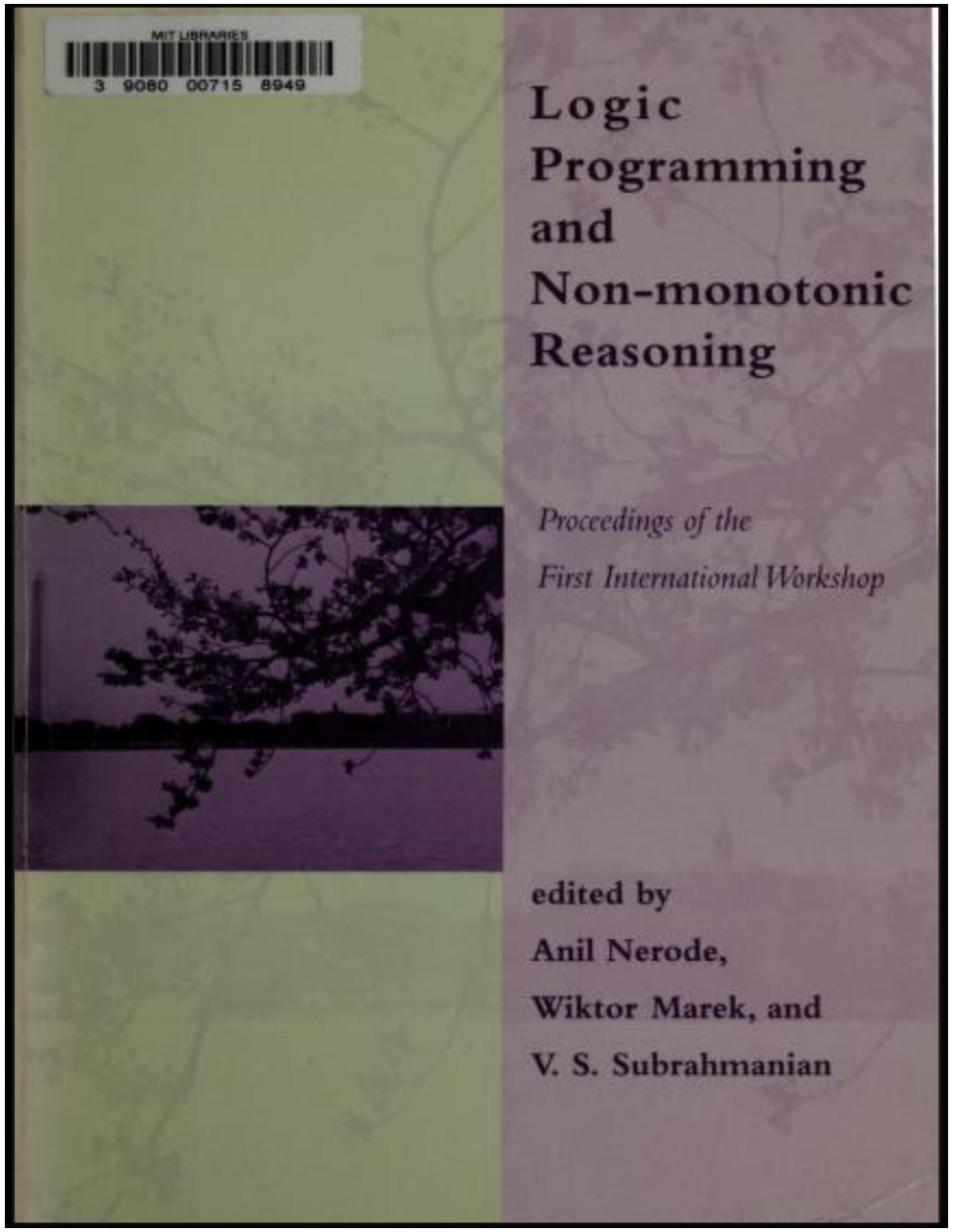
Logic, Machine Learning & Security

V.S. Subrahmanian
Dartmouth College
vs@dartmouth.edu

Joint work with many outstanding students, postdocs, and faculty colleagues.

LPNMR - 1991

- LPNMR started in 1991 with the first international workshop held in Washington DC.
- Founded by
  - Anil Nerode *Cornell U.*
  - Wiktor Marek *U. Kentucky*
  - V.S. Subrahmanian *U. Maryland*
- Proceedings published by MIT Press.
- Based on a workshop held in 1990 chaired by Nerode, Marek, and VS at the North American Conference on LP held in Austin, TX.



Talk Outline

- Temporal Probabilistic Rules
  - A. Dekhtyar, M. Dekhtyar and V.S. Subrahmanian. Temporal Probabilistic Logic Programs, Proc. 1999 Intl. Conf. on Logic Programming, pp. 109-123 New Mexico, Nov. 1999.
- BEEF: Balanced English Explanations of Forecasts
  - Using probabilistic logic to explain forecasts made by virtually any machine learning algorithm
- A Probabilistic Logic of Cyber Deception
  - Using a combination of logic and game theoretic methods to deceive an adversary who has penetrated an enterprise network.

TP-Rules for Terrorist Group Behavior Models

- Terror groups were reduced to a spreadsheet
  - Rows corresponding to months
  - Columns corresponding to
    - Environmental variables
    - Different types of attacks
- TP-rules of the form

$$C_1 \& \ \ldots \& \ C_n \rightarrow^d A$$

Where C_i is a condition on an environmental variable column, $d \geq 0$ is an integer, and A is an attack type.

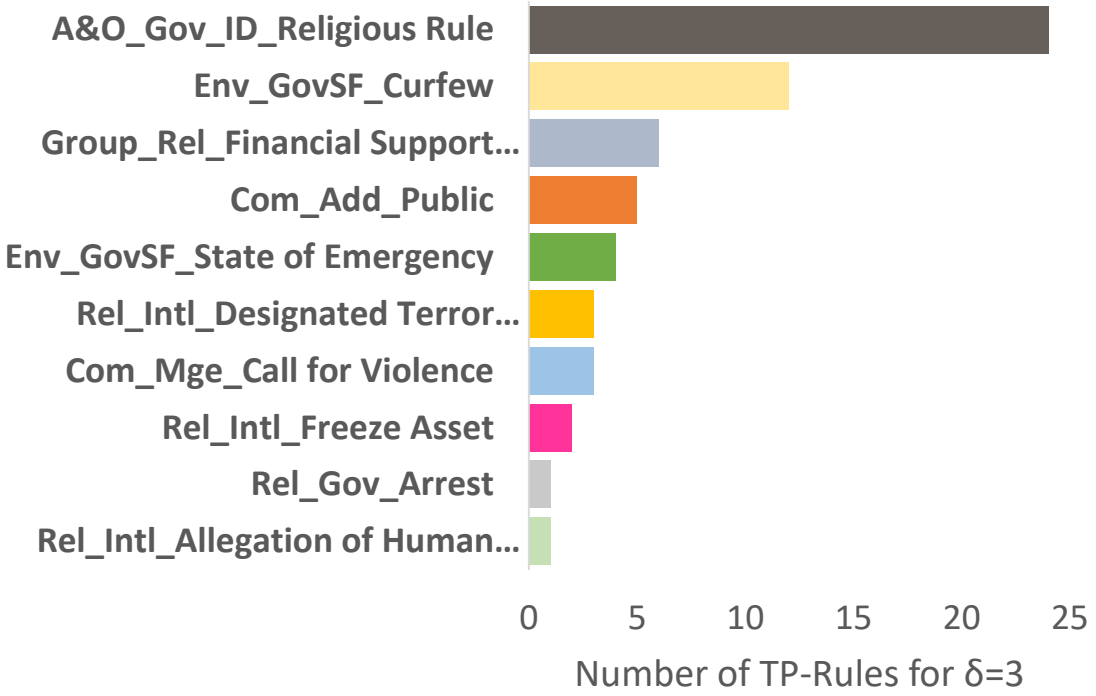
- Precondition $C = C_1 \& \ldots \& \ C_n$

- Rule statistics
  - Confidence: **P**(A@m+d|C@m)
  - Negative Confidence: **P**(A@m+d|~C@m)
  - Inverse Confidence: **P**(C@m|A@m+d)
  - Support: **P**(A@m+d & C@m)
  - Negative Support: **P**(~A@m+d & ~C@m+d)

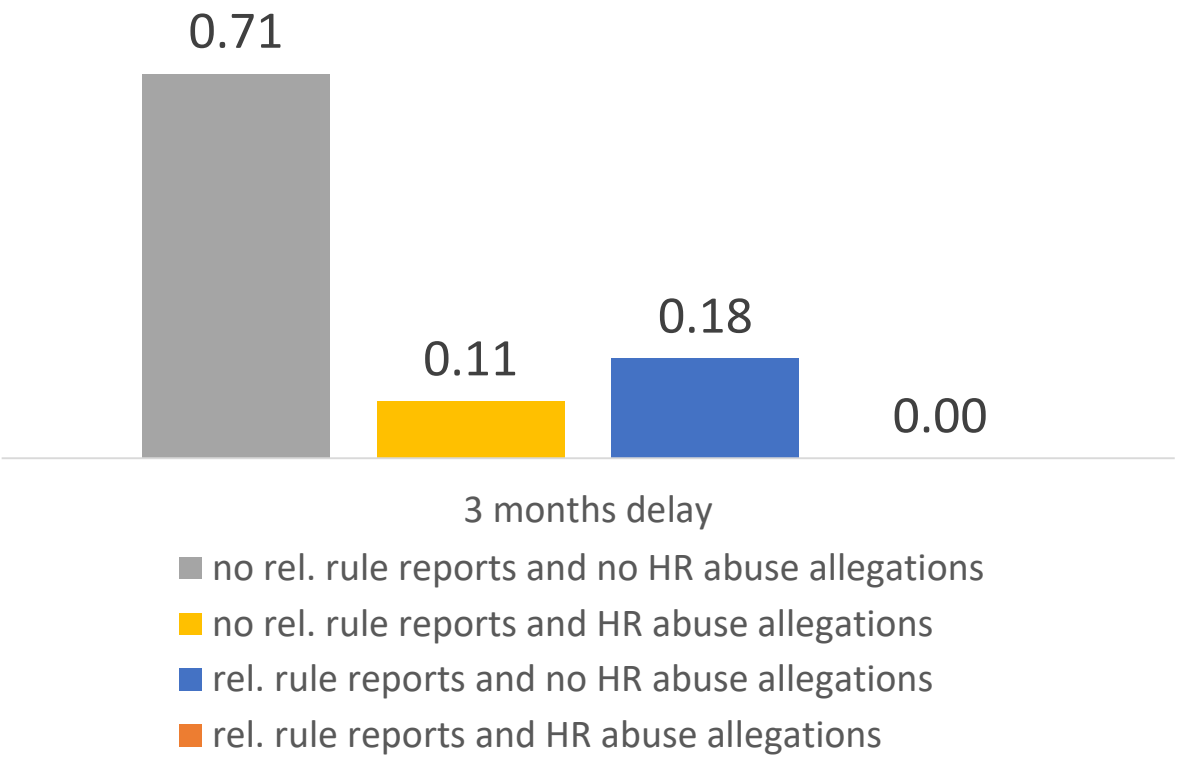
TP-Rules to Express Behavioral Models of Terrorist Groups -- Suicide Bombings

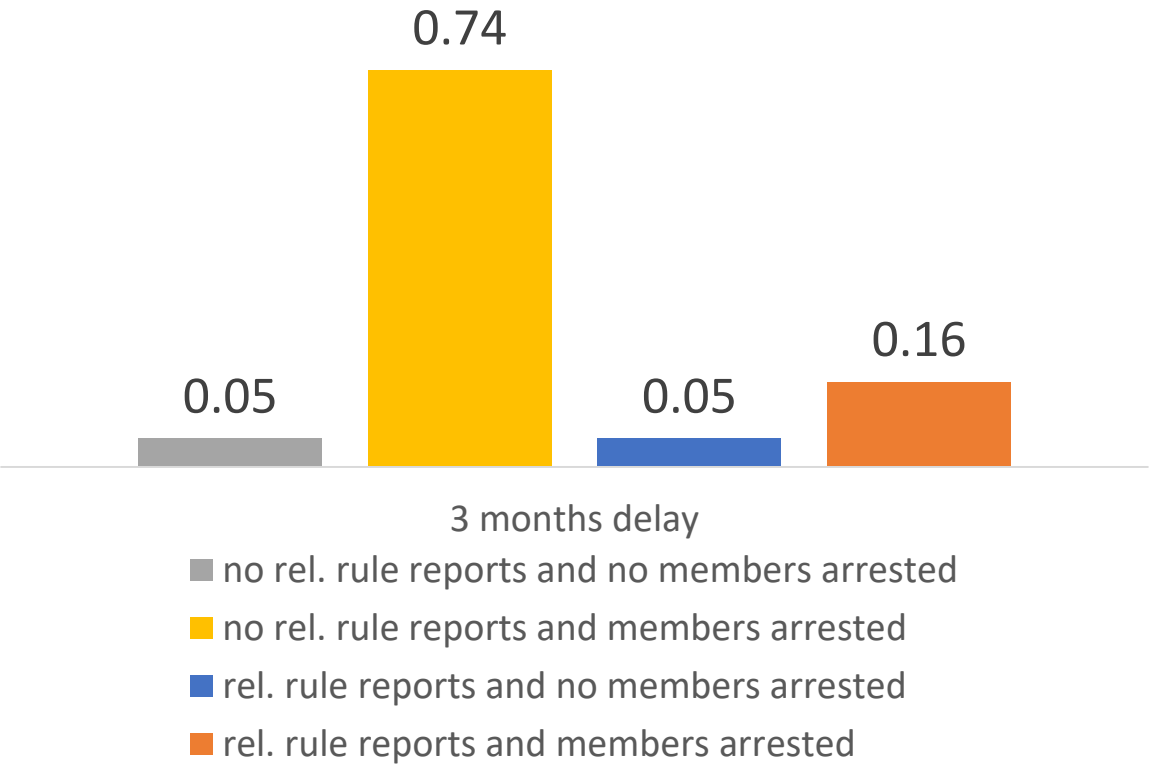
- Boko Haram carries out suicide bombings in month *m* when
  - There are no reports of calls for religious rule in month *m-3* and
  - any *one of the following was true in month m-3:*
    - BH members are arrested or [s=32%,c=65%,nc=23%, ic=74%]
    - International organization(s) reportedly did not make allegations of human rights violations against BH [s=32%,c=65%,nc=23%, ic=71%]

10 Most Important Features for Suicide Bombings



Probability that BH carries out suicide bombings





Success

Apr 2008: Publish first ever predictive behavior model of any terrorist group (Hezbollah)

Oct 2008: Beirut Daily Star publishes scathing article on our work

June 2009: Lebanese elections held, our predictions hold

Dec 2009: We publish our article in *Foreign Policy*

2012: Published first ever comprehensive predictive behavior model of any terrorist group (Lashkar-e-Taiba)…..

Success

2012: Published first ever comprehensive predictive behavior model of any terrorist group (Lashkar-e-Taiba)…..

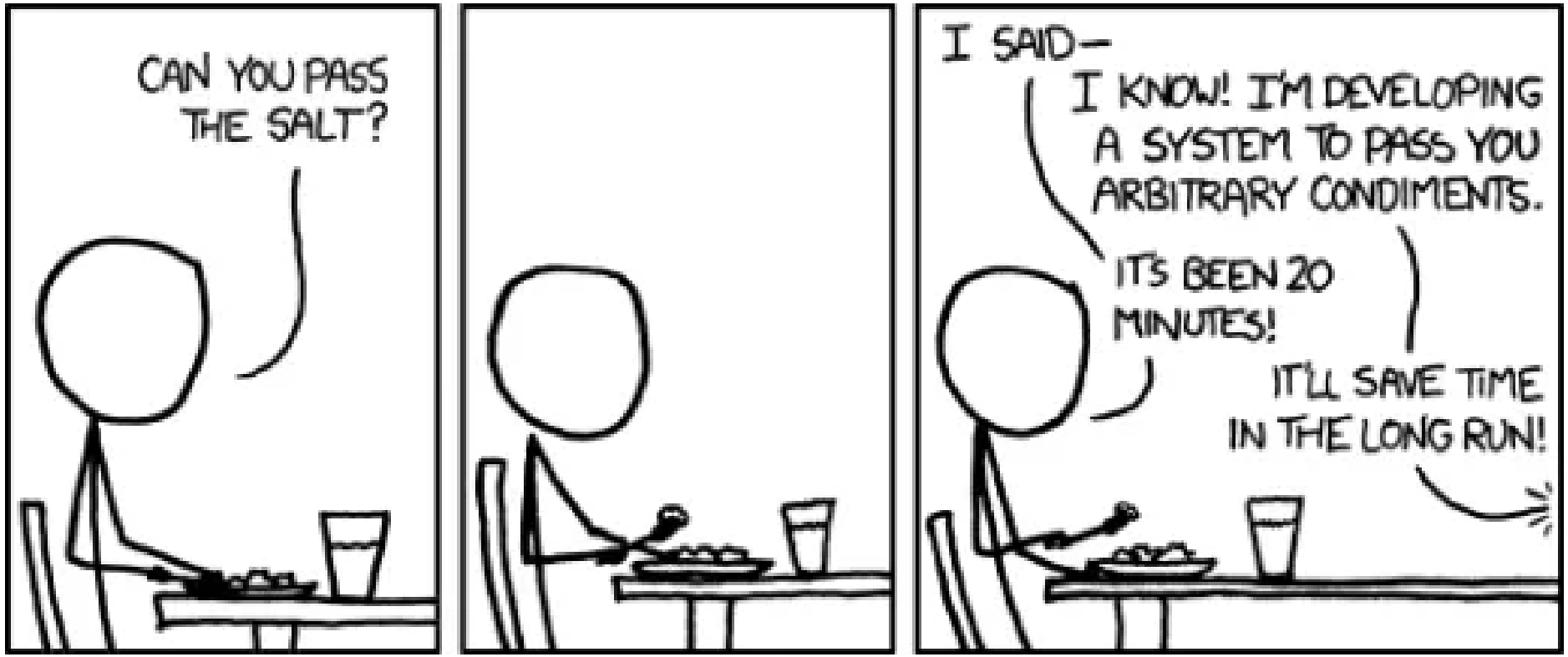
2012: Started putting out regular forecasts

2014: Published second ever comprehensive predictive behavior model of a terrorist group (Indian Mujahideen). Put out several public forecasts.

