Class Imbalance Learning in Software Defect Prediction

Dr. Shuo Wang
s.wang@cs.bham.ac.uk
University of Birmingham

Research keywords: ensemble learning, class imbalance learning, online learning

Outline

1. Problem description: software defect prediction (SDP)
2. Offline class imbalance learning for SDP [1]
   - What is class imbalance learning?
   - How does it help with SDP?
3. Online class imbalance learning [2]
   - Why online?
   - Its potential in SDP?
4. Team work: Learning-to-Rank algorithm for SDP

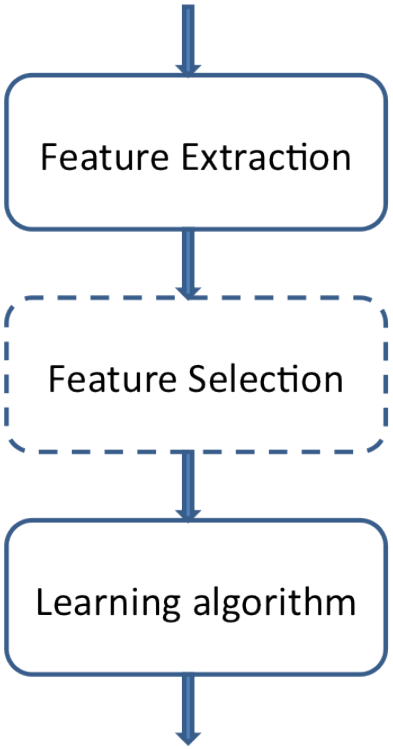
[1] EPSRC-funded project SEBASE (2006-2011)
[2] EPSRC-funded project DAASE (2012-)

Software defect prediction (SDP):

- A learning problem in software testing that aims to locate and analyze which part of software is more likely to contain defects.
- 2-class classification problem: defect and non-defect
- Objective: detect defects as many as possible without losing overall performance.
- When the project budget is limited or the whole software system is too large to be tested completely, a good defect classifier can guide software engineers to focus the testing on defect-prone parts of software.

SDP Main steps:

Input: defect logs from previous releases of software



Output: predict defective modules for the next release

Static code attributes:
•McCabe metrics [McCabe1976]: collect information of the complexity of pathways contained in the module based on a flow graph.
•Halstead metrics [Halstead:1977]: estimate the reading complexity based on the number of operators and operands in the module.

•Different data sets result in very different best feature subsets. [Song2010] [Menzies2007]
•It is suggested to use all available features. [Menzies2007]

A variety of machine learning methods have been discussed:
decision trees[Khoshgoftaar:2002],
neural networks[Zheng:2010][Thwin:2005],
Naïve Bayes[Turhan:2009][Menzies:2007],
support vector machines[Gray2009], etc.

SDP data feature: collected training data contains much more non-defective modules (majority) than defective ones (minority), as shown in the table.

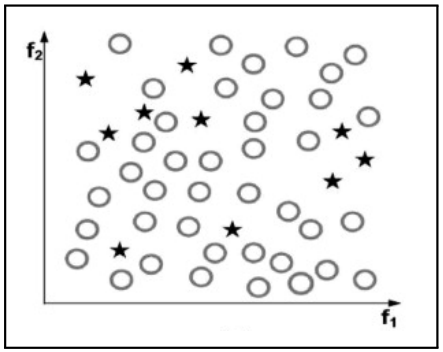
Table: NASA data sets for SDP

data	language	examples	attributes	defect%
cm1	C	498	21	9.83
kc3	Java	458	39	9.38
pc1	C	1109	21	6.94
pc3	C	1563	37	10.23
mw1	C	403	37	7.69

- The rare defective examples are more costly and important.
- Class imbalanced distribution is harmful for classification performance, especially the minority class.

Class Imbalance Learning (machine learning):

Learning from imbalanced data sets, in which some classes of examples (minority) are highly under-represented comparing to other classes (majority).



- Examples: medical diagnosis, risk management, fault detection, etc.
- Learning difficulty: poor generalization on the minority class.
- Learning objective: obtaining a classifier that will provide high accuracy for the minority class without severely jeopardizing the accuracy of the majority class.
- Solutions: resampling techniques, cost-sensitive methods, classifier ensemble methods, etc.

