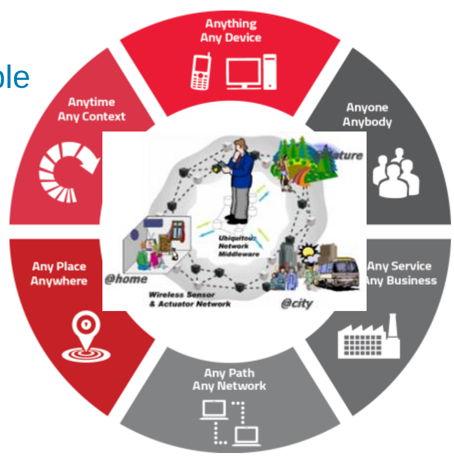
IoT Data Analytics

Vassilis Christophides
christop@csd.uoc.gr
http://www.csd.uoc.gr/~hy562
University of Crete, Fall 2019

The Internet of Things (IoT)

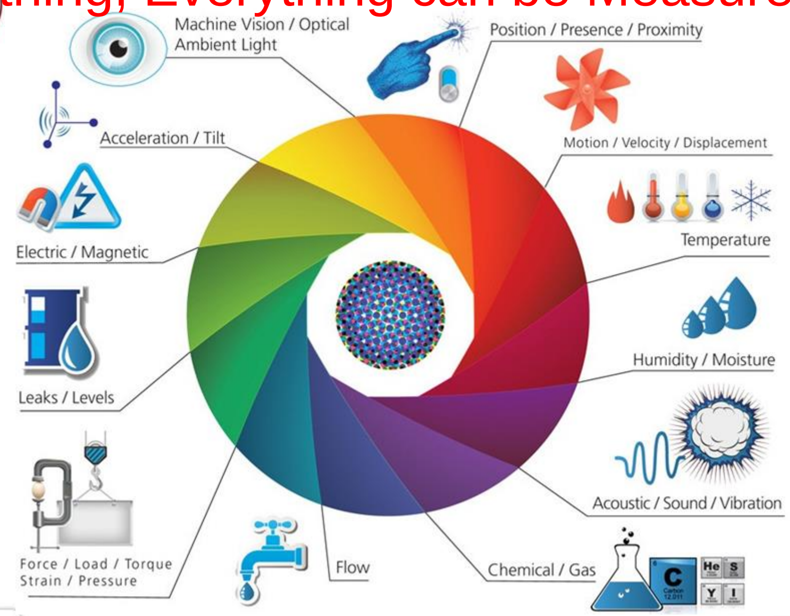
- Networks of physical objects (aka Things) that are uniquely identifiable in the Internet with embedded sensing and actuating along with programmability capabilities

- Information about Things can be collected and the state of Things can be changed from anywhere, anytime, by anything !



A World of Things

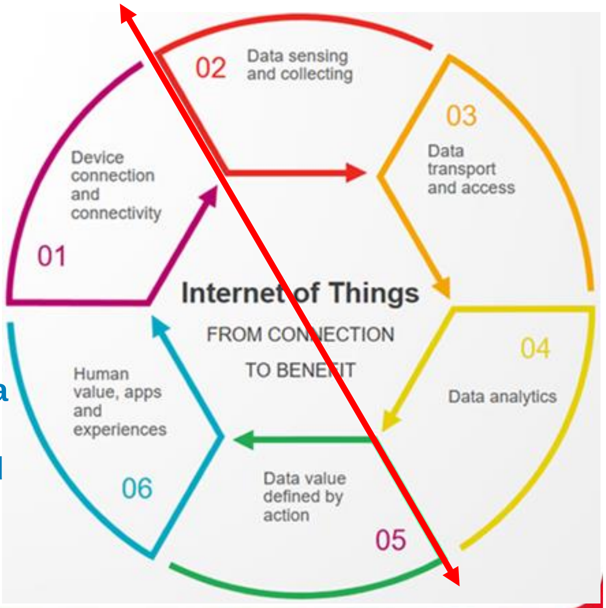
Anything, Everything can be Measured!



https://www.postscapes.com/trackers/video/the-internet-of-things-and-sensors-and-actuators/

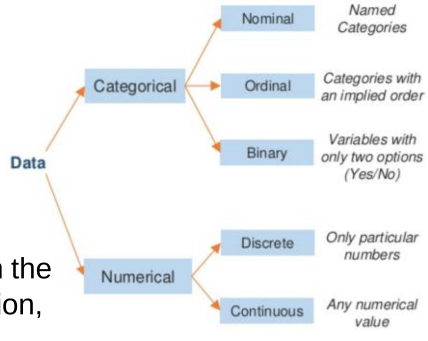
The IoT Value Chain

- IoT has the potential to transform *how* and *when* decisions are made throughout business and our daily lives iff *high quality* **data** can be ***processed*** *efficiently* and **analyzed** *effectively*!



