Descubrimiento de Subgrupos para predecir módulos defectuosos

D. Rodríguez[1], R. Ruiz[2], J.C. Riquelme[3], and J.S. Aguilar–Ruiz[2]

[1] Universidad de Alcalá, Ctra. Barcelona, Km. 31.6, 28871 Alcalá de Henares, Madrid
daniel.rodriguezg@uah.es
[2] Universidad Pablo de Olavide, Ctra. Utrera km. 1, 41013 Sevilla
{robertoruiz,aguilar}@upo.es
[3] Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla
riquelme@us.es

Resumen La aplicación de métodos de Minería de Datos a la Ingeniería del Software tiene una importancia creciente en distintos aspectos del ciclo de vida del software. En este trabajo presentamos una metodología para inducir reglas que nos permitan establecer cuáles son las métricas y los umbrales que caracterizan la aparición de módulos con fallos. Abordamos el problema a partir de modelos de Descubrimiento de Subgrupos (DS) que nos permite buscar patrones sólo para un tipo de dato con alguna propiedad de interés, en nuestro caso, módulos con errores.

Keywords: Predicción de fallos en módulos software, descubrimiento de subgrupos, datos no balanceados, reglas

1. Introduction

La Calidad del Software es una importante área de interés para la comunidad de Ingeniería del Sofware. La fiabilidad del software se puede definir como la probabilidad de que la operatividad de un módulo software esté libre de error en un periodo y entorno determinado.

El uso de técnicas para intentar predecir la aparición de módulos software defectuosos es posible gracias a la aparición de repositorios públicos de proyectos reales, tales como PROMISE o FLOSSMetrics. Sin embargo, hay que tener en consideración características negativas que suelen tener estos datos: desbalanceo, irrelevancia o redundancia e inconsistencia.

El desbalanceo se refiere a que en la mayoría de los conjuntos de datos para predicción de defectos, el número de registros clasificados como error es (muy) minoritario. Esto conlleva que la mayoría de las técnicas de clasificación, al intentar optimizar una tasa de acierto global, generen modelos que desprecian los ejemplos con error, que en este caso son precisamente los que nos interesan. Por otro lado, la presencia de métricas irrelevantes o redundantes perjudica también la fiabilidad de los modelos.

En este trabajo se aplica una técnica de inducción descriptiva denominada Descubrimiento de Subgrupos (DS), que trata de establecer patrones sobre un

subgrupo determinado de datos. Esto es, establecer qué valores de los atributos (métricas en nuestro caso) caracterizan mayoritariamente a una clase de registros (los módulos con errores). DS es por tanto, una técnica muy adecuada para datos no balanceados, ya que puede fijar el interés en los datos de clase minoritaria.

Para representar los subgrupos se ha optado por reglas mediante intervalos de pertenencia. De esta forma, se determinan los umbrales de ciertas métricas que incrementan la probabilidad de que el módulo sea defectuoso. El modelo que hemos denominado EDER-SD se optimiza mediante un algoritmo evolutivo de codificación real, permitiendo usar los valores de las métricas sin discretización previa y seleccionar diferentes medidas de bondad en la función de ajuste.

Los experimentos se han llevado a cabo en varios de los conjuntos de datos del repositorio público PROMISE de proyectos software reales de la NASA. Las reglas encontradas por EDER-SD son modelos fáciles de entender y que pueden ser útiles para gestores de pruebas ya que caracterizan cuáles son las métricas y los valores de éstas que inducen a que un módulo tenga una mayor probabilidad de ser erróneo, y por tanto, deba prestársele una mayor atención. El resto del artículo sigue la siguiente estructura: en la sección 2 se plantean los antecedentes tanto en la aplicación de técnicas de MD a la IS como en DS. En la sección 3 se explica nuestra propuesta que es evaluada en la sección 4 de experimentación.

2. Antecedentes

Recientemente ha habido un auge en el uso de las técnicas de minería de datos para la ingeniería del software y especialmente en la predicción de defectos.

Debido a los problemas de desbalanceo y número de atributos, existen todavía grandes discrepancias con respecto a la evaluación de la bondad de las diferentes técnicas [12]. Por ejemplo, Lessmann et al. [3] compararon múltiples clasificadores y bases de datos del repositorio de la NASA, abogando por el uso de las AUC (Area Under the ROC Curve) como el mejor indicador para comparar los diferentes clasificadores. Otros trabajos en esta línea son [4,5,6]. En [7] aplican CBR (Case Based Reasoning) al problema de desbalanceo.