Using Class Imbalance Learning for Software Defect Prediction

Existing methods to tackle class imbalance in SDP problems:

- undersampling non-defective examples [Menzies et al.,] [Shanab et al.,] [Gao et al., 2012]
- oversampling defective examples [Pelayo and Dick, 2012] [Shatnawi, 2012]
- cost-sensitive: setting a higher misclassification cost for the defect class [Zheng, 2010] [Khoshgoftaar et al.,]

They were compared to the methods without applying any class imbalance techniques, and showed usefulness. However, the following issues have not been answered:

- In which aspect and to what extent class imbalance learning can benefit SDP problems? (E.g. more defects are detected or fewer false alarms?)
- Which class imbalance learning methods are more effective?

Such information would help us to understand the potential of class imbalance learning methods in SDP and develop better solutions.

Using Class Imbalance Learning for Software Defect Prediction

S. Wang and X. Yao, "Using Class Imbalance Learning for Software Defect Prediction", IEEE Transactions on Reliability, vol 62, Pages 434-443, 2012.

Research questions

1. In which aspect and to what extent class imbalance learning can benefit SDP problems?
2. Which class imbalance learning methods are more effective?
3. Can we make better use of them for various software projects efficiently?

Using Class Imbalance Learning for Software Defect Prediction

S. Wang and X. Yao, "Using Class Imbalance Learning for Software Defect Prediction", IEEE Transactions on Reliability, vol 62, Pages 434-443, 2012.

Research questions

1. In which aspect and to what extent class imbalance learning can benefit SDP problems?
2. Which class imbalance learning methods are more effective?
3. Can we make better use of them for various software projects efficiently?

Using Class Imbalance Learning for Software Defect Prediction (Part I)

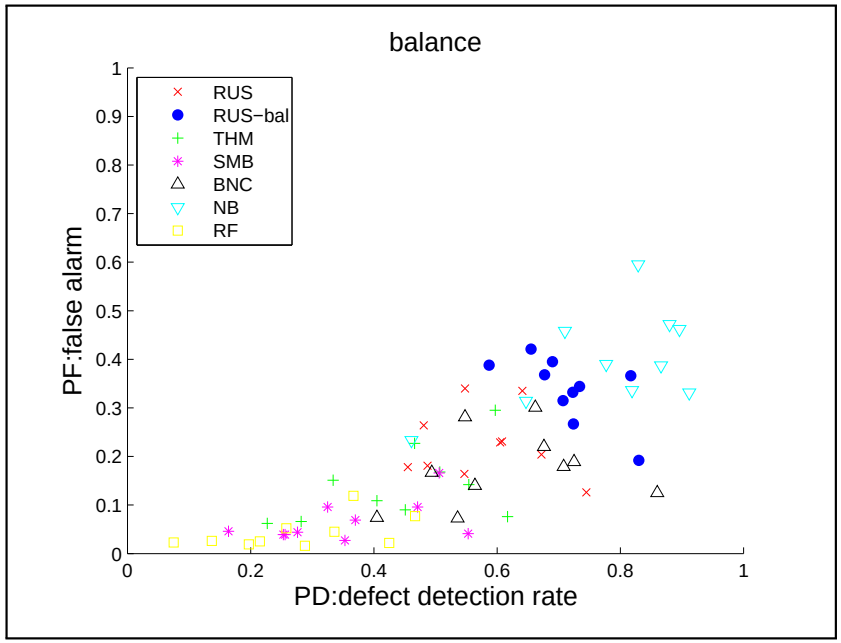
For the first two questions: a comparative study

- Five class imbalance learning methods
  random undersampling (RUS), the balanced version of random undersampling (RUS-bal), threshold-moving (THM), AdaBoost.NC (BNC) and SMOTEBoost (SMB)
- Two top-ranked techniques in SDP field
  Naive Bayes with the log filter [Menzies et al., 2007] and Random Forest [Catal and Diri, 2009]
- Evaluation measures
  - Defect Detection Rate (i.e. recall of the defect class, PD), false alarms (PF)
  - Overall Performance: AUC, G-mean, balance
- Data: 10 practical and commonly used project data sets from the public PROMISE repository.