Types of IoT Data

- Addresses/Unique Identifiers (nominal)
  - IP (IPv4, IPv6), Bluetooth, Zigbee, LoRa, RFID
- Positional Data (continuous)
  - GPS (Longitude, Latitude, Altitude), WiFi (Longitude, Latitude)
- Temporal Data (continuous)
  - Time and Date
- Sensor Data (numerical)
  - Output of a device that detects and responds to some type of input from the physical environment (motion, position, environment, mass, biomarker)
- Descriptive Data (categorical)
  - Objects, Processes, and Systems



IoT Data: Sensing the Physical World!

- IoT data is becoming soon “mega” big (i.e. billions of connected Things)
  - 44EB / months corresponds to ~ 100M hard disks
  - Processing 44 EB with 100M servers would still take > one hour
- IoT “data in motion” as opposed to traditional “data at rest”
  - Data speed is bound by the sensing frequency and connectivity
  - Data throughput (vs processing) has become the limiting factor!
- High variety data e.g., from *heterogeneous Things, embedded in the environment or wearable by persons*
- High veracity data (dropped or unlivable) as IoT data quality depends on *how Things are used and connected in the wild*
- High variability data as Things *frequently change behavior dynamically* and in ways that are not fully known in advance
- Data security, privacy, etc. even more important!

IoT Data Analytics

- Use Cases
  - Real-time monitoring and control
  - Failure detection and predictive maintenance   Anomaly Detection
  - Malicious cyber activity detection
  - Product life-cycle management
  - Operational efficiency, optimization, self-adaptation
- Batch and Online Techniques
  - Descriptive Analytics: aggregation and statistics
  - Predictive Analytics: Supervised, Semi-Supervised, Unsupervised
  - Data series analytics: Spatiotemporal
  - Domain and data type specific: Signal processing

Fall 2019

Real Time Anomaly Detection

Monitor Traffic Conditions

Network and Cyber Security Detect Intrusion

Monitor Biomarkers

Predictive maintenance, Production safety Monitoring

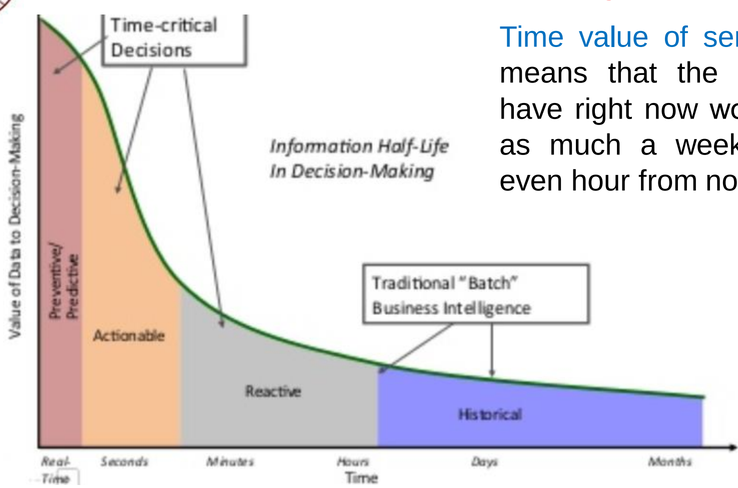
Agriculture Pest, Water Control

Monitor Energy Consumption

Define alerts, predict faults, detect intrusions & threats

Fall 2019

What is Real Time Analytics?



