COUGAR: Clustering Of Unknown malware using Genetic Algorithm Routines

Zachary Wilkins (zachary.wilkins@dal.ca)
Nur Zincir-Heywood (zincir@cs.dal.ca)

Dalhousie University
Halifax, Nova Scotia, Canada

July 2020
RWA @ GECCO

Roadmap

Background

- Malware is a growing, universal problem
- New developments: trouble for signature AV
  - Mutation engines, polymorphic malware
- Manual analysis cannot keep pace with growth
- How do you triage?
- One approach: clustering!

Research Objective

Create a system that:

- Uses observed behaviours to cluster malware
- Predict unseen malware based on seen malware

Lessons from the Literature

Common Approaches (detailed in paper)

- Extracting features from live malware in Cuckoo sandbox
  - Syscalls
  - Return codes
  - Bigrams
- Using TF–IDF (e.g.: malware is document, syscall is term)
- Reducing high dimensional data
- Using machine learning classifiers to predict labels
- Using closed source / industry partner datasets

Lessons from the Literature

Our Approach

- Engage NLP techniques to create intelligent triage system
- Reduce dimensions to speed-up analysis
- Use genetic algorithms to optimize
- To avoid closed dataset / malware distribution issues, use pre-extracted features!

Methodology

- Dataset employed in experiments
- Feature processing
  - Vectorization, dimension reduction
- Algorithms
  - Clustering, optimization, parameter selection

EMBER

- Endgame Malware BEnchmark for Research
- Features extracted from 2,000,000 Windows PEs
  - Samples from 2017 and 2018
  - File characteristics, byte entropy, header info...
  - **Imported functions**
  - AVClass: ground-truth labelling
- Pre-process to Apache Parquet
  - Column-oriented storage: compression, in-memory access

Vectorization

Principles

- Imports are lower-cased, joined with library, added to corpus
  - E.g.: `kernel32.dll,getlasterror`
- Use Bag-of-Words model $\rightarrow$ Count occurences
- EMBER has imports, not usage $\rightarrow$ Matrix of binary vectors

Dimension Reduction

- Need to reduce dimensions to speed-up clustering
- UMAP
  - Uniform Manifold Approximation and Projection
  - Model data with fuzzy topological structure

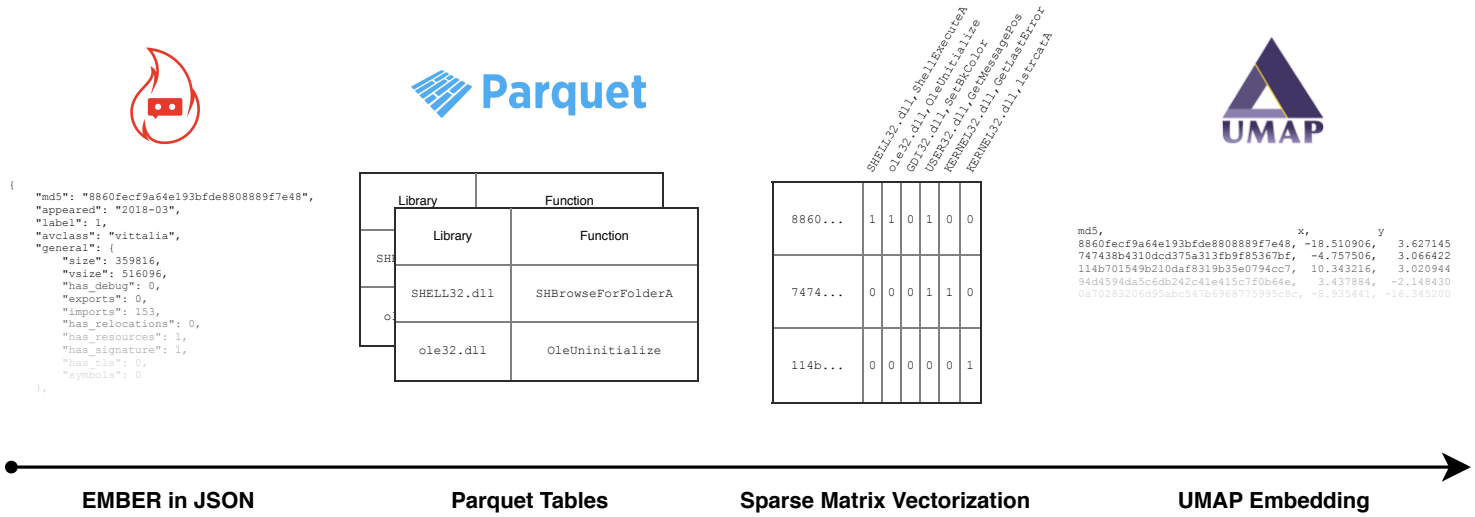
UMAP

Desirable properties

- Preserve local / global structure
- Much faster than similar techniques (e.g.: t-SNE)
- Can transform new data into existing model