2014-2020: Policy Analytics Generation Engine (PAGE) developed. Developed methods to shape terrorist group behavior through disclosures. Helped shape relevant national security policies.

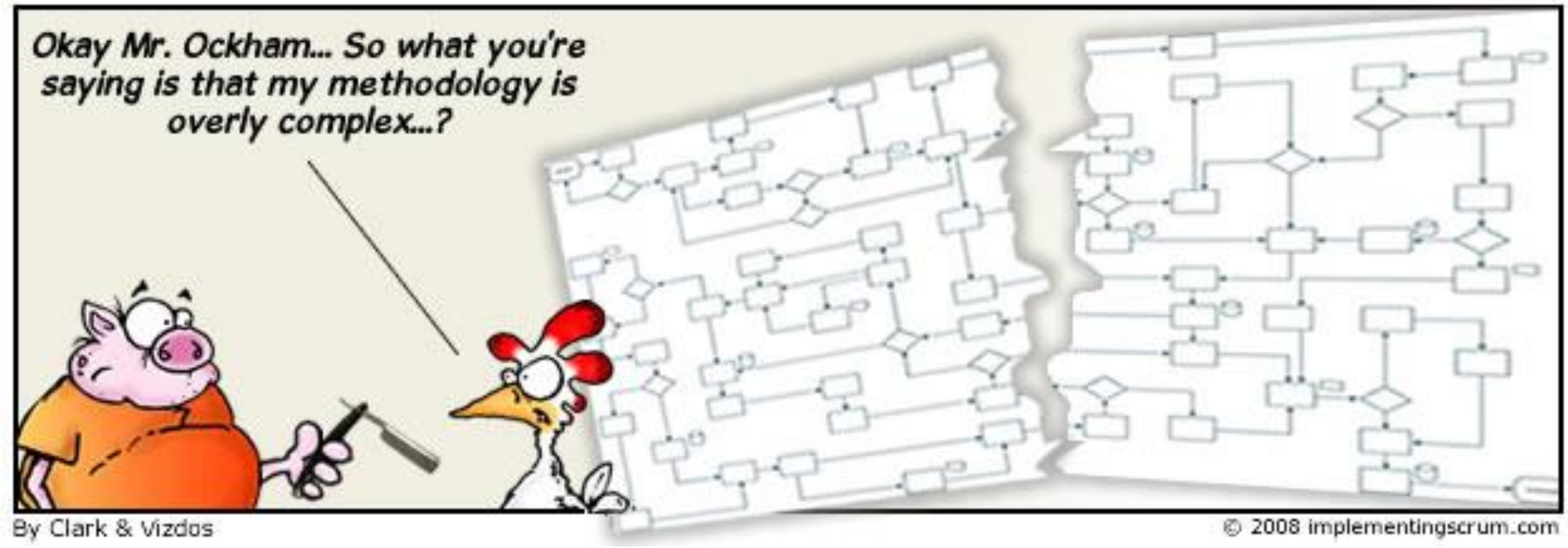
2019-2020: Hope to publish first ever comprehensive predictive behavior model of a rogue intelligence agency (ISI) and a comprehensive behavior model of Boko Haram

KISS: Keep it Simple Stupid



Source: https://www.loadingdeveloper.com/brick-by-brick-planning/

KISS: Keep it Simple Stupid



Source: https://www.implementingscrum.com/2008/05/06/kiss-keep-it-simple-stupid/

Talk Outline

- Temporal Probabilistic Rules
  - A. Dekhtyar, M. Dekhtyar and V.S. Subrahmanian. Temporal Probabilistic Logic Programs, Proc. 1999 Intl. Conf. on Logic Programming, pp. 109-123 New Mexico, Nov. 1999.
- BEEF: Balanced English Explanations of Forecasts
  - Using probabilistic logic to explain forecasts made by virtually any machine learning algorithm
- A Probabilistic Logic of Cyber Deception
  - Using a combination of logic and game theoretic methods to deceive an adversary who has penetrated an enterprise network.

BEEF: Balanced English Explanations of Forecasts

Chiara Pulice
V.S. Subrahmanian

Dartmouth College

vs@dartmouth.edu
www.cs.dartmouth.edu/~vs/

Sachin Grover
Carnegie Mellon University

Gerardo I. Simari
Universidad Nacional Del Sur, Argentina

Reference

Grover, S., Pulice, C., Simari, G. I., & Subrahmanian, V. S. (2019). BEEF: Balanced English Explanations of Forecasts. *IEEE Transactions on Computational Social Systems*, *6*(2), 350-364, April 2019.

https://ieeexplore.ieee.org/abstract/document/8668423

Goals

- Automatically extract **balanced explanations** from forecasts generated by <u>any classifier</u>
  - Traditional classifiers: random forest, naïve bayes, decision trees, SVM,….
  - Deep classifiers: perceptrons, CNNs, RNNs,….
- Provide these explanations in plain English.
- Include not only why the forecast might be correct, but also why it could be wrong.

Evidence for/against the prediction can help decision makers make informed decisions based on the predictions

Why are explanations of predictions important?

- **Invented Features**
  - Some ML algorithms (e.g. RBM, deep learning) introduce new features that are artificially created.
  - Others (e.g. SVM) use “kernel” tricks to map the original set of features to a new set of features, many artificial.

Domain experts with deep knowledge of their data do not understand these new features.

- **Complex Separators**
  - Geometric intuition behind complex ML separators is hard to explain because of the complexity of the separators.
  - Sigmoid functions, cubic separators, logistic regression based classification, are hard to explain to normal human beings.

Combining complexity of separators and newly minted features => CHALLENGES in EXPLAINING FORECASTS

The Balanced Explanation Problem

- The *pseudo*-training data to predict an event can be thought of as a set *P* of objects

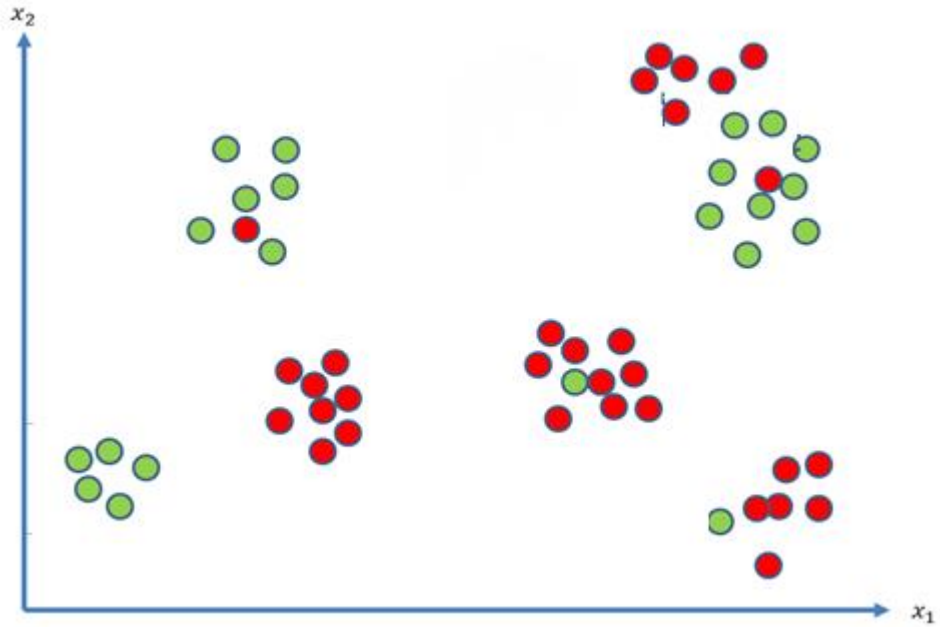
$$p_i = (\overrightarrow{x_i}, y_i)$$

where

$\overrightarrow{x_i}$ is the vector of features associated with object p_i and

$y_i \in \{0,1\}$ is the **prediction** made for object p_i by some classifier (the one we want to explain).

Note that in the original training set, the y_is would represent the ground truth.



We can color the point corresponding to $\overrightarrow{x_i}$ green if $y_i = 0$ (the event does not occur) and red if $y_i = 1$ (the event occurs)

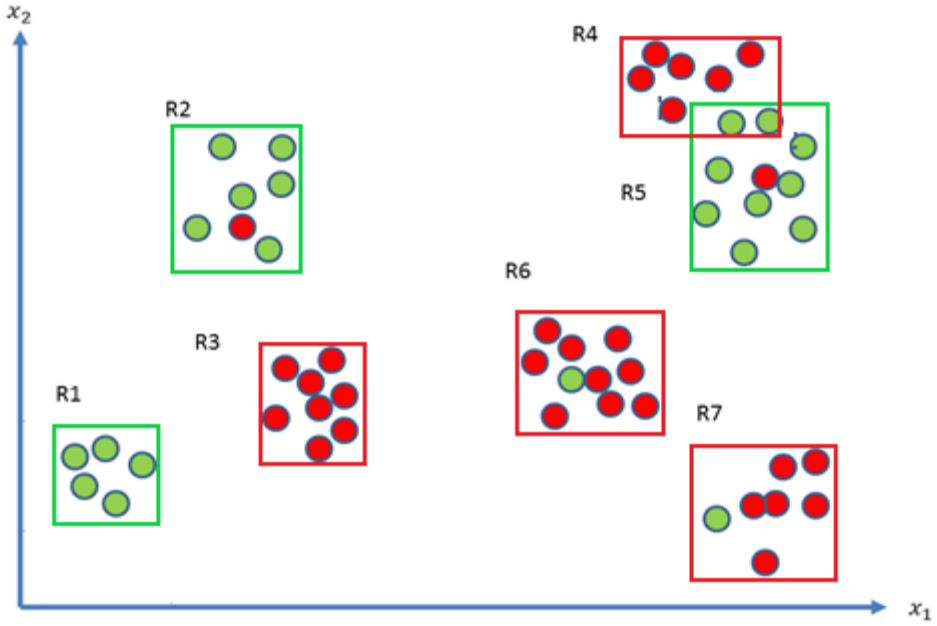
The Balanced Explanation Problem

- Split *P* into two disjoint sets of sets

$$\boldsymbol{C^0} = \{C_1, \dots, C_m\}$$
$$\boldsymbol{C^1} = \{C_{m+1}, \dots, C_h\}$$

  - Green clusters denote class green
  - Red clusters denote class red

- **but…..these sets of clusters must jointly satisfy various desirable properties**



The Balanced Explanation Problem: Desired Properties

- Each cluster is enclosed by a "box" (hyper-rectangle). Hyper-rectangles have the form $\ell_1 \leq x_1 \leq u_1 \& \ \ldots \& \ \ell_n \leq x_n \leq u_n$
- **Minimal overlap:** Green boxes and red boxes must overlap as little as possible
- **Maximal purity:** Each box must mostly consist of objects of one color
- **Maximal inclusion:** Most points in *P* must belong to at least one cluster.
- **Simplicity (Occam's Razor):** We shouldn't have too many boxes.

The Balanced Explanation Problem

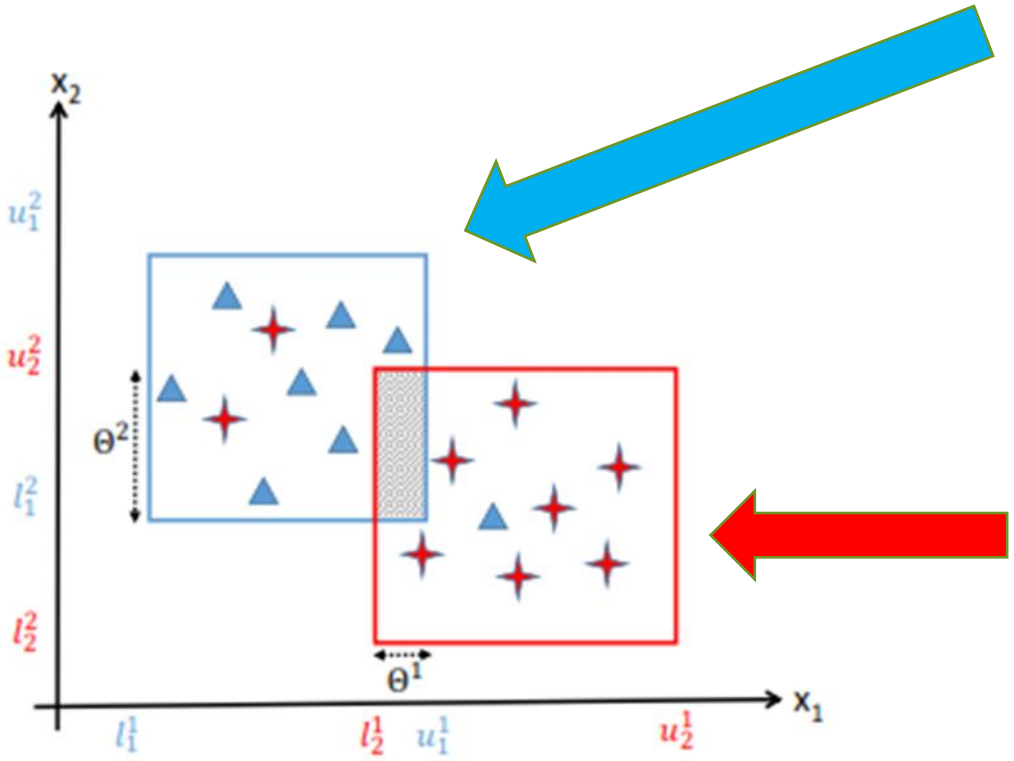
- Find a set of at most *h* boxes such that:
  - purity of each box exceeds a threshold
  - the overlap between a blue box and a red box is always below a threshold
  - overall inclusion exceeds a given threshold.

Theorem: The Balanced Explanation Problem is NP-complete.

Reduction from MAX Independent Set.

We defined BEEF, an efficient algorithm to find near-optimal balanced explanations fast.

Hyper-Rectangles Can be Explained in English !



When x_1 lies between l_1^1 *and* u_1^1 and x_2 lies between l_1^2 and u_1^2, there is an 89% probability that the object is **blue**.

When x_1 lies between l_2^1 *and* u_2^1 and x_2 lies between l_2^2 and u_2^2, there is a 90% probability that the object is **red**.

