Data Analysis Quiz

Match the type of analytics to its characteristic.
(Anomaly (A), Hybrid (H), Misuse (M))

A	model normal network and system behavior and identify deviations from the norm.
H	combination of misuse and anomaly detection
M	detect known attacks using signatures of those attacks
M	can detect known types of attacks without generating a lot of false positives
A	have the ability to detect zero-data attacks

Machine Learning Review

$$y = f(x)$$

Output

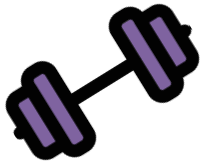
Prediction function

Training/testing example

Machine Learning Review

Training: given a *training set* of labeled examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set

Testing: apply f to a never before seen *test example* x and output the predicted value $y = f(x)$



Test/Training Split

Real-world Process

drawn randomly

split randomly

Data D

split randomly

Training Data D_{train}

$(x_1, y_1), \ldots, (x_n, y_n)$

D_{train}

Learner

f

Test Data D_{test}

$(x_1, y_1), \ldots (x_k, y_k)$

Machine Learning Example

Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

Machine Learning Example

Steps

Training

Training Images

Image Features

Training Labels

Training

Learned Model

Machine Learning Example

Steps

Testing

Test Image

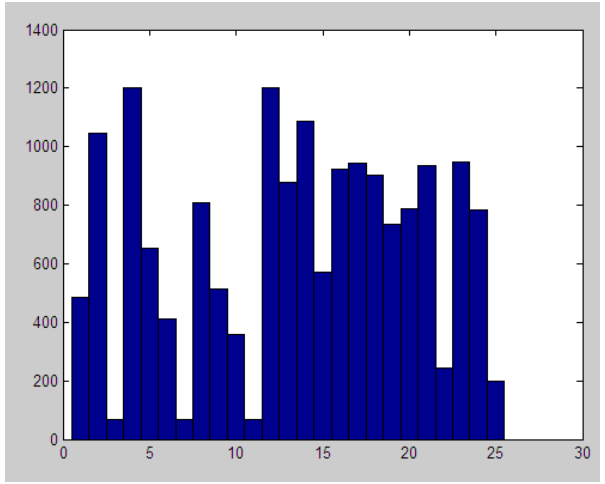
Image Features

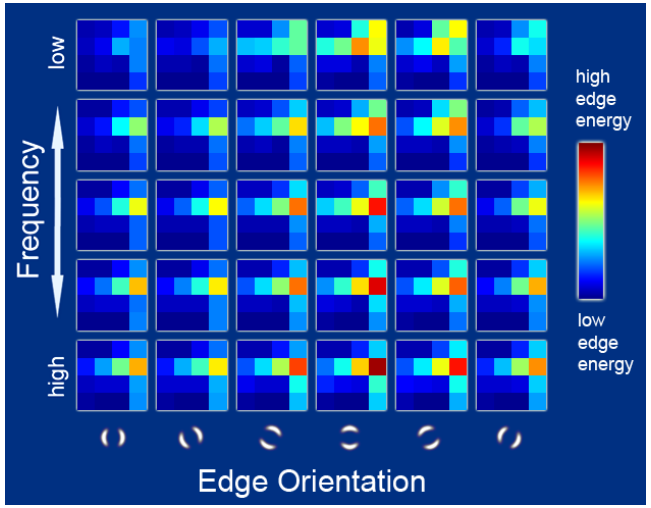
Learned model

Prediction

Machine Learning Features

- Raw pixels
- Histograms
- GIST descriptors
- …

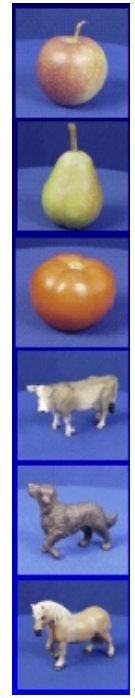




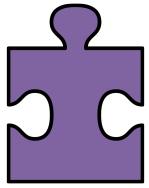
Generalization

How well does a learned model generalize from the data it was trained on to a new test set?

Training set (labels known)



Test set (labels unknown)



ML Types Quiz

Match the ML type to its characteristic.

(Supervised (S), Semi-supervised(SS), Unsupervised(U))

U — the main task is to find patterns, structures, or knowledge in unlabeled data

S — the task is to find a function or model that explains the data

SS — some of the data is labeled during acquisition

Performance Measures

Error Rate

- Fraction (or percentage) of false predictions

Accuracy

- Fraction (or percentage) of correct predictions

(Can be generalized to multi-class case.)

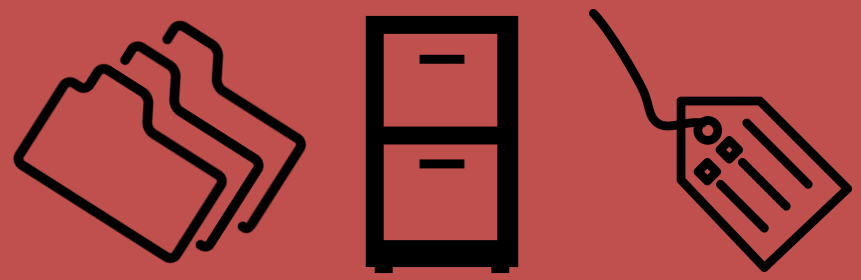
Precision/Recall

- Example: binary classification problems (classes pos/neg)
  - Precision: Fraction (or percentage) of correct predictions among all examples predicted to be positive
  - Recall: Fraction (or percentage) of correct predictions among all real positive examples

Classification Problem

Given a set of example records…

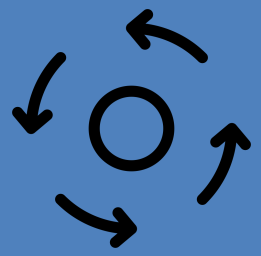
- Each record consists of
  - A set of attributes
  - A class label



Build an accurate model for each class based on the set of attributes

Use the model to classify future data for which the class labels are unknown

Decision Tree Learning



A set of training examples is repeatedly partitioned until all the examples in each partition belong to one class or the partition is sufficiently small

The decision tree can be thought of as a set sentences (in Disjunctive Normal Form) written propositional logic

