Support Vector Machines for Anti-pattern Detection

Abdou Maiga, Nasir Ali, Neelesh Bhattacharya, Aminata Sabané, Yann-Gaël Guéhéneuc, Giuliano Antoniol, Esma Aïmeur

DGIGL et DIRO, École Polytechnique et Université de Montréal, Québec, Canada
E-mails: {abdou.maiga, nasir.ali, n.bhattacharya, a.sabane, giuliano.antoniol}@polymtl.ca, {guehene, aimeur}@iro.umontreal.ca

ASE 2012
August 15, 2013

Introduction

Related Work

Our Approach: SVMDetect

Study Results

Discussions

Conclusion and Future Work

Questions

References

Outline

Introduction

Related Work

Our Approach: SVMDetect

Study Results

Discussions

Conclusion and Future Work

Questions

Introduction

Motivation

- Anti-patterns: **“poor” solutions** to recurring design and implementation problems.
- **Impact** program comprehension, software evolution and maintenance activities [8].
- Important to **detect them early** in software development process, to reduce the maintenance costs

Introduction

Limitations

Current anti-pattern detection approaches have several limitations:

- they require **extensive knowledge** of anti-patterns
- they have **limited precision and recall**
- they cannot be **applied on subsets** of systems.

We propose

- **Apply SVM on subsets** because it considers system **classes one at a time**, not collectively as previous rule-based approaches do.
- To the best of our knowledge, researchers **have not yet studied the potential benefits of using SVM** to detect anti-patterns.

Introduction

Contributions

- **SVMDetect** to detect anti-patterns using SVM
- Use of precision and recall to **compare SVMDetect to DETEX** [13], the best state-of-the-art approach, on 3 programs and 4 anti-patterns.
- The **accuracy** of SVMDetect is **greater** than of DETEX on subsets.
- For whole system, SVMDetect find **more anti-patterns occurrences** than DETEX.

We thus conclude that: a **SVM-based approach** can **overcome** the **limitations** of previous approaches.

Related Work

Smell/Anti-pattern Detection

Many researchers studied anti-patterns detection.

- Alikacem et al. [1] used **meta-model** for representing the source code and **fuzzy thresholds**.
- Langelier et al. [10] used a **visual approach**.
- Marinescu [11] used quantifiable **expression of rules**.
- Sahraoui et al. [7] used **search-based techniques**.
- Moha et al. [13] proposed an approach based on a **set of rules that describes** each anti-pattern.

The works carried out so far suffered from some limitations:

- they have **limited** precision and recall (if reported at all)
- had not been **adopted by practitioners** yet
- cannot be applied on **subsets of systems**
- required **sufficient knowledge** of anti-patterns.

Related Work

SVM Applications

- SVM in several domains for various applications, *e.g.*, bioinformatics [2], object recognition [4].
- SVM is a recent alternative to the classification problems.
- Guihong et al. [3] used SVM, for terms classification.
- SVM used in image retrieval systems by Sethia et al. [12]
- Kim et al. [9] proposed the change classification approach for predicting latent software bugs based on SVM.

To the best of our knowledge, no previous approach used SVM for anti-pattern Detection.

Our Approach: SVMDetect

SVMDetect

SVMDetect is based on Support Vector Machines (SVM) using a polynomial kernel to detect occurrences of anti-patterns.

We use SVMDetect to detect the well-known anti-patterns: Blob, Functional Decomposition, Spaghetti code, and Swiss Army Knife. For each anti-pattern detection, the detection process is identical.

We illustrate the detection process with the Blob anti-pattern for the sake of clarity. We define:

- $TDS = \{C_i, i = 1, \ldots, p\}$, a set of classes C_i derived from an object-oriented system that constitutes the training dataset;
- $\forall i, C_i$ is labelled as Blob (B) or not (N);
- DDS is the set of the classes of a system in which we want to detect the Blob classes.