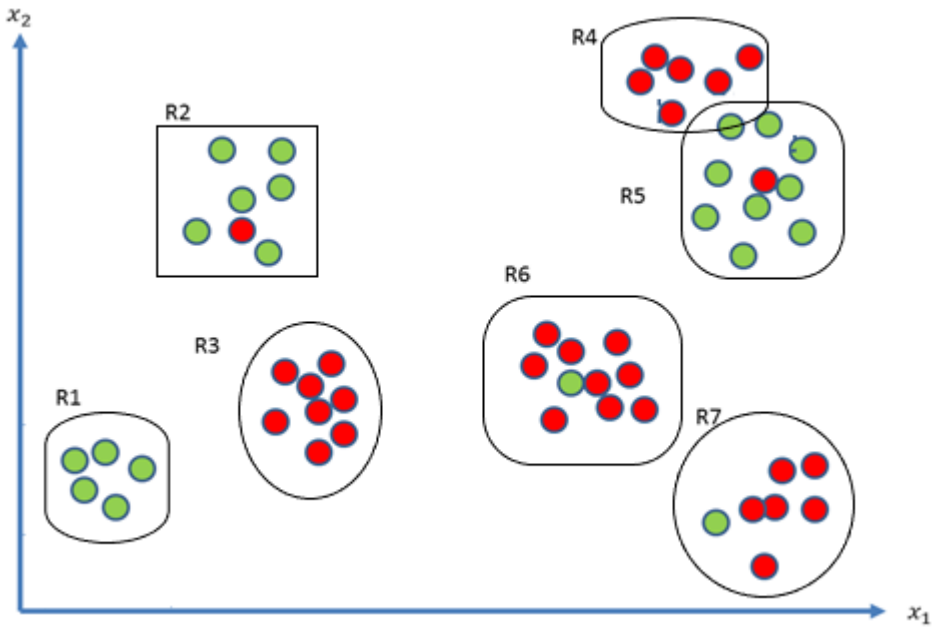
BEEF: Main Steps

Algorithm 1 BEEF

Input: Disjoint sets $P^0, P^1 \in P$, step η, and number of dimensions $\overline{n}$
Output: Sets of clusters $\mathbb{C}^0$, $\mathbb{C}^1$, and $n \times h$ active dimensions matrix B

// Independently run a clustering algorithm on the two training sets
$L^0 \leftarrow \mathit{findClusters}(P^0)$;
$L^1 \leftarrow \mathit{findClusters}(P^1)$;
// Compute the minimum bounding hyper rectangles
$\mathbb{C}^0 \leftarrow \mathit{MBR}(P^0, L^0)$;
$\mathbb{C}^1 \leftarrow \mathit{MBR}(P^1, L^1)$;
// Reduce the overlap between each pair of different-colored clusters
for $C_w \in \mathbb{C}^0$ **do**
  for $C_z \in \mathbb{C}^1$ **do**
    if $\Theta^k_{wz} \geq 0 \ \forall k \in [1, n]$ **then**
      $\langle C'_w, C'_z \rangle \leftarrow \mathit{ReduceOverlap}(C_w, C_z, \eta)$;
      $C_w \leftarrow C'_w$;
      $C_z \leftarrow C'_z$;
    end if
  end for
end for
// Select a subset of features for each cluster
$B \leftarrow \mathit{FeatureSubsetSelection}(\mathbb{C}^0, \mathbb{C}^1, \overline{n})$;
return $\langle C^0, C^1, B \rangle$.

Step 1: Take training data (classes 0 “green”, 1 “red”) and <u>independently</u> cluster 0-items and 1-items.



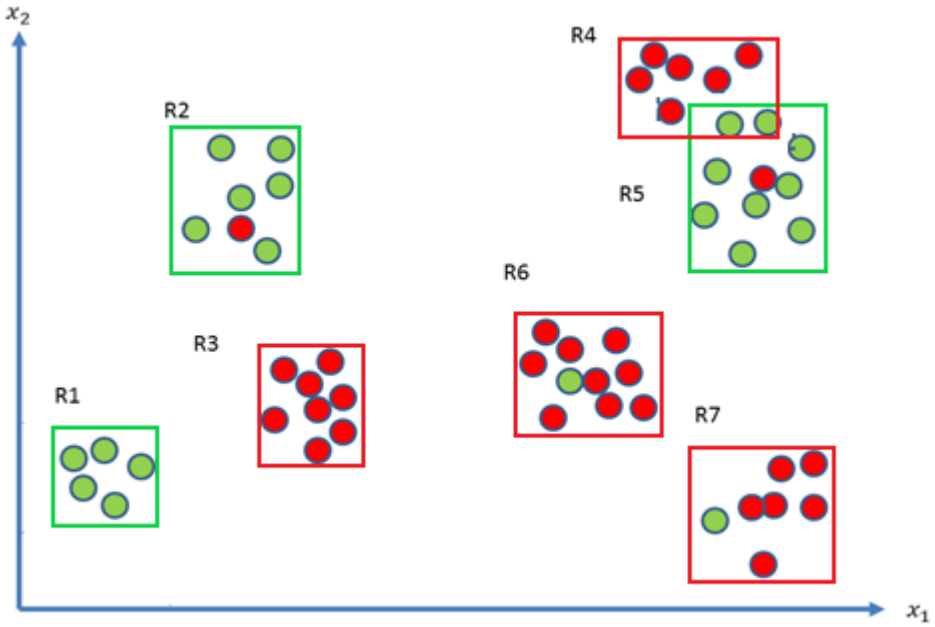
BEEF: Main Steps

Algorithm 1 BEEF

Input: Disjoint sets $P^0, P^1 \in P$, step η, and number of dimensions $\overline{n}$
Output: Sets of clusters $\mathbb{C}^0$, $\mathbb{C}^1$, and $n \times h$ active dimensions matrix B

// Independently run a clustering algorithm on the two training sets
$L^0 \leftarrow \mathit{findClusters}(P^0)$;
$L^1 \leftarrow \mathit{findClusters}(P^1)$;
// Compute the minimum bounding hyper rectangles
$\mathbb{C}^0 \leftarrow \mathit{MBR}(P^0, L^0)$;
$\mathbb{C}^1 \leftarrow \mathit{MBR}(P^1, L^1)$;
// Reduce the overlap between each pair of different-colored clusters
for $C_w \in \mathbb{C}^0$ **do**
  for $C_z \in \mathbb{C}^1$ **do**
    if $\Theta^k_{wz} \geq 0 \ \forall k \in [1, n]$ **then**
      $\langle C'_w, C'_z \rangle \leftarrow \mathit{ReduceOverlap}(C_w, C_z, \eta)$;
      $C_w \leftarrow C'_w$;
      $C_z \leftarrow C'_z$;
    end if
  end for
end for
// Select a subset of features for each cluster
$B \leftarrow \mathit{FeatureSubsetSelection}(\mathbb{C}^0, \mathbb{C}^1, \overline{n})$;
return $\langle C^0, C^1, B \rangle$.

Step 2: For each cluster generated in Step 1, build a green/red minimum bounding hyper-rectangle.



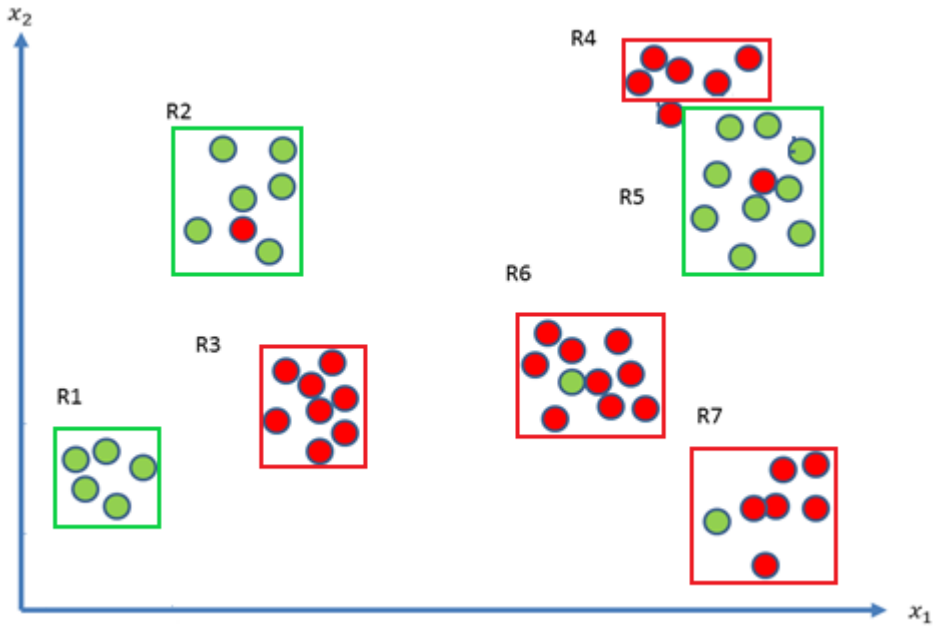
BEEF: Main Steps

Algorithm 1 BEEF

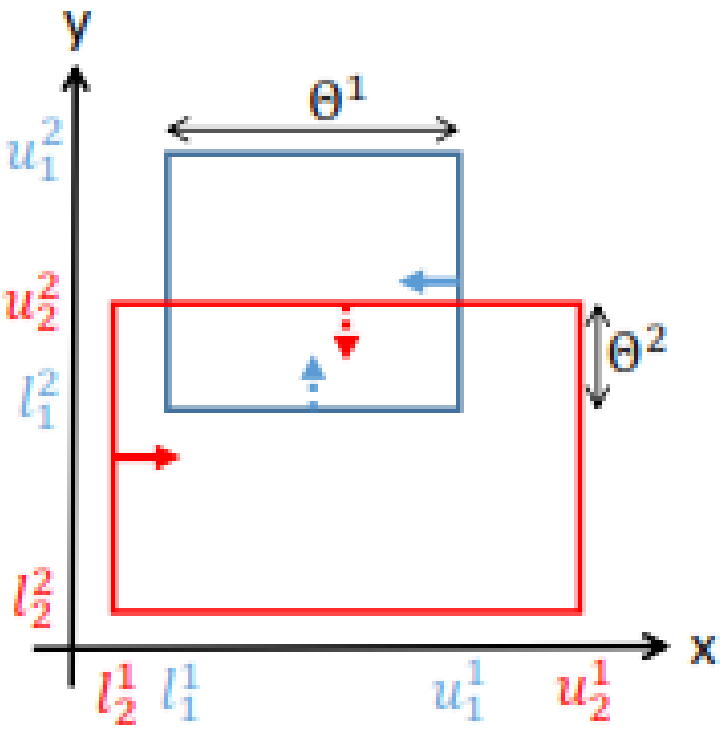
Input: Disjoint sets $P^0, P^1 \in P$, step η, and number of dimensions $\overline{n}$
Output: Sets of clusters $\mathbb{C}^0$, $\mathbb{C}^1$, and $n \times h$ active dimensions matrix B

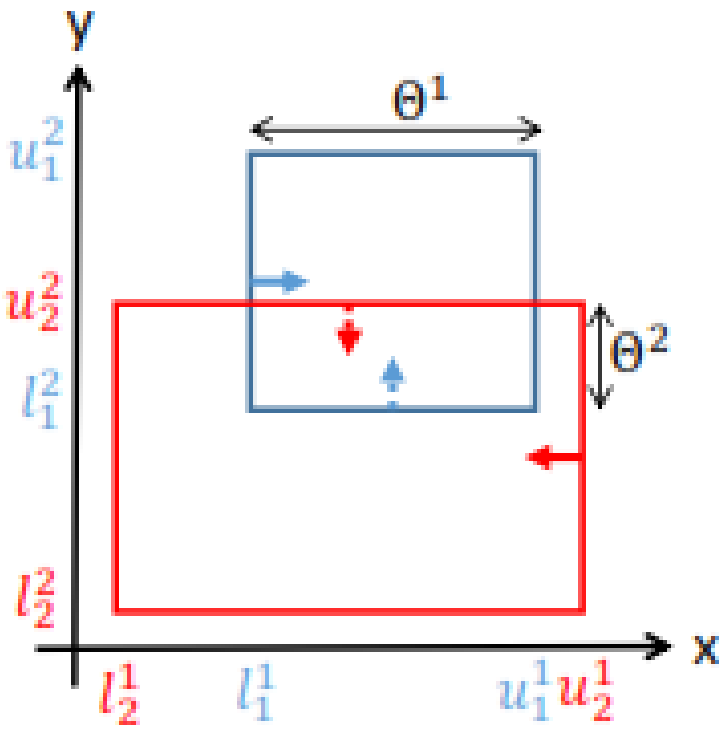
// Independently run a clustering algorithm on the two training sets
$L^0 \leftarrow \mathit{findClusters}(P^0)$;
$L^1 \leftarrow \mathit{findClusters}(P^1)$;
// Compute the minimum bounding hyper rectangles
$\mathbb{C}^0 \leftarrow \mathit{MBR}(P^0, L^0)$;
$\mathbb{C}^1 \leftarrow \mathit{MBR}(P^1, L^1)$;
// Reduce the overlap between each pair of different-colored clusters
for $C_w \in \mathbb{C}^0$ **do**
  for $C_z \in \mathbb{C}^1$ **do**
    if $\Theta^k_{wz} \geq 0 \; \forall k \in [1, n]$ **then**
      $\langle C'_w, C'_z \rangle \leftarrow \mathit{ReduceOverlap}(C_w, C_z, \eta)$;
      $C_w \leftarrow C'_w$;
      $C_z \leftarrow C'_z$;
    end if
  end for
end for
// Select a subset of features for each cluster
$B \leftarrow \mathit{FeatureSubsetSelection}(\mathbb{C}^0, \mathbb{C}^1, \overline{n})$;
return $\langle C^0, C^1, B \rangle$.

Step 3: When a green hyper-rectangle intersects a red one, there is a "conflict". Reduce conflict by shrinking BHRs.



Overlap Reduction

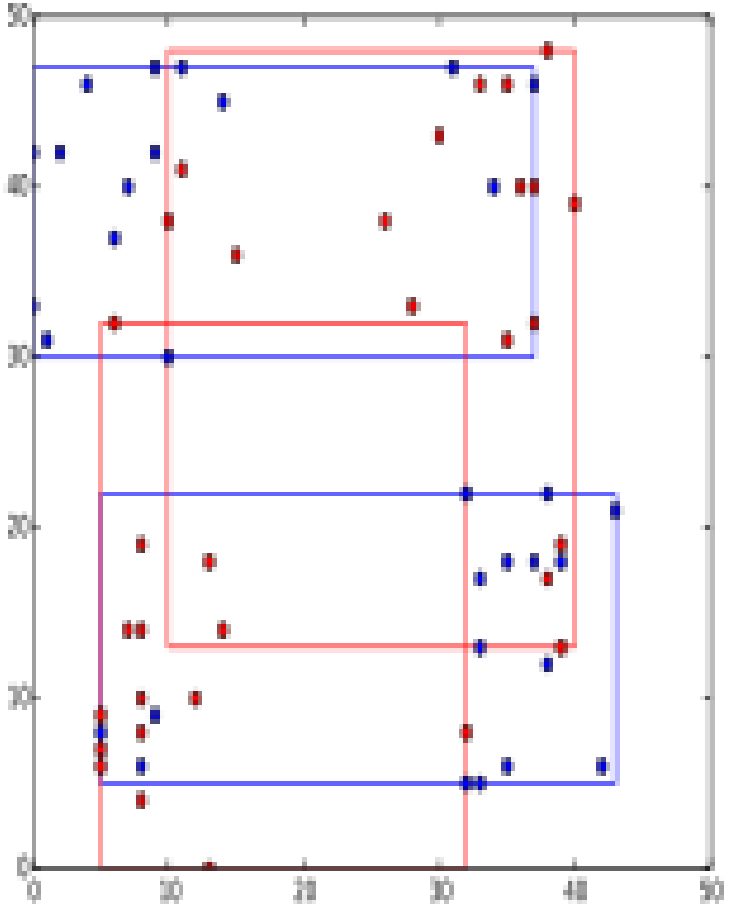




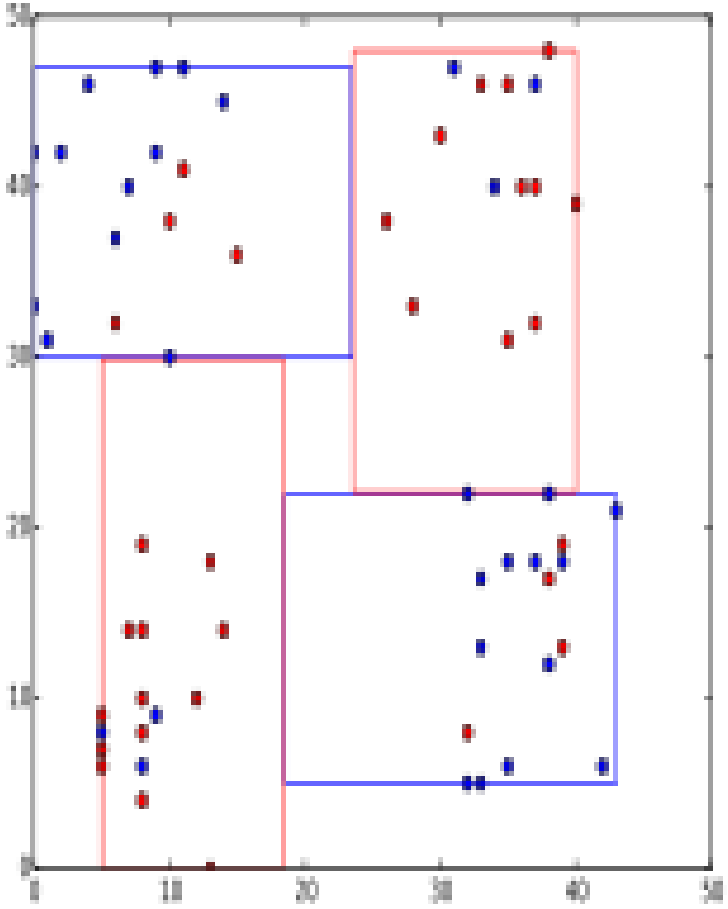
$$\mathcal{R}(C_z, C_w, k, \rho, \Theta_{wz}^k) =$$

$$\begin{cases} \textbf{case 1: } l_z^k > l_w^k \text{ and } u_z^k > u_w^k \textbf{ then} \\ \quad l_{z'}^k = l_z^k + \rho \cdot \Theta_{wz}^k; \\ \quad u_{w'}^k = u_w^k - (1-\rho)\Theta_{wz}^k; \\ \textbf{case 2: } l_z^k \geq l_w^k,\ u_z^k \leq u_w^k, \text{ and } l_z^k - l_w^k \leq u_w^k - u_z^k \textbf{ then} \\ \quad u_{z'}^k = u_z^k - \rho \cdot \Theta_{wz}^k; \\ \quad l_{w'}^k = l_w^k + (1-\rho)(l_z^k - l_w^k + \Theta_{wz}^k); \\ \textbf{case 3: } l_z^k \geq l_w^k,\ u_z^k \leq u_w^k, \text{ and } l_z^k - l_w^k > u_w^k - u_z^k \textbf{ then} \\ \quad l_{z'}^k = l_z^k + \rho \cdot \Theta_{wz}^k; \\ \quad u_{w'}^k = u_w^k - (1-\rho)(u_w^k - u_z^k + \Theta_{wz}^k); \end{cases}$$