Hay una serie de trabajos que aplican clasificadores a los mismos conjuntos de la NASA que los usados en este trabajo: Peng et al. [8] utilizan 13 algoritmos de clasificación con 11 medidas de evaluación en 11 conjuntos de datos; Menzies et al. [9] C4.5 y Naïve Bayes; Elish y Elish [10] máquinas de soporte vectoriales (SVM); Peng et al. [11] destaca *boosting CART* y C4.5.

El descubrimiento de subgrupos (DS) es una técnica de MD que trata de caracterizar subgrupos de ejemplos que son estadísticamente diferentes para una propiedad de interés dada [2]. DS es una técnica a caballo entre la predicción y la descripción. La principal diferencia es que los algoritmos de DS sólo se centran en una determinada propiedad de los datos y por tanto normalmente no describe patrones para todas las instancias.

En nuestra solución los subgrupos son representados por reglas de la forma $Cond \rightarrow Class$, donde $Class$ representa los ejemplos que cumplen una determinada propiedad de interés (en nuestro caso módulos erróneos). El antecedente

Cond está formado por una conjunción de expresiones booleanas de pertenencia de la métrica v_i a un intervalo $v_i \in [u_i, l_i]$.

Un aspecto importante del DS es cómo medir la calidad de las reglas que definen el subgrupo. Esta medida es la que servirá para guiar el proceso de búsqueda mediante la comparación de unas reglas con otras. En general, cuando estamos ante un problema de clasificación binaria casi todas las medidas de bondad se definen a partir de la matriz de confusión. La matriz de confusión son cuatro valores denominados TP (true positives), TN (true negatives), FP (false positives) y FN (false negatives). En el caso de DS y teniendo en cuenta que una regla sólo está pensada para módulos erróneos, dos son las medidas primarias: TP sería el numero de los módulos defectuosos clasificados por la regla como tal, es decir, lo que comúnmente se podría denominar aciertos y FP son los módulos sin error que la regla clasificaría incorrectamente como erróneos.

A partir de esas medidas básicas se pueden definir otras medidas de bondad del clasificador como sensibilidad (o recall) igual a TP/(TP+FN), especificidad (specificity) que se calcula como TN/(FP+TN), tasa de acierto (accuracy) igual a (TP+TN)/N o confianza (confidence o precision) calculada como TP/(TP+FP).

Provenientes de los modelos de reglas de asociación, el DS adopta otras medidas de bondad como soporte = TP/N o cobertura = (TP+FP)/N. Otras medidas más elaboradas son WRAcc que se calcula como cobertura×(confianza-(TP+FN)/N) o interés (lift) igual al cociente entre confianza y soporte.

Todas estas medidas tienen diverso interés, según el objetivo en la búsqueda de las subgrupos. Si el interés es caracterizar subgrupos de módulos erróneos que recojan mayoritariamente sólo módulos maximizando TP y minimizando FP debemos optimizar medidas como la confianza aunque probablemente eso baje otras medidas como cobertura o especificidad. Por el contrario, si queremos caracterizar subgrupos grandes donde la probabilidad de encontrar módulos erróneos sea mucho mayor que en el conjunto total pero sin importarnos en demasía los errores FP, entonces podemos optimizar medidas como sensibilidad.

3. Propuesta

En este trabajo proponemos EDER-SD (Evolutionary Decision Rules for Subgroup Discovery), un algoritmo evolutivo de codificación real para caracterizar mediante reglas clases minoritarias. Para implementarlo se ha partido del algoritmo HIDER [1], un AE de cubrimiento secuencial que produce un conjunto jerárquico de reglas de decisión para clasificación. En concreto EDER-SD hereda de HIDER la representación de los individuos donde un vector de valores reales representan el intervalo de pertenencia de la expresión booleana para cada atributo, y se han introducido las siguientes modificaciones en HIDER:

EDER-SD no genera reglas para todas las clases, sólo para la clase minoritaria. A partir de un ejemplo de esta clase seleccionado aleatoriamente, EDER-SD genera una regla que lo cubra construyendo un intervalo de valores para cada atributo que contenga los valores del ejemplo seleccionado.

Cuadro 1. Conjuntos de datos.