Our Approach: SVMDetect

SVMDetect - Steps

To detect the Blob classes in the set *DDS*, we apply SVMDetect through the following steps:

- **Step 1 (Object Oriented Metric Specification)** SVMDetect takes as input the training dataset *TDS* with object-oriented metrics for classes.
- **Step 2 (Train the SVM Classifier)** Train SVMDetect with *TDS* defined in Step 1.
- **Step 3 (Construction of the dataset *DDS* and detection of the occurrences of an anti-pattern)** Build detection dataset *DDS* and apply SVMDetect trained in step 2 to *DDS*.

We use Weka to implement SVMDetect using its SVM classifier.

Empirical Study

Empirical Study

- *goal*: **validate** that SVMDetect can **overcome** previous approaches' limitations
- *quality focus*: **accuracy** of SVMDetect, in terms of **precision** and **recall**.
- *perspective*: researchers and practitioners interested in **verifying** if SVMDetect can be **effective** in detecting various kinds of anti-patterns, and in **overcoming** the previous limitations.

Empirical Study

Research Questions

- RQ1: How does the accuracy of SVMDetect compare with that of DETEX, in terms of precision and recall? We decompose RQ1 as follows:
  - $RQ1_1$: How does the accuracy of SVMDetect compare with that of DETEX, in terms of precision and recall, when applied on a same subset of a system?
  - $RQ1_2$: How many occurrences of Blob SVMDetect can detect when comparing with that of DETEX on a same entire system?

Empirical Study

Objects

Names	Versions	# Lines of Code	# Classes	# Interfaces
ArgoUML	0.19.8	113,017	1,230	67
A design tool for UML				
Azureus	2.3.0.6	191,963	1,449	546
A peer-to-peer client that implements the protocol BitTorrent				
Xerces	2.7.0	71,217	513	162
A syntaxic analyser				

Table : Description of the objects of the study

Empirical Study

Subjects

The subjects of our study are the following four anti-patterns:

- Blob
- Functional Decomposition (FD)
- Spaghetti Code (SC)
- Swiss Army Knife (SAK)

These four anti-patterns because **known** anti-patterns, **commonly studied** in **previous** work for **comparison**.

Study Results

Subsets of System: RQ1_1

Table : Precision of SVMDetect vs. DETEX in subsets (%)

		ArgoUML	Azureus	Xerces
Blob	DETEX	0.00	0.00	0.00
	SVMDetect	97.09	97.32	95.51
FD	DETEX	0.00	0.00	0.00
	SVMDetect	70.68	72.01	66.93
SC	DETEX	0.00	0.00	0.00
	SVMDetect	85.00	88.00	86.00
SAK	DETEX	10.00	10.00	0.00
	SVMDetect	75.46	84.54	80.76

Study Results

Subsets of System: RQ1$_1$

Table : Recall of SVMDetect vs. DETEX in subsets (%)

		ArgoUML	Azureus	Xerces
Blob	DETEX	0.00	0.00	0.00
	SVMDetect	84.09	91.33	95.29
FD	DETEX	0.00	0.00	0.00
	SVMDetect	57.50	84.28	70.00
SC	DETEX	0.00	0.00	0.00
	SVMDetect	71.00	89.00	86.00
SAK	DETEX	0.00	0.00	0.00
	SVMDetect	77.14	85.71	75.50

Study Results

Complete System: $RQ1_2$

Table : Total recovered occurrences of BLOB by DETEX and SVMDetect

	DETEX	**SVMDetect**
ArgoUML	25	40
Azureus	38	48
Xerces	39	55
Total	102	**143**

We answer RQ1: “How does the accuracy of SVMDetect compare with that of DETEX, in terms of precision and recall?” as follows:

- on subsets of systems, SVMDetect **dramatically outperforms** DETEX.
- on entire systems, SVMDetect **detects more occurrences** of Blob than DETEX.

Discussions

Threats to Validity

Threats to the validity of our results.

