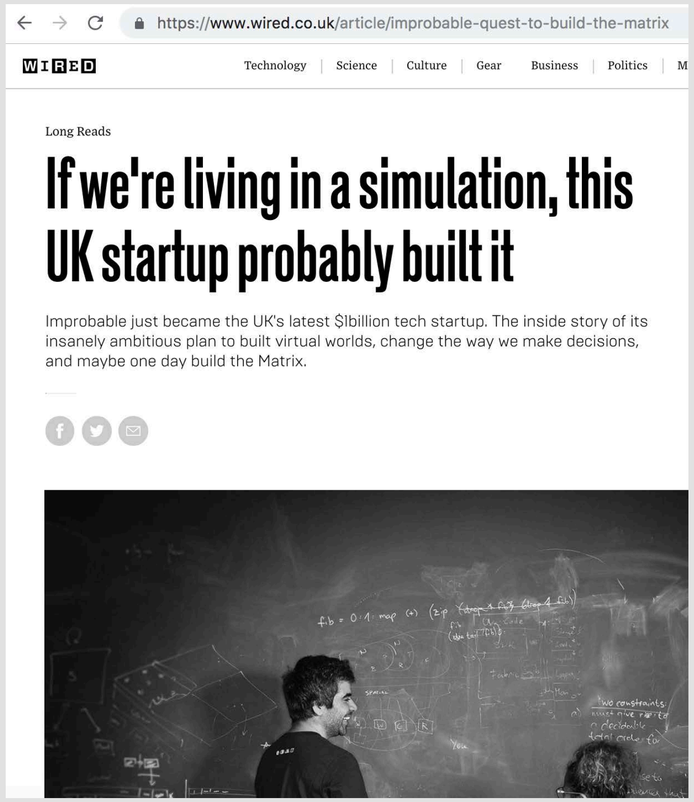
Poorly Supervised Learning

How to engineer labels for machine learning when you don’t have any.



Dr. Chris Anagnostopoulos
Head of Research
www.improbable.io

Honorary Senior Lecturer
Imperial College

Background

Maths
Cambridge

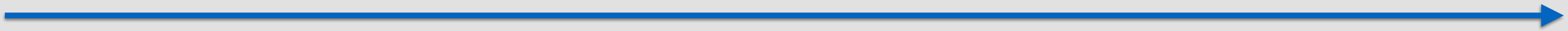
AI
Edinburgh

Logic
Athens

PhD Stats
Imperial

Fellow
Statslab

Lecturer
Imperial Stats



Background

Maths
Cambridge

AI
Edinburgh

Logic
Athens

PhD Stats
Imperial

Fellow
Statslab

Lecturer
Imperial Stats

Chief Data Scientist
Co-Founder Mentat

GOV.UK

CISCO

BARCLAYS

Background

Maths
Cambridge

AI
Edinburgh

Logic
Athens

PhD Stats
Imperial

Fellow
Statslab

Lecturer
Imperial Stats

Chief Data Scientist
Co-Founder Mentat

GOV.UK

CISCO

BARCLAYS

Head of Research
Improbable

Background

Maths
Cambridge

AI
Edinburgh

Logic
Athens

PhD Stats
Imperial

Fellow
Statslab

Lecturer
Imperial Stats

Chief Data Scientist
Co-Founder Mentat

GOV.UK

CISCO

BARCLAYS

Head of Research
Improbable

Background

Maths Cambridge

AI Edinburgh

Logic Athens

PhD Stats Imperial

Fellow Statslab

Lecturer Imperial Stats

Chief Data Scientist
Co-Founder Mentat

GOV.UK

CISCO

BARCLAYS

Head of Research
Improbable

Methodology

interface

Application

```
$ python
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf=RandomForestClassifier()
>>> clf.fit(X,y)
>>> clf.predict(Xnew)

$ R
> library(randomForest)
> clf = randomForest(y~X)
> predict(clf, newdata=Xnew)
```

Background

Maths Cambridge

AI Edinburgh

Logic Athens

PhD Stats Imperial

Fellow Statslab

Lecturer Imperial Stats

Chief Data Scientist
Co-Founder Mentat

GOV.UK

CISCO

BARCLAYS

Head of Research
Improbable

```
$ python
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf=RandomForestClassifier()
>>> clf.fit(X,y)
>>> clf.predict(Xnew)

$ R
> library(randomForest)
> clf = randomForest(y~X)
> predict(clf, newdata=Xnew)
```

(X, y)

image courtesy of Stanford DAWN

(X, y)

Predict the class **y** of an object, given a description **x** of that object

Function approximation:

$$\tilde{f} = \underset{f \in \Phi}{\operatorname{argmin}} \sum_{i=1}^{n} L(f(x_i), y_i)$$

Parametric estimation:

$$\tilde{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{n} L(f_\theta(x_i), y_i)$$

learning by example

Relies on a classifier (family of functions) and a labelled dataset.