Using Class Imbalance Learning for Software Defect Prediction (Part I)

Conclusions:

- Naive Bayes (NB) find more defects, but suffers from more false alarms.
- AdaBoost.NC (BNC) can better balance the performance between defect and non-defect classes.
- Choosing appropriate parameters for class imbalance learning methods are crucial to their ability of finding defects.
- Optimal parameters vary with different data sets.



Next challenges?

- Can we find a predictor that combines the strengths of Naive Bayes and AdaBoost.NC?
- A solution with adaptive parameters is desirable.

AVG	PD	PF
NB	78%	40%
BNC	62%	17%

*PD: detection rate

*PF: false alarm

Using Class Imbalance Learning for Software Defect Prediction

S. Wang and X. Yao, "Using Class Imbalance Learning for Software Defect Prediction", IEEE Transactions on Reliability, vol 62, Pages 434-443, 2012.

Research questions

1. In which aspect and to what extent class imbalance learning can benefit SDP problems?
2. Which class imbalance learning methods are more effective?
3. Can we make better use of them for various software projects efficiently?

Using Class Imbalance Learning for Software Defect Prediction (Part II)

Dynamic version of AdaBoost.NC (DNC) (third research question)

- Advantage: adaptively adjust its main parameter during training, to maximumly emphasize the defect class, reduce the time of looking for the best parameter and make it applicable to various software projects.
- Idea:
  - *increase* the parameter (more learning bias to the minority class) if the performance gets better during the sequential training of AdaBoost;
  - *decrease* the parameter if the performance gets worse.
- Results:
  - Average PD = 64%, PF = 21%.
  - Dynamic AdaBoost.NC is a more effective and efficient method than AdaBoost.NC.
  - It improves defect detection rate and overall performance without deciding the best parameter prior to learning.

Outline

1. Problem description: software defect prediction (SDP)
2. Offline class imbalance learning for SDP [1]
   - What is class imbalance learning?
   - How does it help with SDP?
3. Online class imbalance learning [2]
   - Why online?
   - Its potential in SDP?
4. Team work: Learning-to-Rank algorithm for SDP

[1] EPSRC-funded project SEBASE (2006-2011)
[2] EPSRC-funded project DAASE (2012-)

Why Online?

- Software projects are becoming more dynamic.
- Software codes are evolving between the releases.

New challenges

- The type of defects can be evolving alone with the system development (concept drift) [Harman et al., 2014]
- The class imbalance status for the defect class can be changing (changing imbalanced rate)

Online Class Imbalance Learning

Some recent work

1. S. Wang, L.L. Minku and X. Yao, "Resampling-Based Ensemble Methods for Online Class Imbalance Learning", IEEE Transactions on Knowledge and Data Engineering, vol.27, Pages 1356-1368, 2015.
2. S. Wang, L.L. Minku and X. Yao, "Online Class Imbalance Learning and Its Applications in Fault Detection", Special Issue of International Journal of Computational Intelligence and Applications, vol. 12, Pages 1340001(1-19), 2013.
3. S. Wang, L.L. Minku and X. Yao, "A Multi-Objective Ensemble Method for Online Class Imbalance Learning", in International Joint Conference on Neural Networks (IJCNN '14) 2014.
4. S. Wang, L.L. Minku, D. Ghezzi, D. Caltabiano, P. Tino and X. Yao, "Concept Drift Detection for Online Class Imbalance Learning", in International Joint Conference on Neural Networks (IJCNN '13) 2013.
5. S. Wang, L.L. Minku and X. Yao, "Learning Framework for Online Class Imbalance Learning", in IEEE Symposium Series on Computational Intelligence (SSCI) 2013, Singapore. Pages 36-45, 2013.

Aim to deal with data streams under dynamic environments:

- Data arrive in real time.
- Concept drift can happen.
- Data streams tend to be imbalanced.