Overlap Reduction I

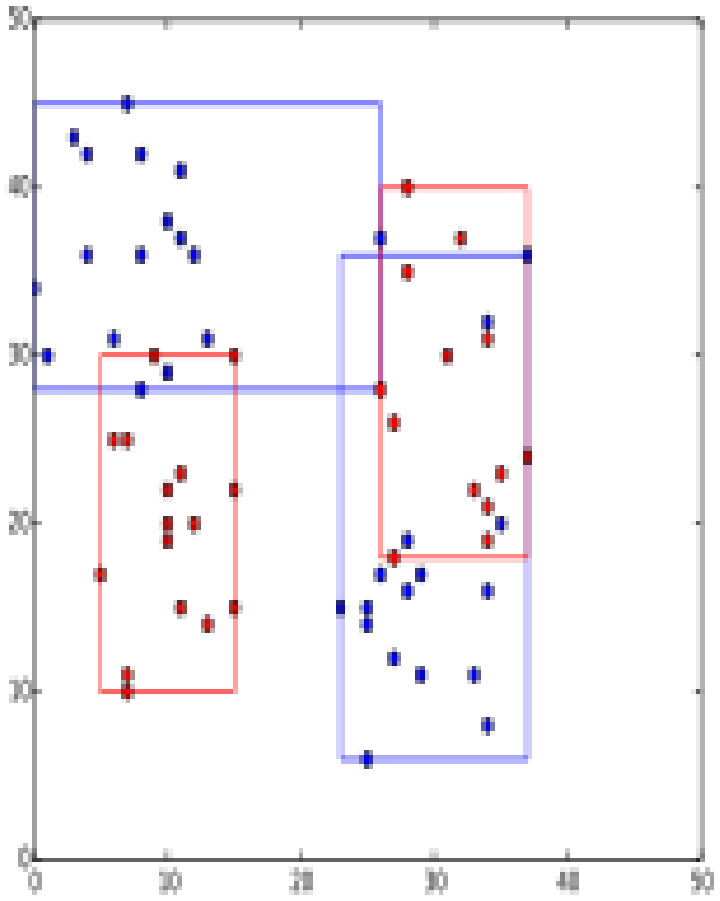


Before

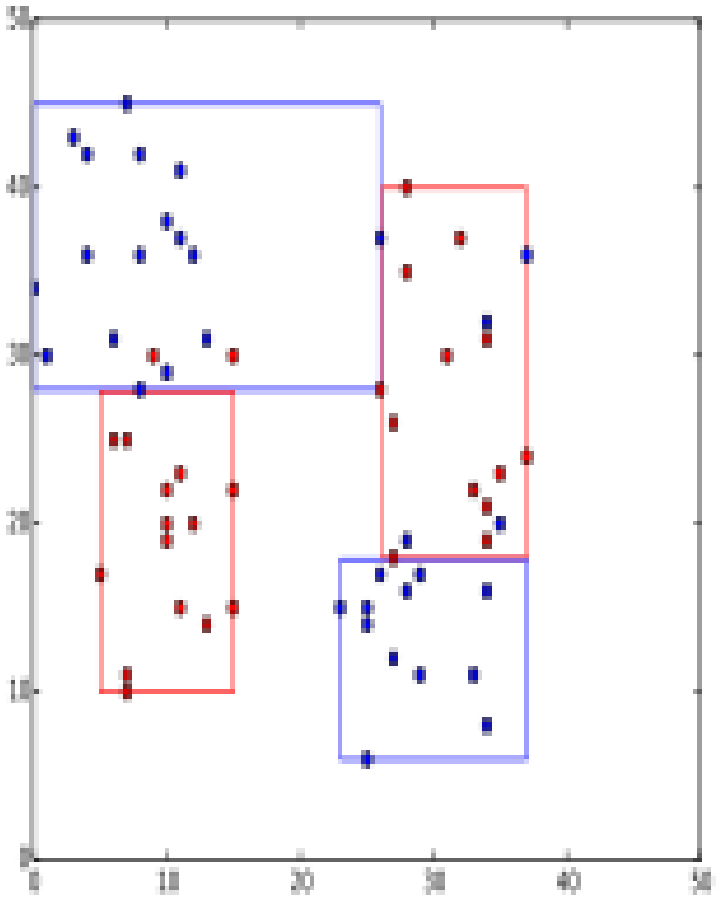


After

Overlap Reduction II



Before



After

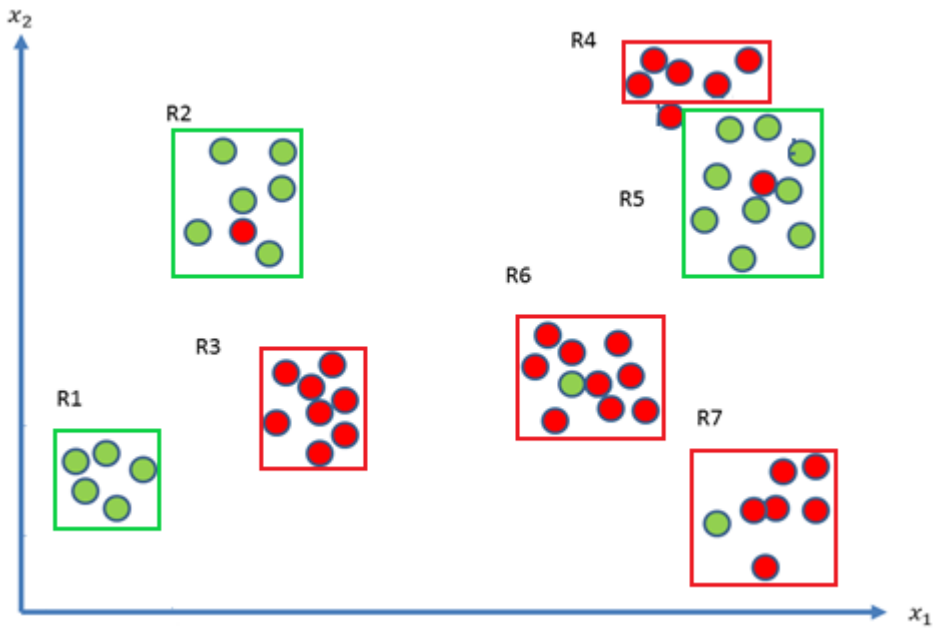
BEEF: Main Steps

Algorithm 1 BEEF

Input: Disjoint sets $P^0, P^1 \in P$, step η, and number of dimensions $\overline{n}$
Output: Sets of clusters $\mathbb{C}^0$, $\mathbb{C}^1$, and $n \times h$ active dimensions matrix B

// Independently run a clustering algorithm on the two training sets
$L^0 \leftarrow \mathit{findClusters}(P^0)$;
$L^1 \leftarrow \mathit{findClusters}(P^1)$;
// Compute the minimum bounding hyper rectangles
$\mathbb{C}^0 \leftarrow \mathit{MBR}(P^0, L^0)$;
$\mathbb{C}^1 \leftarrow \mathit{MBR}(P^1, L^1)$;
// Reduce the overlap between each pair of different-colored clusters
for $C_w \in \mathbb{C}^0$ **do**
  for $C_z \in \mathbb{C}^1$ **do**
    if $\Theta^k_{wz} \geq 0 \; \forall k \in [1, n]$ **then**
      $\langle C'_w, C'_z \rangle \leftarrow \mathit{ReduceOverlap}(C_w, C_z, \eta)$;
      $C_w \leftarrow C'_w$;
      $C_z \leftarrow C'_z$;
    end if
  end for
end for
// Select a subset of features for each cluster
$B \leftarrow \mathit{FeatureSubsetSelection}(\mathbb{C}^0, \mathbb{C}^1, \overline{n})$;
return $\langle C^0, C^1, B \rangle$.

Step 4: Simplify explanation by choosing just a few features for the hyper-rectangles, i.e. project onto a lower-dimension space



Feature Selection: Backward Elimination

Greedy iterative algorithm that is parametrized by a strategy.

- **Backward Elimination:**
  - **Starts with all features active.**
  - **Deactivates the feature that, when switched off, provides the smallest value of**

$$\underbrace{(1 - purity) + overlap}$$

Cost: big value for this is bad

Iteratively get rid of features whose removal leads to the lowest value of this objective function

Feature Selection: Forward Selection

Greedy iterative algorithm that is parametrized by a strategy.

- **Forward Selection:**
  - **Starts with all features inactive.**
  - **Activates the feature that, when switched on, provides the largest value of**

$$\underbrace{\boldsymbol{purity} + (\mathbf{1} - \boldsymbol{overlap})}$$

Cost: big value for this is good

Iteratively select features whose addition leads to the highest value of this objective function

Feature Selection: Highest Penalty Selection

Greedy iterative algorithm that is parametrized by a strategy.

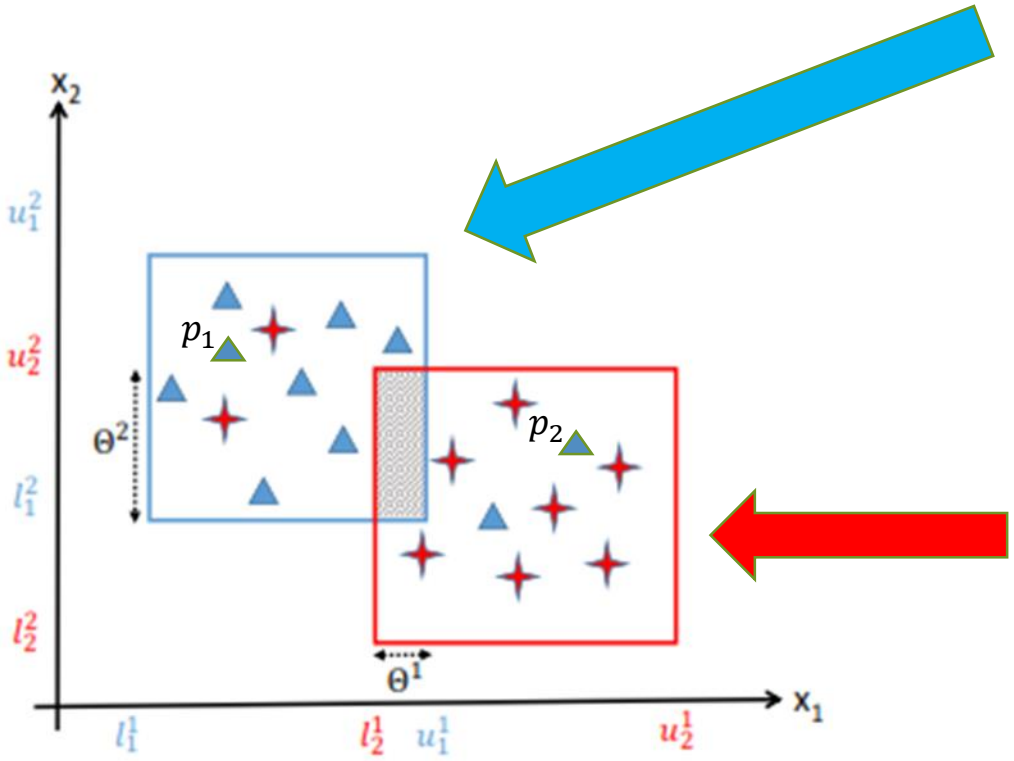
- **Highest Penalty Selection:**
  - **Starts with all features active.**
  - **Computes the penalty incurred for turning off each feature (individually)**
  - **Selects the *n* features with the lowest penalty**

$$\underbrace{(1 - purity) + overlap}_{\textbf{Cost: big value for this is bad}}$$

Balanced Explanation

- A balanced explanation of a forecast $(\overrightarrow{x_i}, y_i)$ generated by any classifier *C* consists of three parts:
  - *Primary* explanation: describes cluster with highest purity containing $\overrightarrow{x_i}$ – confidence of the explanation is purity of the cluster.
  - *Supporting* explanation(s): describes other clusters that support the forecast.
  - *Alternative* explanation(s): describes clusters supporting the opposite of the primary explanation.

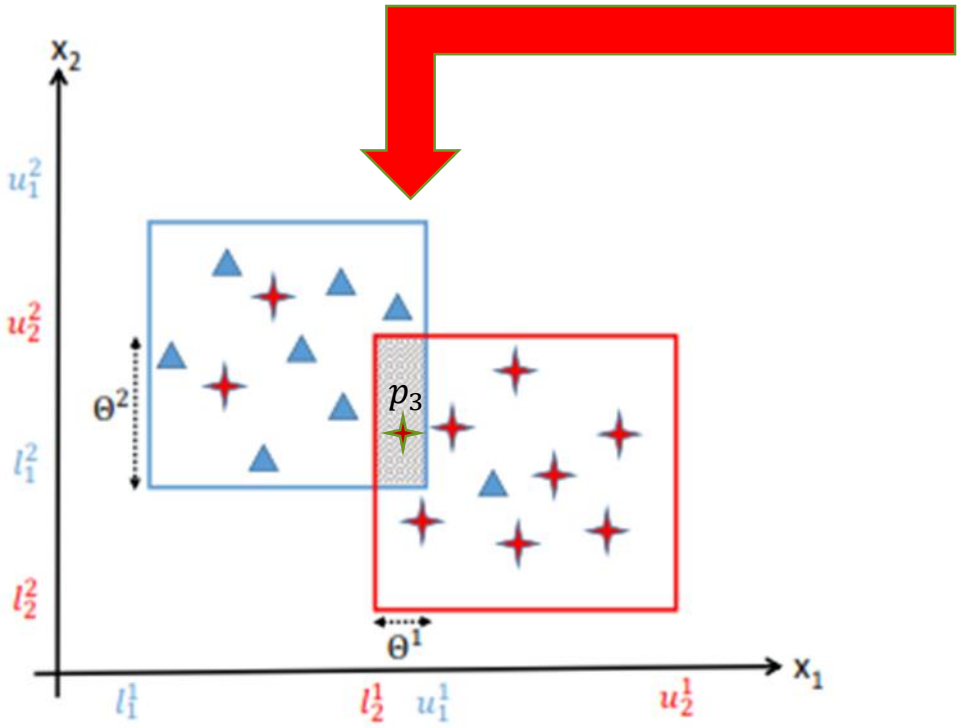
Balanced Explanation



Since object p_1 falls in the **blue** cluster and is classified as **blue**, the *primary* explanation provided by BEEF will be for the classifier's prediction.

The object p_2 is in the **red** cluster. However, since it is classified as **blue**, the *primary* explanation will be against the classifier's prediction.

Balanced Explanation



Object p_3 is in the overlapping region.

The **red** cluster is used to extract the *primary* explanation because it has the highest purity.

The **blue** cluster is used to extract the *alternative* explanation.

Validation

Data: 8 standard datasets from the UCI repository.

Clustering: Used k-Means, DBSCAN, MeanShift

Classifiers: SVM, KNN, Logistic Regression, Multinomial NB, Bernoulli NB, Random Forest, AdaBoost, Multi-Layer Perceptron.

