Exceptional service in the national interest

Sandia National Laboratories

SAND2013-9107C



Streaming Malware Classification in the Presence of Concept Drift and Class Imbalance

W. Philip Kegelmeyer, Ken Chiang, Joe Ingram

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Malware Overview

- One of the most prevalent cybersecurity threats.
- Most organizations rely on anti-virus software to identify malware, which utilize signatures.
- Signature-based detection only allows anti-virus software to detect known malware (cannot generalize to unseen instances).
- Instead, use supervised machine learning for learning more robust malware detection rules from the data.

Malware Dataset

ID	Dates	Goodware count	Malware count
2010	10-2010 to 01-2011	10260	8501
2011	11-2011 to 02-2012	3409	13011
2012	01-2012 to 03-2012	54153	16911

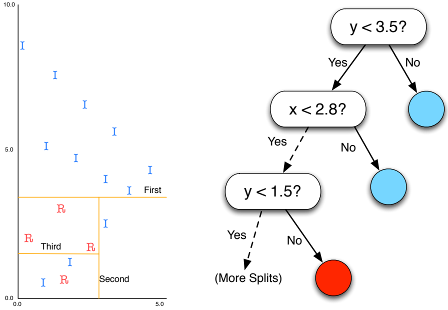
- Malware collected from Arbor Networks[1].
- Goodware collected from live feeds of all files that crossed a corporate network border. Filtered through multiple antivirus scanners to reduce the risk of contaminating the feed with malware.

[1] http://www.arbornetworks.com/

Feature Extraction

- Features based on Portable Executable (PE) headers, which specify the layout of executable files for the Windows operating system.
- Approximately 100 features.
- Some example features:
  - CheckSum
  - number_of_sections
  - file_info
  - RT_DIALOG
  - REMOVABLE_RUN_FROM_SWAP
  - SizeOfHeapReserve
  - HeaderCharacteristicsValue

Batch Decision Trees



Tree growing procedure:

- Exhaustively check *all* possible splits (requires revisiting data).
- Pick best split (based on objective function), split node, and recurse.

Streaming Decision Trees

Observation:
If you relax the notion of the“best” split, only a small subset of data may be needed to find best attribute at a given node.

Idea:

- Maintain list of leaves in current tree.
- Filter examples from stream into appropriate leaf.
- Expand a leaf only when it contains enough examples to “reliably” pick the “best” attribute for splitting.

Hoeffding Trees

Let $\overline{G}(A)$ be the quality of splitting a leaf using attribute A.

Suppose that A_1 and A_2 are the 1^{st} and 2^{nd} best attributes with respect to $\overline{G}(\cdot)$, so far. Define

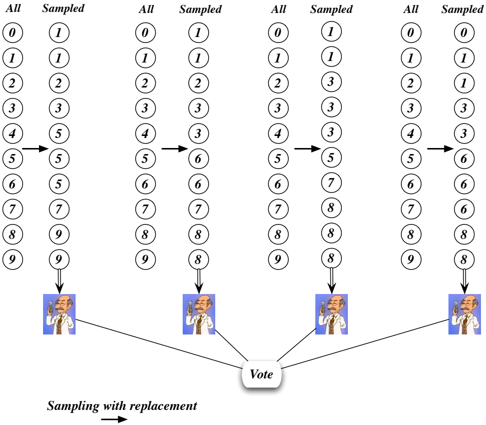
$$\Delta\overline{G} = \overline{G}(A_1) - \overline{G}(A_2) \geq 0.$$

If $\Delta\overline{G} > \epsilon$, then A_1 is best attribute with probability $1 - \delta$ (based on Hoeffding bound).

Tree growing procedure:

- Pick δ. (Remember: ϵ is a function of δ and n.)
- Accumulate samples at a node (that is, increase n) until the best split is ϵ better than the second best split.
- Then make the best split, and recurse.

Batch Ensembles – Bagging



- Bagging seems to require revisiting data.