Feature Processing Overview

Transformation from EMBER to embedding



Clustering Algorithms

- DBSCAN
  - Density-based spatial clustering of applications with noise
  - Well suited for non-convex clusters
  - Can mark data as noise $\rightarrow$ *Not* clustered
- OPTICS
  - Ordering Points To Identify the Clustering Structure
  - Better suited to large datasets
- Neither requires number of clusters!
- k-means
  - Good for baseline comparison

Clustering Algorithm Parameters

- Most consequential parameters per implementation docs
- DBSCAN
  - `eps` and `min_samples`
  - Neighbourhood distance between samples
  - Sample count to be considered as core point
- OPTICS
  - `max_eps` and `min_samples`
  - Maximum neighbourhood distance between samples
- *k*-means
  - `n_clusters`, `n_init`, `max_iter`
- Initialize randomly in proximity to default values

Optimization Objectives

Intuitively, we want...

- as many good clusters as possible
- containing the right points in the right clusters
- those clusters to be large

Optimization Objectives

More formally...

1. Maximize the count of high-quality clusters
   - $cos(\theta)$ over all points in cluster
   - Sum and scale to $[0.0, 1.0]$
   - High quality if scaled summed similarity (SSS) is > 0.8
2. Minimize the sum of the sum squared error (SSE) for each cluster
   - $SSE = (1 - SSS)^2$
3. Maximize the median cluster size

Optimization Algorithm

- NSGA-III
  - Non-dominated Sorting Genetic Algorithm III
  - Use reference points to achieve uniform Pareto front
- Implementation from DEAP
  - Out of box parallelization with SCOOP

DISTRIBUTED
EVOLUTIONARY
ALGORITHMS IN
PYTHON

Optimization Algorithm Parameters

- Partitions (p) of the reference plane: 4 $\rightarrow$ 16 points (H)
- Number of objectives (M): 3
- 100 generations
- Crossover probability: 1.0
- Ind. mutation prob.: 1.0
- Gene mutation prob.: 0.25

Individual Representation

- Two / three floating point numbers, per clustering algorithm
- Variable scales, so conversion necessary from individual to parameters
- E.g.: Default DBSCAN individual <0.005, 0.04>
  - First gene multiplied by 100
  - Second gene multiplied by 1000, truncated to Integer
  - eps= 0.5 and `min_samples`= 40

Experiment Setup

- Train on 2,000 EMBER samples ($\times 10$)
  - Rank individuals according to objective scores
- Statistical analysis on objectives (ANOVA)
- Test on 1,000 holdout samples
  - For each output cluster, label samples with majority class from training data

Evaluation Metrics

Given true positives (T_p), false positives (F_p), true negatives (T_n), and false negatives (F_n):

- Precision ($P = \frac{T_p}{T_p + F_p}$)
  - Relevancy of results
- Recall ($R = \frac{T_p}{T_p + F_n}$)
  - Quantity of relevant results returned
- F-Score ($F = 2 \cdot \frac{P \cdot R}{P + R}$)
  - Harmonic mean of precision and recall

Evaluation Metrics

Given a set of F-Scores for all n classes (F_C), from N data samples, and a set of support values (S) for each class:

- Weighted F-Score ($F_w = \frac{1}{N} \sum_{i=0}^{n} F_{C\,i} \cdot S_i$)
- Not every possible AVClass in training
- Mitigate achieving F-Score of 0.0 when no samples present

Evaluation Metrics

Given H is Entropy:

- Homogeneity

$$h = \begin{cases} 1 & \text{if } H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{else} \end{cases}$$