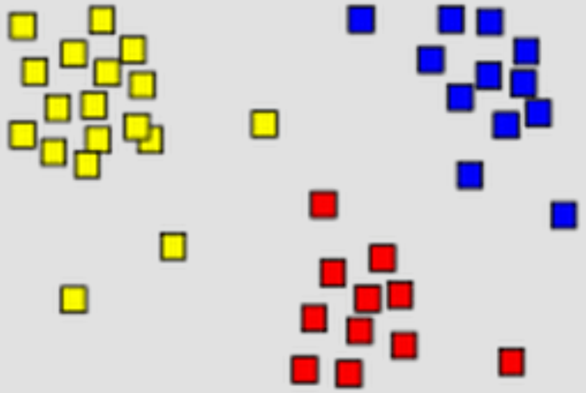
(X, y)

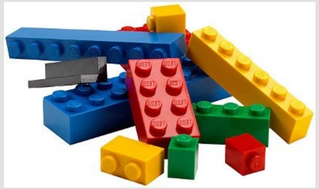
Few labels

Multiple noisy labels

Not missing
at random

vs





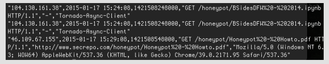
Unit of analysis /
missing context

Class imbalance

Expert prior rules

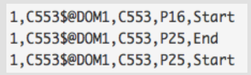
The X and the y in Cybersecurity

The X in Cybersecurity

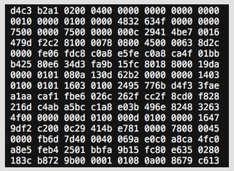


web proxy logs

windows authentication logs

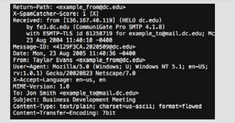


process logs

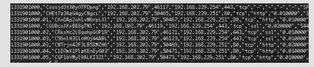


packet capture

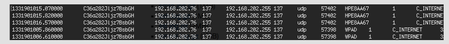
A broad array of data formats, often capturing different aspects of the same activity.



email headers



network flow



DNS logs

firewall logs

Unix syslog

An example: exfiltration

social engineering
spearphishing

LinkedIn

Email

malicious
infrastructure

spam filter

User clicks

web proxy

malware
download

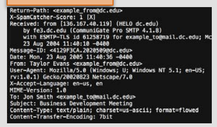
antivirus

privilege
escalation

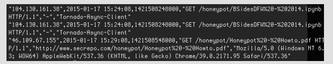
login

file scanning
and encryption

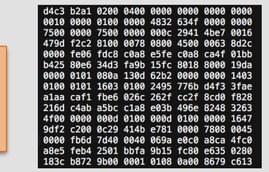
firewall



email headers

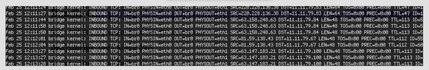


web proxy logs

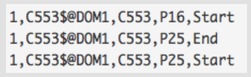


packet capture

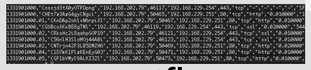
Unix syslog



firewall logs



process logs



netflow

windows
authentication logs

The y in Cybersecurity

actual attacks are
(thankfully) very rare

but plenty of signals worth surfacing:
near-misses, early attack stages,
risky behaviours, “suspect” traffic

The y in Cybersecurity

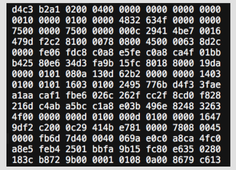
Mountains of “soft” labelling rules captured as search queries, automated rules (NIDS) or look-ups (intel)

vs

Very little to no time to produce manual labels at any reasonable scale

The y in Cybersecurity

Benign vs malware in pcap



Network intrusion detection systems (NIDS) logs

Threat intel: observables, indicators of compromise

Threat intel: kill chain

Threat intel: TTPs

no **y** exists for the vast majority of traffic

even when **y** is available, it's often about a complex **X** obtained by data fusion