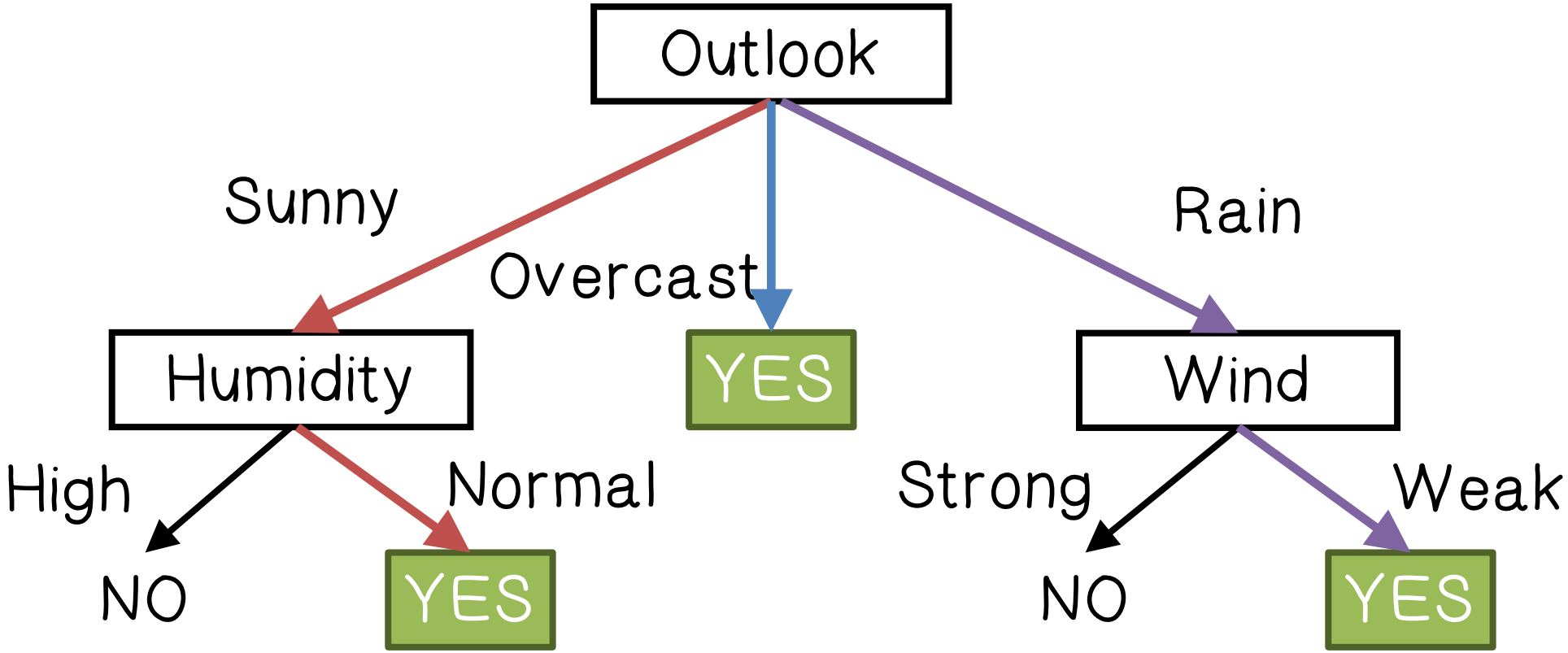
Decision Tree Learning

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

Decision Tree Learning

$$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal}) \vee (\text{Outlook} = \text{Overcast}) \vee (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$$



Tom M. Mitchell, *Machine Learning,* McGraw-Hill, 1997

Building a Decision Tree

First test all attributes and select the one that would function as the best root;

Break-up the training set into subsets based on the branches of the root node;

Test the remaining attributes to see which ones fit best underneath the branches of the root node;

Continue this process for all other branches until

- all examples of a subset are of one type (i.e., same class label)
- there are no more attributes left (default value should be majority classification)

Building a Decision Tree

Determining which attribute is best (Entropy & Gain)

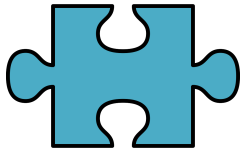
Entropy (E) is the minimum number of bits needed represent the examples according to their class labels; or roughly, how "pure" the examples are, that is, if the examples are evenly distributed into different classes, the entropy is the maximum and if the examples are all in a single class the entropy is minimum.

Building a Decision Tree

The information gain **G(S,A)** where **A** is an attribute

$$G(S,A) = E(S) - \sum_{v \text{ in } Values(A)} (|S_v| / |S|) * E(S_v)$$

Higher **G(S,A)** means that **A** is better at separating samples in **S** according to their classification, that is, **S** are partitioned into "purer" subsets



Decision Tree Quiz

Select the true statements with regards to decision tree based detection models:

- [x] Can supplement honeypot analysis
- [x] Can supplement penetration testing
- [ ] Cannot highlight malicious traffic
- [ ] Cannot characterize known scanning activity
- [x] Can detect previously unknown network anomalies

Clustering

Construct a partition of training examples that optimizes the chosen partitioning criterion, e.g., a “distance” or ”similarity” function (e.g., Euclid or Mahalanobis distance)

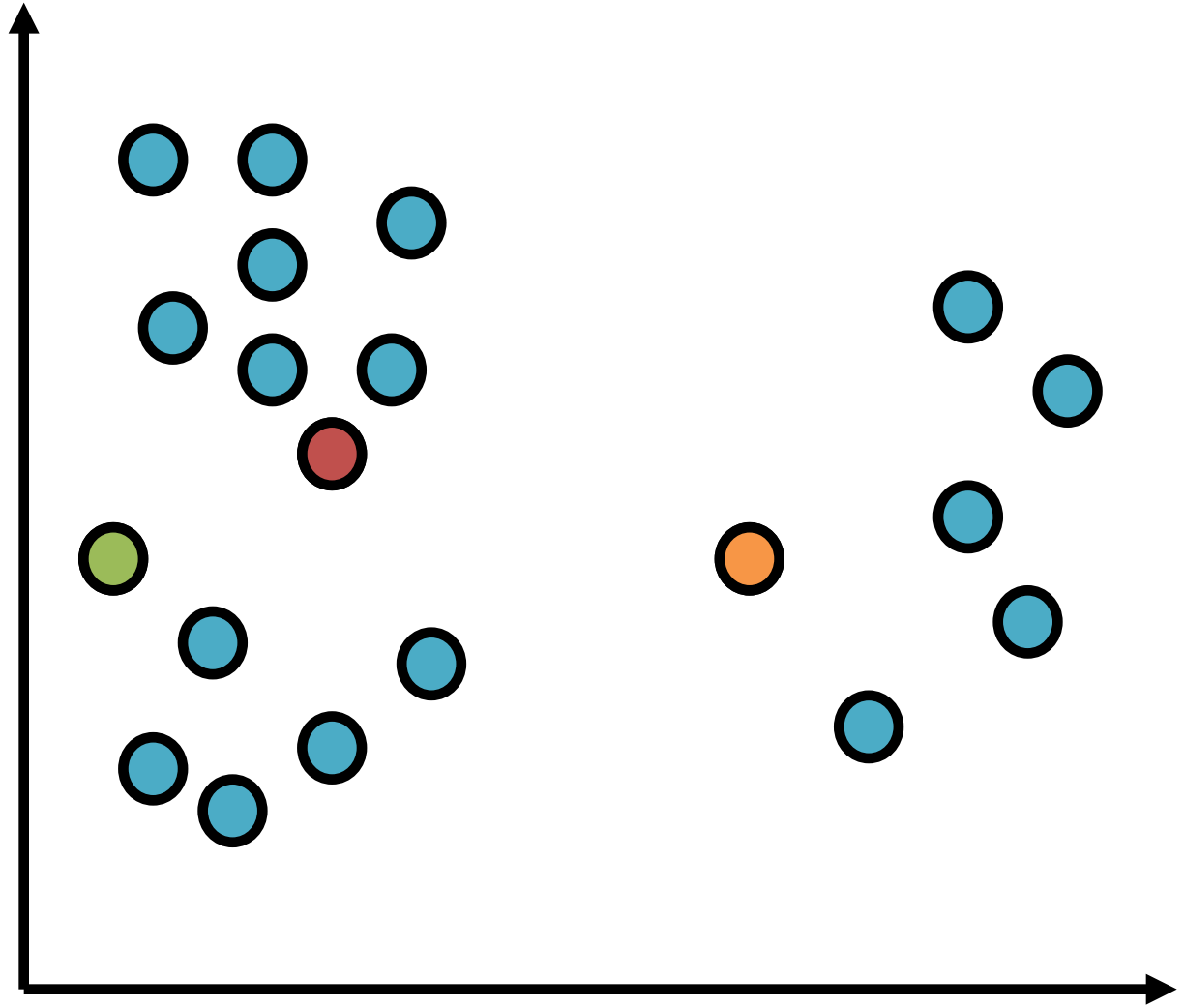
Clustering

Examples in a cluster are more similar to each other than examples from different clusters

- i.e., the distance between two examples in a cluster is smaller than the distance between examples from two clusters