Datos	# *inst*	*No-def*	*Def*	% *def*	*Dupl*	*Inconst*	*Leng*
CM1	498	449	49	9.83	56	1	C
KC1	2,109	1,783	326	15.45	897	20	C++

Cada vez que una regla es generada HIDER quita del conjunto de entrenamiento los ejemplos cubiertos por esta regla, condición que da lugar a conjunto jerárquico de reglas. Por el contrario EDER-SD no elimina completamente esos ejemplos, sino que los penaliza. Para ello, EDER-SD añade pesos a los ejemplos de entrenamiento. Estos pesos se inicializan a uno y cada vez que una regla se genera, los ejemplos cubiertos decrementan su peso. Este peso es usado en la función de bondad de forma que los ejemplos ya cubiertos valen menos y las siguientes reglas tienden a cubrir otros ejemplos. Este mecanismo da lugar a una estructura de reglas que hemos llamado piramidal, en el que la primera regla tiene pocas condiciones y por tanto tiene un mayor soporte pero baja precisión. Conforme se generan nuevas reglas se añaden más condiciones que decrecen el soporte pero incrementan la precisión.

Finalmente la función de fitness es totalmente diferente. HIDER es un clasificador y por tanto, su función de bondad está basada exclusivamente en maximizar el valor de TP para el conjunto final de reglas. EDER-SD evalúa las reglas de manera individual (aunque influidas por las anteriores debido a la actualización de pesos del apartado anterior) por lo que se ha optado por medidas de bondad como WRAcc o interés, más apropiadas para un objetivo descriptivo.

4. Experimentación

En este trabajo se pretende analizar el comportamiento de nuestra propuesta con los datos CM1 y KC1 disponibles en el repositorio PROMISE [13] para predecir módulos defectuosos. Estos conjuntos de datos se originaron a partir de proyectos llevados a cabo por la NASA[4]. La tabla 1 muestra el número de instancias, el número de módulos con y sin defectos y su porcentaje, inconsistencias (igual valor para todos los atributos pero distinta clase) y lenguaje de programación en el que fueron escritos los módulos.

Como se puede ver en la tabla 2 en relación con el conjunto de datos CM1, la ratio entre módulos con y sin defectos para la primera regla es entorno al 26 % (22/84). Aunque parezca un balor bajo, hay que resaltar que el porcentaje de desbalanceo es del 10 %, con sólo 49 módulos defectuosos de 498 casos. La segunda y tercera regla cubren menos módulos que la primera pero la ratio entre módulos de una y otra clase se incrementa al 38 % y 43 % respectivamente.

Las reglas de la 4 a la 6 muestran el efecto de decrementar el peso de las instancias ya cubiertas por el algoritmo evolutivo, añadiendo un nueva condición a la regla precedente y mostrando un efecto piramidal. La regla 4 sólo considera

[4] `http://mdp.ivv.nasa.gov/`

Cuadro 2. Selected EDER-SD Rules for the CM1 dataset

#	*Rule*	# *Def*	# *Non Def*
1	$6 \leq v(g) \wedge 35 \leq uniqueOpnd \wedge 64 \leq totalOpnd$	22	62
2	$82 \leq LoC \wedge 22 \leq uniqueOp$	13	21
3	$82 \leq LoC \wedge 22 \leq uniqueOp \wedge 190 \leq totalOpnd$	12	16
4	$71 \leq LoC$	17	27
5	$71 \leq LoC \wedge 22 \leq uniqueOp$	16	25
6	$71 \leq LoC \wedge 22 \leq uniqueOp \wedge 190 \leq totalOpnd$	12	18

Cuadro 3. Selected EDER-SD Rules for KC1 dataset

#	*Rule*	# *Def*	# *Non Def*
1	$93 \leq LoC$	40	33
2	$93 \leq LoC \wedge 17 \leq uniqOp$	39	29
3	$4 \leq iv(g) \wedge 69 \leq totalOpnd$	76	54
4	$4 \leq iv(g) \wedge 69 \leq totalOpnd \wedge LoC \leq 78$	27	10
5	$3 \leq ev(g) \wedge 4 \leq v(g)$	100	159
6	$3 \leq ev(g) \wedge 4 \leq v(g) \wedge 17 \leq uniqOp$	71	72

LoC como condición logrando una precisión del 38 %, lo que significa que casi el 40 % de los módulos con más de 71 *LoC* tendrá defectos. Este umbral está relativamente cerca de los 60 *LoC* sugeridos por la herramienta McCabe IQ[5] y el repositorio de la NASA. En la regla 5 y 6, cuando se añaden nuevas condiciones, la precisión se incrementa (64 % y 67 %) pero el soporte disminuye (41 y 30).