often a noisy **y** is available by checking against a DB

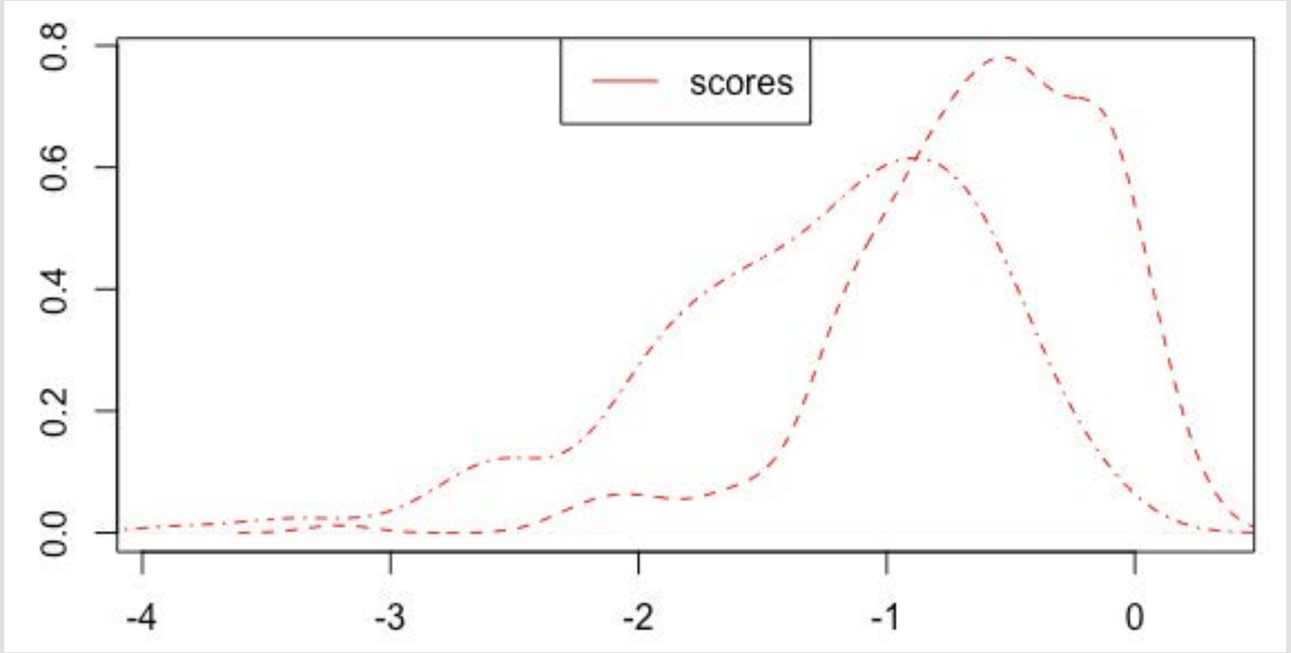
the user is looking for patterns and lookalikes

Solution: label engineering

Standard tools: data fusion and feature engineering

	label	max run	entropy	prop distinct
fvmiu	0	3	2.07	1.00
owg	0	2	2.31	1.00
delbert	1	2	1.10	0.86
ucamky	0	2	1.91	1.00
russell	1	2	1.37	0.71
yzba	0	2	3.07	1.00
paul	1	2	1.93	1.00
rdhts	0	5	1.18	1.00
dennis	1	2	1.11	0.83
william	1	2	1.55	0.71

	AUC
logReg on characters	0.93
logReg on features	0.96
randomForest on characters	0.97
single feature: max run	0.82
single feature: entropy	0.78
single feature: prop distinct	0.72



expert-defined features
are usually meant as scores

A middle-ground between rules and classifiers

cat

dog

“cats are smaller than dogs”

- hard to explicitly describe how cats differ from dogs.
- learning by example avoids that, but needs a gazillion labels
- human learning is a sensible mix of examples and imperfect rules

- the trick is to interpret rules as **noisy, partial labels**
- a feature engineering step is needed to map rules to raw data

A middle-ground between rules and classifiers

Raw Data

01011010110101000
10101101010111010
101101010111101…

Features

```
{
   ‘size’: ‘large’,
   ‘ear shape’: ‘pointy’,
   ’color:’ [‘white’, ‘orange’],
   ‘nose shape: …
}
```

Expert Ruleset

if it is very large, then it is not a cat,

if its ears are pointy, then it is likely a cat

…

Noisy Labels

Example ID	Rule 1	Rule 2	Rule 3	Ground Truth
01.jpeg	Cat	Dog	Cat	Cat
02.jpeg	Cat	-	Cat	-
03.jpeg	-	-	Dog	-
04.jpeg	-	Cat	-	Cat

A middle-ground between rules and classifiers

Raw Data

01011010110101000
10101101010111010
101101010111101…

Features

```
{
   ‘size’: ‘large’,
   ‘ear shape’: ‘pointy’,
   ’color:’ [‘white’, ‘orange’],
   ‘nose shape: …
}
```

Expert Ruleset

if it is very large, then it is not a cat,

if its ears are pointy, then it is likely a cat

…

Noisy Labels

Example ID	Rule 1	Rule 2	Rule 3	Ground Truth
01.jpeg	Cat	Dog	Cat	Cat
02.jpeg	Cat	-	Cat	-
03.jpeg	-	-	Dog	-
04.jpeg	-	Cat	-	Cat

Data Programming **Manual**

A middle-ground between rules and classifiers

Raw Data

01011010110101000
10101101010111010
101101010111101…

Features

{
‘size’: ‘large’,
‘ear shape’: ‘pointy’,
’color:’ [‘white’, ‘orange’],
‘nose shape: …
}

Expert Ruleset

if it is very large, then it is not a cat,

if its ears are pointy, then it is likely a cat

…

Noisy Labels

Example ID	Rule 1	Rule 2	Rule 3	Ground Truth
01.jpeg	Cat	Dog	Cat	Cat
02.jpeg	Cat	-	Cat	-
03.jpeg	-	-	Dog	-
04.jpeg	-	Cat	-	Cat

Data Programming **Manual**

Semantically Easy

Semantically Tough

A middle-ground between rules and classifiers

Raw Data

Case Report
Class: Confidential
Jack Black interviewing suspect John Doe
Date of interview: 11/01/2015

Summary: Regarding the incident in London on the 6th of December 2010, the main suspect was John Doe.