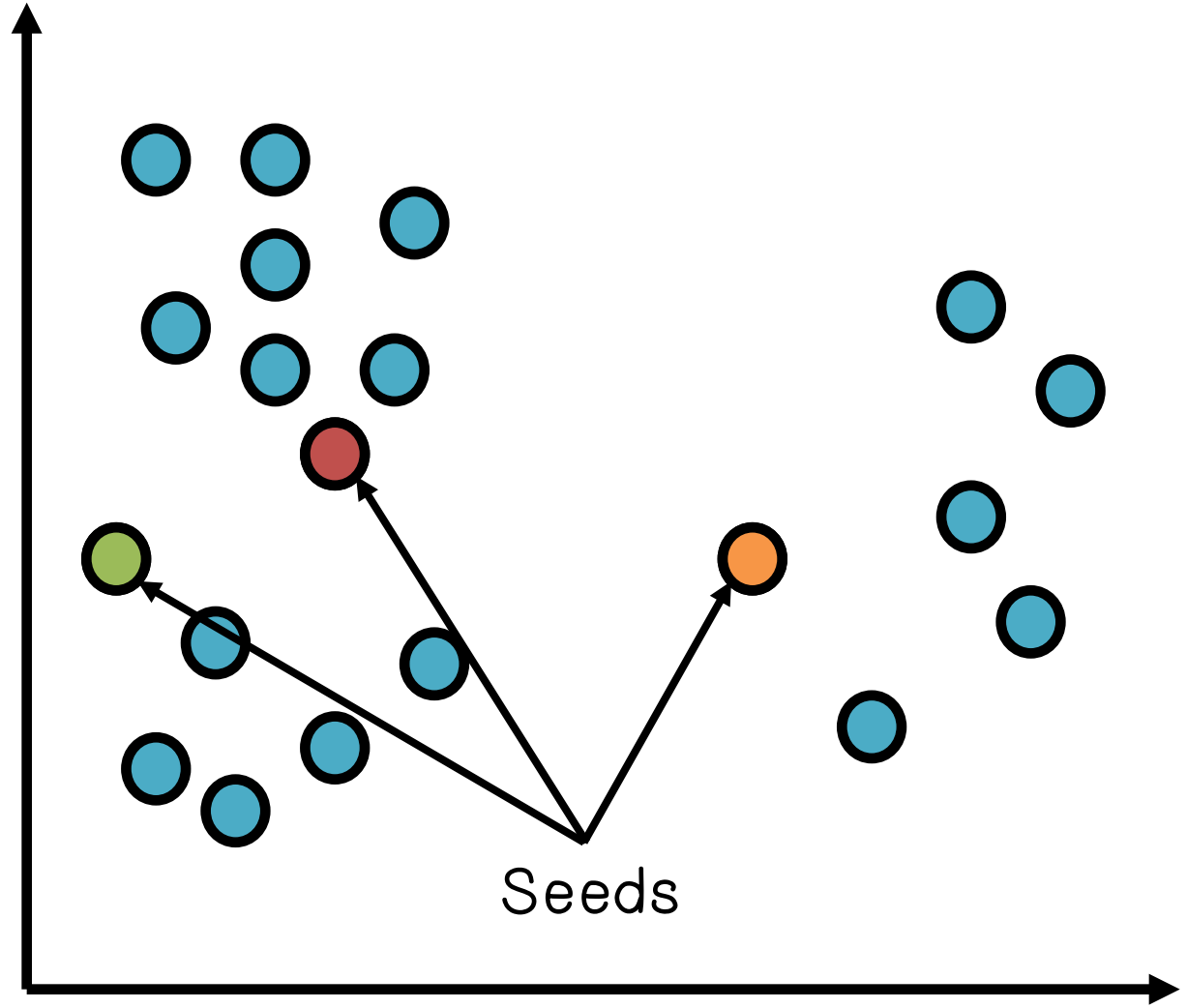
Clustering

Predetermined number of clusters



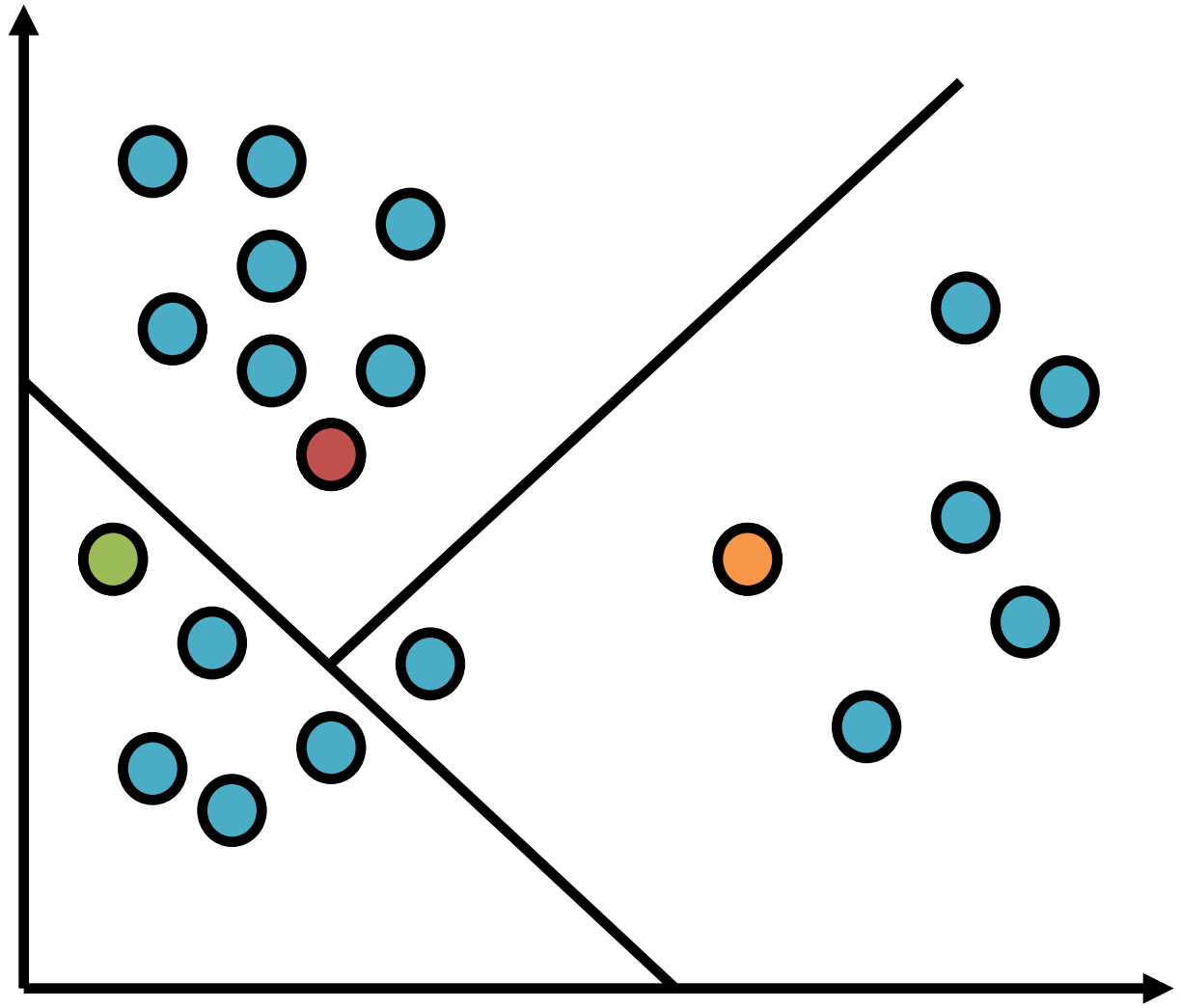
Clustering

Start with seed clusters of one element



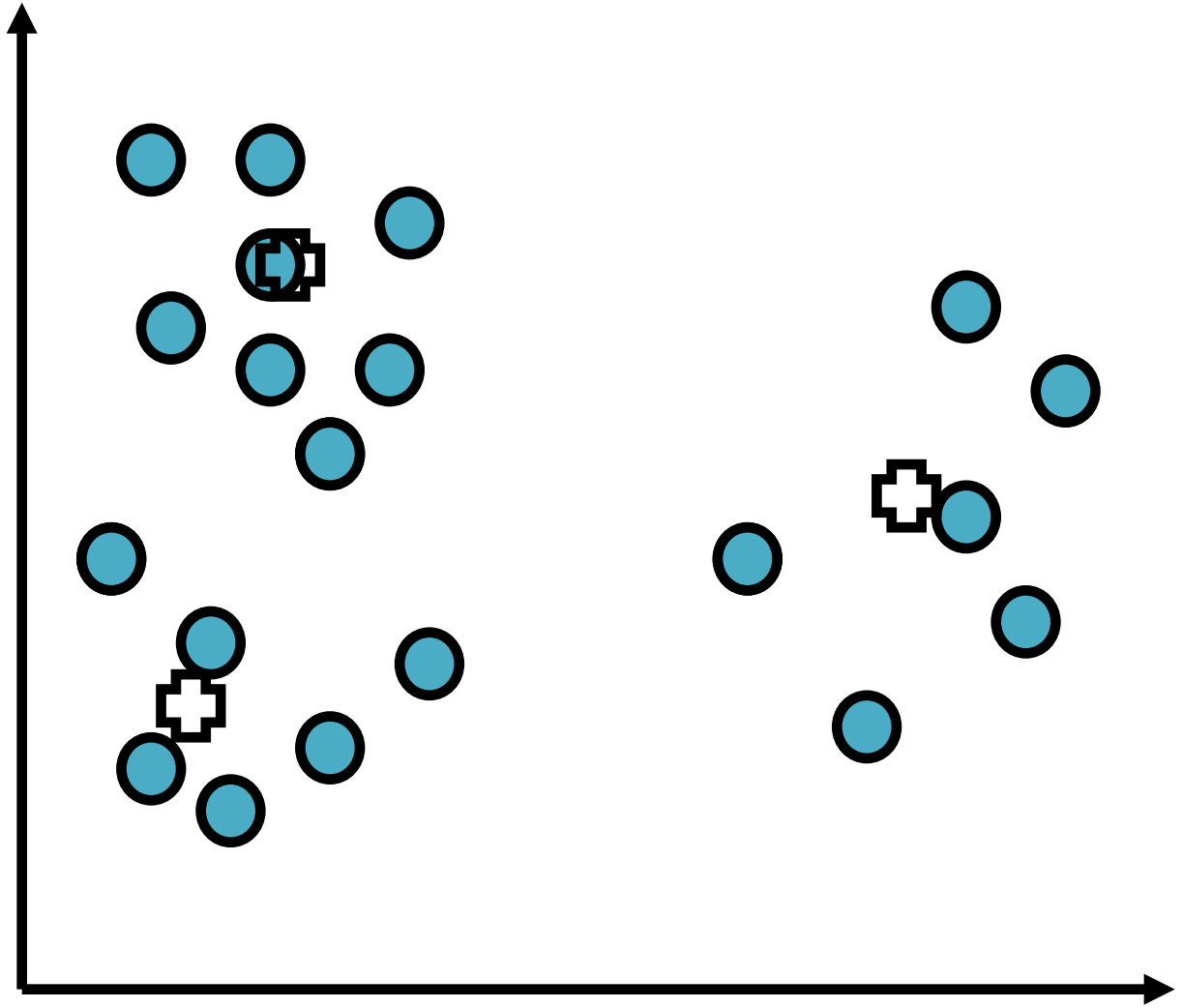
Clustering

Assign Samples to Clusters