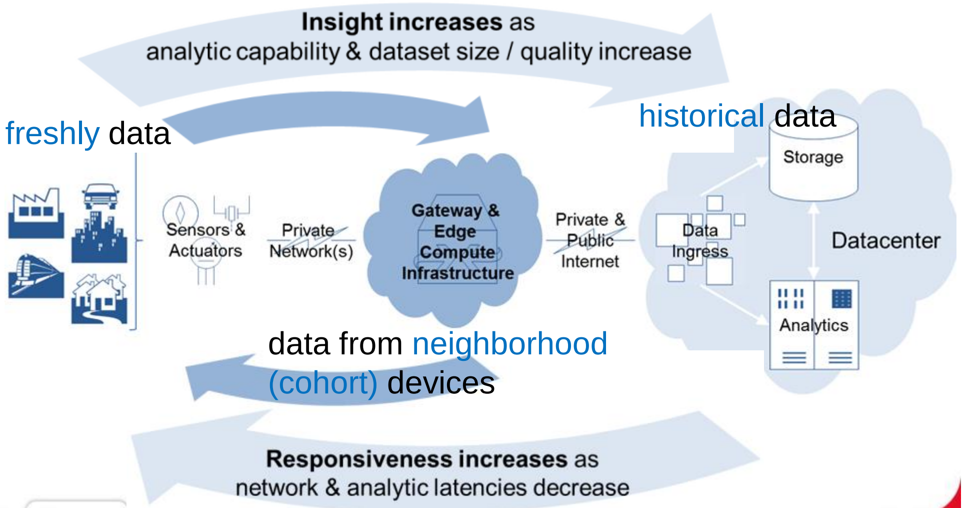
near real time | medium-term | long-term | historical

Time value of sensor data means that the data you have right now won't mean as much a week, day or even hour from now !

https://www.slideshare.net/AmazonWebServices/introduction-to-realtime-streaming-data-and-amazon-kinesis-streaming-data-ingestion-with-firehose

A 3-tier Architecture for IoT Analytics

- To support analytics with different time horizons (near-real time, medium, long terms), we need to a *3-tier architecture* for IoT

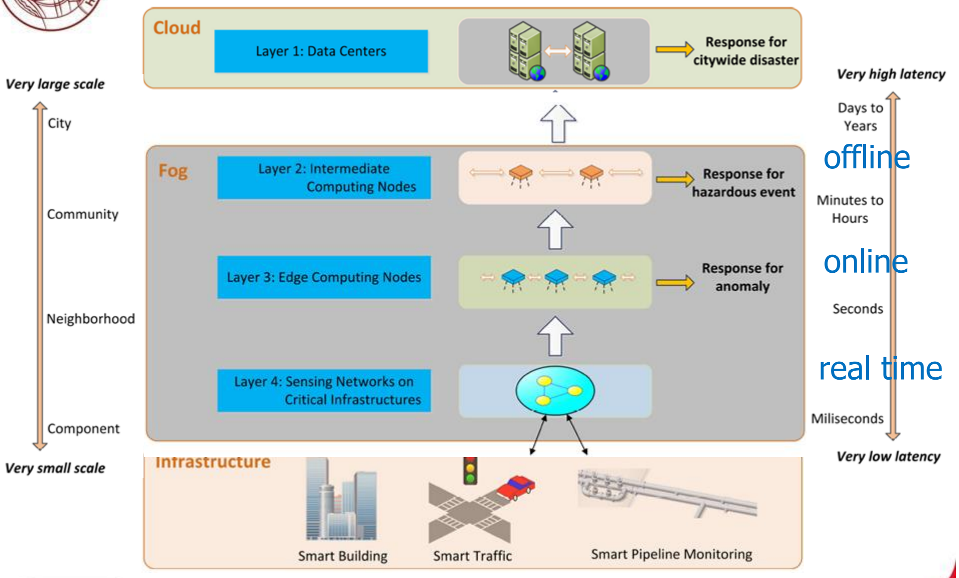


http://pjtec.info/how-the-internet-of-things-will-shape-the-datacenter-of-the-future-2/iot-20150804/

Edge Processing/Computing

- Objective: push data processing away from the core and towards the edge of the network
  - help ensuring that the right data processing task takes place at the right time and place
- Motivating factors:
  1. Reduce latency: run data computations directly on IoT devices or gateways, and only interact with the Cloud off the critical path (e.g. to continuously train ML models with freshly data)
  2. Be robust to connectivity issues: applications are not disrupted in case of limited or intermittent network connectivity
  3. Preserve privacy: ensures that sensitive data is pre-processed *on-site*, and only data that is *privacy compliant* is sent to the Cloud for further analysis, after having passed through a first layer of anonymizing aggregation
- Wide range of technologies: Local Cloud/Fog computing, Grid/Mesh Computing, mobile edge computing, cloudlets, etc.