Labelling Functions

Features

Expert Rules

- whichever name appears first in the document is the name of the suspect.

- if in the first paragraph the pattern “Suspect: [Name]” appears, then that is the suspect name. It can also be called “Person of Interest”.

- if the term “incident” or “crime” or “investigation” and the term “suspect” appears in the summary, followed by a name, and that name also appears in the first paragraph

Noisy Tags

Example ID	Rule 1	Rule 2	Rule 3	Ground Truth
01.doc	J. Black	-	J. Doe	J. Doe
02.doc	…	…	…	…
03.doc	…	…	…	…
04.doc	…	…	…	…

Data Programming **Manual**

Weakly supervised learning and data programming

snorkel

A training data creation and management system focused on information extraction

View the Project on GitHub
HazyResearch/snorkel

Download ZIP File | Download TAR Ball | View On GitHub

This project is maintained by HazyResearch

Part of the Stanford DAWN project

Snorkel: A System for Fast Training Data Creation

Snorkel is a system for rapidly **creating, modeling, and managing training data**, currently focused on accelerating the development of *structured or "dark" data extraction applications* for domains in which large labeled training sets are not available or easy to obtain.

Today's state-of-the-art machine learning models require *massive* labeled training sets--which usually do not exist for real-world applications. Instead, Snorkel is based around the new data programming paradigm, in which the developer focuses on writing a set of *labeling functions*, which are just scripts that programmatically label data. The resulting labels are noisy, but Snorkel automatically models this process—learning, essentially, which labeling functions are more accurate than others—and then uses this to train an end model (for example, a deep neural network in TensorFlow).

Surprisingly, by modeling a noisy training set creation process in this way, we can take potentially low-quality labeling functions from the user, and use these to train high-quality end models. We see Snorkel as providing a general framework for many weak supervision techniques, and as defining a new *programming model for weakly-supervised machine learning systems*.

Spark Summit Attendees: See how to run Snorkel on Spark!

Weakly supervised learning and data programming

Labelling functions is a middle ground between feature engineering and labelling. It is a very information-efficient use of analyst time.

Challenge: domain primitives

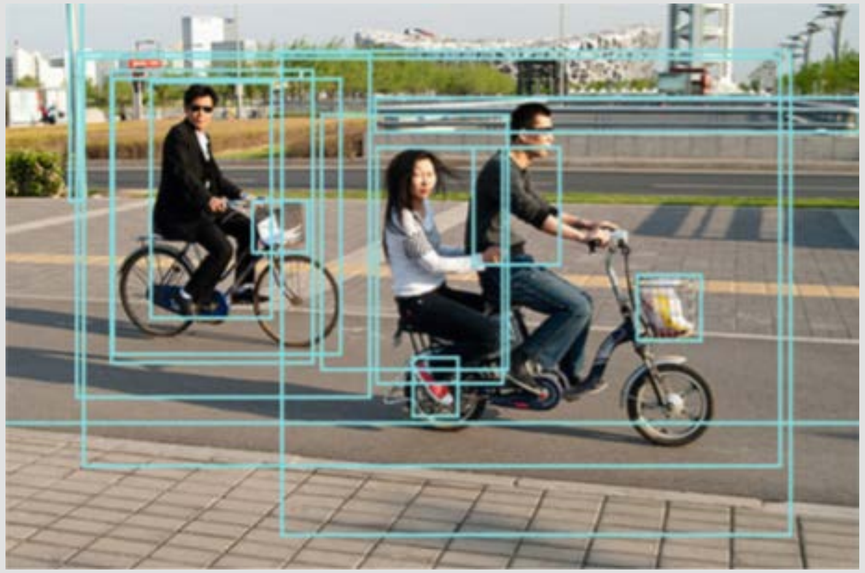
This was most striking in the chemical name task, where complicated surface forms (e.g., “9-acetyl-1,3,7-trimethyl-pyrim- idinedione") caused tokenization errors which had significant effects on system recall. This was corrected by utilizing a domain-specific tokenizer.

Data Programming with DDLite:
Putting Humans in a Different Part of the Loop

Henry R. Ehrenberg, Jaeho Shin, Alexander J. Ratner, Jason A. Fries, Christopher Ré
Stanford University

Weakly supervised learning and data programming

Is all expert labelling heuristic? (as opposed to ground truth labelling)



vision and other types of cognition are often hard to reduce to a number of heuristics

but sometimes they are heuristics on top of “deep” domain primitives (e.g., a “circle” and a “line”)

Weakly supervised learning and data programming

Is all expert labelling heuristic? (as opposed to ground truth labelling)

in non-sensory domains, we would expect expert labelling to be much more rules-based.

eliciting the full decision tree voluntarily is impossible, however; in practice, it seems experts use a forest methodology, rather than one relying on a single deep tree

regulation plays a role here. If you are expected to follow a playbook and document your work, then a procedural description is natural

Weakly supervised learning and data programming

Domain Primitives

Log analysis: sessions, action sequences, statistical profiling (“*unusually high*”), data fusion (job roles), integration with threat intel knowledge DBs, jargon (“*C2-like behaviour*”, “*domain-flux*”)

Threat intel: formal representation of TTPs, regex for cyber entities (hashes, IPs, malware names …),

The whole point is to empower the expert to express their heuristics programmatically in an easy, accurate and scalable fashion

Weakly supervised learning and data programming


```
{
    "IP": "77.75.78.160",
    "datetime" : "2016-06-03 23:20:51",
    "datetime_unixms" : 1464996051000,
    "request" : "GET /robots.txt HTTP/1.1",
    "domain" :"-",
    "agent" : "Mozilla/5.0 (compatible; SeznamBot/3.2)"
}
```


“Hey. We’re building a model for user agents. Does this one look suspicious?”