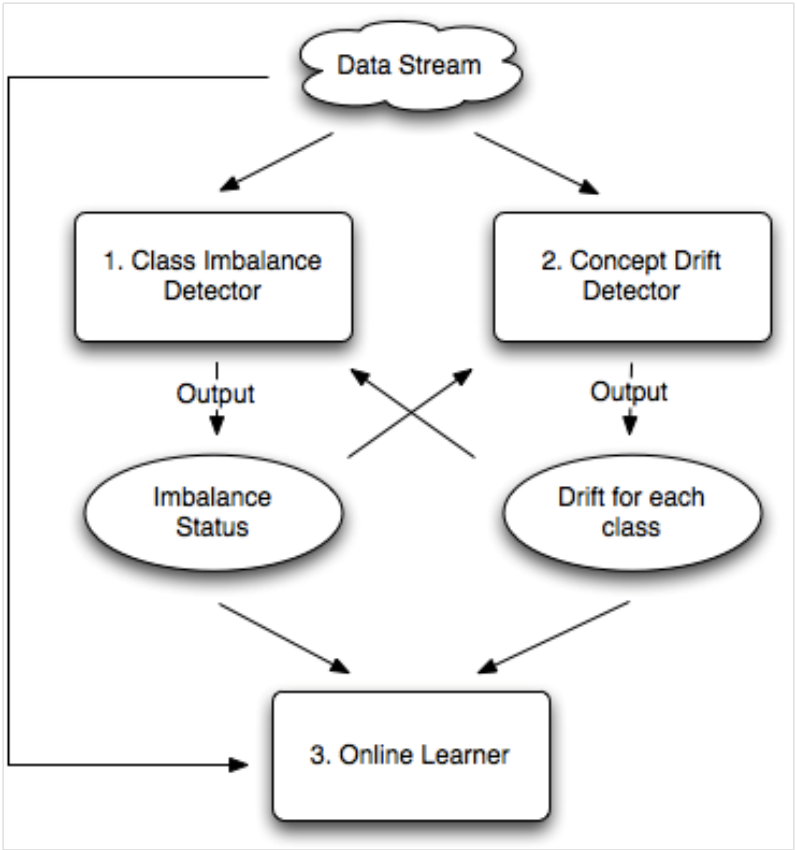
Online Class Imbalance Learning

1. Define class imbalance online
   - A learning framework to decompose the problem
   - Two online indicators to describe real-time data status
2. Detect changes in data
   - Class imbalance detector to sense changes in class prior probabilities
   - Concept drift detector to sense changes in class-conditional probability density function
3. Online solutions for imbalanced data
   - Resampling-based ensemble methods: OOB, UOB

Online Class Imbalance Learning

Three learning modules:

- Module 1: a class imbalance detector
  - Report class imbalance status.
  - Capture changes in class priors.
- Module 2: a concept drift detector
  - Capture changes involving classification boundary shifts.
- Module 3: Adaptive online learner
  - Respond to detected class imbalance and concept drift, and make predictions.



Online Class Imbalance Learning

S. Wang, L.L. Minku and X. Yao, "Online Class Imbalance Learning and Its Applications in Fault Detection", Special Issue of International Journal of Computational Intelligence and Applications, vol. 12, Pages 1340001(1-19), 2013.

Applied Data: fault prediction

- Sensor faults in smart buildings and robotic systems.
- Software defect data from PROMISE repository.

Four Scenarios in Data Streams

- Constant imbalance rate
- A sudden decrease of imbalance rate
- A sudden increase of imbalance rate
- A gradual increase of imbalance rate

Assumption: no concept drift, only 2 classes

Online Class Imbalance Learning

Proposed methods:

- Online indicators: real-time imbalance rate and minority-class recall

$$w_k^{(t)} = \eta w_k^{(t-1)} + (1-\eta)[(x_t, c_k)](k = 1, 2, \ldots, |Y|),$$
$$R_k^{(t)} = \eta' R_k^{(t-1)} + (1-\eta')[x_t \leftarrow c_k](k = 1, 2, \ldots, |Y|).$$

