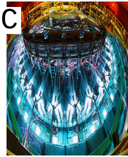
Active Learning for Alert Triage

SAND2013-10143C



JD Doak, Joe Ingram, Jeff Shelburg, Josh Johnson, Brandon Rohrer

Sandia National Laboratories

Albuquerque, New Mexico

{*jedoak, jbingra, jsshelb, jajohn, brrohre*}*@sandia.gov*

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories

Exceptional service in the national interest

Motivation

- Typical cyber security operations monitor multiple sensor feeds.
- When certain conditions in the data are met, an alert is generated in a Security Event and Incident Management system (SEIM).
- Analysts inspect alerts and close or promote to an event.
- Triage process is manual, time-consuming, and detracts from in-depth investigations.

Proposed Solution

- Prioritize alerts using supervised machine learning.
- Efficiently use unlabeled alerts via active learning. Do we outperform passive learning?
- Demonstrate effectiveness of active learning on large, real-world dataset of cyber security alerts.

Operational Challenges

- As more sensor feeds are added, more data is available and the flow of alerts increases.
- Asymmetric nature of cyber defense.

Hypothesis

Automatic alert prioritization would benefit analysts.

- Crucial alerts would not be missed.
- More time for in-depth investigations
- Reduce time gap betweeen event and response, potentially mitigating impact.

Motivating Active Learning

- **Challenge**: supervised learning requires labeled data.
  - Potentially expensive in terms of time and cost.
- **Proposed solution**: active learning
  - Query analysts for labels on alerts predicted to best train models.
- **Desired Result**: need fewer labeled training examples than passive learning.
  - Match or beat accuracy of passive learning with fewer labeled instances.

Related Work

- Literature review suggests that active learning theory is well established.
- Also indicates that the application of active learning to real-world problems is in its infancy
- Relatively small number of papers employing active learning in cyber security settings

Datasets

- Lack of publically available datasets relevant to cyber security
- Many papers present analyses on the KDD-CUP'99 dataset, which has known issues:
  - High redundancy hinders generalization.
  - Not sufficiently challenging as even the worst model attained 86% accuracy
  - Many attacks appearing in the dataset are no longer relevant.

Aladin: Active Learning of Anomalies to Detect Intrusions

- Real-world application of active learning to network traffic classification, anomaly detection, and malware detection.
- Queries for labels to discover new categories and improve accuracy.
- Results:
  - Reduced the number of queries required to attain acceptable accuracy and coverage.
  - Discovered new trojan missed by rule-based methods

Detecting Adversarial Advertisements in the Wild

- Used active learning to discover real-world, adversarial advertisements (e.g., counterfeit goods)
- Only needed a few dozen queries to build accurate one-vs-good models

Data Collection - Feature Extraction

- An alert in the SEIM contains both metadata and the raw alert text.
- The metadata contains information about where the alert came from, when it was created, etc.
- Named-entity recognition (NER) is used to extract, e.g., filenames and URLs
- Latent Dirichlet Allocation used to extract topic-based features.
  - Used NER output as vocabulary to build model

Data Collection - Implicit Label Extraction

- *Explicit* labels obtained from active learning queries.
- *Implicit* labels based on alert life-cycle.
  - Augment explicit labels
  - Allow us to build models before obtaining any explicit labels (i.e., bootstrapping)
  - Sample implicit labels: False Positive, Promoted False Positive, Promoted, and Incident
  - Mapped all labels to Closed or Promoted to allow binary classification

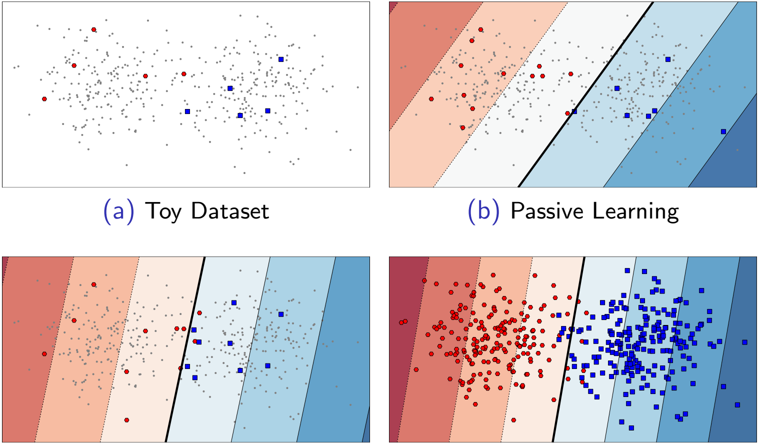
Tested Models

- Linear methods
  - Linear Support Vector Machine (SVM)
  - Logistic Regression
  - Linear Discriminant Analysis (LDA)
  - Naïve Bayes
- Non-linear methods
  - SVM with Radial Basis Function (RBF) Kernel
  - *k*-Nearest Neighbors (*k*NN)
  - Quadratic Discriminant Analysis (QDA)
  - Multilayer Perceptron (MLP)
  - Random Forest

Active Learning

- Only small percentage of data is labeled in many real-world applications.
- Obtaining labels can be costly in time and effort, generally requiring human annotator.
- Active learning tries to maximize utility of labeled data.
- Learner can match or beat the performance obtained via passive learning with less training data if it can choose the data from which it learns.

Example



(a) Toy Dataset (b) Passive Learning

(c) Active Learning (d) Full Learning

Figure: Passive vs. Active Learning on a Toy Dataset.