“Yep! That’s a bad one.”

“Great! Out of curiosity, why?”

“The SeznamBot version is wrong”

Where you had one additional label, now you have a labelling rule.

Weakly supervised learning and data programming


```
{
    "IP": "77.75.78.160",
    "datetime" : "2016-06-03 23:20:51",
    "datetime_unixms" : 1464996051000,
    "request" : "GET /robots.txt HTTP/1.1",
    "domain" :"-",
    "agent" : "Mozilla/5.0 (compatible; SeznamBot/3.2)"
}
```


“Hey. We’re building a model for user agents. Does this one look suspicious?”

“Yep! That’s a bad one.”

“Great! Out of curiosity, why?”

“The IP is blacklisted”

This is a case of distant learning that might or might not induce a signal on the user agent itself (a small fraction of all attacks will have anomalous user agents, so label is very noisy)

Weakly supervised learning and data programming


```
{
    "IP": "77.75.78.160",
    "datetime" : "2016-06-03 23:20:51",
    "datetime_unixms" : 1464996051000,
    "request" : "GET /robots.txt HTTP/1.1",
    "domain" :"-",
    "agent" : "Mozilla/5.0 (compatible; SeznamBot/3.2)"
}
```


“Hey. We’re building a model for user agents. Does this one look suspicious?”

“Yep! That’s a bad one.”

“Great! Out of curiosity, why?”

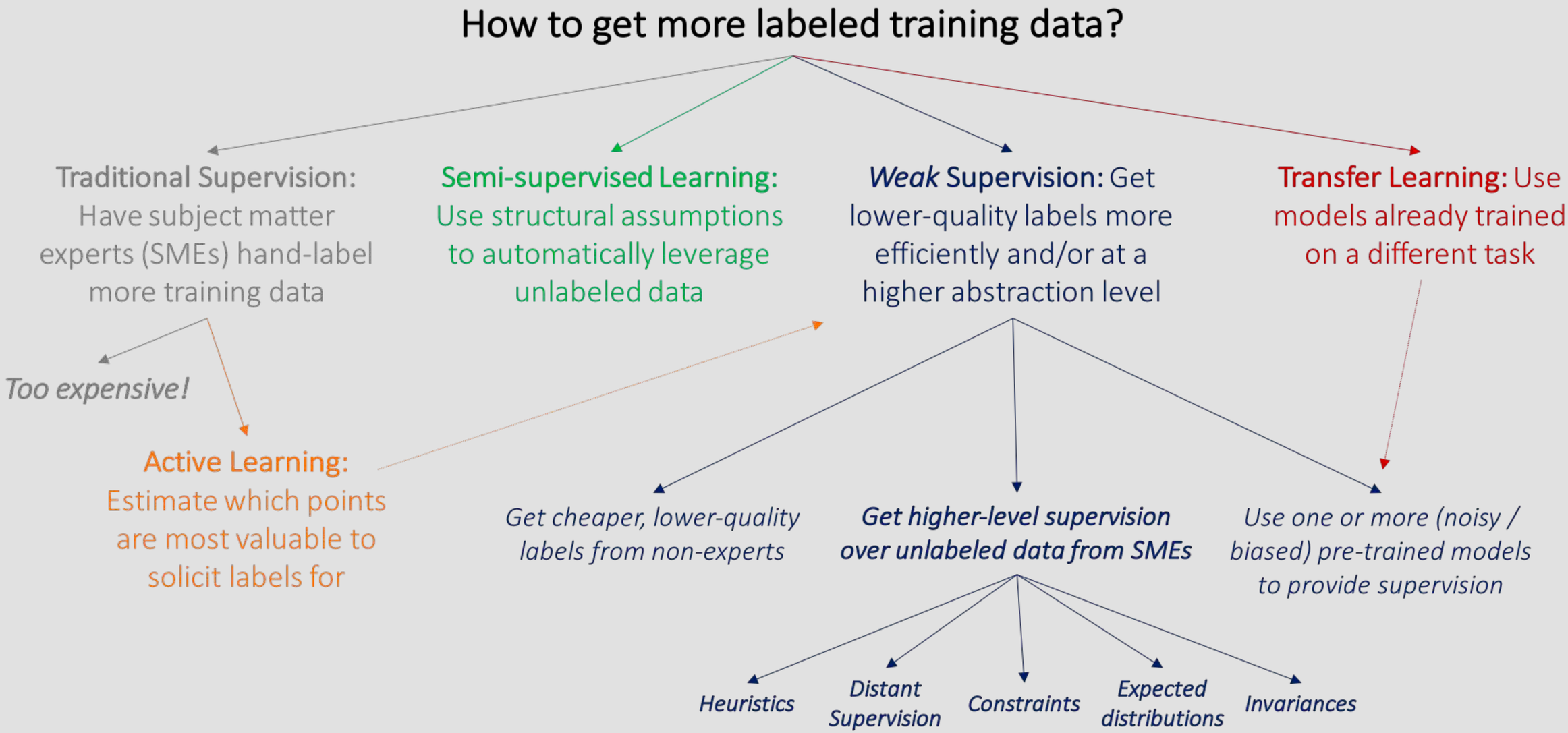
“The IP is from Russia”