  - Clusters only contain data points from a single class

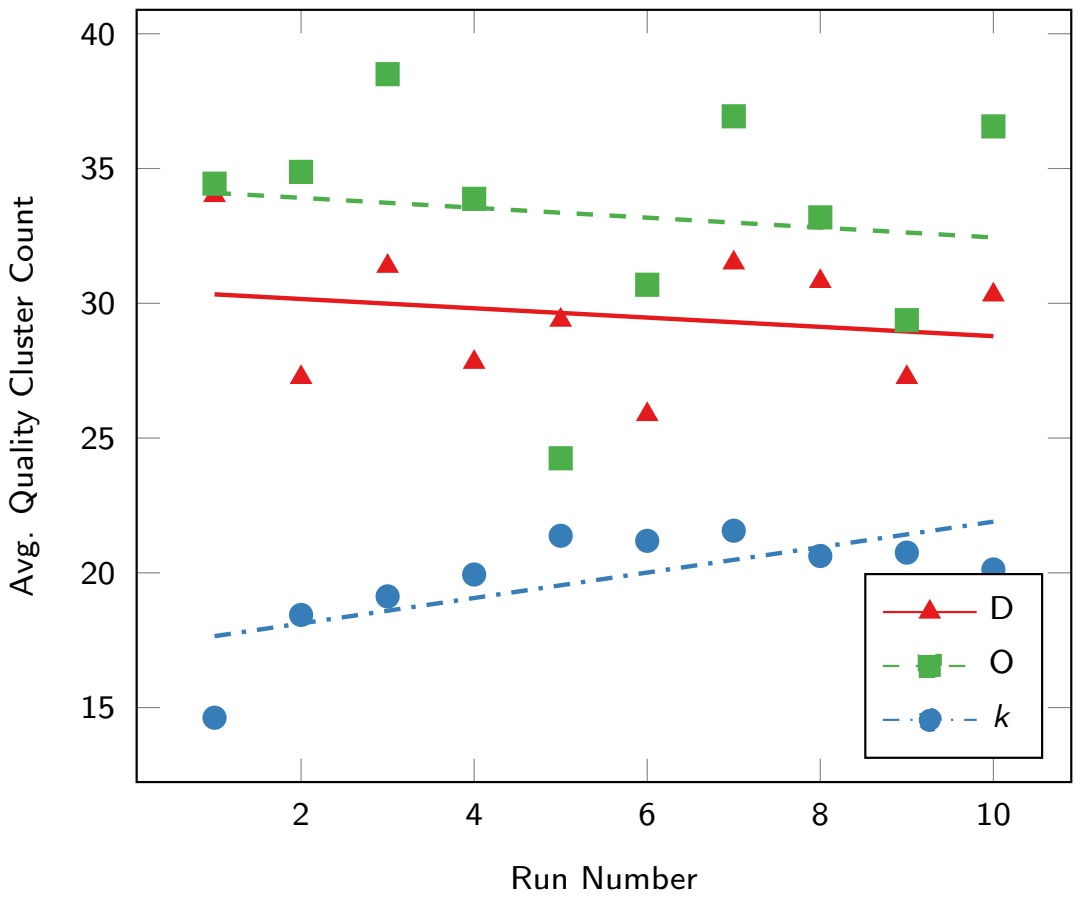
- Completeness

$$c = \begin{cases} 1 & \text{if } H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{else} \end{cases}$$

  - All data points from given class are in same cluster

- V-Measure ($v = 2 \cdot \frac{h \cdot c}{h + c}$)
  - Harmonic mean of homogeneity and completeness

