyo objetivo es la obtención de un conjunto óptimo de reglas mediante la optimización simultánea de diferentes medidas.

2. Estudio realizado

2.1. Background

En predicción de fallos, la minería de datos tiene como objetivo la generación de modelos de clasificación que predigan si un módulo software será defectuoso en base a las métricas de los datos históricos de proyectos. Existen múltiples representaciones posibles del aprendizaje basado en modelos, sin embargo, cuando la salida va a ser interpretada directamente por usuarios finales debe prevalecer la legibilidad y la comprensibilidad de la representación, por eso en este trabajo empleamos las reglas como representación, al ser más simples e intuitivas.

En el campo de la minería de datos, las reglas de asociación (AR) se utilizan para describir las relaciones existentes entre los atributos o propiedades que describen los datos [1]. En este trabajo, nos centraremos en encontrar AR que relacionen las medidas de complejidad de un módulo software con la propiedad de módulos defectuosos. Como métricas de bondad de las reglas, se utilizará la cobertura como medida para calcular el porcentaje de módulos defectuosos que se explican mediante una regla y la confianza para medir la probabilidad de que al aplicar esa regla efectivamente el módulo sea defectuoso.

2.2. Algoritmo multiobjetivo

En este trabajo aplicamos un algoritmo evolutivo multiobjetivo llamado MOQAR [4] que sigue el esquema del algoritmo NSGA-II [3] capaz de encontrar AR cuantitativas en conjuntos de datos con atributos continuos evitando la discretización previa de los mismos. La búsqueda de los intervalos más apropiados se lleva a cabo mediante un proceso evolutivo multiobjetivo. En dicho proceso, los intervalos se ajustan para encontrar un conjunto de reglas de calidad a partir de la optimización simultánea de varias funciones objetivo.

La optimización simultánea de diferentes objetivos en conflicto se lleva a cabo mediante la evolución de la población basada en la ordenación de soluciones en frentes de dominancia o frentes de Pareto [5]. Se considera que una solución a domina a otra b, si cada objetivo de a es mejor o igual que el correspondiente objetivo de b y en al menos un objetivo es mejor. El primer frente se compone de soluciones no dominadas por ninguna otra solución de la población (primer frente de Pareto), el segundo se compone de soluciones dominadas por una solución y así sucesivamente. En cada frente, los individuos se ordenan de mayor a menor según la distancia de las distintas funciones objetivo de él mismo a sus vecinos [3]. Una vez alcanzada la condición de parada, el algoritmo multiobjetivo devuelve las soluciones no dominadas que se encuentran en el primer frente de Pareto.

Tabla 1. Reglas obtenidas para el conjunto de datos arAll con defect = TRUE

ID	*Regla*	*Cobertura(%)*	*Confianza (%)*
1	$unOp \in$ [16-29]	58,33	50,72
2	$unOp \in$ [17-31] $\wedge$ $dd \in$ [0-1.4]	53,33	61,54
3	$bl \in$ [43-142] $\wedge$ $dd \in$ [0-1.4]	38,33	62,16
4	$bl \in$ [43-142] $\wedge$ $hv \in$ [61-139] $\wedge$ $dd \in$ [0-1.4]	30	85,71
5	$tl \in$ [114-477] $\wedge$ $mcc \in$ [9-25] $\wedge$ $dd \in$ [0-1.162] $\wedge$ ncc $\in$ [0.082-0.231]	21,67	92,86
6	$mcc \in$ [9-12] $\wedge$ dd [0-1.162] $\wedge$ fp [0-0.21]	15	100

$unOp$ - unique_operators, dd - design_density, bl - blank_loc, hv - halstead_vocabulary, mcc - multiple_condition_count, fp - formal_parameters, tl - total_loc

Tabla 2. Reglas obtenidas para el conjunto de datos D'Ambros con bugsBoolean = 1

ID	*Regla*	*Cobertura(%)*	*Confianza (%)*
1	$cbo \in$ [14-153] $\wedge$ $rfc \in$ [78-1,156]	34,23	50,34
2	$cbo \in$ [13-144] $\wedge$ $noc \in$ [0-11] $\wedge$ $rfc \in$ [157-1,156] $\wedge$ $wmc \in$ [209-972]	6,8	86,57
3	$dit \in$ [1-4] $\wedge$ $noc \in$ [0-5] $\wedge$ $rfc \in$ [163-1,156]	18,76	62,75
4	$dit \in$ [1-4] $\wedge$ $noc \in$ [0-5] $\wedge$ $rfc \in$ [185-1,156]	15,94	69,04
5	$fanOut \in$ [19-61] $\wedge$ $noa \in$ [2-182]	18,87	59,85
6	$fanOut \in$ [17-61] $\wedge$ $nopa \in$ [1-36]	9,38	76,19

cbo - Coupling Between Objects, rfc - Response For Class, noc - No. of Children, wmc - Weighted Method Count, dit - Depth of Inheritance Tree, $fanOut$ - No. of other classes referenced by the class, noa - No. Of Attributes, $nopa$ No. Of Public Attributes

