Cost-Sensitive Decision Forest: CSForest

Michael J. Siers and Md Zahidul Islam
School of Computing and Mathematics, Charles Sturt University

Charles Sturt University

Our CSForest Publications

The technique CSForest was a products of Michael's Honours in 2014.

- ERA Rank A* Journal Information Systems (2015) [1].
- CORE Rank B Conference AI 2014 [2].

Introduction

- CSForest is a cost-sensitive classification algorithm. It uses the methodology of the decision forest algorithm SysFor [3] with the splitting criteria used in CSTree [4].
- In our two related publications, we applied CSForest to the software defect prediction problem. This is an application domain in which classification is used to determine which sections of a programs source code are likely to contain defects (bugs).

Software Defect Prediction Datasets

- We use the publicly available NASA-MDP software defect datasets. Features of the dataset include:
  - Various line of code counts (logical, whitespace etc.)
  - Halstead's Complexity Metrics
  - Cyclomatic Complexity Metrics
- The number of records of the datasets range from roughly 500 to 1500 records.
- The number of attributes/features of the datasets are roughly 30.
- Each record in the dataset describes a software module (typically a function/method).

```
public int getHeight() {
    if (measure == "cm")
        return 182;
    else
        return 1820;
}
```

Figure 1: Example module

CSForest

- CSForest first finds a set of "good" splitting points for a dataset based on the total expected cost criterion. The good splitting points are constrained by two user-defined settings:
  - Separation: If it is a numerical split, the splitting value is not too close to another good split.
  - Goodness: The split is close enough to the best split for minimizing the splitting cost criterion.
- Buliding the trees
  - Each good splitting point is used a root node in a CSTree decision tree.
  - If the user requested more trees, then more trees are built using the good splitting points at the next level in the tree.
- Classifying new records (CSVoting)
  - The set of leaves L that the record r falls in is found.
  - $\forall l \in L$, the total expected cost is found for each possible classification.
  - Finally, r is classified as the cheapest classification.

The splitting criterion (From CSTree)

- Consider a distribution d with records which belong to either the positive class or the negative class. Given that we know the cost of making true positive C_{TP}, true negative C_{TN}, false negative C_{FN}, and false positive C_{FP} predictions, we can calculate the cost of labelling every record in the distribution as negative C_N or positive C_P.

$$C_N = N_{TN} \times C_{TN} + N_{FN} \times C_{FN}$$

$$C_P = N_{TP} \times C_{TP} + N_{FP} \times C_{FP}$$

  Where N_{TN}, N_{TP}, N_{FN}, and N_{FP} are the number of true negative, true positive, false negative, and false positive predictions respectively.
- The total expected cost is then calculated as:

$$E = \frac{2 \times C_P \times C_N}{C_P + C_N}$$

- Favoured splits reduce the total cost the most after splitting compared to without splitting.

Results: Table

- Performance Comparison (lower cost, the better)

Dataset	CSTree	SysFor (Voting 1)	CSForest
$MC2'$	165	164	129
$PC1'$	290	255	276
$KC1'$	1187	1404	1168
$PC3'$	586	664	521
$MC1'$	291	275	261
$PC2'$	84	81	80

Table 1: Cost comparison

- We used the following cost-matrix in our comparison.