Avg. Count of Quality Clusters Over 10 Runs



DBSCAN

- Relatively flat
- $\sim$ 30 quality clusters

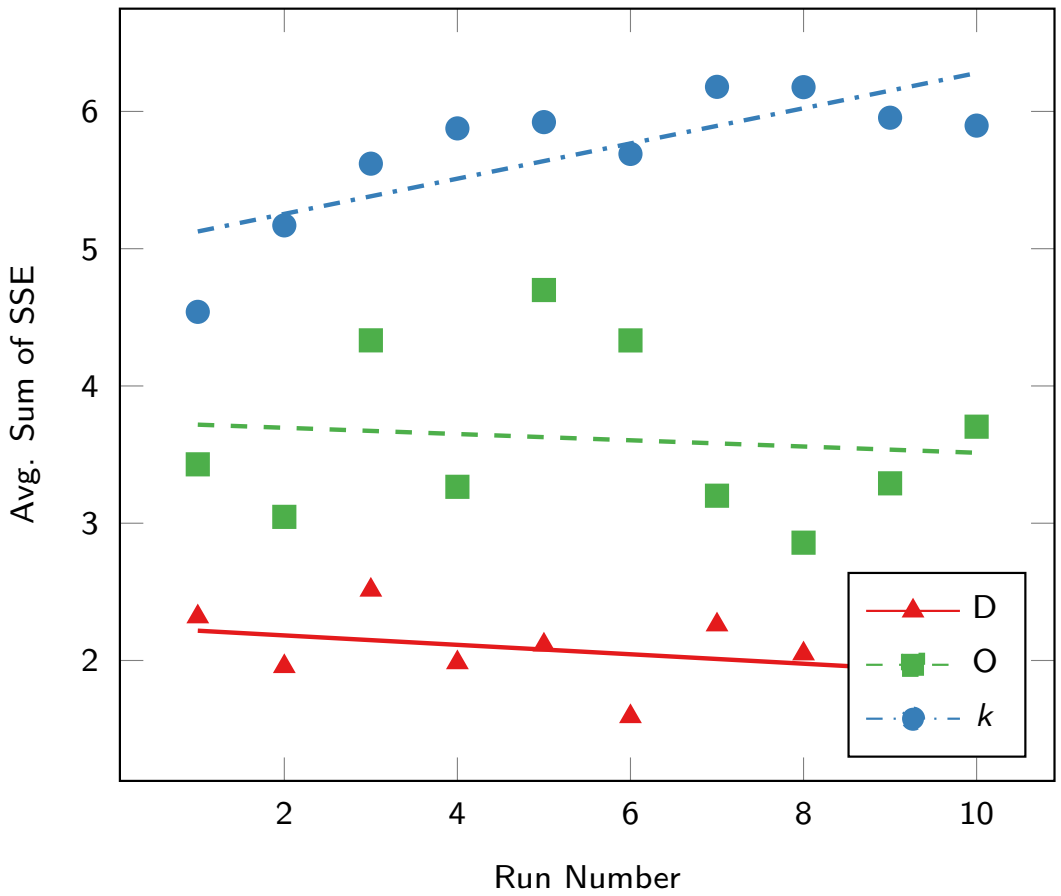
OPTICS

- Marginally higher than DBSCAN
- More variable: 25-40

k-means

- Most consistent (90% $\approx$ 20)
- Low performance

Avg. Sum of SSE Over 10 Runs



DBSCAN

- Still flat
- ~ 2.0

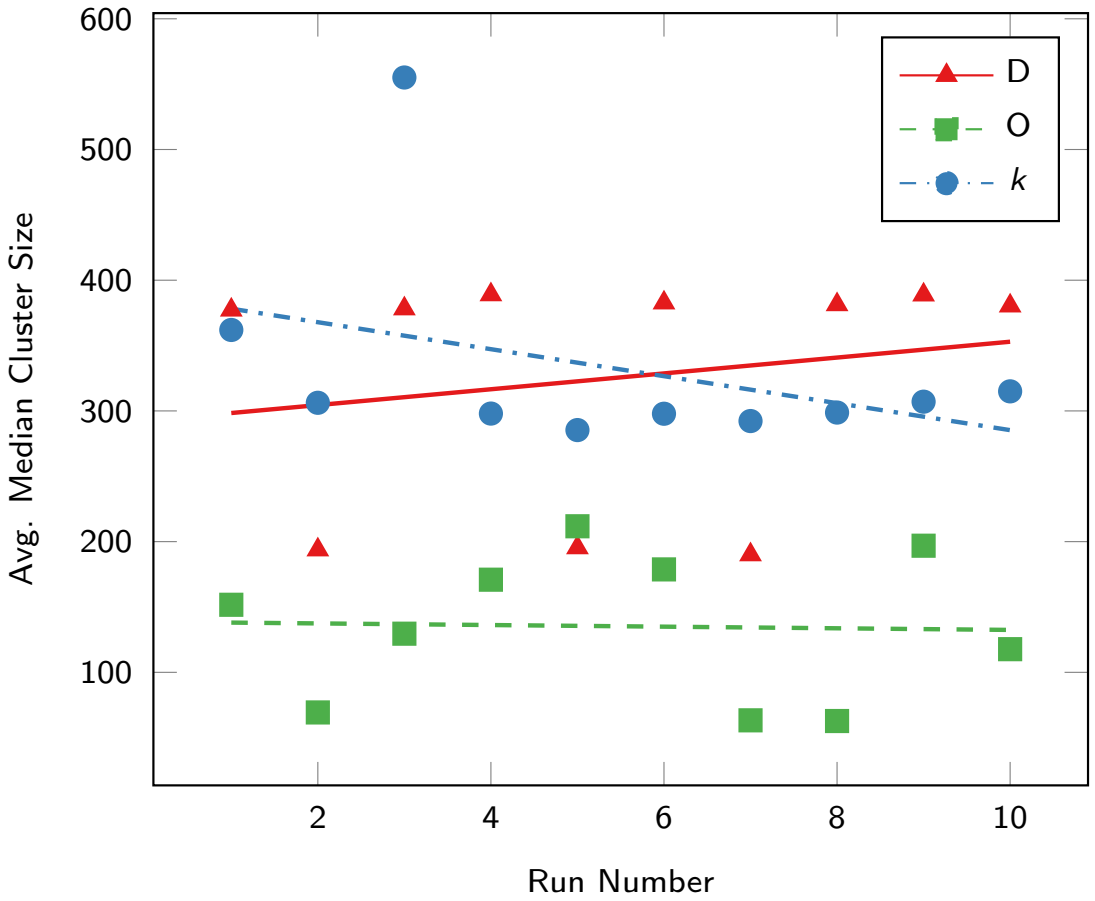
OPTICS

- Still variable
- ~ 4.0

k-means

- Noise $\rightarrow$ Error++
- ~ 6.0

Avg. Median Cluster Size Over 10 Runs



DBSCAN

- Most variable
- ~ 325

OPTICS

- Smaller spread
- ~ 140

k-means

- $\approx$ DBSCAN!
- All integer parameters

Top 3 Unique Clusterings per Algorithm

Algo.	Qual. Cnt.	Sum of SSE	Med. Size	Wgtd. F-Score	Homo.	Comp.	V-Meas.
DBSCAN 1	**65**	**3.88**	**8**	**0.668**	**0.725**	**0.862**	**0.788**
DBSCAN 2	62	3.87	9	0.678	0.748	0.872	0.805
DBSCAN 3	62	4.42	8.5	0.673	0.733	0.864	0.793
OPTICS 1	**84**	**10.83**	**8**	**0.549**	**0.675**	**0.792**	**0.729**
OPTICS 2	83	10.86	8	0.554	0.677	0.795	0.731
OPTICS 3	83	13.35	7	0.539	0.671	0.780	0.722
k-means 1	**52**	**13.62**	**15**	**0.520**	**0.593**	**0.774**	**0.671**
k-means 2	50	13.52	15	0.528	0.588	0.773	0.668
k-means 3	49	12.78	16	0.526	0.588	0.767	0.666

DBSCAN

- $>$ 60 Clusters, Avg. Error 4.06, Avg. Median Size 8.5
- Med. size much lower than average (30), small clusters
- Comp.: Groupings contain most instances of label

Top Unique Clusterings

Algo.	Qual. Cnt.	Sum of SSE	Med. Size	Wgtd. F-Score	Homo.	Comp.	V-Meas.
DBSCAN 1	**65**	**3.88**	**8**	**0.668**	**0.725**	**0.862**	**0.788**