A Fog/Edge Computing Architecture



B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, Q. Yang. A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities. ASE BD&SI 2015

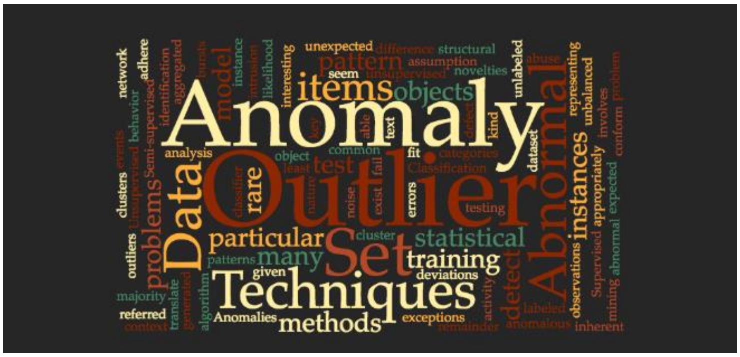
So to Recap

- The Internet of Things Is more About Data, than Things!
  - data in motion usually with spatiotemporal semantics
  - produced by several, distributed, heterogeneous devices
  - in the wild, hence data prone to errors and noise

- Low-latency, privacy-aware IoT data processing in post-cloud architectures
  - Edge/fog computing for effectively and efficiently ingesting and analyzing data *when* and *where* it is needed

- Real time IoT Data Analytics requires data streams support
  - Limited ground truth calls for unsupervised techniques

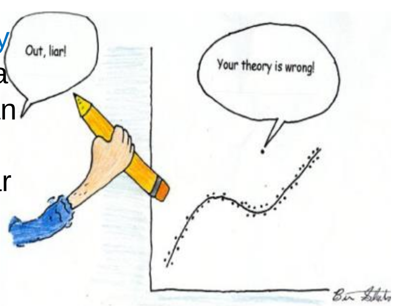
Data Anomaly Detection



Data Anomalies

- What are anomalies?
  - "An outlier is an observation in a dataset that appears to be inconsistent with the remainder of that dataset" [Johnson 1992]
  - "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism" [Hawkins 1980]

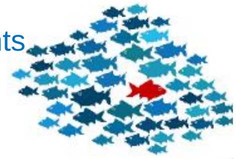
- Anomaly, or outlier, or deviation, or novelty detection, aims to measure whether a data point (or a subset) considerably differs than the remainder of the data points
  - On a scatter plot of the data, they lie far away from other data
  - Anomalies = Outliers + Novelties



http://www.kdd.org/kdd2016/topics/view/outlier-and-anomaly-detection

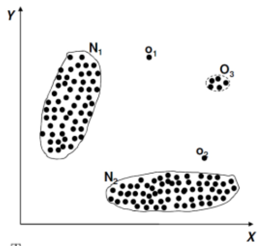
Cause of Data Anomalies

- Errors in the data collection or measurement process
  - Because of human error (intentional or not), a problem with a measuring device or the presence of noise
  - Insights extracted from “dirty data” are probably erroneous and thus the decisions to be made are likely unsound (e.g., incur a high rate of *false positives* and *false negatives*)
- Changes in the “data generation” process, e.g. some given statistical process (not an error, novelties in data)
  - Abnormal data deviate from this generating mechanism because it is of a different type
  - Novel class of observations provide valuable insights



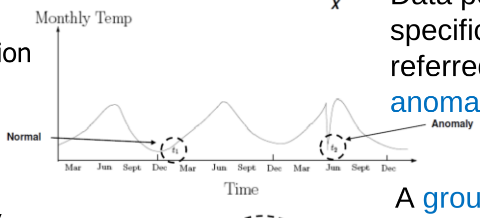
Types of Anomalies

Point Anomaly Detection



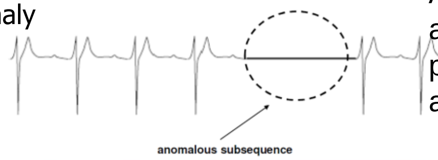
Data point anomalous w.r.t. to the rest of the data