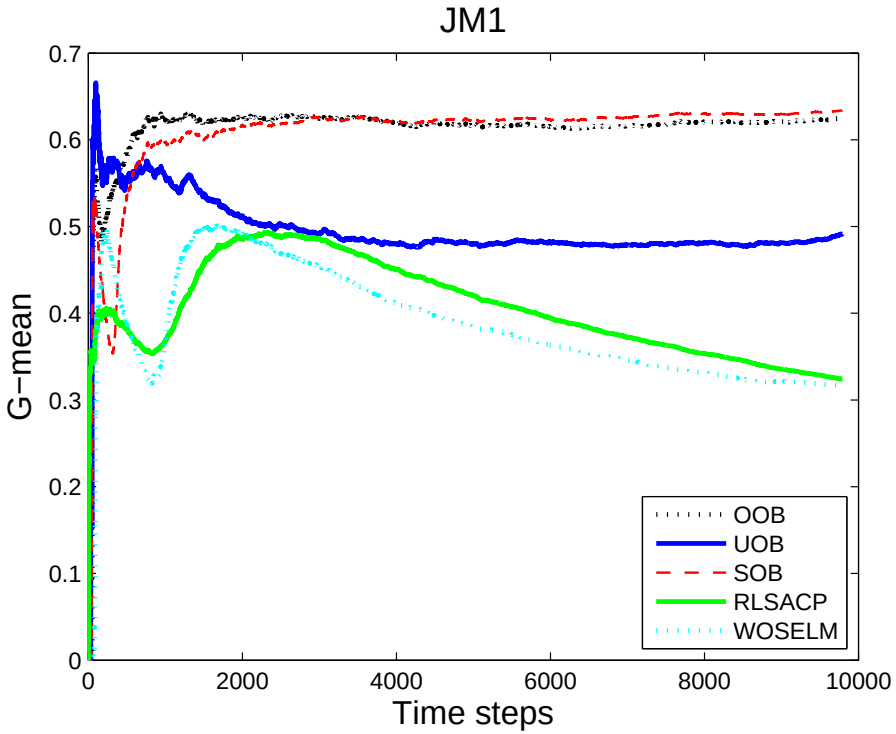
- Online predictive model: Sampling-based Online Bagging

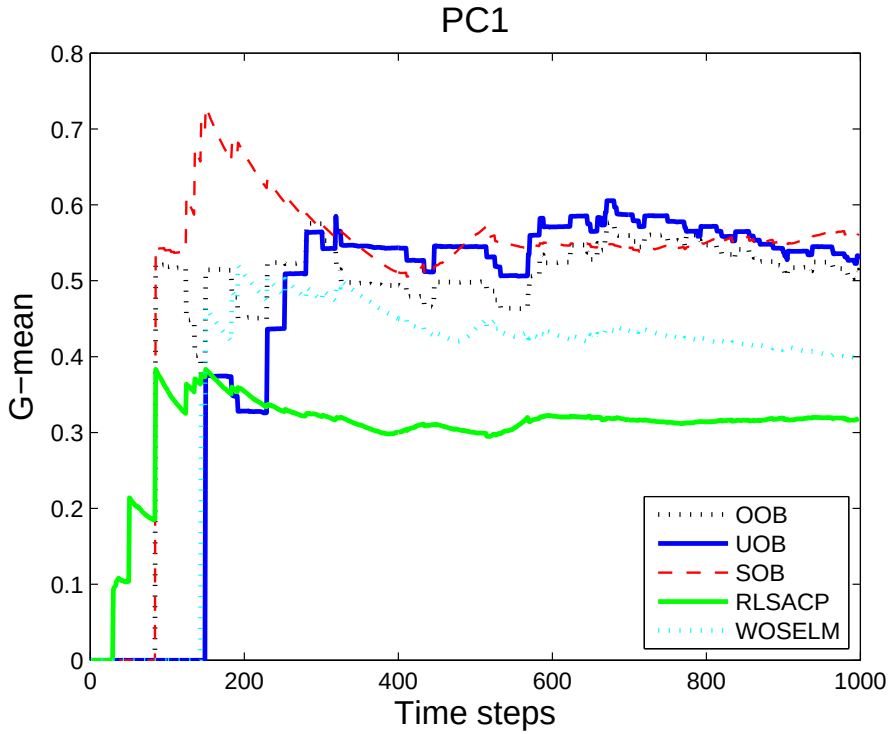
Table 1. Sampling-based online bagging.