2.3. Experimentación

Entre los conjuntos de datos más estudiados se encuentran el repositorio PROMISE[4] y Bug Prediction Dataset (BPD)[5] creado por D'Ambros et al. [2]. Cada uno de ellos trata con múltiples conjuntos de datos que corresponden a un proyecto software. Para extraer reglas generales que no estén ajustadas a los datos de un solo proyecto software, hemos unido conjuntos de datos de cada grupo que están escritos en el mismo lenguaje, resultando dos conjuntos. El conjunto de datos arAll reúne datos de cinco conjuntos disponibles en el repositorio PROMISE (AR1-AR5), cuyos valores corresponden con métricas estáticas como las de McCabe, de Halstead y líneas de código. Contiene 428 instancias de las cuales 60 corresponden a módulos fallidos. La segunda recopilación de conjuntos es D'Ambros, que contiene datos sobre proyectos software conocidos (Eclipse JDT Core, Eclipse PDE UI, Equinox Framework, Lucene, Mylyn). En este caso, los valores corresponden a métricas de código orientado a objetos siendo el 15,88 % de los módulos defectuosos (853 de 5371).

Con el objetivo de obtener un conjunto amplio de reglas para cada conjunto de datos, MOQAR ha sido ejecutado utilizando diferentes umbrales mínimos de soporte (1 % y 5 %) y confianza (50 %) así como diferentes combinaciones de objetivos a optimizar basados en estas medidas. Posteriormente, un experto del dominio ha seleccionado un conjunto reducido de reglas teniendo en cuenta las que tienen pocos operandos y buenos resultados según las métricas estudiadas.

La Tabla 1 estudia el fichero de datos arAll. Estos datos recogen un total de 428 módulos software de los cuales 60 son fallidos. Así, a priori, la probabilidad de que un módulo sea defectuoso es del 14 %. Si nos fijamos en la primera regla, el valor de la cobertura (58,33 %) indica que esta regla cubre 35 de los 60

[4] `http://openscience.us/repo/`

[5] `http://bug.inf.usi.ch/`

módulos con error. Esto es, la condición *uniq_operator* entre 16 y 29 explica casi el 60 % de los errores y la confianza del 50,72 % indica que aproximadamente la mitad de los módulos que cumplen esa restricción son defectuosos. Por tanto, la probabilidad inicial del 14 % se dispara al 50,72 %. Las siguientes reglas muestran que con condiciones más restrictivas (por ejemplo, añadiendo condiciones sobre *design_density*) la probabilidad de acierto de módulos erróneos se incrementa, aunque lógicamente los módulos cubiertos disminuyen. Por ejemplo, el cumplimiento de la última regla prácticamente asegura que el módulo tendrá errores y eso se cumple en el 15 % de todos los módulos defectuosos.

Para las reglas obtenidas para los datos D'Ambros de la Tabla 2, las métricas seleccionadas como decisivas para que un módulo sea erróneo son principalmente *cbo*, *noc*, *dist* y *rfc*. La probabilidad inicial es 15,8 % y se consigue aumentar a valores entre 50 % y 86 %, con coberturas entre 6,8 % y 34,2 %.

3. Conclusiones y trabajo futuro

En este trabajo se ha pretendido mostrar los primeros resultados de la aplicación de una técnica de optimización multiobjetivo para analizar la relación entre las medidas de complejidad del software y la existencia de errores. La herramienta de análisis empleada es capaz de encontrar reglas que maximizan dos medidas contrapuestas: porcentaje de módulos defectuosos explicados y probabilidad de acierto, ambos en cada regla. Como trabajo futuro pretendemos proporcionar a la herramienta MOQAR un segundo módulo también multiobjetivo, para la automatización de la selección final del conjunto de reglas que optimice no solo las medidas individuales de cada regla sino del conjunto total.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad bajo los proyectos TIN2014-55894-C2, TIN2013-46928-C3-2-R Y TIN2016-76956-C3-3-R.

Referencias

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 207–216 (May 1993)
2. D'Ambros, M., Lanza, M., Robbes, R.: An extensive comparison of bug prediction approaches. In: Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR07). pp. 31–41. IEEE CS Press (2010)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182 –197 (Apr 2002)
4. Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F., Riquelme, J.C.: Improving a multi-objective evolutionary algorithm to discover quantitative association rules. Knowledge and Information Systems 49(2), 481–509 (2016)
5. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)