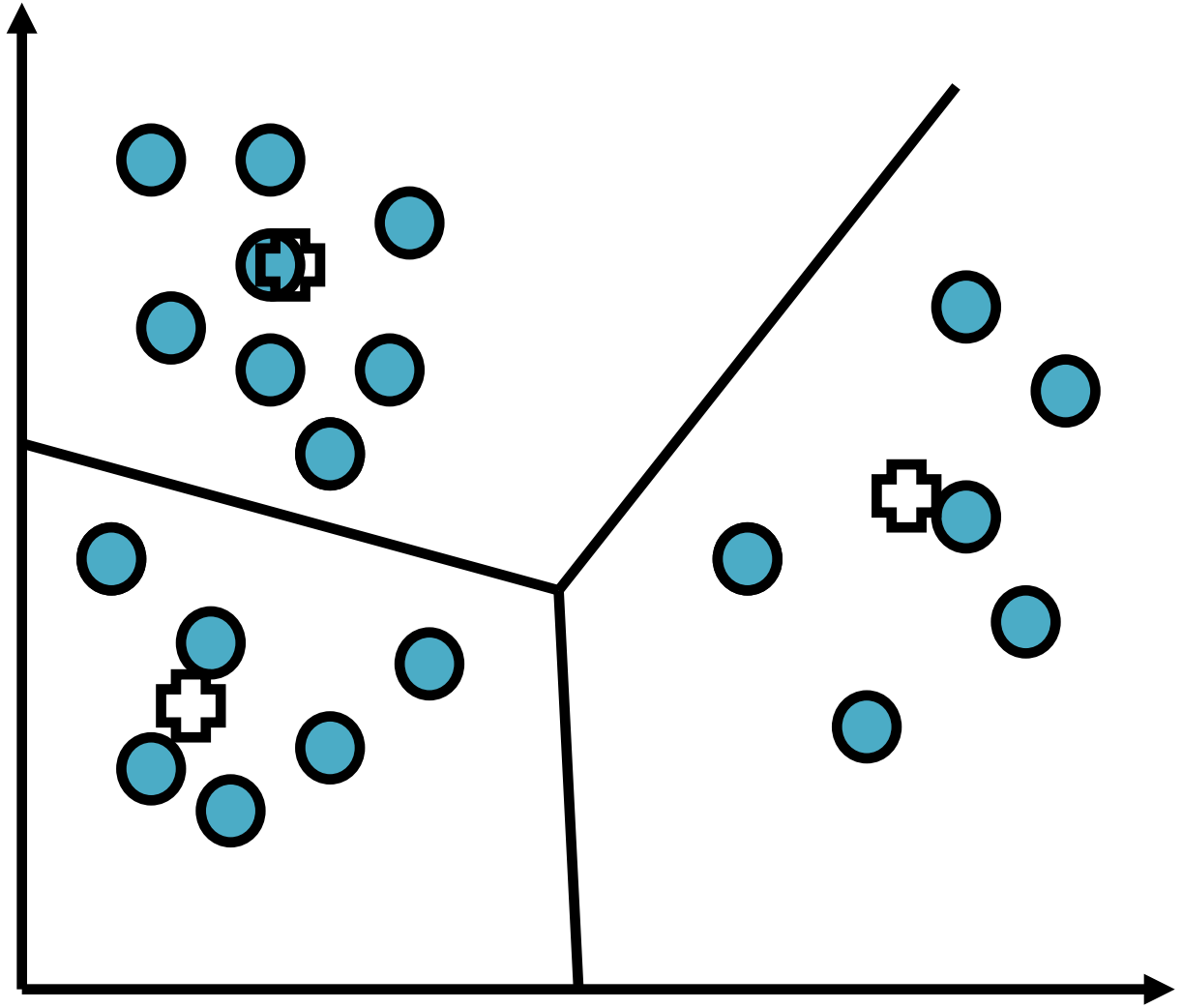
Clustering

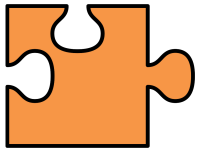
Find new centroids



Clustering

New clusters;
This process continues to iterate until the clusters “converge”, that is, no more changes in the clusters.