$C_{TP} = 1$	$C_{TN} = 0$
$C_{FP} = 1$	$C_{FN} = 5$

Table 2: Cost-Matrix

Results: Figure

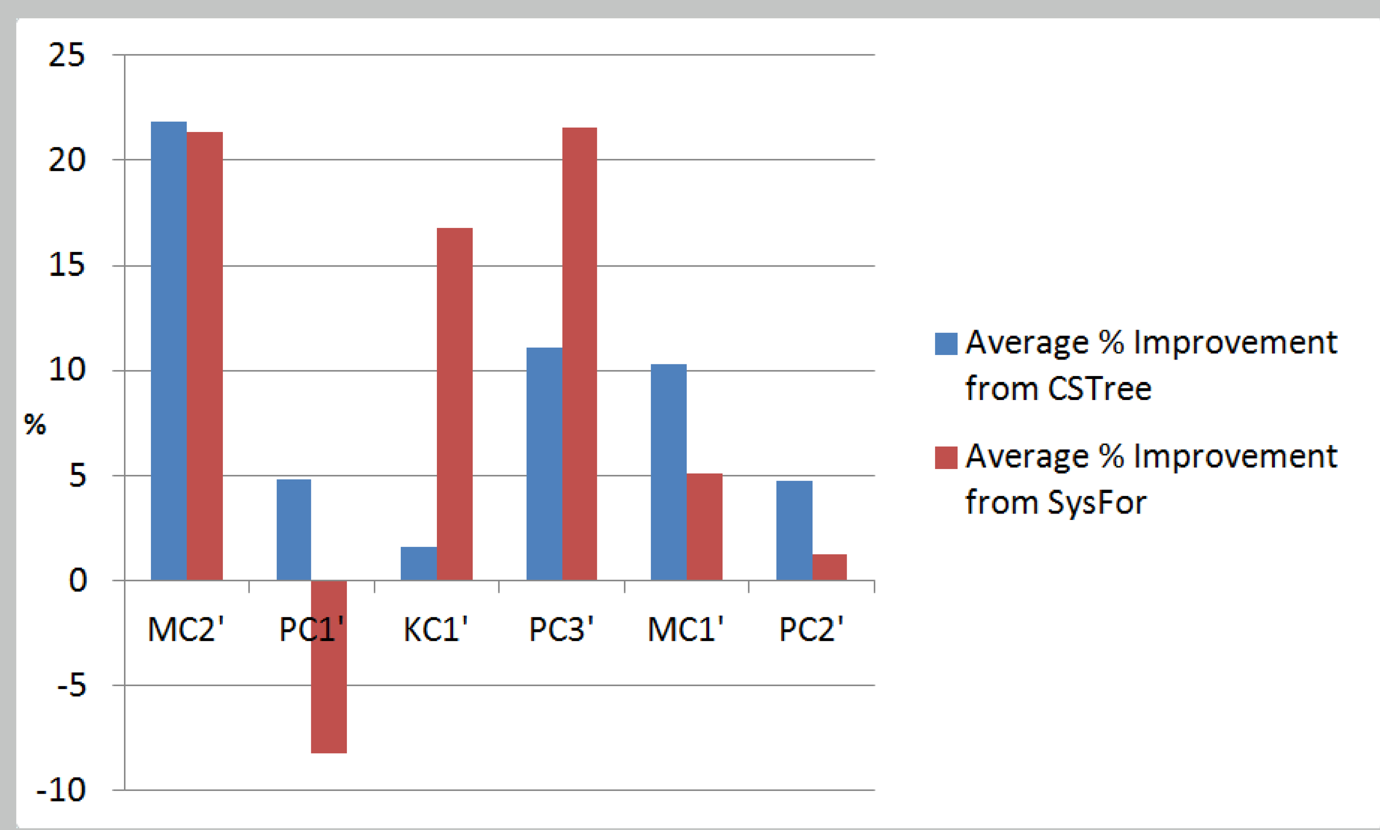

Figure 2: Average CSForest Performance Increase from Existing Techniques

References

1. Siers, M. J., Islam, M. Z. (2015). Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Information Systems, 51, 62-71.
2. Siers, M. J., Islam, M. Z. (2014). Cost sensitive decision forest and voting for software defect prediction. In PRICAI 2014: Trends in Artificial Intelligence (pp. 929-936). Springer International Publishing.
3. Islam, Z., Giggins, H. (2011, December). Knowledge discovery through SysFor: a systematically developed forest of multiple decision trees. In Proceedings of the Ninth Australasian Data Mining Conference-Volume 121 (pp. 195-204). Australian Computer Society, Inc..
4. Ling, C. X., Sheng, V. S., Bruckhaus, T., Madhavji, N. H. (2006, August). Maximum profit mining and its application in software development. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 929-934). ACM.
5. Halstead, M. H. (1977). Elements of Software Science (Operating and programming systems series). Elsevier Science Inc..
6. McCabe, T. J. (1976). A complexity measure. Software Engineering, IEEE Transactions on, (4), 308-320.

Acknowledgments

- Thank you to the various reviewers for the two CSForest related publications.

Contact Information

- Web: mikesiers.com/software, https://csusap.edu.au/ zislam
- Email: msiers@csu.edu.au, zislam@csu.edu.au
- Phone: Michael: +61435608521