This is case of missing context that could be turned into a labelling rule if we enrich the data. Again, the label might be noisy as far as the user agent is concerned - but that’s OK!

Breaking free of (X,y): label engineering

semi-supervised	cost-sensitive learning
label-noise robustness	**weakly supervised**
active learning	**data programming**
unsupervised learning	**distant learning**
multi-view learning	importance reweighting
boosting	co-training

Breaking free of (X,y): label engineering



Maths and algos

Same loss as before: $\mathbb{E}_{(X,y)\sim\pi} L(f_\theta(x_i), y_i)$

Enter labelling functions: $\lambda_i : \mathcal{X} \to \{-1, 0, 1\}$

missing value

“an IP that we have never seen before, and the TLD is not a .com, .co.uk, .org”

“an IP that appears in our blacklist”

“an IP that appears in our greylist”

Generative model: $\mu_{\alpha,\beta}(\Lambda, Y) = \frac{1}{2} \prod_{i=1}^{m} \left(\beta_i \alpha_i \mathbf{1}_{[\Lambda_i = Y]} + \beta_i (1 - \alpha_i) \mathbf{1}_{[\Lambda_i = -Y]} + (1 - \beta_i) \mathbf{1}_{[\Lambda_i = 0]}\right)$

“each labelling function λ has a probability β of labelling an object and α of getting the label right”

$\mu_{\alpha,\beta}(\Lambda, Y)$ remember: we do not have any Ys

Maths and algos

Maximum Likelihood:

$$\mu_{\alpha^\star,\beta^\star}(\Lambda, Y) \text{ where } \alpha^\star, \beta^\star \leftarrow \underset{\alpha,\beta}{\operatorname{argmax}} \sum_{x \in X} \log P_{(\Lambda,Y)\sim\mu_{\alpha,\beta}}(\Lambda = \lambda(x))$$

“what values of α and β seem most likely given our unlabelled data X?”.
Average over all possible labels.

Noise-aware empirical loss:

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{x \in X} \mathbb{E}_{(\Lambda,Y)\sim\mu_{\alpha^\star,\beta^\star}} [L(x, y) \mid \Lambda = \lambda(x)]$$

Assumptions:

$$y \perp (f(x) \mid \lambda(x))$$

“the labelling functions tell you all you need to know about the label”

$$\lambda_i(x) \perp \lambda_j(x) \mid y$$

“the labelling functions are independent”
So overlaps are very informative!