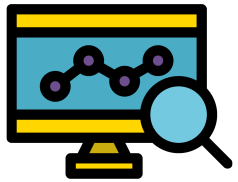
Define the ML Problem Quiz

C&C Protocol Detection:

- Task- recognize and attribute C&C communication on networks
- Training - supervised learning
- Performance measures - percentage of network communication correctly classified

What type data will we need for C&C Protocol Detection?

We will need known C&C communication



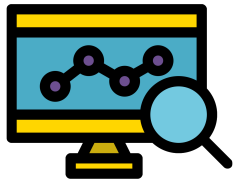
Classifiers

No free lunch: machine learning algorithms are tools, not dogmas

Try simple classifiers first

Better to have smart features and simple classifiers than simple features and smart classifiers

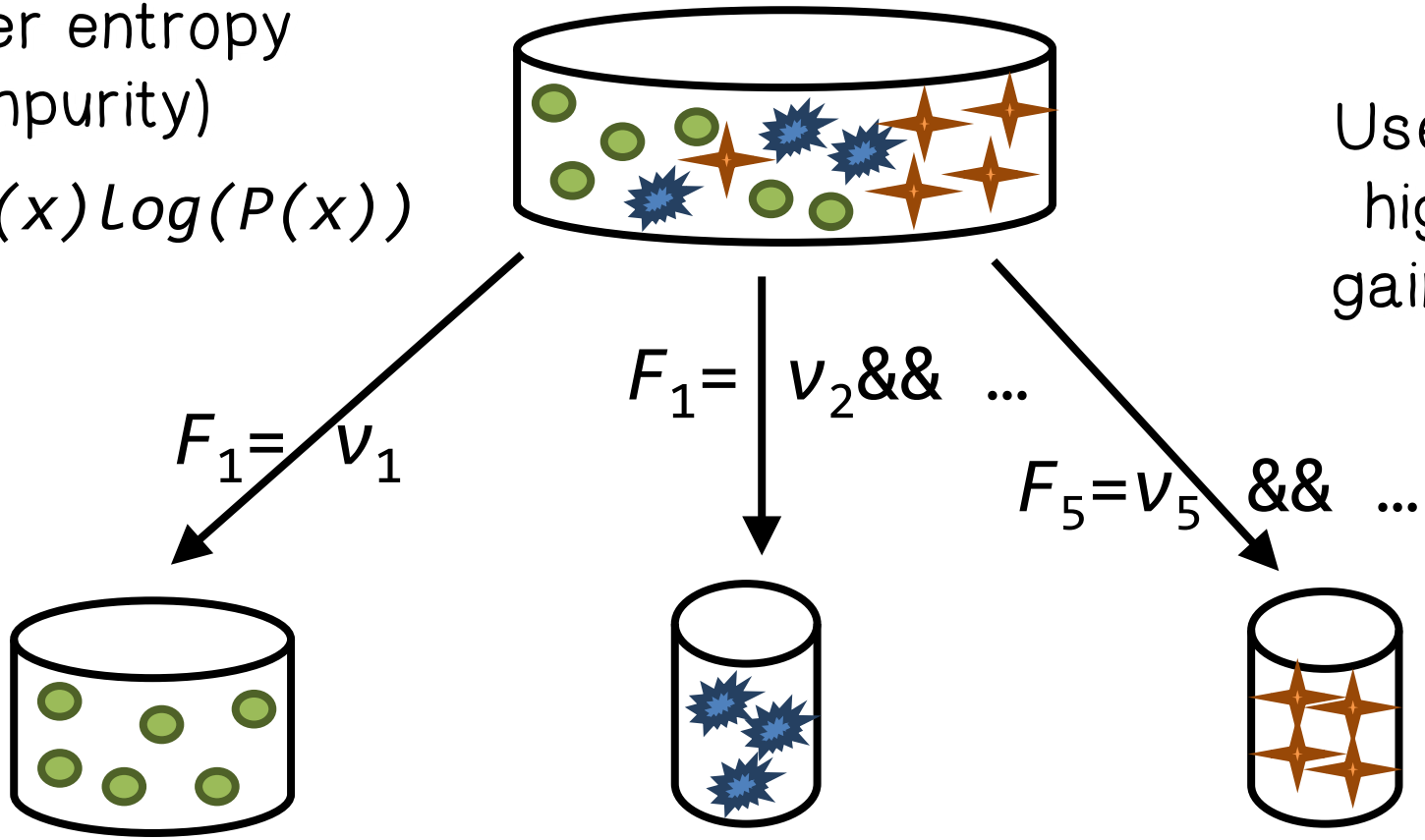
Use increasingly powerful classifiers with more training data (bias-variance tradeoff)



Classifiers
Intrusion Detection

Higher entropy (impurity)

$E(X) = -\Sigma_x P(x) log(P(x))$



Use features with high information gain – reduction in entropy

Lower entropy (purer)

Audit Data Preprocessing

tcpdump packet data

```
10:35:41.5 A > B: . 512:1024(512) ack 1 win 9216
10:35:42.2 C > D: . ack 1073 win 16384
10:35:45.6 E > F: . ack 2650 win 16225
...
```

connection records

time	dur	src	dst	bytes	srv	flag	…
10:35:39.1	5.2	A	B	42	http	SF	…
10:35:40.4	20.5	C	D	22	user	REJ	…
10:35:41.2	10.2	E	F	1036	ftp	SF	…
…	…	…	…	…	...	…	…

Feature Construction Problem

dst …	*service …*	*flag*	
h1	http	S0	syn flood
h1	http	S0	
h1	http	S0	
h2	http	S0	normal
h4	http	S0	
h2	ftp	S0	

existing features not very useful

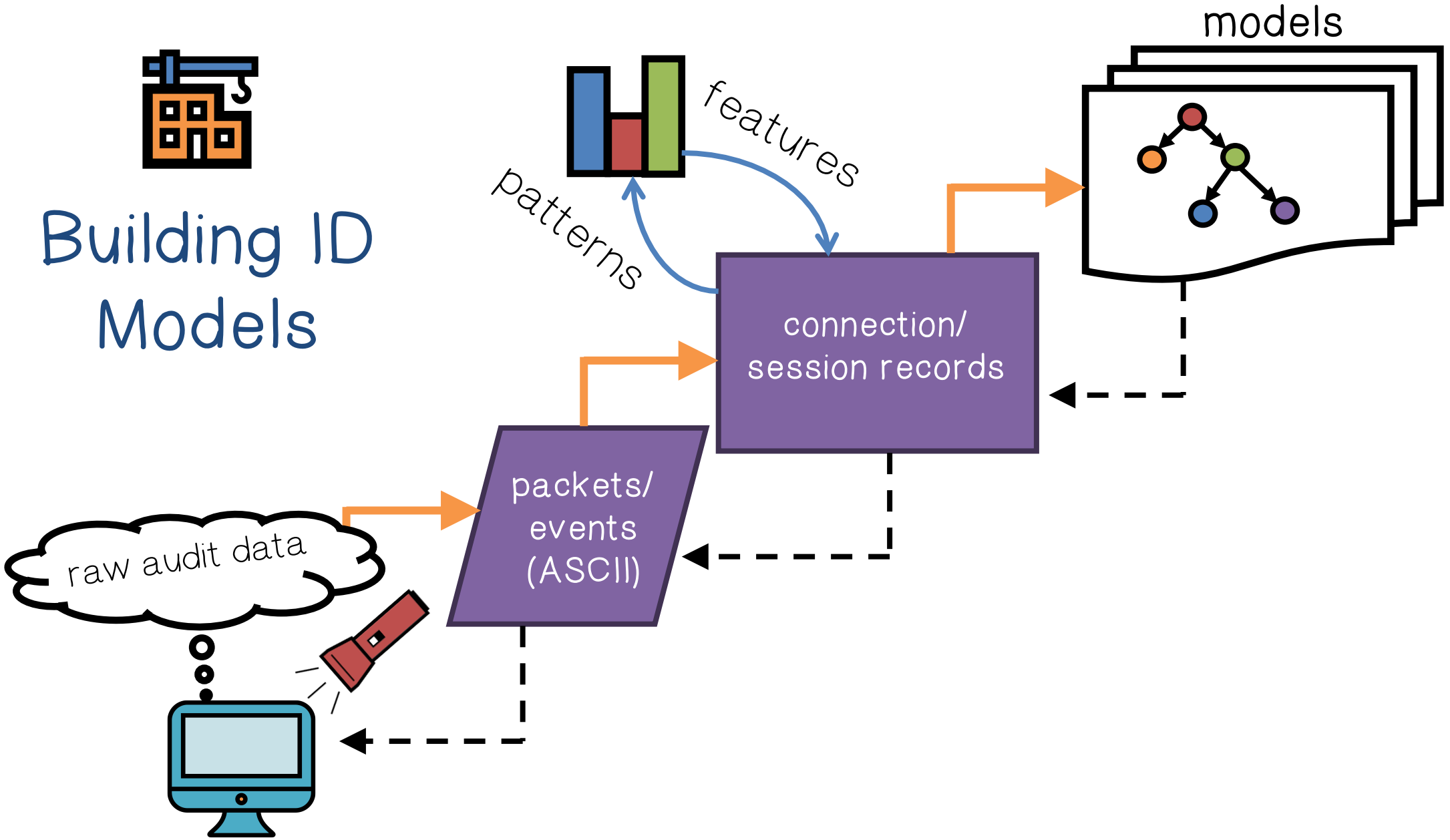
dst …	*service …*	*flag*	*%S0*	
h1	http	S0	70	syn flood
h1	http	S0	72	
h1	http	S0	75	
h2	http	S0	0	normal
h4	http	S0	0	
h2	ftp	S0	0	