DBSCAN 2	62	3.87	9	0.678	0.748	0.872	0.805
DBSCAN 3	62	4.42	8.5	0.673	0.733	0.864	0.793
OPTICS 1	**84**	**10.83**	**8**	**0.549**	**0.675**	**0.792**	**0.729**
OPTICS 2	83	10.86	8	0.554	0.677	0.795	0.731
OPTICS 3	83	13.35	7	0.539	0.671	0.780	0.722
k-means 1	**52**	**13.62**	**15**	**0.520**	**0.593**	**0.774**	**0.671**
k-means 2	50	13.52	15	0.528	0.588	0.773	0.668
k-means 3	49	12.78	16	0.526	0.588	0.767	0.666

OPTICS

- Count is excellent (20 more than DBSCAN)
- Error is triple DBSCAN
- More high quality clusters, more noise

Top Unique Clusterings

Algo.	Qual. Cnt.	Sum of SSE	Med. Size	Wgtd. F-Score	Homo.	Comp.	V-Meas.
DBSCAN 1	**65**	**3.88**	**8**	**0.668**	**0.725**	**0.862**	**0.788**
DBSCAN 2	62	3.87	9	0.678	0.748	0.872	0.805
DBSCAN 3	62	4.42	8.5	0.673	0.733	0.864	0.793
OPTICS 1	**84**	**10.83**	**8**	**0.549**	**0.675**	**0.792**	**0.729**
OPTICS 2	83	10.86	8	0.554	0.677	0.795	0.731
OPTICS 3	83	13.35	7	0.539	0.671	0.780	0.722
k-means 1	**52**	**13.62**	**15**	**0.520**	**0.593**	**0.774**	**0.671**
k-means 2	50	13.52	15	0.528	0.588	0.773	0.668
k-means 3	49	12.78	16	0.526	0.588	0.767	0.666

k-means

- Count is impressively close to DBSCAN
- Higher median cluster size $\rightarrow$ Higher error
- Worst overall, as expected, but not *that* bad!

Testing Setup

- DBSCAN is overall top performer
  - Supported by statistical analysis test, see paper
- 10 more runs on 2,000 samples
- Include 1,000 samples in vectorization as holdout
- Choose one from top 10% of 160 individuals
  - 60, 3.212, 9.50
- Reload embedding, fit new data, cluster, score!