Semi-supervised learning and co-training

Expectation-Maximisation: treat unlabelled examples as missing labels

Bootstrapping: retrain the model on a noised-up version of its own predictions

Feature discovery: cluster the unlabelled data, use the labels as features

Entropy regularisation: place the decision boundary in low density regions

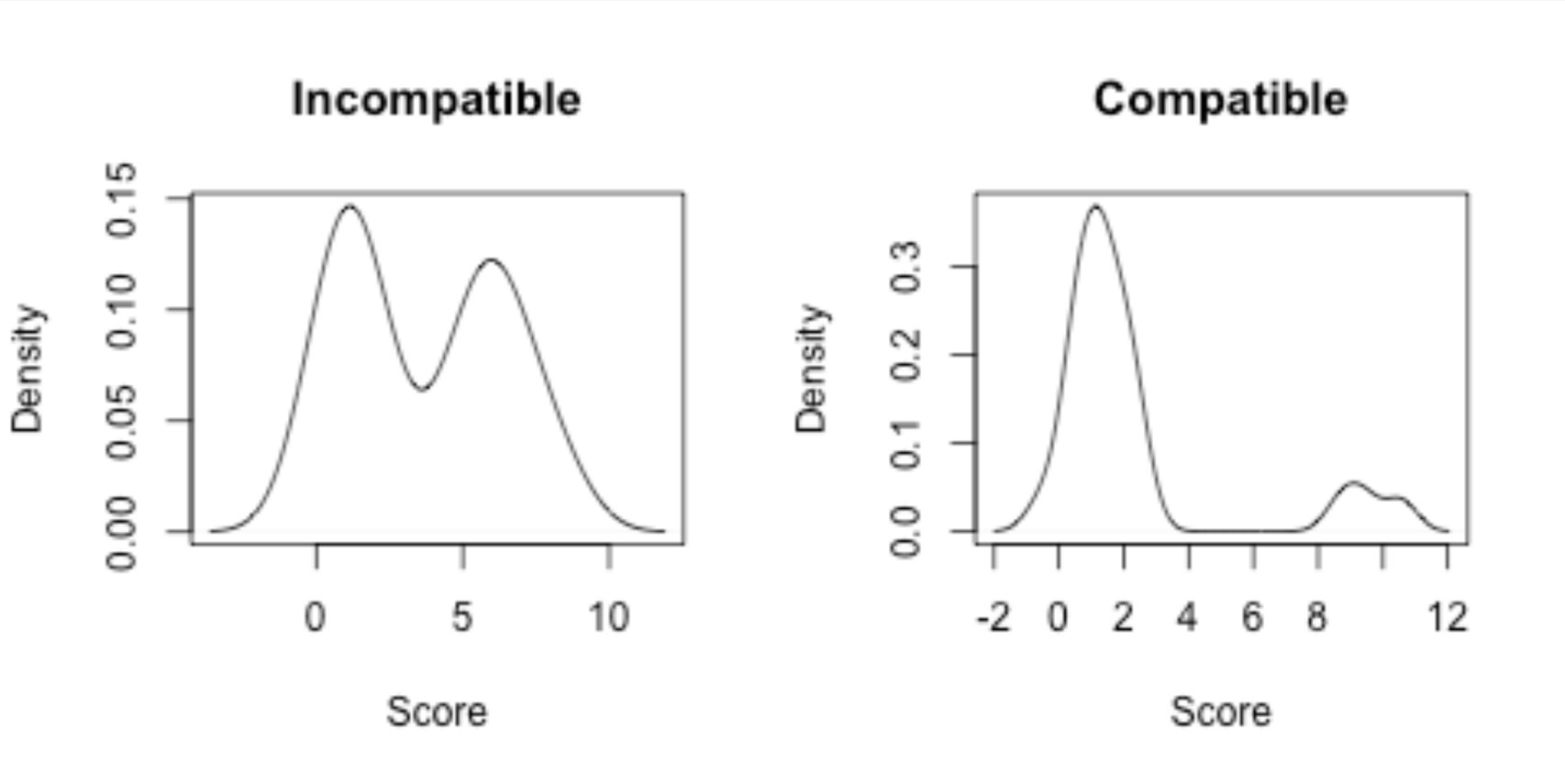
Co-training: train two classifiers using labelled data using different features for each one. Then retrain one using the labels produced by the other on unlabelled data.

Semi-supervised learning in conjunction with
weak learning is a very powerful technique

Generalised Expectation Criteria

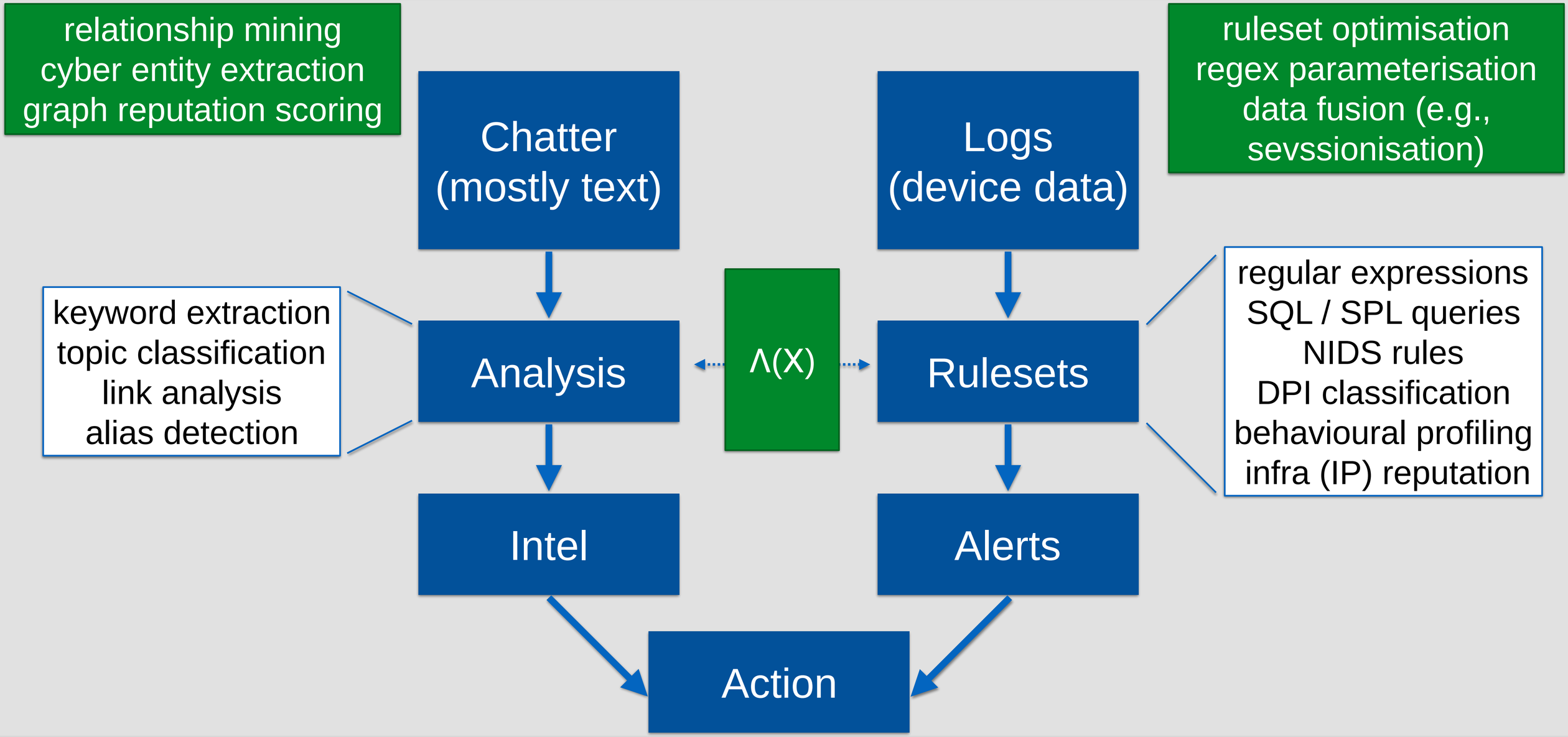
We know attacks are rare. This is a constraint on our model.