Input: an ensemble classifier F composed of M base learners f_m, and current training example (x_t, y_t).
for each base learner f_m $(m - 1, 2, \ldots, M)$ **do**
  if y_t is $+1$
    set $K \sim \text{Poisson}(N/P)$
  else
    set $K \sim \text{Poisson}(R_p)$
  end if
  update f_m K times
end for

Online Class Imbalance Learning

Some results on PROMISE data streams with constant imbalance rate:





Outline

1. Problem description: software defect prediction (SDP)
2. Offline class imbalance learning for SDP [1]
   - What is class imbalance learning?
   - How does it help with SDP?
3. Online class imbalance learning [2]
   - Why online?
   - Its potential in SDP?
4. Team work: Learning-to-Rank algorithm for SDP

[1] EPSRC-funded project SEBASE (2006-2011)
[2] EPSRC-funded project DAASE (2012-)

A Learning-to-Rank Algorithm for SDP

Learning-to-Rank Algorithms for SDP:

- Task: predict a ranking of the modules based on their number of defects (defect-proneness)
  input: module features with defect numbers
  output: a ranking of new modules
- When to use Learning-to-Rank: need to find out which software modules contain more defects.
- Significance: help to guide the assignment of testing resources.
  E.g. When the resource is limited, only the modules with high rankings will be tested.
- Problem of existing methods:
  They build defect prediction models by optimizing their prediction on defect numbers, such as maximum likelihood and minimizing least square errors. It causes problems because **prediction on defect numbers** is an indirect objective, while the **module ranking** should be the real objective for learning-to-rank.

A Learning-to-Rank Algorithm for SDP

Example:

	Defect Number		
Module	Real Number	Model1	Model2
A	4	1	1
B	2	2	0

- Model 1 is seen as a better model in terms of the distance to the real defect number of modules A and B.
- Model 2 is more desired because it gives the correct ranking.

Idea in the proposed method:

- It optimizes the ranking performance measure directly based on defect numbers.
- Objective function to be optimized: Fault-percentile-average – the average proportion of actual defects in the top i predicted modules ($i = 1, \ldots, k$, considering k modules).

X. Yang, K. Tang, X. Yao, "A Learning-to-Rank Approach to Software Defect Prediction", IEEE Transactions on Reliability, Volume 64, pp.234-246, 2015.

Future Work

- Study incremental learning techniques, which may be more suitable for SDP than online learning techniques.
- The learning problem with multiple types of faults (multi-class problems) in both offline and online scenarios
  - Offline: the imbalanced status is aggravated.
  - Incremental/Online: the number of classes can be changing.

References I

Catal, C. and Diri, B. (2009).

Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem.

Information Sciences, 179(8):1040–1058.

Gao, K., Khoshgoftaar, T., and Napolitano, A. (2012).

A hybrid approach to coping with high dimensionality and class imbalance for software defect prediction.

In *11th International Conference on Machine Learning and Applications (ICMLA)*, pages 281–288.

Harman, M., Islam, S., Jia, Y., Minku, L. L., Sarro, F., and Srivisut, K. (2014).

Less is more: Temporal fault predictive performance over multiple hadoop releases.

In *Symposium on Search-Based Software Engineering (SSBSE'14), Lecture Notes in Computer Science*, pages 240–246.

Khoshgoftaar, T. M., Geleyn, E., Nguyen, L., and Bullard, L.

Cost-sensitive boosting in software quality modeling.

In *Proceedings of 7th IEEE International Symposium on High Assurance Systems Engineering*, pages 51–60.

Menzies, T., Greenwald, J., and Frank, A. (2007).

Data mining static code attributes to learn defect predictors.

IEEE Transactions on Software Engineering, 33(1):2–13.

Menzies, T., Turhan, B., Bener, A., Gay, G., Cukic, B., and Jiang, Y.

Implications of ceiling effects in defect predictors.

In *The 4th International Workshop on Predictor Models in Software Engineering (PROMISE 08)*, pages 47–54.

Pelayo, L. and Dick, S. (2012).

Evaluating stratification alternatives to improve software defect prediction.

IEEE Transactions on Reliability, 61(2):516–525.

References II

Shanab, A. A., Khoshgoftaar, T. M., Wald, R., and Hulse, J. V.

Comparison of approaches to alleviate problems with high-dimensional and class-imbalanced data.
In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 234–239.

Shatnawi, R. (2012).

Improving software fault-prediction for imbalanced data.
In *International Conference on Innovations in Information Technology (IIT)*, pages 54–59.

Zheng, J. (2010).

Cost-sensitive boosting neural networks for software defect prediction.
Expert Systems with Applications, 37(6):4537–4543.