construct features with high information gain

Feature Construction Problem

Use temporal and statistical patterns, e.g., “a lot of S0 connections to same service/host within a short time window”

Building ID Models



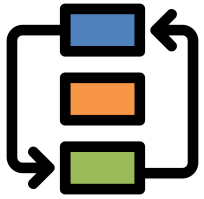
Mining Patterns

Associations of features

- e.g. **(service=http, flag=S0)**
- Basic algorithm: association rules

Sequential patterns in activity records

- Se.g. **(service=http, flag=S0), (service=http, flag=S0) → (service=http, flag=S0) [0.8,2s]**
- Basic algorithm: frequent episodes



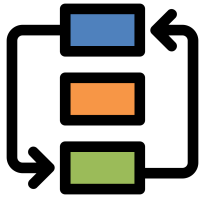
Basic Algorithms

dst …	*service …*	*flag*	*sbytes*	*dbytes*
h1	http	S0	100	100
h1	http	S0	200	100
h1	http	S0	200	100
h2	http	S0	200	100
h4	http	S0	100	100
h2	ftp	S0	200	100

Too MANY USELESS patterns!

Do not consider schema/structure of data

Compute frequent long patterns by combining frequent short patterns



Basic Algorithms
Extensions

- Designating “essential” features to compute “relevant” patterns
  - <time, source, destination, source port, service>, flag
  - Relevant patterns must describe the essential features
    - *axis* attribute(s)
    - *reference* attribute(s)

Axis Attributes

axis attribute(s) as a "constraint":

- the most important attribute, e.g., *service*
- patterns must contain *axis* attribute values

axis attributes (e.g., service)	non-essential attributes (auxiliary information, e.g., sbytes, dbytes)

Axis Attributes

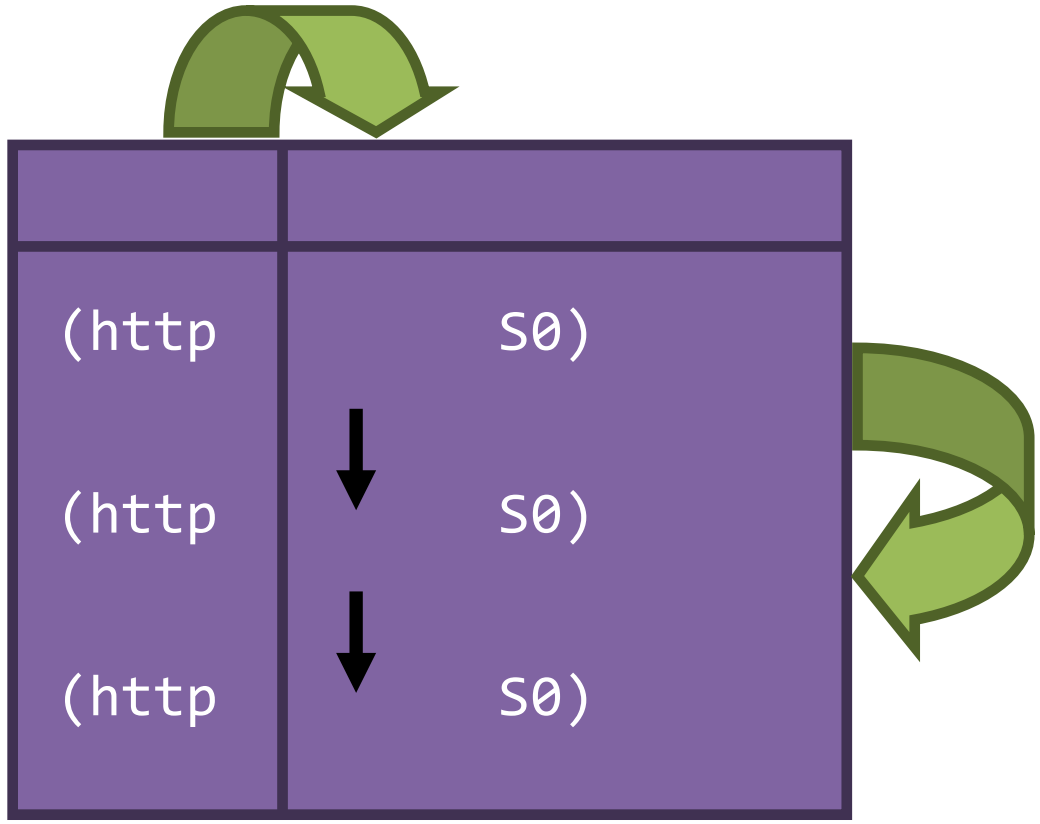
Compute sequential patterns in two phases:

Associations using the axis attribute(s)

Sequential patterns from the associations

Axis Attributes

Example (*service* is the axis attribute): (service=http, flag=S0), (service=http, flag=S0) (service=http, flag=S0)



Reference Attribute(s)