Contextual Anomaly Detection



Data point anomalous in a specific context; also referred to as conditional anomalies

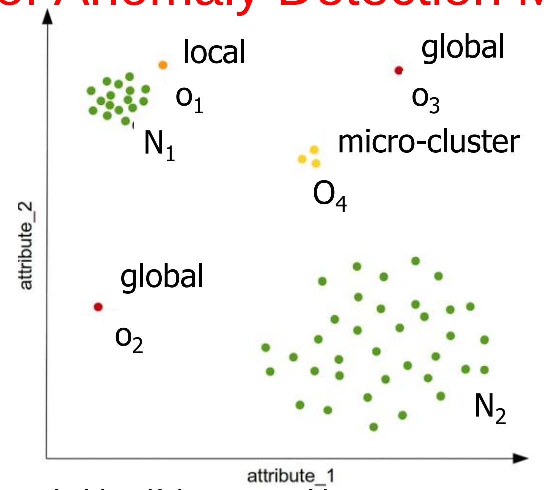
Group Anomaly Detection



A group of data points is anomalous; the individual points within a group are not anomalous by themselves

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58.

Scope of Anomaly Detection Methods



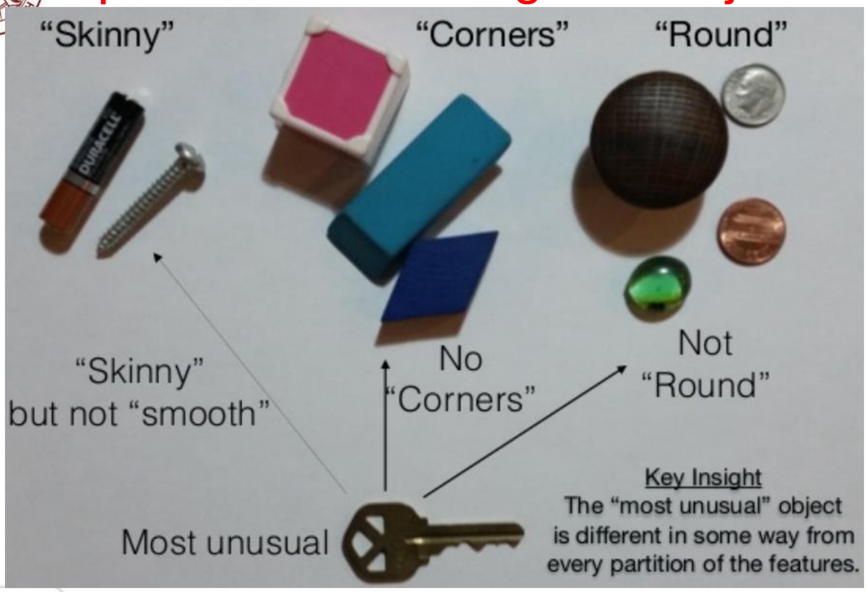
- Swamping: wrongly identifying normal instances as anomalies when normal instances are too close to anomalies
- Masking : an anomaly is close to another point, anomalous or not

M. Goldstein, S. UchidaA Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PLoS One. 2016;11(4)

Example: Most Unusual Objects?

https://www.slideshare.net/mlvlc/114-anomaly-detection

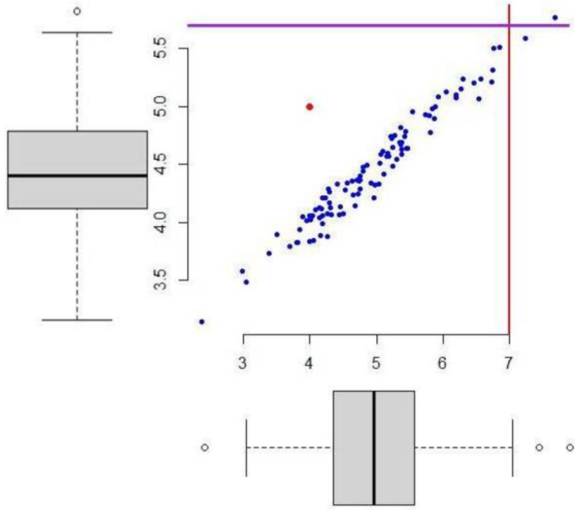
Example: How to Distinguish Objects?