Measures of Efficacy

- **Quality** – a linear combination of inclusion, purity, and overlap

$$quality(\mathbb{C}^0, \mathbb{C}^1) = w_1 \cdot inclusion(\mathbb{C}^0 \cup \mathbb{C}^1) + \\ + \frac{w_2}{|\mathbb{C}^0 \cup \mathbb{C}^1|} \sum_{c \in \mathbb{C}^0 \cup \mathbb{C}^1} P_c + \\ + w_3 \left[1 - \sum_{y \in \{0,1\}} \frac{1}{|\mathbb{C}^y|} \sum_{c_z \in \mathbb{C}^y} \sum_{c_w \in \mathbb{C}^{(1-y)}} \frac{overlap(c_z, c_w)}{Area(c_z)} \right]$$

- **Amazon Mechanical Turk** – 100 subjects

Run-Time & Quality (n=3)

	Backward Elim.			Forward Selection			Highest Penalty		
Dataset	Quality-μ	Quality-σ^2	Time	Quality-μ	Quality-σ^2	Time	Quality-μ	Quality-σ^2	Time
Adult	0.555	0.040	124024.939	0.550	0.040	4651.00	0.554	0.040	1245.00
Bank	0.692	0.045	0.760	0.677	0.048	0.53	0.678	0.048	0.22
Creditcard	0.590	0.038	114509.481	0.590	0.038	4329.00	0.570	0.040	1454.00
Diabetes	0.537	0.040	62502.000	0.530	0.040	4335.00	0.531	0.040	3204.00
Magic	0.635	0.044	94.100	0.619	0.041	44.00	0.610	0.041	22.75
Spambase	0.646	0.045	2943.000	0.620	0.043	198.00	0.615	0.042	64.18
Titanic	0.597	0.041	2.270	0.596	0.041	1.91	0.414	0.018	0.69
Vandal	0.610	0.042	226.000	0.581	0.039	81.00	0.585	0.040	29.20

- Use number of features selected n=3 because there is no statistically significant difference with *n=4* and *n=3* yields simpler explanations.
- The feature selection step is the bottleneck.
- Backward Elimination gives better results but it is the slowest.
- Though run-times appear high, this computation is done only once at training time.

Quality of Explanations with Forward Selection

(a) Bank (b) Adult (c) Credit Card (d) Diabetes

- The choice of coefficients w_1, w_2 and w_3 affects the quality of the explanations.
- The configuration that yielded the best quality for the maximum number of cases was $w_3 > w_1 > w_2$, i.e.,

  weight_overlap > weight_inclusion > weight_purity

BEEF vs IDS*

- Comparison of our quality measurements to the quality of interpretable decision sets (IDS) classifier
- IDS requires continuous variables to be discretized during the preprocessing step → no overlap
- The first step of IDS is frequent itemset generation using Apriori algorithm → very slow

	BEEF		IDS	
Dataset	**Purity**	**Inclusion**	**Purity**	**Inclusion**
Adult	0.610	1.00	0.417	0.667
Bank	0.984	0.99	0.54	0.99
Creditcard	0.741	1.00	0.605	0.653
Diabetes*	0.522	1.00		
Magic	0.875	0.997	0.51	0.92
Spambase*	0.904	0.995		
Titanic	0.713	0.9713	0.52	1.00
Vandal	0.723	0.921	0.52	0.99

* H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction", ACM SIGKDD 2016

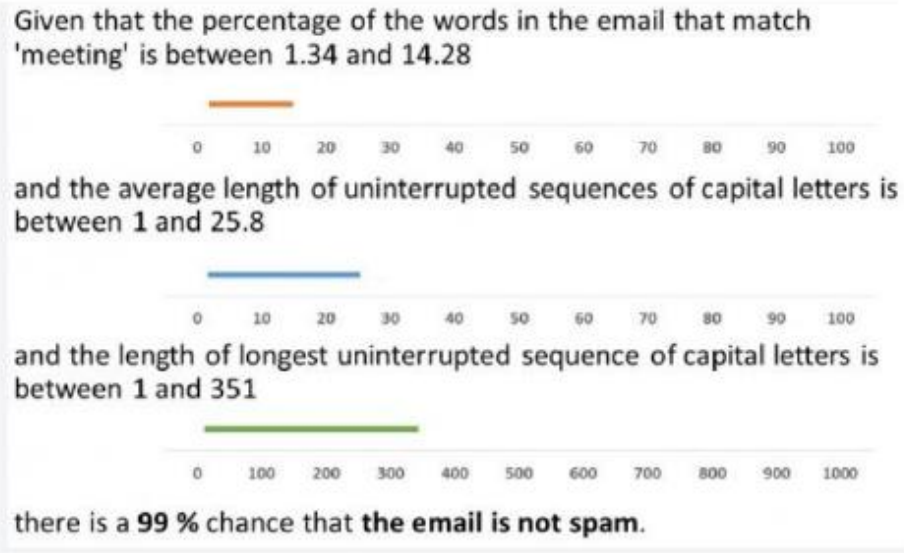
BEEF vs LIME*: MTurk evaluation

- Chose 5 datasets
- Provided 5 predictions for each dataset (25 in all).
- Gave both BEEF and LIME explanations.
- Participants rated explanations on a 7-point Likert scale.
- Order of questions and choices were randomized.

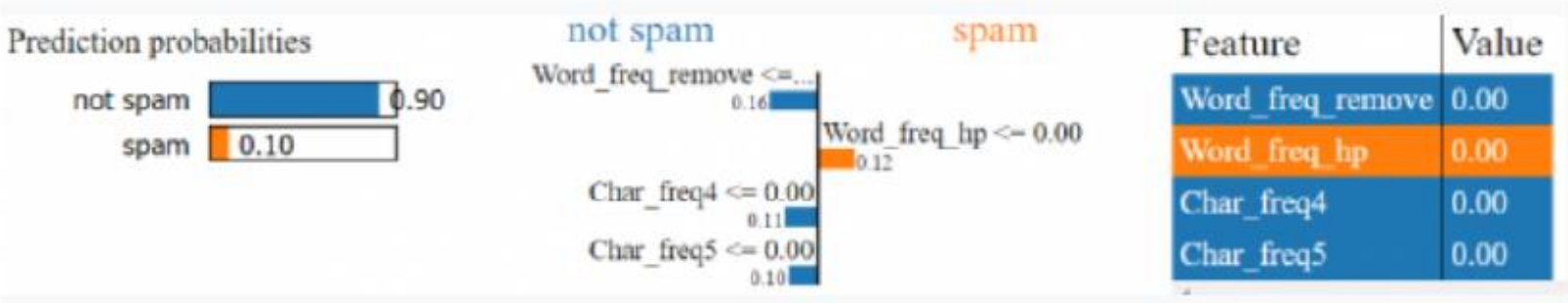
LIME is a local explainer because it builds approximated linear models for each observation (data point).

* M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," KDD 2016

BEEF vs LIME: MTurk evaluation



(a) BEEF



(b) LIME

Sample Question

BEEF vs LIME: MTurk evaluation

	Positive		Neutral		Negative	
	BEEF	LIME	BEEF	LIME	BEEF	LIME
Adult	**63.6**	48	25.8	32.4	**10.6**	19.6
Creditcard	**66.2**	46.2	21.6	34.2	**12.6**	19.6
Vandal	**68**	51.8	21	28.2	**11**	20
Spambase	**60**	48	29.2	29.8	**10.8**	22.2
Magic	53.4	**58.6**	28.4	26.2	18.2	**15.2**
Total	**62.24**	50.52	25.12	30.16	**12.64**	19.32

For 4 out of 5 datasets, users preferred BEEF to LIME.

Percentage of positive, neutral, and negative responses

Conclusions.

- BEEF does a good job in generating English language explanations of predictions made by many different binary classifiers.
- Many English templates can be used to generate explanations.
- Can also be used to generate explanations of forecasts in other languages.

A Probabilistic Logic of Cyber-Deception

V.S. Subrahmanian
Dartmouth College
vs@dartmouth.edu

Joint work with
S. Jajodia, George Mason University
N. Park, Univ. of North Carolina
F. Pierazzi, Univ. of Modena
A. Pugliese, Univ. of Calabria
E. Serra, Boise State U.
G. I. Simari, Univ. of Bahia

Reference

Jajodia, S., Park, N., Pierazzi , F., Pugliese, A., Serra, E., Simari , G. I., & Subrahmanian , V. S. (2017). A Probabilistic Logic of Cyber Deception. IEEE Transactions on Information Forensics and Security , 12 (11), 2532-2544, Nov 2017

http://ieeexplore.ieee.org/abstract/document/7937934

PLD Outline

- **Motivation**
- PLD-Logic: Syntax & Semantics
- Defender's Optimal Actions
  - Attacker behavior
  - Possible Answers & State Update
  - Quantifying (Expected) Damage
- Computational Algorithms
  - Naïve-PLD
  - Fast-PLD
- Experiments

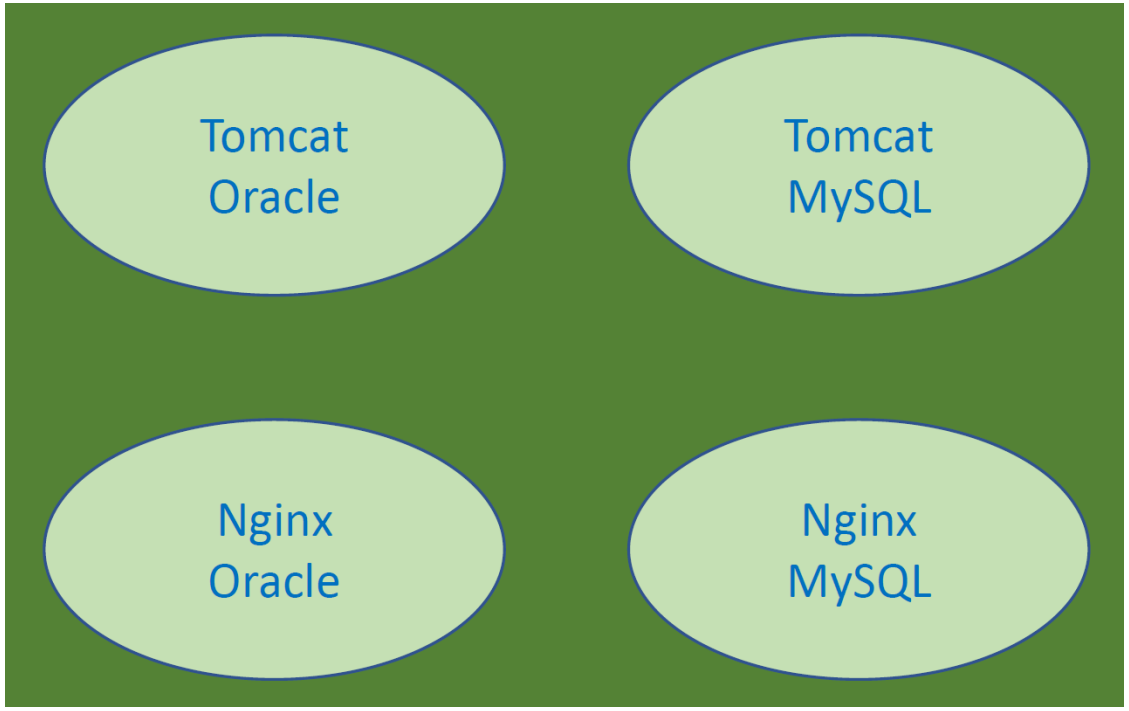
Motivation

- Attackers navigate through an enterprise network by **scanning** nodes.
- Typical types of scan operations:
  - NMAP commands
  - TCP connection scans
  - Stealth SYN scans
  - UDP scans
  - IP protocol scans
- Scanning enables an attacker to decide which node to attack next and the type of attack to mount.

Increases attacker's knowledge of the network (nodes, protocols, connections) and helps him exploit vulnerabilities

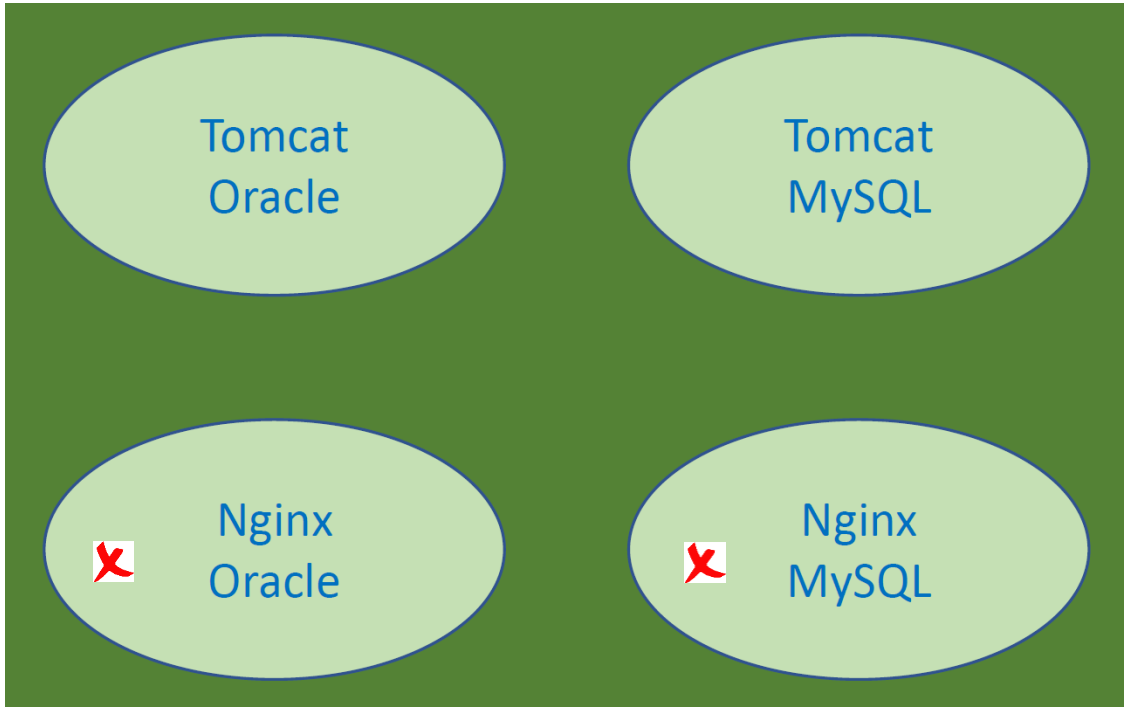
A Simple Example

- Attackers know that node *n* has a
  - Webserver –either *Tomcat* or *Nginx* running on it
  - Database –*either Oracle* or *MySQL* running on it
- *4* possible worlds for the attacker
- Attacker asks *n*:
  - "Are you running a Tomcat webserver?"



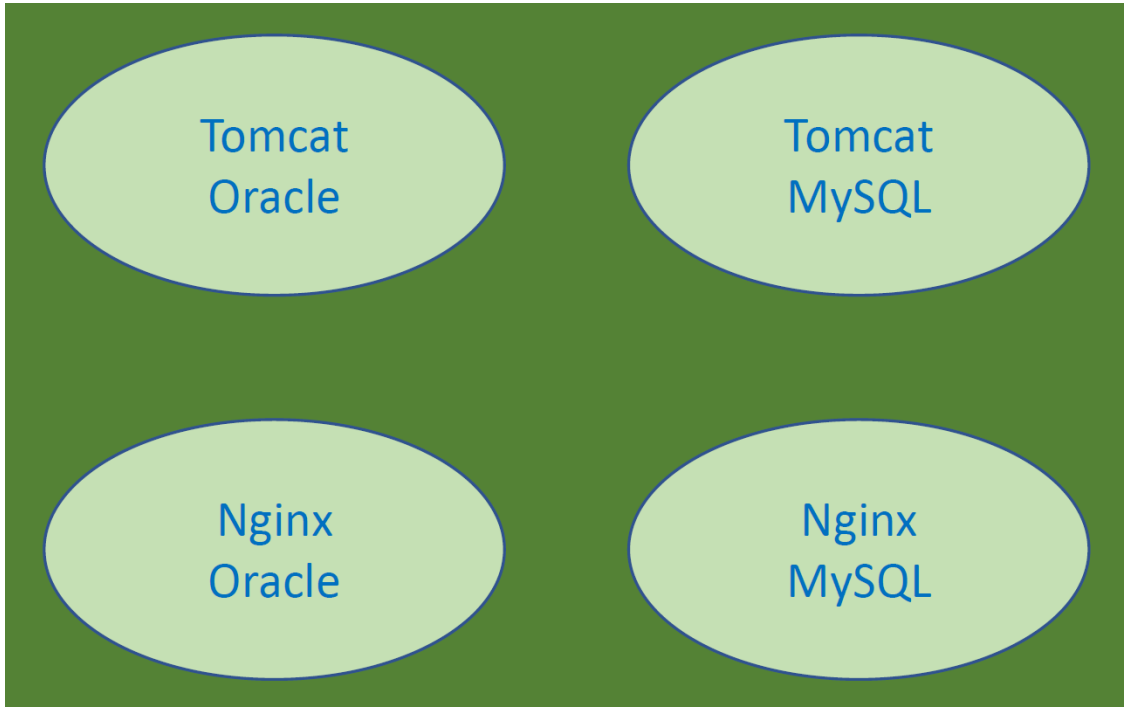
A Simple Example

- Attackers know that node *n* has a
  - Webserver –either *Tomcat* or *Nginx* running on it
  - Database –*either Oracle* or *MySQL* running on it
- *4* possible worlds for the attacker
- Attacker asks *n*:
  - "Are you running a Tomcat webserver?"
- **Honest answer** (e.g. "yes")–reduces attacker's uncertainty by 50%, eliminating two possible worlds.