Streaming Ensembles – "Oza" Bagging

Sampling with replacement from a data set of size N means each bag has K copies of a sample, where K is binomial:

$$P(K = k) \;=\; \binom{N}{k}\left(\frac{1}{N}\right)^{k}\left(1 - \frac{1}{N}\right)^{N-k}$$

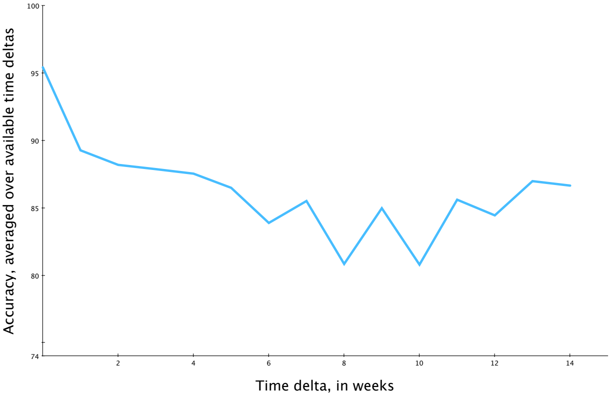
$$\text{As } N \to \infty, P(K = k) \;\approx\; \frac{e^{-1}}{k!}$$

That is, K tends to a $Poisson(1)$ distribution as the number of samples increases. So:

- Choose to start C base classifiers.
- Consider each sample x_i. For each base classifier c_j , pick K_{ij} from a $Poisson(1)$ distribution and give K_{ij} copies of x_i to c_j.

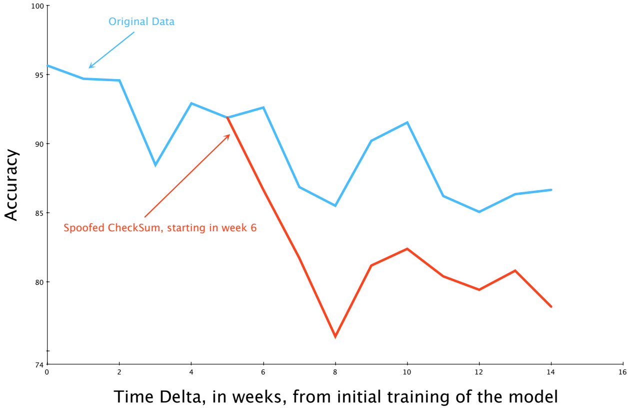
Each sample is handled only once.

Gradual Concept Drift



- Malware detection accuracy degrades over time.

Sudden Concept Drift



- Result of a suddenly invalidated malware detector.

Baseline Performance without Concept Drift

Data	Batch Ensemble	Streaming Ensemble
2010	98.80	96.77
2011	98.45	96.51
2012	98.60	96.73

- Ten-fold cross-validated bagged ensemble analysis of each year individually.
- Both batch and streaming decision tree ensembles are pretty good at distinguishing between malware and goodware.
- Streaming ensemble is somewhat less accurate than its batch counterpart, because the trees have to commit to splits early (i.e., after only seeing a fraction of the data).

Performance with Concept Drift and Copious Malware

Train	Test	Type	Batch Ensemble	Streaming Ensemble
2010	2011	Standard	90.97	91.87
2010	2011	Interleaved	—	95.69
2010	2012	Standard	90.03	85.80
2010	2012	Interleaved	—	92.82
2011	2012	Standard	91.25	81.36
2011	2012	Interleaved	—	93.32

- Impact of concept drift when *all* of the additional malware data is available for training and testing.
- Interleaved streaming ensemble always outperforms the batch ensemble, because it is able to make use of new data not available to the batch model.
- So, one should always add in new data as it arrives?

Performance with Concept Drift and Class Imbalance

Train	Test	Type	Batch Ensemble	Streaming Ensemble
2010	2011	Standard	91.96	90.63
2010	2011	Interleaved	—	90.70
2010	2012	Standard	87.46	86.77
2010	2012	Interleaved	—	82.09
2011	2012	Standard	93.63	84.38
2011	2012	Interleaved	—	75.60

- Train on the full set of malware and goodware, but thin test malware to only 180 samples. Repeated ten times, average performance reported.
- Here, interleaved streaming ensemble did worse than batch.
- Proportion of new data input is so skewed that the interleaved model learns it can do best by predicting that nothing is malware.

Conclusion and Future Work

- Characterized the effect of concept drift and class imbalance on batch and streaming decision tree ensembles using a malware dataset collected from live feeds.
- Demonstrated how bagged ensembles of decisions trees can be well-adapted to the streaming data case.
- Illustrated a perhaps surprising vulnerability stemming from updating a model based on new data.
- Need to investigate performance of algorithms designed to handle concept drift.
- Need to investigate efficacy of skew-correction techniques.

Questions?