High-Dimensional Data: Example

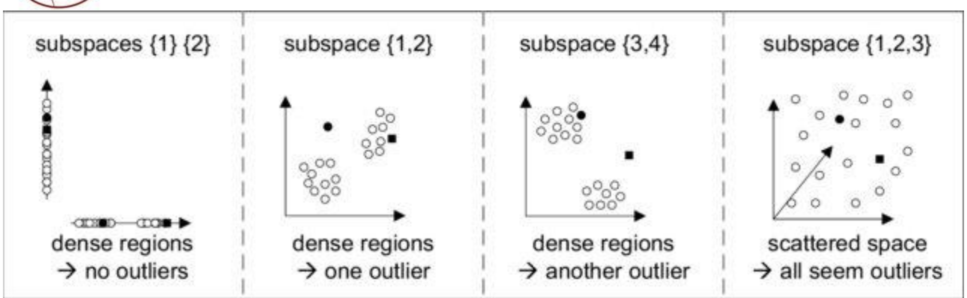
	Object	Length/Width	Num Surfaces	Smooth	
round	penny	1	3	true	
round	dime	1	3	true	
round	knob	1	4	true	
corners	box	1	6	true	
corners	eraser	2.75	6	true	
corners	block	1.6	6	true	
	screw	8	3	true	skinny
	battery	5	3	true	skinny
	key	4.25	3	false	skinny
	bead	1	2	true	

Multi-Dimensional Anomaly Detection



- Anomaly detection in multi-dimensional datasets reveals more accurate behavior of the data but at the same time poses various challenges

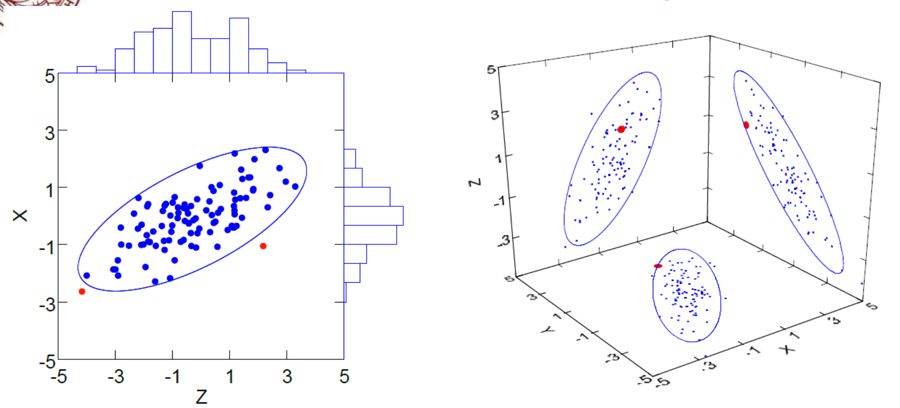
Multi-Dimensional Anomaly Detection



- Space concentration of distances: feature-wise distances of i.i.d. data samples approximately converge to a normal distribution
- The number of feature subspaces grows exponentially with the increasing dimensionality of data: data density functions not easily computed
- Data-snooping bias: given enough features, at least one feature subspace can be found for each data point such that it appears as an anomaly

D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality," AMS Math Challenges Lecture, pp. 1–32, 2000.

Multi-Dimensional Anomaly Detection



- Challenge: select a subspace that highlights *relevant features*, i.e. in which anomalies exhibit significantly different values from normal data
  - A robust anomaly detector should be able to detect anomalies with a *high proportion of irrelevant features creating noise in the input data, which masks the true anomalies*

L. Wilkinson Visualizing Big Data Outliers through Distributed Aggregation IEEE Trans. Vis. Comput. Graph. 24(1): Jan. 2018