Top DBSCAN Testing Results

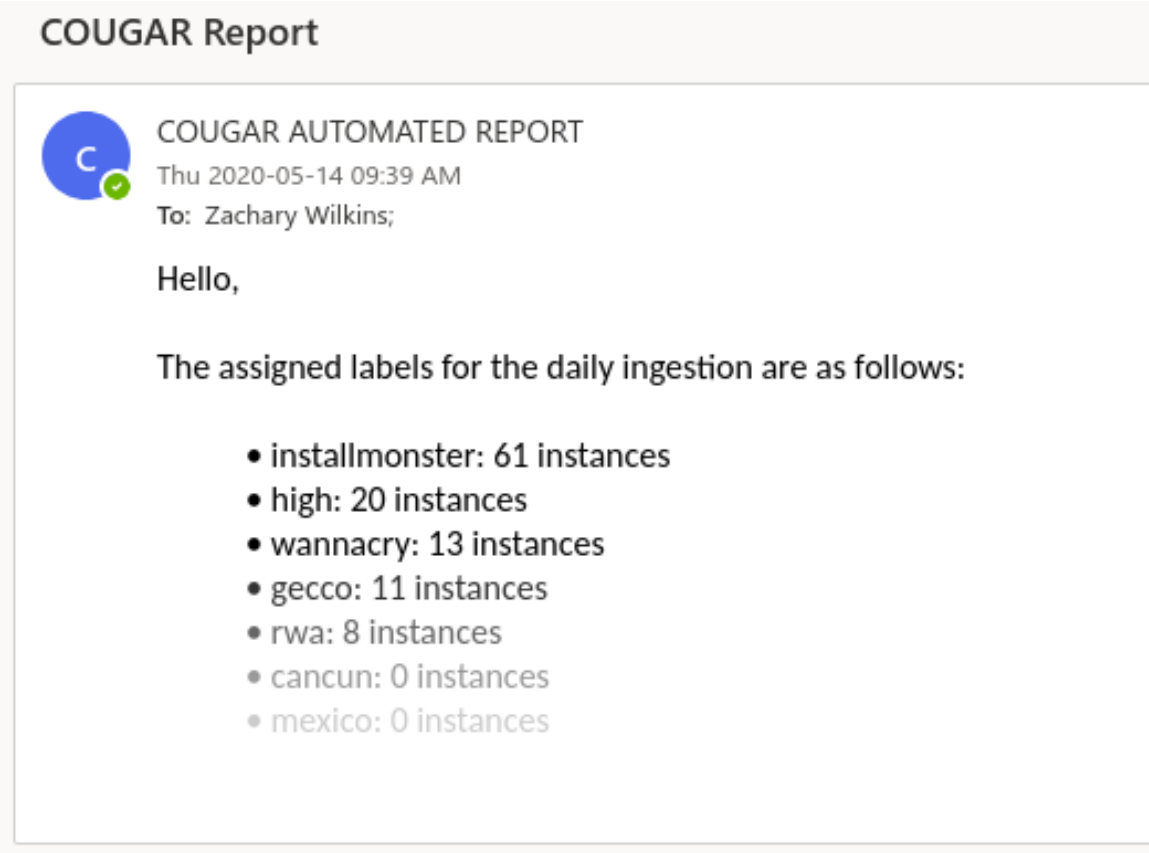
Data	Qual. Cnt.	Sum of SSE	Med. Size	Wgtd. F-Score	Homo.	Comp.	V-Meas.
Training Only	60	**3.21**	**9.5**	**0.680**	0.734	0.870	0.796
Testing Only	–	–	–	0.671	**0.821**	**0.922**	**0.868**
Combined	**74**	5.42	9.0	0.673	0.727	0.870	0.792

- Testing and training very similar: ~ 0.67
- Testing only: highest reported homo., comp., V-Meas.
  - Fairly homogeneous, highly complete
- Takeaway: very small performance drop from training to testing!

An Illustrated Example

- Massive influx of new samples
- Analysts need to triage the malware
- Does COUGAR help?

A typical day at work...


COUGAR Report

C

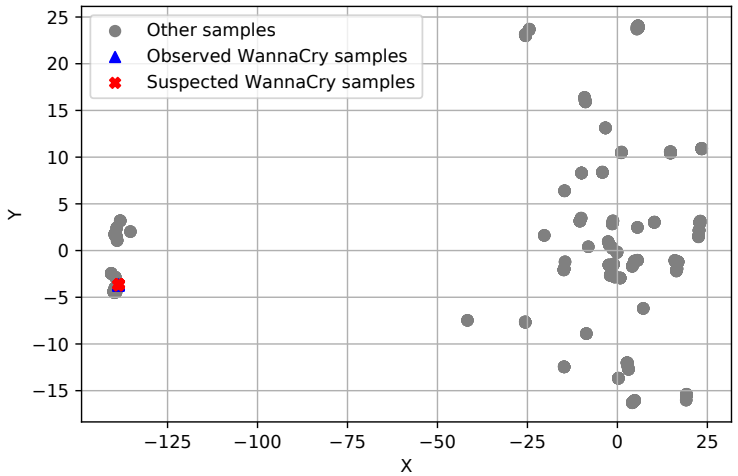
COUGAR AUTOMATED REPORT
Thu 2020-05-14 09:39 AM
To: Zachary Wilkins;