- **Construct validity** (Measurement errors, subjectivity): occurrences of anti-patterns **manually validated**.
- **Internal Validity** (dependence of the obtained results on chosen anti-patterns and systems.): used four **well-known and representative** anti-patterns. used in previous works. used 3 **open-source systems with different sizes**, used in **previous** works.
- **Reliability Validity** (possibility of replication): used 3 **open-source** systems **available on-line**.
- **External Validity** (Generalisability): 3 systems with **different** sizes and different domains. Representative subset of anti-patterns.

Conclusion and Future Work

Conclusion

- introduced a novel approach to detect anti-patterns, SVMDetect, based on SVM.
- SVMDetect performs on 3 systems (ArgoUML v0.19.8, Azureus v2.3.0.6, and Xerces v2.7.0) and 4 anti-patterns (Blob, Functional Decomposition, Spaghetti Code, and Swiss Army Knife)
- the accuracy of SVMDetect is greater than that of DETEX on a subset of classes.
- on whole system, SVMDetect is able to find more anti-patterns occurrences than DETEX
- SVM-based approach can overcome the limitations of the previous approaches and could be more readily adopted by practitioners.

Conclusion and Future Work

Future Work

Future work includes:

- use SVMDetect in real-world environments.
- reproduce the study with other systems and anti-patterns to increase our confidence in the generalisability of our conclusions.
- take into account the user feedback.
- evaluate the impact of the quality of training dataset and feedback set on SVMDetect results.

Any Question?

References

E. H. Alikacem and H. A. Sahraoui.
Détection d'anomalies utilisant un langage de règle de qualité.
In LMO. Hermes Science Publications, 2006.

J. Bedo, C. Sanderson, and A. Kowalczyk.
An efficient alternative to svm based recursive feature elimination with applications in natural language processing and bioinformatics.
In A. Sattar and B.-h. Kang, editors, AI 2006: Advances in Artificial Intelligence, volume 4304 of Lecture Notes in Computer Science, pages 170–180. Springer Berlin Heidelberg, 2006.

References

G. Cao, J.-Y. Nie, J. Gao, and S. Robertson.
Selecting good expansion terms for pseudo-relevance feedback.
In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008, pages 243–250. ACM, 2008.

M. J. Choi, A. Torralba, and A. S. Willsky.
A tree-based context model for object recognition.
IEEE Trans. Pattern Anal. Mach. Intell., 34:240–252, Feb. 2012.

C. Cortes and V. Vapnik.
Support-vector networks.
Machine Learning, 20:273–297, 1995. 10.1007/BF00994018.

References

Y.-G. Guéhéneuc, H. Sahraoui, and Farouk Zaidi.
Fingerprinting design patterns.
In E. Stroulia and A. de Lucia, editors, Proceedings of the 11th Working Conference on Reverse Engineering (WCRE). IEEE Computer Society Press, November 2004.

M. Kessentini, S. Vaucher, and H. Sahraoui.
Deviance from perfection is a better criterion than closeness to evil when identifying risky code.
In Proceedings of the IEEE/ACM international conference on Automated software engineering, ASE '10, pages 113–122, New York, NY, USA, 2010. ACM.

References

F. Khomh, M. D. Penta, and Y.-G. Guéhéneuc.
An exploratory study of the impact of antipatterns on class change-and fault-proneness.
Journal of Empirical Software Engineering (EMSE), 2011.

S. Kim, E. J. W. Jr., and Y. Zhang.
Classifying software changes: Clean or buggy?
IEEE Trans. Software Eng., 34(2):181–196, 2008.

G. Langelier, H. Sahraoui, and P. Poulin.
Visualization-based analysis of quality for large-scale software systems.
In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, ASE '05, pages 214–223, New York, NY, USA, 2005. ACM.

References

R. Marinescu.
Detection strategies: Metrics-based rules for detecting design flaws.
In In Proceedings of the IEEE 20th International Conference on Software Maintenance. IEEE Computer Society Press, 2004.

L. Setia, J. Ick, and H. Burkhardt.
Svm-based relevance feedback in image retrieval using invariant feature histograms.

Naouel Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. L. Meur.
DECOR: A method for the specification and detection of code and design smells.
Transactions on Software Engineering (TSE), 2009.