A Simple Example

- Attackers know that node *n* has a
  - Webserver –either *Tomcat* or *Nginx* running on it
  - Database –*either Oracle* or *MySQL* running on it
- *4* possible worlds for the attacker
- Attacker asks *n*:
  - "Are you running a Tomcat webserver?"
- **If the defender is allowed to lie** (answer honestly and dishonestly) –no worlds are eliminated.



PLD Outline

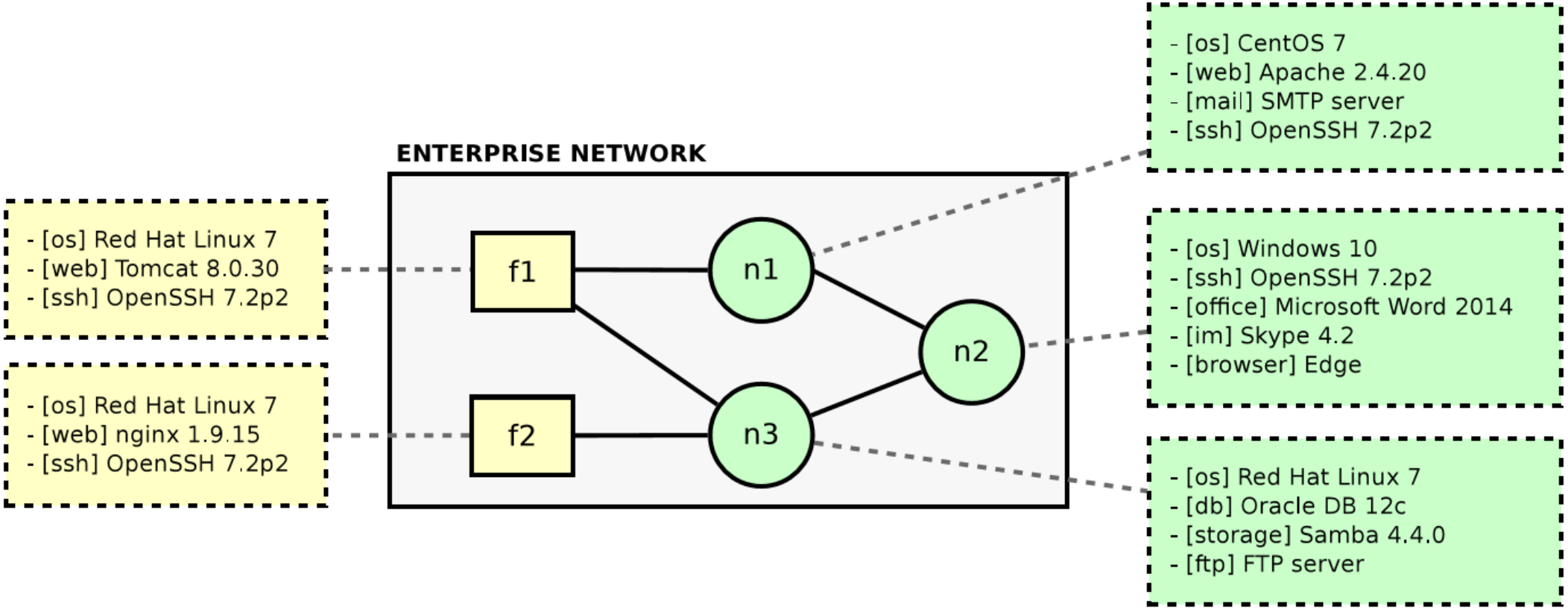
- Motivation
- **PLD-Logic: Syntax & Semantics**
- Defender's Optimal Actions
  - Attacker behavior
  - Possible Answers & State Update
  - Quantifying (Expected) Damage
- Computational Algorithms
  - Naïve-PLD
  - Fast-PLD
- Experiments

Our Proposal

Allow the defender to provide a mix of true and false responses to attacker scans in order to

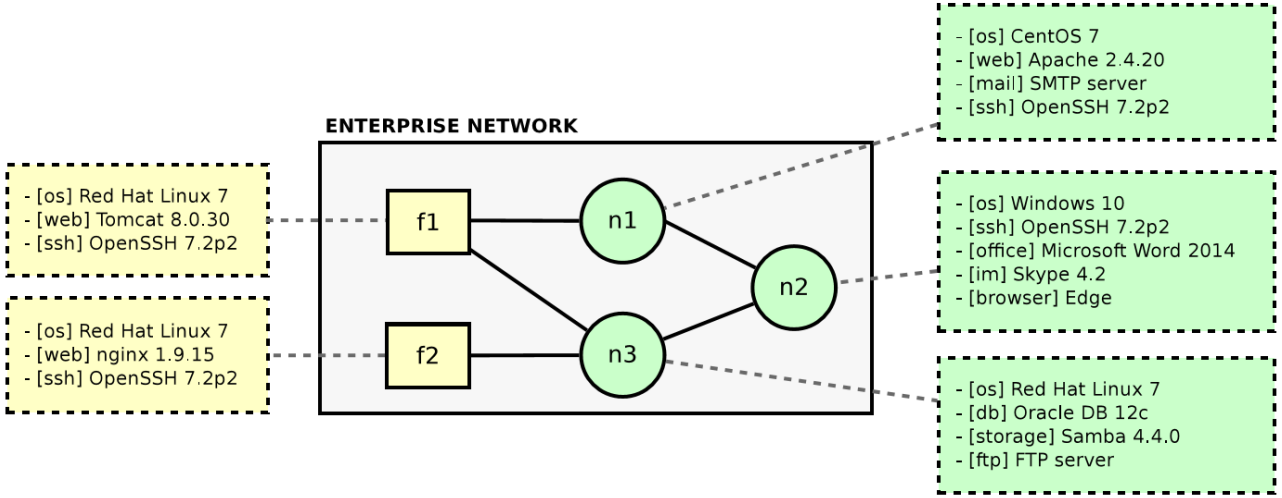
- Increase time spent by attacker inside the network;
- Increase costs/frustration on the attacker;
- Reduce damage

Sample Enterprise Network



Predicates/Atoms

- *node(n)*–*n* is a node
- *edge (n1,n2)* –*n1* and *n2* are connected
- *runs(n,t,s,x)*–node *n* runs version *x* of software *s* that is of type *t*
- *vuln(s,x,v)*–version *x* of software *s* had vulnerability *v*
- ***Possible world:*** set of ground atoms.



runs(n1,os,Centos,7)
runs(n2,ssh,Open SSH, 7.2p2)
Vuln(tomcat,v8.0.30,cve-2016-0763).

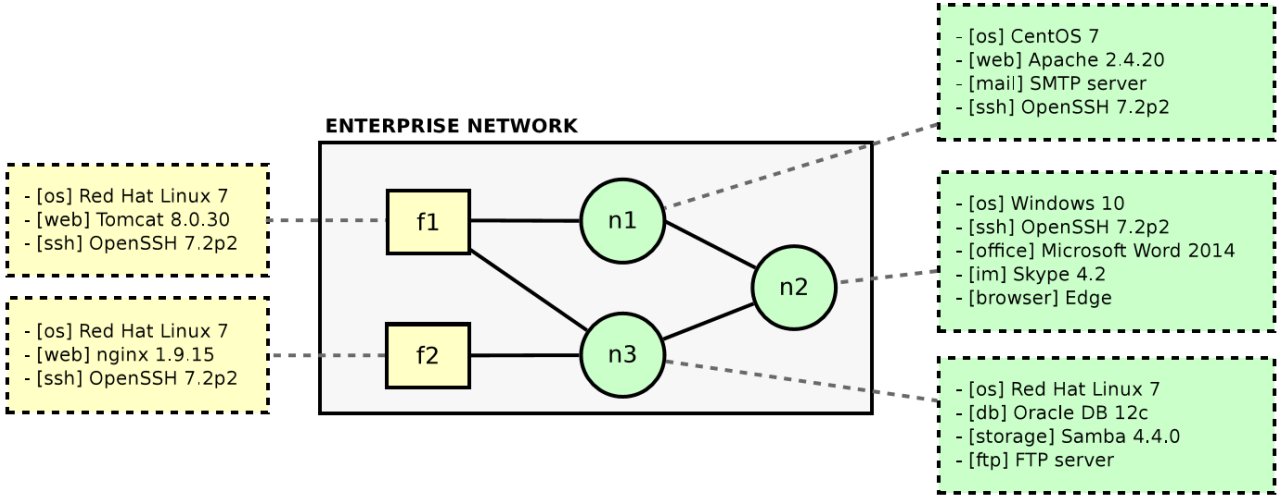
Integrity Constraints

- **Each node must have an OS.**

$\forall N \exists S (\exists X) [runs(N,os,S,X) \leftarrow]$

- **No node may have multiple OSs, e.g. if network has no virtual machines.**
- $\forall N \forall S_1 \forall S_2 \forall X_1 \forall X_2$

  $[S_1 = S_2 \& X_1 = X_2 \leftarrow runs(N,os,S_1,X_1) \ \& \ runs(N,os,S_2,X_2).]$



Theorem: Checking if there exists a possible world satisfying an input set of integrity constraints is NP-hard.

Proof. Reduction from Max Independent Set.

PLD Framework: Logic, Linear Programs, and more…

- **PLD-framework** $M=(N,IC)$
  - *N* is a set of nodes, edges, *runs*, and *vuln* atoms
  - *IC* is a set of integrity constraints.
- **Probabilistic state:** set of *runs*-atoms of the form $A{:}[l,u]$
  - **E.g,** *runs(n,t,s,x)*:$[l,u]$ says node *n* runs version *x* of software *s* with probability in the interval $[l,u]$
- **Probabilistic state** induces constraints on the set of possible worlds.

Example

- Suppose f_1 runs
  - OS: windows or linux
  - Webserver: Tomcat or nginx
- If there are no ICs, then the possible worlds are shown on the right.

$w_0 = \{\emptyset\}$, $w_1 = \{\mathsf{w}\}$, $w_2 = \{\mathsf{l}\}$,
$w_3 = \{\mathsf{t}\}$, $w_4 = \{\mathsf{n}\}$, $w_5 = \{\mathsf{w}, \mathsf{l}\}$,
$w_6 = \{\mathsf{w}, \mathsf{t}\}$, $w_7 = \{\mathsf{w}, \mathsf{n}\}$, $w_8 = \{\mathsf{l}, \mathsf{t}\}$,
$w_9 = \{\mathsf{l}, \mathsf{n}\}$, $w_{10} = \{\mathsf{t}, \mathsf{n}\}$, $w_{11} = \{\mathsf{w}, \mathsf{l}, \mathsf{n}\}$,
$w_{12} = \{\mathsf{w}, \mathsf{l}, \mathsf{t}\}$, $w_{13} = \{\mathsf{w}, \mathsf{t}, \mathsf{n}\}$, $w_{14} = \{\mathsf{l}, \mathsf{t}, \mathsf{n}\}$,
$w_{15} = \{\mathsf{w}, \mathsf{l}, \mathsf{t}, \mathsf{n}\}$.

Example

- Suppose f_1 runs
  - OS: windows or linux
  - Webserver: Tomcat or nginx
- But if the integrity constraints state that a node has
  - Exactly one OS and
  - Exactly one webserver.

w_0 ✗ $\{\emptyset\}$,	$w_1 = \{w\}$,	$w_2 = \{l\}$,
w_3 ✗ $\{t\}$,	w_4 ✗ $\{n\}$,	w_5 ✗ $\{w, l\}$,
$w_6 = \{w, t\}$,	$w_7 = \{w, n\}$,	$w_8 = \{l, t\}$,
$w_9 = \{l, n\}$,	w_{10} ✗ $\{t, n\}$,	w_{11} ✗ $\{w, l, n\}$,
w_{12} ✗ $\{w, l, t\}$,	w_{13} ✗ $\{w, t, n\}$,	w_{14} ✗ $\{l, t, n\}$,
w_{15} ✗ $\{w, l, t, n\}$.		

PLD: Logic & Linear Programs

- Suppose the defender thinks the attacker believes that node f_1 runs
  - Linux as the OS with probability [0.6,0.85] and
  - Windows as the OS with probability [0.4,0.8]
  - Tomcat as the webserver with probability [0.6,0.9]
  - Nginx as the webserver with probability [0.5,0.65]

$$0.6 \leq p_2 + p_8 + p_9 \leq 0.85$$
$$0.6 \leq p_6 + p_8 \leq 0.9$$
$$0.4 \leq p_1 + p_6 + p_7 \leq 0.8$$
$$0.5 \leq p_7 + p_9 \leq 0.65$$
$$p_1 + p_2 + p_6 + p_7 + p_8 + p_9 = 1$$

w_0 ✗ $\{\emptyset\}$,	$w_1 = \{\mathsf{w}\}$,	$w_2 = \{\mathsf{l}\}$,
w_3 ✗ $\{\mathsf{t}\}$,	w_4 ✗ $\{\mathsf{n}\}$,	w_5 ✗ $\{\mathsf{w}, \mathsf{l}\}$,
$w_6 = \{\mathsf{w}, \mathsf{t}\}$,	$w_7 = \{\mathsf{w}, \mathsf{n}\}$,	$w_8 = \{\mathsf{l}, \mathsf{t}\}$,
$w_9 = \{\mathsf{l}, \mathsf{n}\}$,	w_{10} ✗ $\{\mathsf{t}, \mathsf{n}\}$,	w_{11} ✗ $\{\mathsf{w}, \mathsf{l}, \mathsf{n}\}$,
w_{12} ✗ $\{\mathsf{w}, \mathsf{l}, \mathsf{t}\}$,	w_{13} ✗ $\{\mathsf{w}, \mathsf{t}, \mathsf{n}\}$,	w_{14} ✗ $\{\mathsf{l}, \mathsf{t}, \mathsf{n}\}$,
w_{15} ✗ $\{\mathsf{w}, \mathsf{l}, \mathsf{t}, \mathsf{n}\}$.		

PLD Outline

- Motivation
- PLD-Logic: Syntax & Semantics
- **Defender's Optimal Actions**
  - **Attacker behavior**
  - **Possible Answers & State Update**
  - **Quantifying (Expected) Damage**
- Computational Algorithms
  - Naïve-PLD
  - Fast-PLD
- Experiments

NIST: National Vulnerability Database

07937934.pdf | NVD - CVE-2017-3444

nvd.nist.gov/vuln/detail/CVE-2017-3444

Source: MITRE **Last Modified:** 10/19/2017 + View Analysis Description

Impact

CVSS Severity (version 3.0):

CVSS v3 Base Score: 8.2 High
Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N (legend)
Impact Score: 4.7
Exploitability Score: 2.8

CVSS Version 3 Metrics:

Attack Vector (AV): Network
Attack Complexity (AC): Low
Privileges Required (PR): None
User Interaction (UI): Required
Scope (S): Changed
Confidentiality (C): High
Integrity (I): Low
Availability (A): None

CVSS Severity (version 2.0):

CVSS v2 Base Score: 5.8 MEDIUM
Vector: (AV:N/AC:M/A
Impact Subscore: 4.9
Exploitability Subscore: 8.6

CVSS Version 2 Metrics:

Access Vector: Network expl
with attack m
Access Complexity: Medium
Authentication: Not required
Impact Type: Allows unauth
unauthorized

Utility of a Node & Sub-rationality Factor

- Attacker's utility is a function $util: Nodes \rightarrow \boldsymbol{R}^{\geq 0}$.
- Suppose we fix a possible world ω.
- Example, building on NIST National Vulnerability DB

$$util(n) = \sum_{runs(n,t,s,x) \in \omega \ \& \ vuln(s,x,v) \in N} [impact(v) * exploitability(v)]$$

- Other definitions are also possible
- Attacker scans neighbors of nodes he has compromised and builds a **utility array** $[\boldsymbol{u_1}, \dots, \boldsymbol{u_m}]$
- Attacker may choose to attack ***a*** neighbor with a utility greater than or equal to $SUB * \max[u_1, \dots, u_m]$ => ***so the adversary is only partly rational.***

Possible Answers & State Updates

- *M=(N,IC)* is a PLD-framework.
- *H:* a set of answers provided in the past only involving *runs* atoms.
- **Possible answers** to *Q* are sets A_i of *runs*-atoms s.t.
  - **if** $H \cup N \vDash IC$, **then** $H \cup N \cup A_i \vDash IC$.

Other models also possible with some modifications.

- **Many possible answers can exist.**

State update

- $\boldsymbol{\sigma}$ is a probabilistic state
- *A* is a possible answer
- New state obtained via 3 steps:
  1) Compute $\sigma' = \{b: [l,u] | b: [l,u] \in \sigma, \& \, b \in A\}$. **We will remove things b the attacker had doubts about before, but that defender's answer *A* says are true**.
  2) Compute $\sigma'' = \{a: [1,1] | a \in A\}$. **Attacker believes everything he is now told [unless he has prior reason to doubt it].**
  3) Compute $(\sigma - \sigma') \cup \sigma''$ ensuring the above.
  4) **Consolidate**$((\sigma - \sigma') \cup \sigma'')$ to remove inconsistency.