Query Frameworks

These are common settings in which active learning is applied.

- **Query Synthesis**
  - Learner allowed to query any point in input space
  - Synthesized points may be nonsensical (e.g., digit recognition)
- **Selective Sampling**
  - Instances arrive sequentially.
  - Learner chooses to query or discard.
  - Typically only applicable to streaming settings
- **Pool-based Sampling**
  - Set of labeled instances and pool of unlabeled instances
  - Learner allowed to look at all instances in unlabeled pool to select optimal query
  - Most common framework in practical settings

Query Strategies

A query strategy identifies the points to label.

- **Random Sampling**
  - No info about input space or model used to select instances (passive)
  - Baseline for comparison (not a type of active learning)
  - Sometimes outperforms active learning
- **Uncertainty Sampling**
  - Selects instances the model is least certain how to classify
  - Getting labels for least confident points may yield more info
  - Involves estimating distance to decision boundary
  - Initial model trained on little data. May bias sampling.
- **Other Strategies**
  - Hypothesis-space search
  - Expected error or variance reduction
  - Exploiting structure in data

Types of Uncertainty Sampling

- **Least Confident**
  - Queries instance whose predicted output is least confident
  - Query Instance $\leftarrow \underset{x}{\operatorname{argmax}} \Big[1 - P_\theta(\hat{y}|x)\Big]$
- **Margin**
  - Margin is difference between two most likely predictions.
  - Queries instance with smallest margin
  - Query Instance $\leftarrow \underset{x}{\operatorname{argmax}} \Big[P_\theta(\hat{y}_2|x) - P_\theta(\hat{y}_1|x)\Big]$
- **Entropy**
  - Entropy is a measure of average information content.
  - Queries instance with highest entropy
  - Query Instance $\leftarrow \underset{x}{\operatorname{argmax}} \Big[-\sum_y P_\theta(y|x)\log\big(P_\theta(y|x)\big)\Big]$

Experimental Results

- 8905 alerts
- 1436 promoted (approximately 16%)
- scikit-learn used for all models except MLP (PyBrain)

Baseline Performance

- Used class-averaged accuracy (CAA) as evaluation metric
  - Can mitigate effects of class skew
- 3 runs of 10-fold stratified cross-validation
- Best model: random forest using 100 base decision trees
- Wilcoxon signed-rank test with $\alpha = 0.05$ confirmed statistical significance

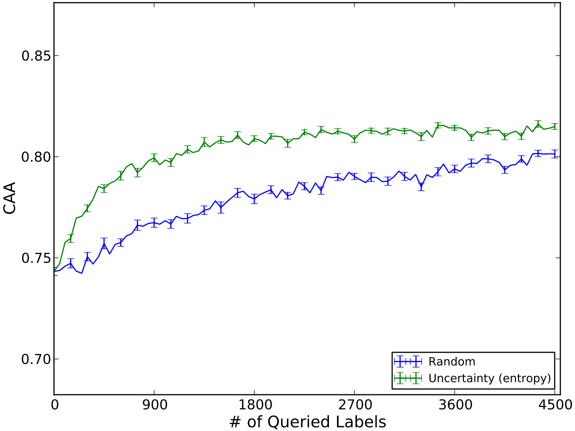
Baseline Performance

	Method	average CAA (SD)
Linear	LDA	0.774 (0.019)
	Naïve Bayes	0.684 (0.016)
	Linear SVM	0.585 (0.015)
	Logistic Regression	0.556 (0.014)
Nonlinear	**Random Forest**	**0.814 (0.020)**
	QDA	0.753 (0.074)
	MLP	0.560 (0.021)
	SVM w/ RBF	0.516 (0.007)
	*k*NN	0.457 (0.014)

Active Learning Performance

- Pool-based sampling
- Variation on 10-fold stratified cross-validation
  - 10% of data in training folds used to build initial model
  - Active learning strategies sample from remaining 90%.
  - Per iteration of active learning
    - Query 50 instances.
    - Retrain model and evaluate against test fold.
  - Repeat 10x for every fold.
- Plot on next page shows average over 100 iterations. (10 folds x 10 initial models)

Active Learning Performance



Approaches baseline performance using only 30% of the data

Deployment to Enterprise Security

- Integrate ranking model into SEIM to automatically prioritize alerts for analysts
- Batch processing (perform at off-peak times)
  - Rebuild model using all labeled alerts.
  - Relabel open alerts.
  - Periodically revisit closed alerts.
- Near real-time (processing new and modified alerts)
  - Extract features.
  - Predict label.
  - Insert into prioritized alert list.
- Query interface for SEIM

References

- E. Bloedorn, L. Talbot, and D. DeBarr, "Data Mining Applied to Intrusion Detection: MITRE Experiences," in *Machine Learning and Data Mining for Computer Security*, ser. Advanced Information and Knowledge Processing, M. Maloof, Ed. Springer London, 2006, pp. 65–88.
- J. W. Stokes, J. C. Platt, J. Kravis, and M. Shilman, "Aladin: Active Learning of Anomalies to Detect Intrusions," *Technique Report. Microsoft Network Security Remond, WA*, vol. 98052, 2008.
- D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, "Detecting Adversarial Advertisements in the Wild," in *Proceedings of the 17th ACM SIGKDD International Conference on Data Mining and Knowledge Discovery*, 2011.