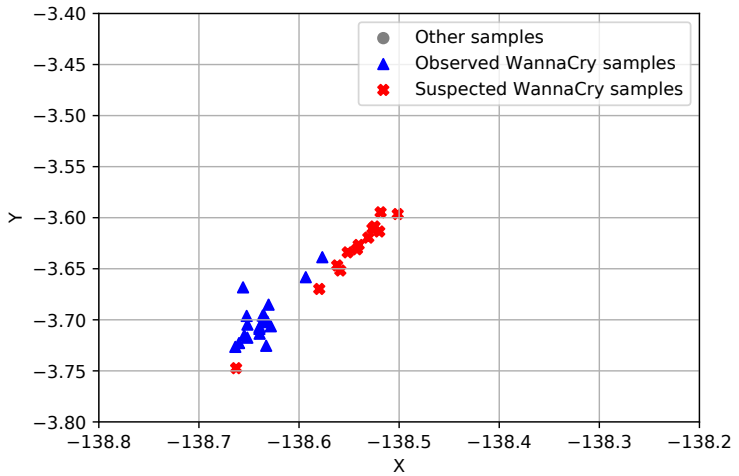
Hello,

The assigned labels for the daily ingestion are as follows:

- installmonster: 61 instances
- high: 20 instances
- wannacry: 13 instances
- gecco: 11 instances
- rwa: 8 instances
- cancun: 0 instances
- mexico: 0 instances

Load the COUGAR!





But did it work?

- Yes!
- 12 of 13 samples were indeed WannaCry ransomware
- Last one?
  - "Behaves like ransomware" — McAfee Gateway Edition

Research Objective

Create a system that:

- Uses observed behaviours to cluster malware
- Predict unseen malware based on seen malware

Contributions

COUGAR

- Reduce high-dimensional behavioural data
- Optimize clustering with MOGA
- Label unknown malware

Experiments

- Evolve optimal clustering configurations
- Predict holdout samples
- Apply to real world scenario

Conclusions

Clustering Algorithms

- OPTICS: Highest number of clusters
- DBSCAN: Lowest error
- k-means: Still here!

Predicted Labels

- Weighted F-Scores need more work (≤ 0.68)
- Training / testing very close!
- When labelling improves, so too will predictions

Conclusions

Real-World Scenario

- Analyst identified the WannaCry samples
- 12 / 13 correct!

Future Work

Increase Label Metrics

- Fuzzy clustering logic
- Multi-label clusters

Increase Speed

- Scale up system size
- Distance calculation considerations

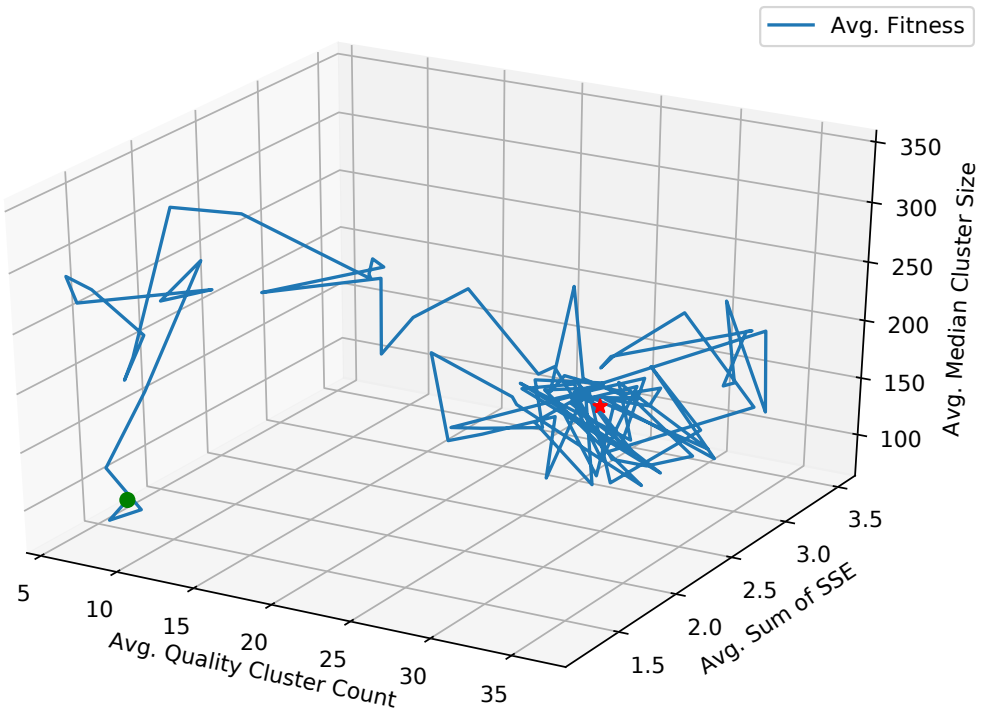
Thank you!