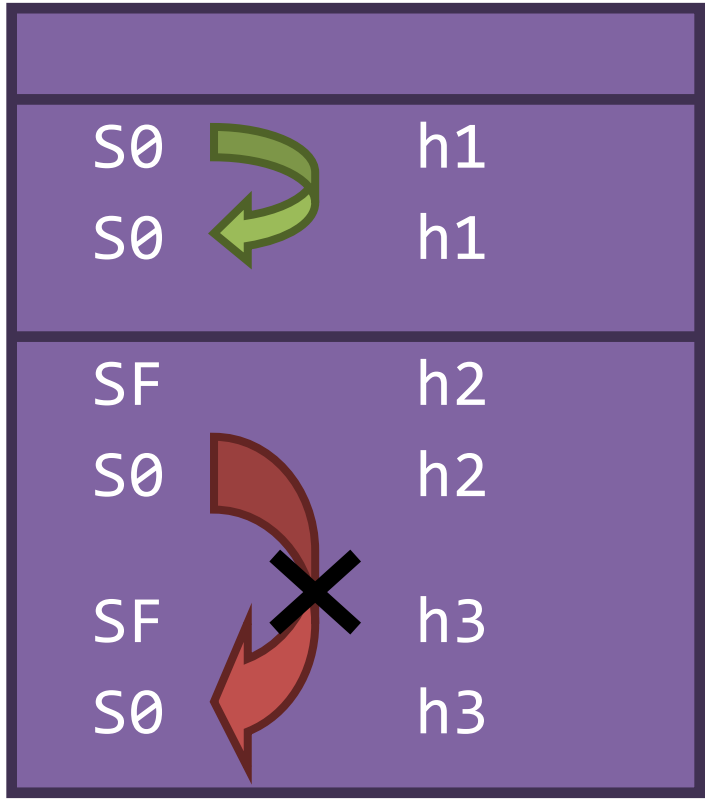
The “reference subject” of a sequence of related “actions”, e.g., connections to the same destination host:

(service=http, flag=S0), (service=http, flag=S0) → (service=http, flag=S0)

Reference Attribute(s)

Reference attribute(s) as an constraint:

- frequent patterns must refer to the same *reference* attribute value



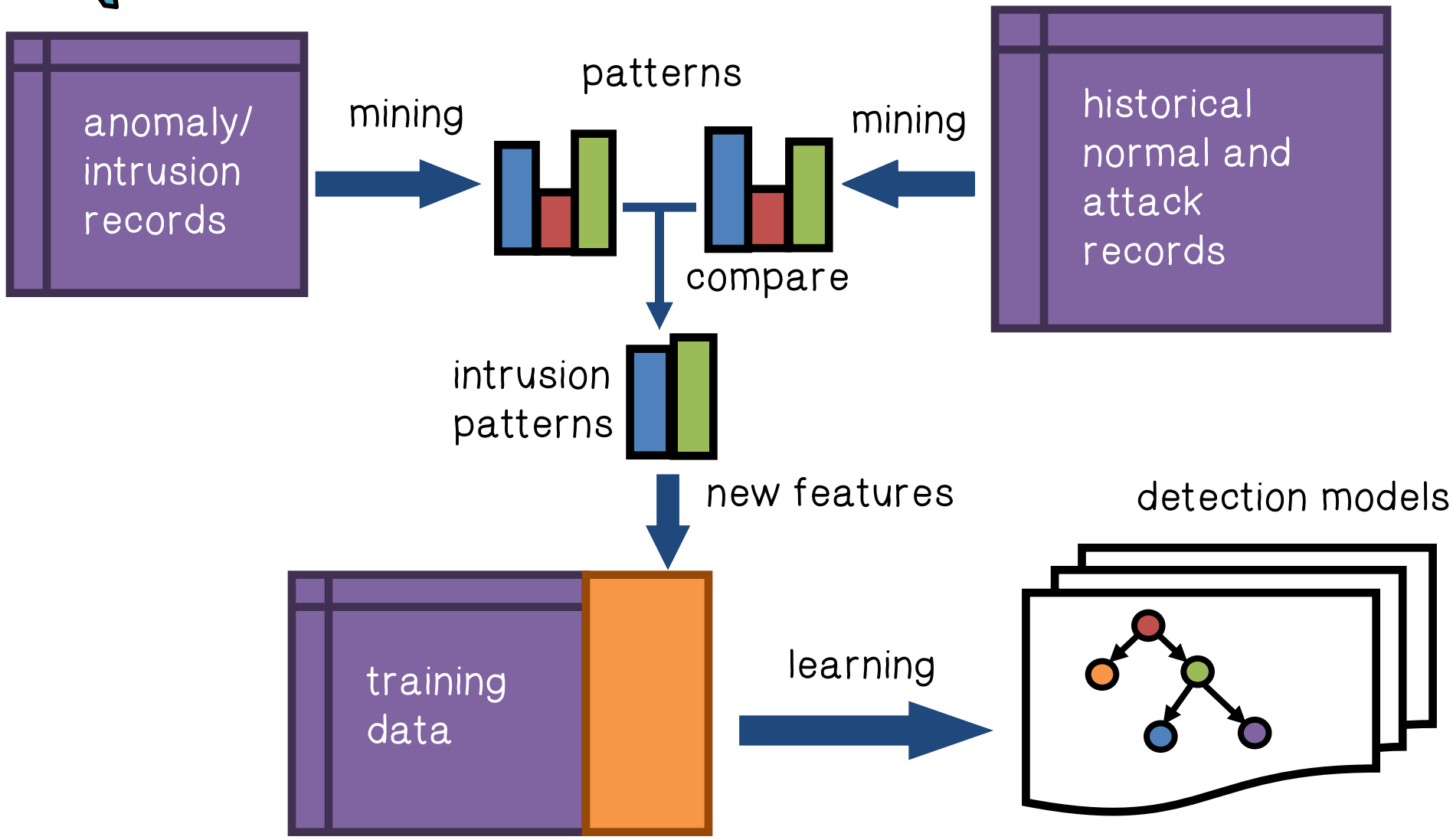
Feature Construction from Patterns

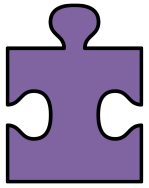
Compare and identify “intrusion-only” patterns

Parse each intrusion pattern:

- Identify the *anatomy* (reference and axis) and *invariant* information of an attack;
- Apply *count*, *percent*, and *average* operations to add temporal and statistical features.

Feature Construction from Patterns





Dataset Selection Quiz

List some considerations when selecting a dataset for training:

- There is no perfect way of labelling data, therefore there is really no perfect IDS dataset.
- Selecting a correct baseline dataset for your network.
- Selecting a dataset that has a range of intrusion attacks

Feature Construction Example

Connection Records (syn flood):

```
...
11:55:15 19468 http 1.2.3.4 172.16.112.50 ? ? S0 ...
11:55:15 19724 http 1.2.3.4 172.16.112.50 ? ? S0 ...
...
11:55:15 18956 http 1.2.3.4 172.16.112.50 ? ? S0 ...
11:55:15 20492 http 1.2.3.4 172.16.112.50 ? ? S0 ...
11:55:15 20748 http 1.2.3.4 172.16.112.50 ? ? S0 ...
11:55:15 21004 http 1.2.3.4 172.16.112.50 ? ? S0 ...\
...
11:55:15 21516 http 1.2.3.4 172.16.112.50 ? ? S0 ...
11:55:15 21772 http 1.2.3.4 172.16.112.50 ? ? S0 ...
...
```

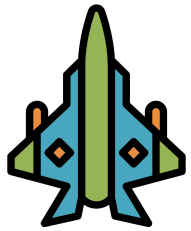
Feature Construction Example

"syn flood" patterns (**dst_host** is *reference* attribute):

(flag = S0, service = http), (flag = S0, service = http) → (flag = S0, service = http) [0.6, 2s]

Add features:

- count the connections to the same ***dst_host*** in the past 2 seconds, and among these connections,
- the percentage with the same *service*,
- the percentage with S0



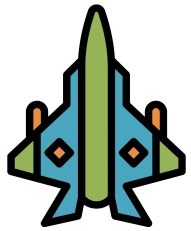
1998 DARPA Evaluation

The data (prepared by MIT Lincoln Lab):

Total 38 attack types, in four categories:

- DOS (denial-of-service), e.g., SYN flood
- Probing (gathering information), e.g., port scan
- r2l (remote intruder illegally gaining access to local systems), e.g., guess password
- u2r (user illegally gaining root privilege), e.g., buffer overflow

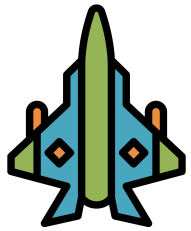
40% of attack types are in test data only, i.e., "new" to intrusion detection systems



DARPA Evaluation
Features

"intrinsic" features:

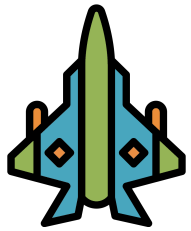
- protocol (service),
- protocol type (TCP, UDP, ICMP, etc.),
- duration of the connection,
- flag (connection established and terminated properly, SYN error, rejected, etc.),
- # of wrong fragments,
- # of urgent packets,
- whether the connection is from/to the same IP/port pair.