Possible Answers & State Updates: Consolidation

1. Compute $X = ((\sigma - \sigma') \cup \sigma'')$
2. If $A_i : [l_i, u_i] \in X$, then write the constraints:
   1. $l'_i \leq \sum_{w_j \in W_{IC} \& A_i \in w_j} p_j \leq u'_i$ **[widens the original constraint]** and
   2. $l'_i \leq l_i$
   3. $u_i \leq u'_i$
3. Solve the optimization problem

   maximize $\sum_i (l'_i - u'_i) * \beta_i$ **[$\boldsymbol{\beta_i}$ is the believability of atom $\boldsymbol{A_i}$]**

   subject to above constraints
4. $LC'(\sigma) = LC(\sigma) \cup \{\sum_i (u'_i - l'_i) * \beta_i$=M*} where M* is the solution in step (3).

Example

$$\{runs(n_3, \text{os}, \text{linux}, \text{rhel7}) : [0.5, 1],$$
$$runs(n_3, \text{os}, \text{windows}, \text{v8}) : [0.8, 1]\}$$

IC says only one OS.
w1={linux}, w2={windows}

$$0.5 \leq p_1 \leq 1$$
$$0.8 \leq p_2 \leq 1$$
$$p_1 + p_2 = 1$$

NO SOLUTION !

Suppose $\beta_1 = 1$ $\beta_2 = 50$

$$\ell'_1 \leq p_1 \leq u'_1$$
$$\ell'_2 \leq p_2 \leq u'_2$$
$$\ell'_1 \leq 0.5, u'_1 \geq 1$$
$$\ell'_2 \leq 0.8, u'_2 \geq 1$$
$$p_1 + p_2 = 1$$

$M^* = \ell'_1 - u'_1 + 50 \times (\ell'_2 - u'_2)$ to $LC(\sigma)$.

Add this constraint to get the new LC.
M* is opt solution

Attacker Strategy

- ***Attacker Strategy***: A sequence of nodes $\lambda = (n_1, \dots, n_k)$ that have been compromised by the attacker such that the restriction of the edges is a connected subgraph.
- Let $\boldsymbol{A}$ be the set of all strategies
- ***Probability Distribution over A w.r.t. world w***. Function $PR_w: \boldsymbol{A} \rightarrow [0,1]$ such that $\sum_{\lambda \in \boldsymbol{A}} PR_w(\lambda) = 1$.
- How should we (defender) get an attacker strategy?
  - Simulate attacker behavior upto *NSTEP* times using a given choice of behavior.
  - If $\lambda = (n_1, \dots, n_k)$ is the strategy executed by the attacker thus far, then we look *NSTEP* nodes into the future.

Example Simulation of Attacker Strategy

1. Attacker strategy thus far. $\lambda = (\mathrm{n}_1, \mathrm{n}_2)$.
2. Attacker issues *scan* queries to all neighbors (n_3, n_4, n_5, n_6).
3. Values (to defender)

n1	n2	n3	n4	n5	n6
10	5	1	5	100	10

4. Defender needs to answer these queries – what to do?

Example Simulation of Attacker Strategy

1. Attacker strategy thus far. $\lambda = (\mathrm{n}_1, \mathrm{n}_2)$.
2. Attacker issues *scan* queries to all neighbors (n_3, n_4, n_5, n_6).
3. Values

n1	n2	n3	n4	n5	n6
10	5	1	5	100	10

4. Defender needs to answer these queries – what to do?
   - Sample answers from the space of possible answers.
   - Suppose one such answer has the following utilities
   - Suppose SUB=0.8

n1	n2	n3	n4	n5	n6
		40	50	10	35

Example Simulation of Attacker Strategy

1. Attacker strategy thus far. $\lambda = (\mathrm{n}_1, \mathrm{n}_2)$.
2. Attacker issues *scan* queries to all neighbors (n_3, n_4, n_5, n_6).
3. Values

n1	n2	n3	n4	n5	n6
10	5	1	5	100	10

4. Defender needs to answer these queries – what to do?
   - Sample answers from the space of possible answers.
   - Suppose one such answer has the following utilities. Fully rational attacker would target n4 next.
   - Suppose SUB=0.8.
   - In this case, attacker would target either n3 or n4.

n1	n2	n3	n4	n5	n6
		40	50	10	35

Example Simulation of Attacker Strategy

n1	n2	n3	n4	n5	n6
10	5	1	5	100	10

1. Defender needs to answer these queries – what to do?
   - Sample answers from the space of possible answers.
   - Suppose one such answer has the following utilities. Fully rational attacker would target n4 next.
   - Suppose SUB=0.8.
   - In this case, attacker would target either n3 or n4.
   - Suppose NSTEPS = 2, so we want to simulate his next 2 steps.
   - Attacker chooses between *n3, n4*.
     - In one simulation run, suppose he chooses n4.
     - Next, he chooses either n3 or n6. Why?
   - Repeat the previous step *NSIM* times to get pdf

n1	n2	n3	n4	n5	n6
		40	50	10	35

Damage

- Given world *w,* **damage w.r.t. a strategy** $damage(\lambda) = \sum_{n \in \lambda} value(n)$. So damage w.r.t. a strategy from the defender's point of view is the sum of the values of the nodes compromised.
- Expected damage **w.r.t. a world** $exp\_damage(w) = \sum_{\lambda \in \boldsymbol{A}} PR_w(\lambda) * damage(\lambda)$. Determined by simulation.
- Damage w.r.t. a probabilistic state σ:

$$damage(\sigma) = \sum_{w_i \in \mathcal{W}_{IC}} (p_i \cdot exp\text{-}damage(w_i)) .$$

- **But probability p_i of a world is determined by our linear constraints which may not have a unique solution.**

Damage Theorems

- **Theorem:** Computing the probability assignment that corresponds to the worst damage is NP-hard.

- **Theorem:** Both the following problems are NP-hard:

i) Does there exist an answer to a scan query?

ii) Finding the answer that minimizes damage is NP-hard.

Talk Outline

- Motivation
- PLD-Logic: Syntax & Semantics
- Defender's Optimal Actions
  - Attacker behavior
  - Possible Answers & State Update
  - Quantifying (Expected) Damage
- **Computational Algorithms**
  - **Naïve-PLD**
  - **Fast-PLD**
- Experiments

Algorithms: Naïve-PLD

Algorithm Naive-PLD

Input: PLD framework M, current probabilistic state σ, scan query Q, history H of answers given to past scan queries, $NSTEPS, NSIM \in \mathbb{N}$, $SUB \in [0, 1]$.

Output: Set of ground *runs* atoms.

```
worlds:= compute the set of all possible worlds w.r.t. M;
possAns:= possAnswers(Q, H);
leastDamage:= +∞;
leastDamageAns:= ∅;
for (i:= 0; i ≤ |possAns|; i++) {
    currAns:= possAns.get(i);
    newState:= updateState(σ, currAns);
    if damage(newState, NSTEPS, NSIM, SUB) ≤ leastDamage then {
        leastDamageAns:= currAns;
        leastDamage:= damage(newState);
    }
}
return leastDamageAns.
```

1. Find all possible worlds.
2. Find set of all possible answers compatible with the history.

Algorithms: Naïve-PLD

Algorithm Naive-PLD

Input: PLD framework M, current probabilistic state σ, scan query Q, history H of answers given to past scan queries, $NSTEPS, NSIM \in \mathbb{N}$, $SUB \in [0, 1]$.

Output: Set of ground *runs* atoms.

```
worlds:= compute the set of all possible worlds w.r.t. M;
possAns:= possAnswers(Q, H);
leastDamage:= +∞;
leastDamageAns:= ∅;
for (i:= 0; i ≤ |possAns|; i++) {
    currAns:= possAns.get(i);
    newState:= updateState(σ, currAns);
    if damage(newState, NSTEPS, NSIM, SUB) ≤ leastDamage then {
        leastDamageAns:= currAns;
        leastDamage:= damage(newState);
    }
}
return leastDamageAns.
```

1. Find all possible worlds.
2. Find set of all possible answers compatible with history.
3. Loop through answers
   - Compute new state if that answer is given
   - Estimate damage
   - Update best solution if needed.
4. Return the best.

$$O(|Nodes| \times \binom{S}{2^{MaxSW}} \times NSIM \times P \times |\mathcal{W}_{IC}|)$$

Algorithms: Fast-PLD

Algorithm Fast-PLD

Input: PLD framework M, current probabilistic state σ, scan query Q, history H of answers given to past scan queries, $S \in \{1, \ldots, |\mathcal{W}|\}$, $SUB \in [0, 1]$, $NA, NW, NSTEPS, NSIM, NIDE \in \mathbb{N}$.

Output: Set of ground *runs* atoms.

1. let *Ans* be a mapping from sets of atoms to real numbers (initialized to empty);
2. $leastAvgDamage := +\infty$;
3. *prAns*:= new uniform prob. dist. over answers;
4. for (n:= 1; $i \leq NIDE$; n++) {
5. for (i:= 1; $i \leq NA$; i++) {
6. $spa := sample\text{-}possAnswer(Q, H, prAns)$;
7. $damage := 0$;
8. for (j:= 1; $j \leq NW$; j++) {
9. $sw := sample\text{-}worlds(M, \sigma, S, prWorlds)$;
10. $newState := updateState(\sigma, spa)$;
11. $avgDamage := \Big(avgDamage * (j-1) + damage(newState, NSTEPS, NSIM, SUB)\Big)/j$;
12. } // end for (world sampling)
13. if $avgDamage < leastAvgDamage$ then
14. $leastAvgDamageAns := spa$;
15. $leastAvgDamage := avgDamage$;
16. Add $(spa, avgDamage)$ to *Ans*;
17. } // end for (answer sampling)
18. Update *prAns* according to best samples in *Ans*;
19. Reset *Ans* to an empty mapping;
20. } // end for (I.D.E. iterations)
21. return *leastAvgDamageAns*.

- Sample answer to a query.
- Then sample many different possible worlds.
- Worlds not in the sample implicitly have probability set to 0 => ***inconsistency possible!***
- Fixed in step 10

Algorithms: Fast-PLD

Algorithm Fast-PLD

Input: PLD framework M, current probabilistic state σ, scan query Q, history H of answers given to past scan queries, $S \in \{1, \ldots, |\mathcal{W}|\}$, $SUB \in [0, 1]$, $NA, NW, NSTEPS, NSIM, NIDE \in \mathbb{N}$.

Output: Set of ground *runs* atoms.

1. let *Ans* be a mapping from sets of atoms to real numbers (initialized to empty);
2. $leastAvgDamage := +\infty$;
3. $prAns :=$ new uniform prob. dist. over answers;
4. for ($n := 1$; $i \leq NIDE$; n++) {
5. for ($i := 1$; $i \leq NA$; i++) {
6. $spa := sample\text{-}possAnswer(Q, H, prAns)$;
7. $damage := 0$;
8. for ($j := 1$; $j \leq NW$; j++) {
9. $sw := sample\text{-}worlds(M, \sigma, S, prWorlds)$;
10. $newState := updateState(\sigma, spa)$;
11. $avgDamage := \Big(avgDamage * (j-1) + damage(newState, NSTEPS, NSIM, SUB)\Big)/j$;
12. } // end for (world sampling)
13. if $avgDamage < leastAvgDamage$ then
14. $leastAvgDamageAns := spa$;
15. $leastAvgDamage := avgDamage$;
16. Add $(spa, avgDamage)$ to *Ans*;
17. } // end for (answer sampling)
18. Update *prAns* according to best samples in *Ans*;
19. Reset *Ans* to an empty mapping;
20. } // end for (I.D.E. iterations)
21. return *leastAvgDamageAns*.

- Estimate damage using simulation approach
- Keep best answer seen so far
- But update the probability distributions – see step 18 - used for sampling every time the loop within (4)-(20) is executed.

$$O\left(NIDE \times NA \times |Nodes| \times MaxSW \times NSIM \times P \times NW\right)$$

Talk Outline

- Motivation
- PLD-Logic: Syntax & Semantics
- Defender's Optimal Actions
  - Attacker behavior
  - Possible Answers & State Update
  - Quantifying (Expected) Damage
- Computational Algorithms
  - Naïve-PLD
  - Fast-PLD
- **Experiments**

Experiments

DATA

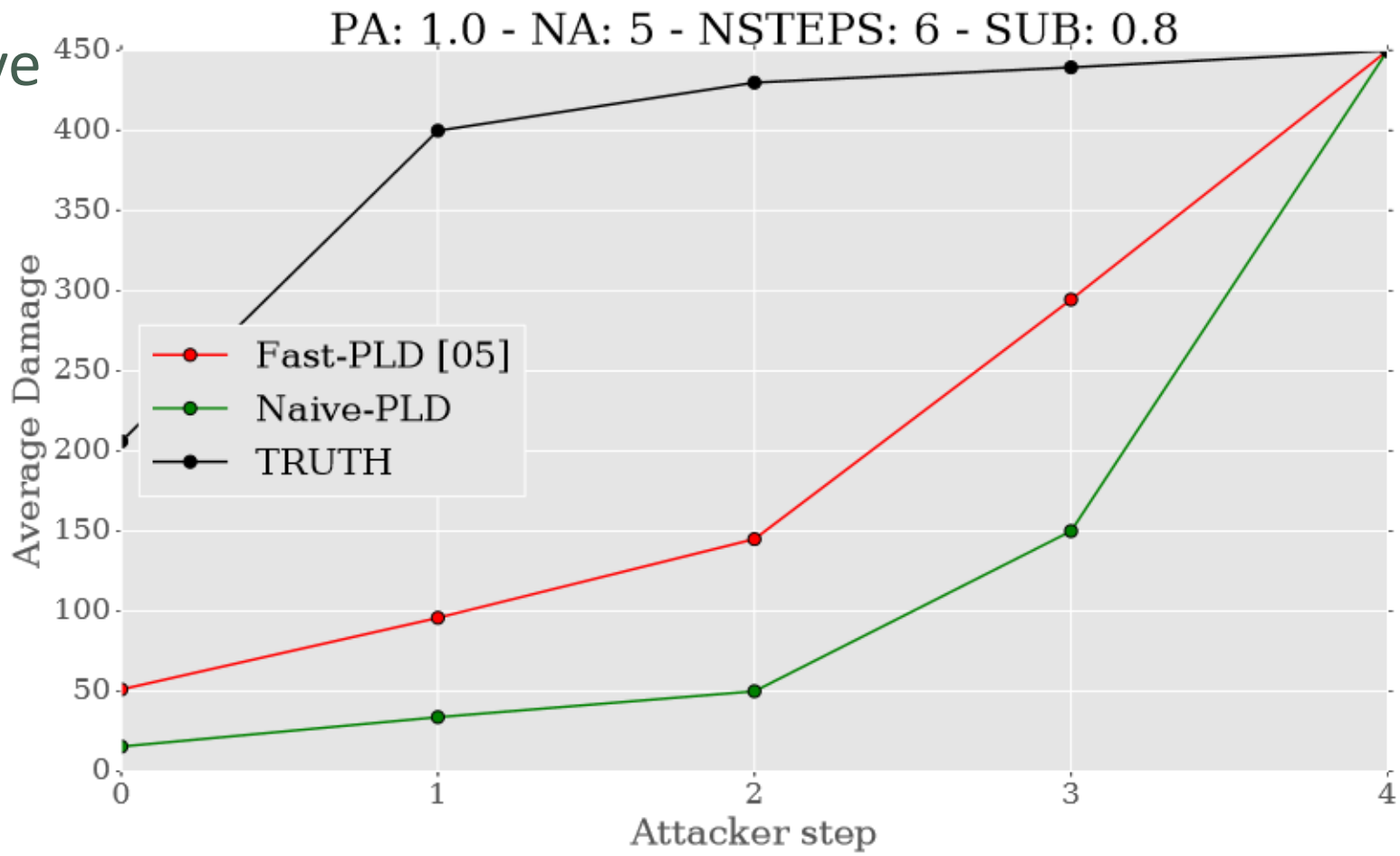
- Real enterprise network from Cauldron – 60 nodes;
- Synthetic 600-node network by replicating Cauldron using NS2 network simulator.
- **Values** assigned to nodes sampled from both
  - Gaussian distribution and
  - Zipf distribution

Samples involved just 50 worlds + real one.