Como se muestra en las dos primeras reglas de la tabla 3, los módulos con un número alto de *LoC* o *uniqOp* tienen una alta probabilidad de tener defectos. Aunque el número de modulos defectuosos cubiertos por las reglas is mayor que el de no defectuosos, ambas reglas tienen un soporte bajo al cubrir un número relativamente pequeño de módulos: 73 para la primera regla con una condición ($93 \leq LoC$) y 68 para la segunda con dos condiciones ($93 \leq LoC \wedge 17 \leq uniqOp$).

EDER-SD también extrae reglas combinando líneas de código y complejidad. Para módulos complejos pero con un número pequeño de *LoC*, la probabilidad de que el módulo sea defectuoso incrementa. Por ejemplo, para la regla 3 en la tabla 3, su ratio es 58 % (76 de 130). Sin embargo, cuando el tamaño del módulo se limita a 78 *LoC*, la ratio de módulos con defectos es 72 %. En otras palabras, la regla 4 establece que módulos pequeños con alta complejidad tienden a ser defectuosos. Los umbrales para las métricas de complejidad $ev(g)$ y $v(g)$ son 3 y 4 respectivamente, logrando un porcentaje de 38 % para módulos con defectos. Añadiendo una nueva restricción ($17 \leq uniqOp$)se incrementa el porcentaje a 50 % con más de 70 módulos cubiertos por la regla.

[5] `http://www.mccabe.com/`

5. Conclusiones

En este trabajo se ha presentado una técnica de Descubrimiento de Subgrupos (DS), EDER-SD (Evolutionary Decision Rules for Subgroup Discovery), que nos permite buscar patrones sólo para un tipo de dato con alguna propiedad de interés, en nuestro caso, módulos con errores. EDER-SD se ha aplicado a dos conjuntos de datos del repositorio PROMISE, y los resultados muestran que las reglas inducidas son capaces de caracterizar subgrupos de modulos con defectos. Además, la representación mediante reglas permite su fácil comprensión y aplicación por los gestores de proyectos.

Como trabajo futuro, nos proponemos sacar reglas que sean más generales y que sirvan para cualquier tipo de módulo software mediante su aplicación a datos provenientes de múltiples fuentes, e intentar mejorar el algoritmo genético mediante una perspectiva multiobjetivo para poder maximizar a la vez varias de las medidas que hemos propuesto.

Referencias

1. Aguilar-Ruiz J., Ramos I., Riquelme J., Toro M.: An evolutionary approach to estimating software development projects. Information and Software Technology 43 (14), 875–882 (2001)
2. Herrera, F., Carmona del Jesus, C. J., González, P., del Jesus, M. J.: An overview on subgroup discovery: Foundations and applications. Knowledge and Information Systems. (Pendiente)
3. Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering **34** 485–496 (2008)
4. Arisholm, E., Briand, L.C., Johannessen, E.B.: A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software **83** 2–17 (2010)
5. Mende, T., Koschke, R.: Effort-aware defect prediction models. In: 14th European Conference on Software Maintenance and Reengineering (CSMR'10) (2010)
6. Koru, A.G., Liu, H.: Building effective defect-prediction models in practice. IEEE Software **22** 23–29 (2005)
7. Khoshgoftaar, T.M., Seliya, N.: Analogy-based practical classification rules for software quality estimation. Empirical Software Engineering **8** 325–350 (2003)
8. Peng, Y., Kou, G., Wang, G., Wang, H., Ko, F.: Empirical evaluation of classifiers for software risk management. International Journal of Information Technology & Decision Making (IJITDM) **08** 749–767 (2009)
9. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering **33** 2–13 (2007)
10. Elish, K., Elish, M.: Predicting defect-prone software modules using support vector machines. Journal of Systems and Software **81** 649–660 (2008)
11. Peng, Y., Wang, G., Wang, H.: User preferences based software defect detection algorithms selection using mcdm. Information Sciences (Pendiente)
12. Zhang, H., Zhang, X.: Comments on "data mining static code attributes to learn defect predictors". IEEE Transactions on Software Engineering 33 (9), 635–637 (2007)
13. Boetticher, G., Menzies, T., Ostrand, T.: Promise repository of empirical software engineering data. http://promisedata.org/ (2007)