- Happy to answer your questions at this time
- Source code: `https://github.com/znwilkins/cougar`

Contact Info

Zachary Wilkins (zachary.wilkins@dal.ca)
Nur Zincir-Heywood (zincir@cs.dal.ca)
Dalhousie NIMS Lab (https://projects.cs.dal.ca/projectx/)

Avg. objectives over DBSCAN run



- All three algorithms: Quick convergence to a local optimum, regular scrubbing until completion.

Analysis of Variance

ANOVA on final avg. of objectives over 10 runs

- 5.83e-10, 5.52e-15, 3.61e-06
- $P \ll 0.05$

Tukey's Honestly Significant Difference (HSD) Test

Obj.	Algo. 1	Algo. 2	$\Delta\mu$	P-Adj.	Reject?
Qual. Cnt.	DBSCAN	k-means	-9.7812	0.001	True
Qual. Cnt.	DBSCAN	OPTICS	3.7125	0.0302	True
Qual. Cnt.	k-means	OPTICS	13.4938	0.001	True
Sum of SSE	DBSCAN	k-means	3.6401	0.001	True
Sum of SSE	DBSCAN	OPTICS	1.5532	0.001	True
Sum of SSE	k-means	OPTICS	-2.0869	0.001	True
Med. Size	DBSCAN	k-means	6.0812	0.9	False
Med. Size	DBSCAN	OPTICS	-190.4063	0.001	True
Med. Size	k-means	OPTICS	-196.4875	0.001	True

Confirms earlier observations

- Count: OPTICS $>$ DBSCAN $\gg$ k-means
- Error: DBSCAN $<$ OPTICS $\ll$ k-means
- Size: DBSCAN $\approx$ k-means $\gg$ OPTICS

Malware Clustering Survey

Faridi et al., ICDMW, 2018

- Find optimal algos, parameters, features
- Run 5,600 malware in Cuckoo sandbox: 94 clusters
  - Collect network requests, syscalls
- Extract features, group into categories
- Vectorize most important strings using TF–IDF
- Best algos: DBSCAN, spectral, hierarchical

Clustering on API Calls

Duarte-Garcia et al., IWBF, 2019

- System to characterize malware behaviour
- Run malware in Cuckoo sandbox
- Remove noise: calls in 90% of malware
- TF–IDF as input to k-means, then classifiers
  - Evaluate cluster quality using silhouette coefficient
- Best algo: Gradient-boosting decision trees

Clustering on API Calls

Lee et al., The Journal of Supercomputing, 2018

- System to guide malware analysis triage
- Run malware in Cuckoo sandbox
- Use API calls to create bigrams
- Compare bigrams using cosine similarity
  - If above threshold t, join as graph
- 210 groups of malware found
  - Top 10: 43% of samples

Clustering on API Calls

Pirscoveanu et al., CyberSA, 2016

- System to cluster malware by behaviour
- Run malware in Cuckoo sandbox, labelled by VirusTotal
- Extract features from API calls
  - Successful/unsuccessful calls, return codes
- Use PCA to reduce, gap statistics to choose cluster count
- Project each sample with SOM matching cluster count
- Results on top five Microsoft AV labels
  - 80% on 2x2 map
  - 70% on 4x4 map

Distributed Clustering

BitShred, Jang et al., CCS '11, 2011

- Malware triage to identify mutants
- Create malicious fingerprints by hashing many features
- Using approx. Jaccard similarity, cluster hierarchically
- Co-cluster to discover correlated features in malware sub-groups
- Parallelize across Hadoop cluster to scale
- Approach is 'much faster' than SOTA with comparable accuracy

Fuzzy Clustering

Acampora et al., FUZZ-IEEE, 2018

- Discover Android malware phylogeny
- Run 3,000 samples on Android phone
  - Collect syscall traces, represents behaviour
- Use process modelling language to transform to fingerprints
- Fingerprints $\rightarrow$ Dissimilarity matrix $\rightarrow$ FANNY
- Compare to reference clustering
  - Precision: 0.8
  - Recall: 0.73