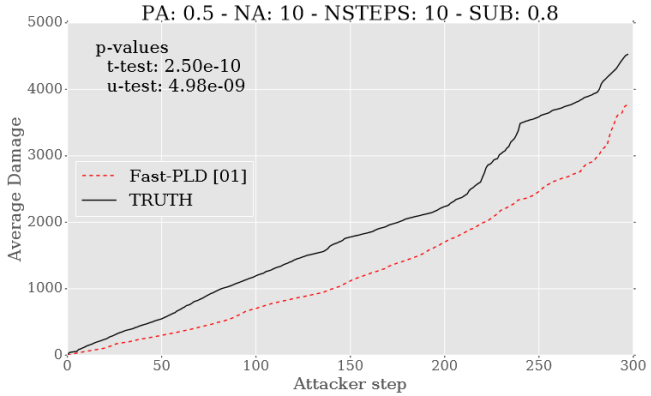
- **Truth** baseline answers scan queries honestly
- Evaluated
  - **Expected damage**
  - **Actual run time**
- **Statistical validation:**
  - Student t-test
  - Mann-Whitney U-test
- Varied
  - NA = number of answers (in Fast-PLD)
  - NSTEPS
  - NIDE
  - SUB

Experiment 1

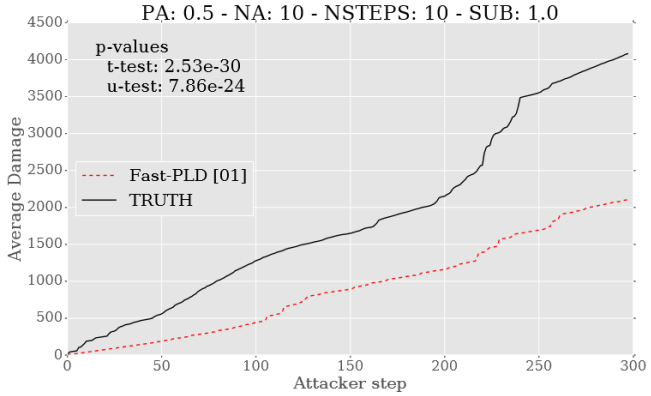
- Just 5-node network as Naïve PLD is expected to be slow!
- **Damage**:
  - TRUTH is worst.
  - Fast-PLD is second best.
  - Naïve-PLD is best.
- **Run-time**:
  - TRUTH = 0.2 sec
  - Fast-PLD = 3.2 sec
  - Naïve-PLD = 10+ hours!



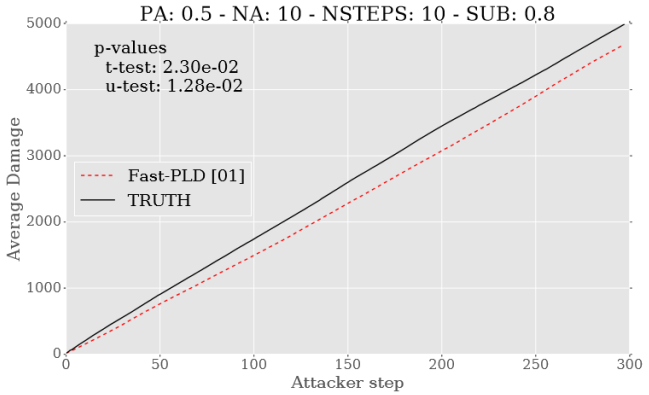
Experiment 2: Zipf vs. Gaussian



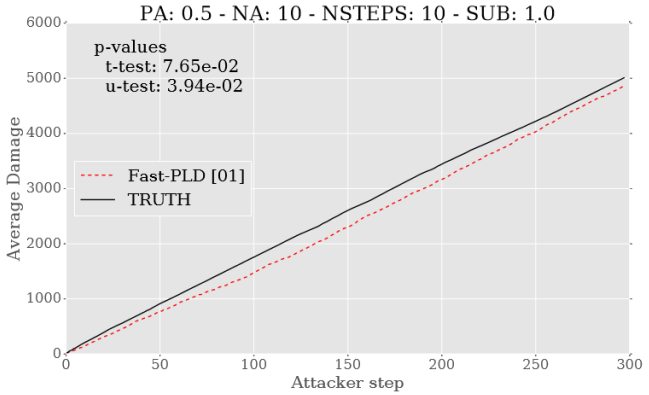
(a) Zipf ($SUB = 0.8$)



(b) Zipf ($SUB = 1.0$)



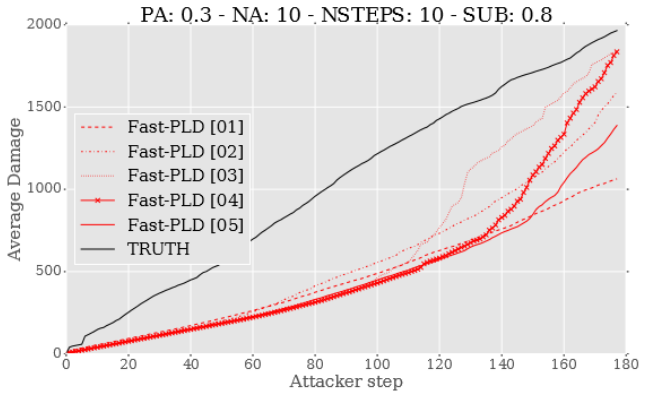
(c) Gaussian ($SUB = 0.8$)



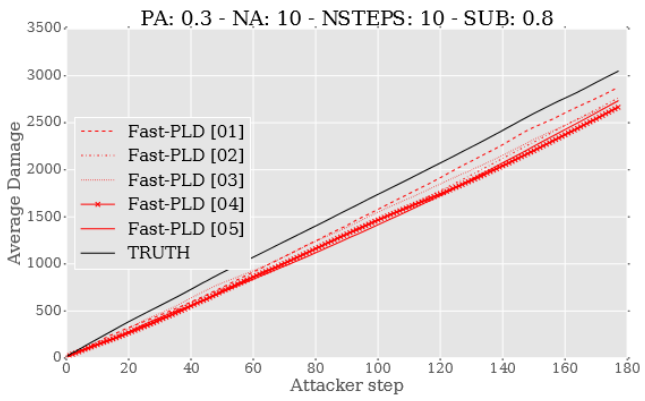
(d) Gaussian ($SUB = 1.0$)

- Fast-PLD beats TRUTH for both fully rational and subrational attackers.
- Fast-PLD performs much better at deception when assets are distributed according to a Zipf distribution compared to Gaussian.
- ***Suggests that putting important information in a small number of well protected nodes might be a better strategy.***
- **Topic for further exploration.**

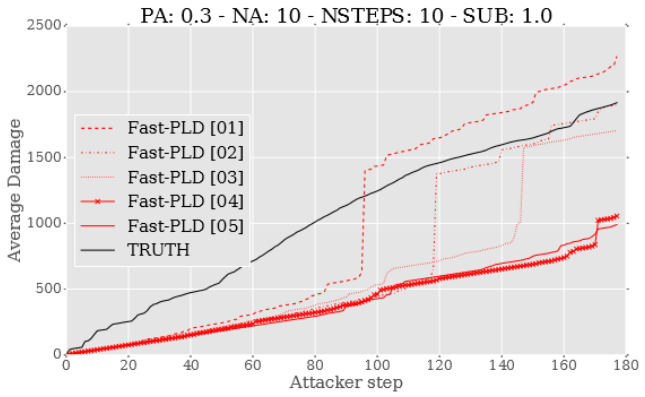
Experiment 3: Number of IDE Iterations



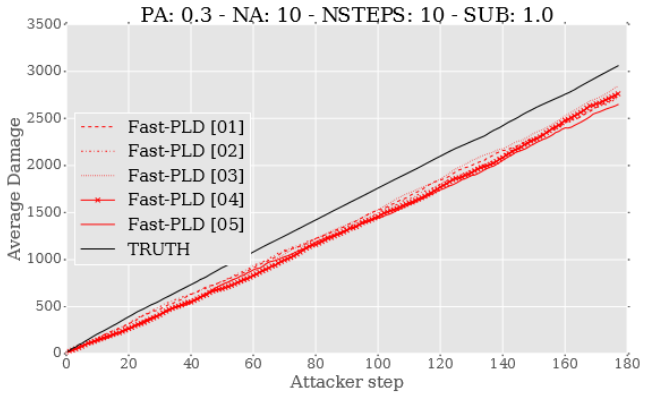
(a) Zipf ($SUB = 0.8$)



(c) Gaussian ($SUB = 0.8$)



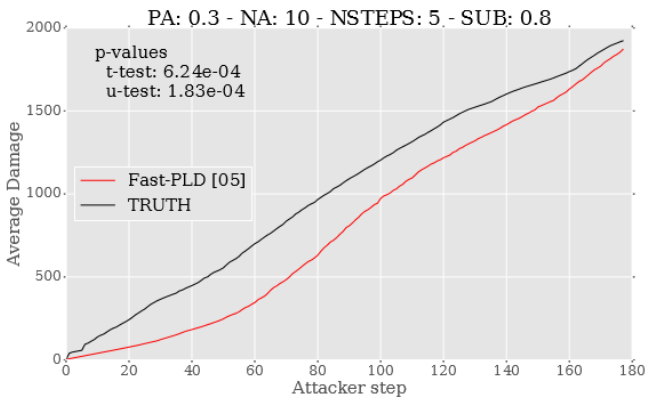
(b) Zipf ($SUB = 1.0$)



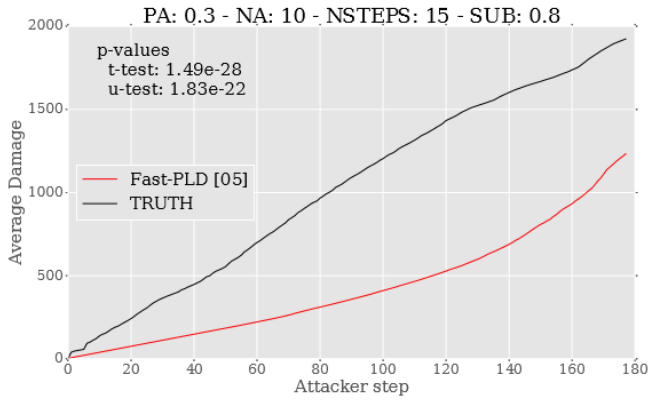
(d) Gaussian ($SUB = 1.0$)

- We expected that high NIDE => increased prob of sampling “good” atoms => reduced exp. damage
- **Zipf:** increasing NIDE significantly reduces damage. Spikes correspond to cases where attacker compromised high value nodes despite deception. *Rarely happens with NIDE > 3.*
- Fully rational attackers can be better deceived.
- NIDE > 5 has little impact.

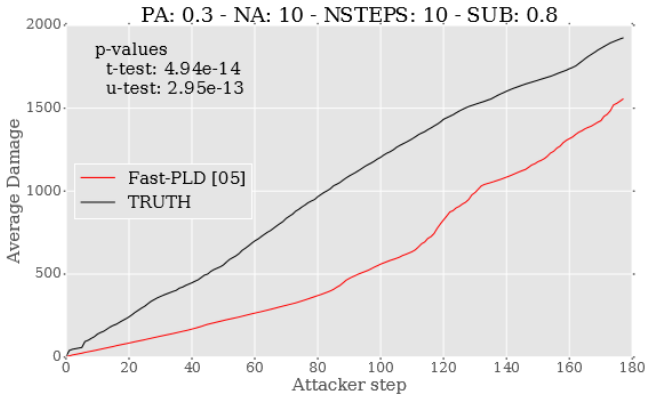
Experiment 4: Number of Steps



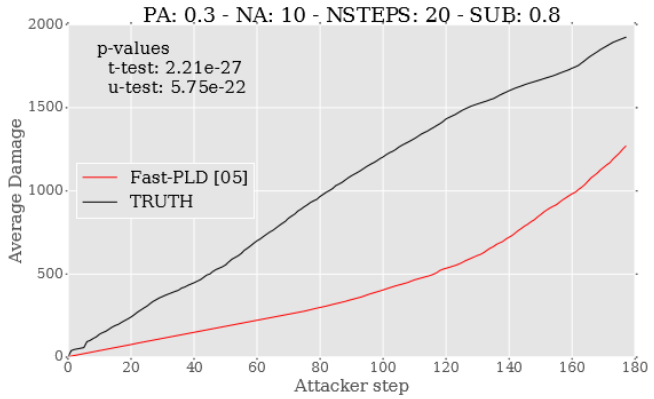
(a) $NSTEPS = 5$



(c) $NSTEPS = 15$



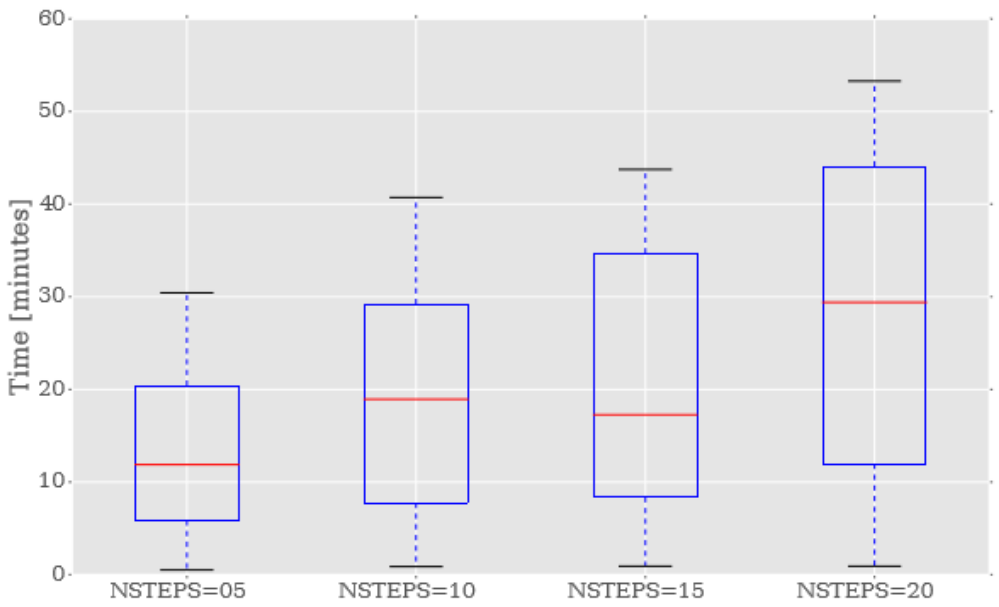
(b) $NSTEPS = 10$



(d) $NSTEPS = 20$

- NSTEPS
  - Low values => more damage
  - NSTEPS > 10 => significant reduction in damage
  - But increasing upto 100 yields little benefit

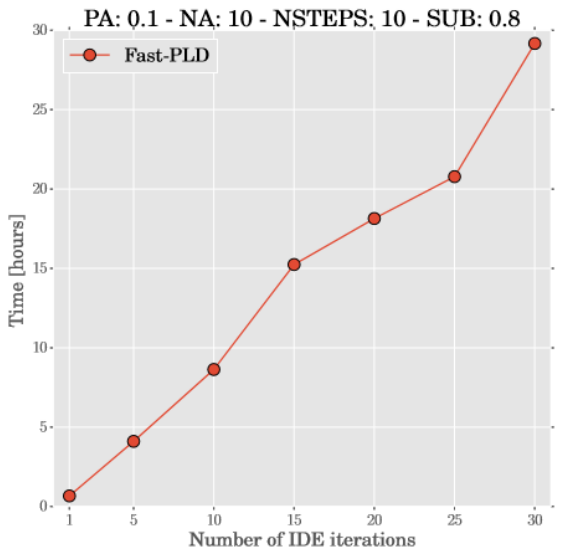
Experiment 5: Run-time of Fast-PLD



- Significant increase in execution time for NSTEPS > 15.
- NSTEPS in the 10-15 range seems to give good performance with reasonable run time.

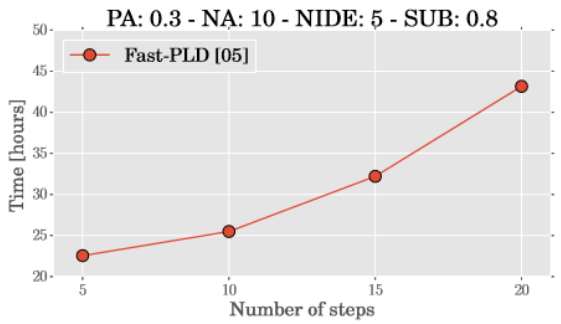
Experiment 5: Run-time of Fast-PLD

60 nodes



(a)

180 nodes



- Fast-PLD is an offline algorithm that can take ~1 day to run for the best parameter settings (NIDE=5, NSTEPS = 10-15).
- Computes everything once and then serves up answers as the attacker scans.
- Only need to update when something changes in the network.
- Online time is 5-6 milliseconds.

Conclusion

- Logic has an important role to play in many real world settings.
- Choice of settings where a real impact can be made is key.
- Logic is great for
  - Explanations
  - Reasoning about adversaries' state of mind
- But
  - KISS (Keep it Simple Stupid)

Contact Information

V.S. Subrahmanian
Dept. of Computer Science
Dartmouth College
Hanovr, NH 03755.
vs@dartmouth.edu
@vssubrah
www.cs.dartmouth.edu/vs/