[Mann & McCallum 2010]

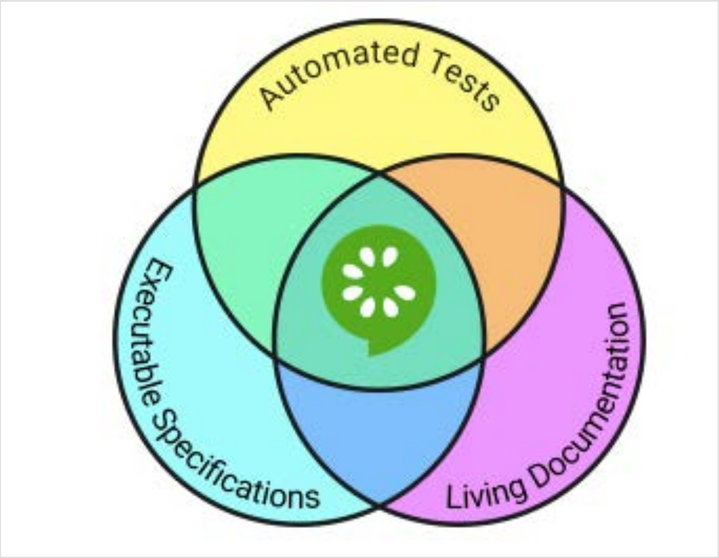


Combination of GE with weak learning is an open problem

Software


relationship mining
cyber entity extraction
graph reputation scoring
ruleset optimisation
regex parameterisation
data fusion (e.g., sevssionisation)
Chatter (mostly text)
Logs (device data)
keyword extraction
topic classification
link analysis
alias detection
Analysis
Λ(X)
Rulesets
regular expressions
SQL / SPL queries
NIDS rules
DPI classification
behavioural profiling
infra (IP) reputation
Intel
Alerts
Action

Open Work: BDD for Data Driven Adaptive Development



Gherkin is a "Business Readable,
Domain Specific Language"

From Unit Tests
to
Behavioural Tests

```
Feature: Serve coffee
  Coffee should not be served until paid for
  Coffee should not be served until the button has been pressed
  If there is no coffee left then money should be refunded

Scenario: Buy last coffee
  Given there are 1 coffees left in the machine
  And I have deposited 1$
  When I press the coffee button
  Then I should be served a coffee
```

Use Cases in Defence and Security

Anomaly Detection in Cyber Logs

< 10 true positives
billions of log lines

vast amounts of threat intel and custom Splunk queries

Relation Extraction from Investigator Reports

~10K case reports
2 annotated ones

semi-standardised language and HR databases allows for easy creation of labelling rules

“Mr. John SMITH”
↓
suspect

Detect New Buildings in Aerial Photography

annotated data for UK,
0 annotations for territories of interest

huge rulesets already exist in legacy software

Conclusion

Experts often give you one label where they could simply tell you the rule they are using to produce it

Labelling functions are more intuitive than feature engineering; they can yield vastly more labels than manual annotation; and they capture know-how.

Forcing the user to deal with the tradeoff between accuracy and coverage is unnecessary. Boosting has taught us how to combine weak heuristics.

At the end of all this, you can still use your favourite classifier. Focus on the real pain point: **dearth** of labels.

Glossary

machine learning	αυτόματη μάθηση
supervised classification	επιβλεπόμενη αυτόματη ταξινόμηση
weakly semi-supervised learning	αδύναμα ημι-επιβλεπόμενη αυτόματη ταξινόμηση
unsupervised learning	μάθηση δίχως επίβλεψη
K-means clustering	K-μέσων ομαδοποίηση
kernel density estimation	αλγόριθμοι εκτίμησης πυρήνα
outlier and anomaly detection	σύστημα ανίχνευσης περιπτώσεων άτυπης ή ανώμαλης συμπεριφοράς