DARPA Evaluation
Content Features

"content" features (for TCP connections only):

- # of failed logins,
- successfully logged in or not,
- # of root shell prompts,
- "su root" attempted or not,
- # of access to security control files,
- # of compromised states (e.g., "Jumping to address", "path not found" ...),
- # of write access to files,
- # of outbound commands,
- # of hot (the sum of all the above "hot" indicators),
- is a "guest" login or not,
- is a root login or not.

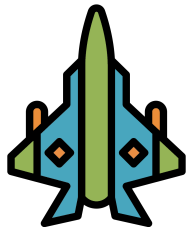


DARPA Evaluation

Features constructed from mined patterns:

Temporal and statistical "traffic" features that describe connections within a time window:

- # of connections to the same *destination host* as the current connection in the past 2 seconds, and among these connections,
- # of rejected connections,
- # of connections with "SYN" errors,
- # of different services,
- % of connections that have the same service,
- % of different (unique) services.

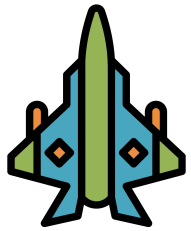


DARPA Evaluation

Features constructed from mined patterns:

Temporal and statistical "traffic" features that describe connections within a time window (cont'd.):

- # of connections that have the same *service* as the current connection, and among these connections,
- # of rejected connections,
- # of connection with "SYN" errors,
- # of different destination hosts,
- % of the connections that have the same destination host,
- % of different (unique) destination hosts.

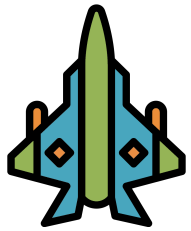


DARPA Evaluation

Features constructed from mined patterns:

- Example "content" connection records:

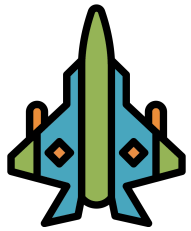
dur	p_type	proto	flag	l_in	root	su	compromised	hot	…	label
92	tcp	telnet	SF	1	0	0	0	0	…	normal
26	tcp	telnet	SF	1	1	1	0	2	…	normal
2	tcp	http	SF	1	0	0	0	0	…	normal
149	tcp	telnet	SF	1	1	0	1	3	…	buffer
2	tcp	http	SF	1	0	0	1	1	…	back



DARPA Evaluation

Example rules produced by machine learning:

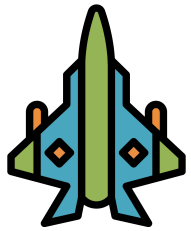
```
buffer_overflow :- hot >= 3, compromised >= 1,
su_attempted <= 0, root_shell >= 1.

back :- compromised >= 1, protocol = http.
```



DARPA Evaluation

Example “traffic” connection records:

dur	p_type	proto	flag	count	srv_count	r_error	diff_srv_rate	…	label
0	icmp	ecr_i	SF	1	1	0	1	…	normal
0	icmp	ecr_i	SF	350	350	0	0	…	smurf
0	tcp	other	REJ	231	1	198	1	…	satan
2	tcp	http	SF	1	0	0	1		normal

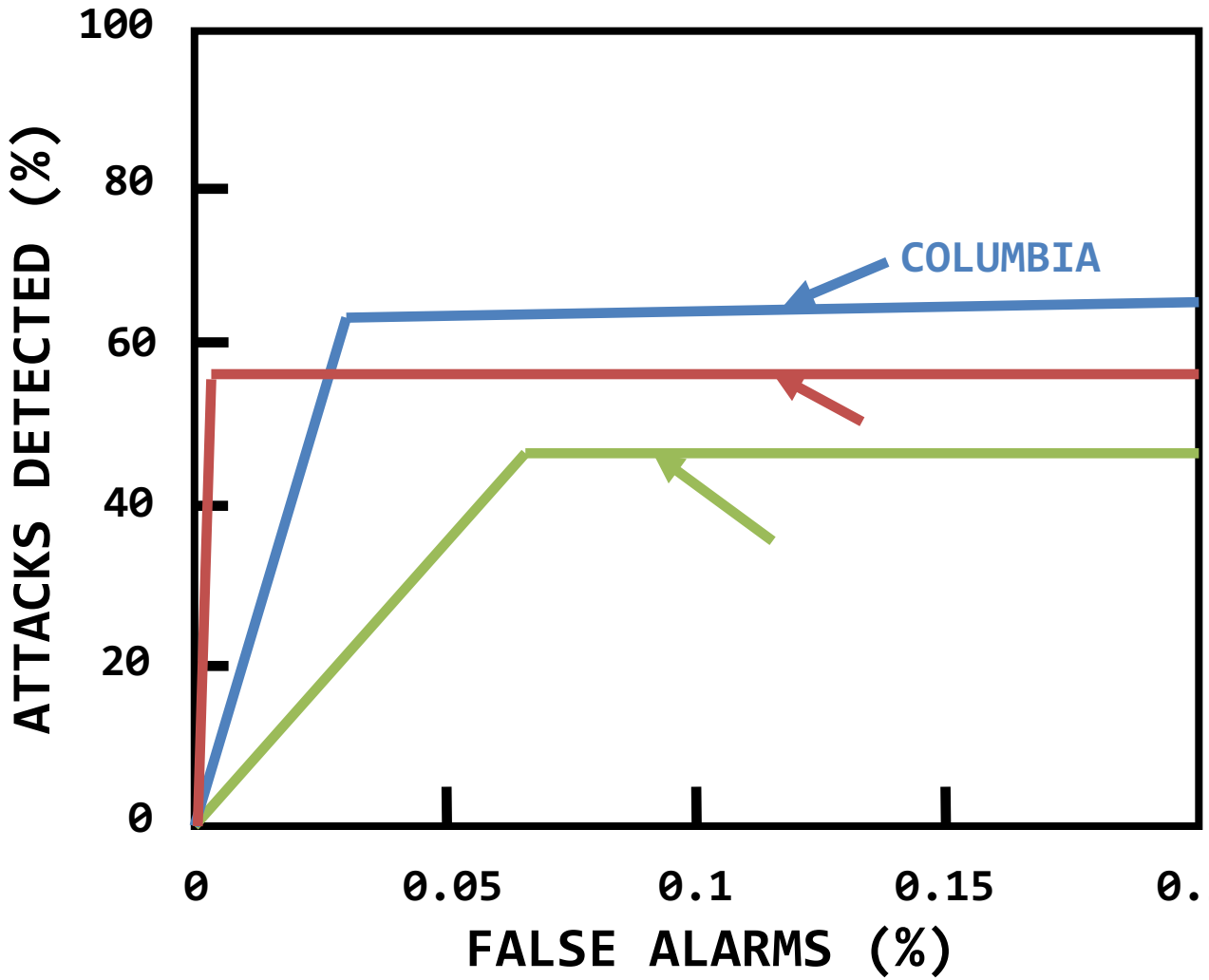


DARPA Evaluation

Example rules produced by machine learning:

- `smurf :- protocol = ecr_i, count >= 5, srv_count >= 5.`
- `satan :- r_error >= 3, diff_srv_rate >= 0.8`

TCP Dump Overall



Attacks: 120
Normal: 660,049