Anomaly Detection Methods

Machine Learning

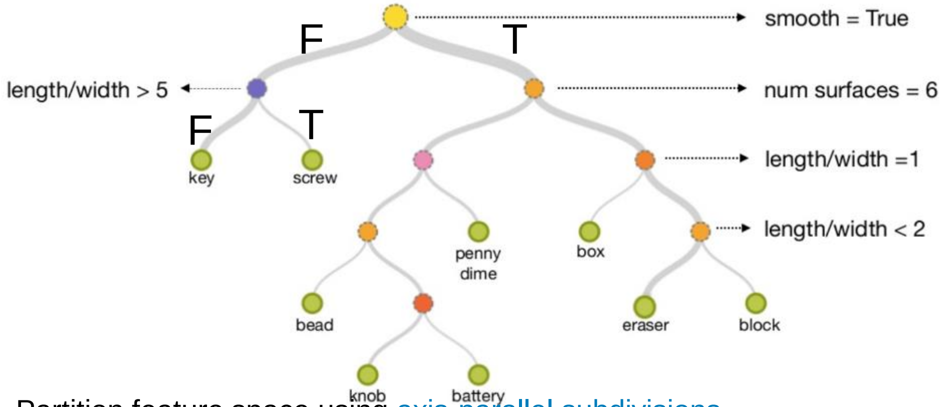
- Supervised
  - Classification
    - Tree-based
      - Hoeffding
- Unsupervised
  - Nearest Neighbor based
    - Distance-based
      - DB-Outlier
      - k^{th}-NN
      - Storm
    - Density based
      - LOF
      - LOCI
      - ABOD
  - Clustering based
    - CBLOF–uCBLOF
    - COD - MCOD
  - Ensemble based
    - Isolation Forest
    - Half-Space
    - RRCF
- Semi-Supervised
  - One class SVM
    - PSVM
    - SVDD
    - QSSVM
    - CESVM
  - Streaming Algorithms

Goldstein M, Uchida S (2016) A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PLOS ONE 11(4)

Fall 2019

Random Trees

- Build a decision tree from a random data subsample

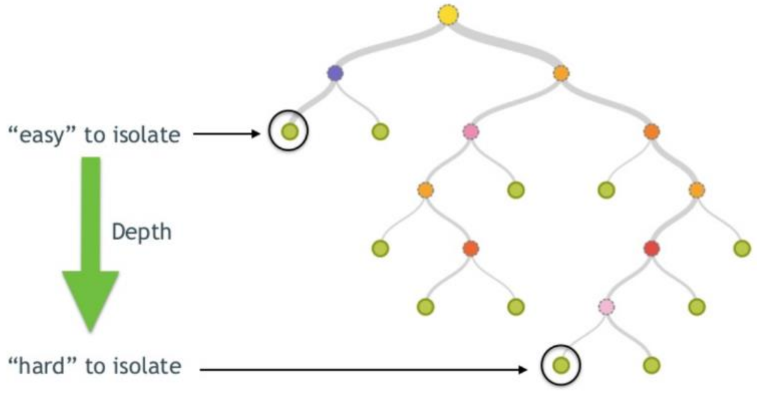


- Partition feature space using axis-parallel subdivisions
  - Select the split feature *A*, randomly and uniformly
  - Select the split value V_A, uniformly as the min(A)+(max(A)–min(A))*rand(1)
- Grow a random tree until each data point is in its own leaf or the tree reaches a maximum height

Fall 2019

Isolation Forest [Liu et al. 2008]

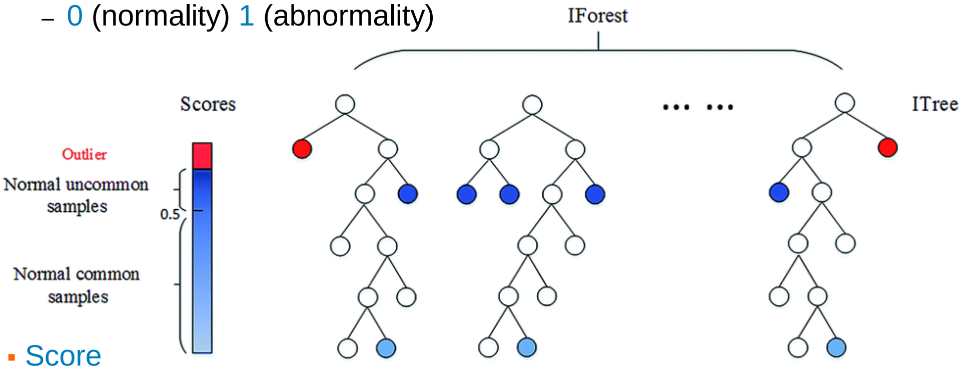
- To score a data point, find the height of the leaf node
  - The smaller the height the more anomalous is the data



- Build an ensemble of decision trees from randomly selected subsamples of size n

Isolation Forest Scores

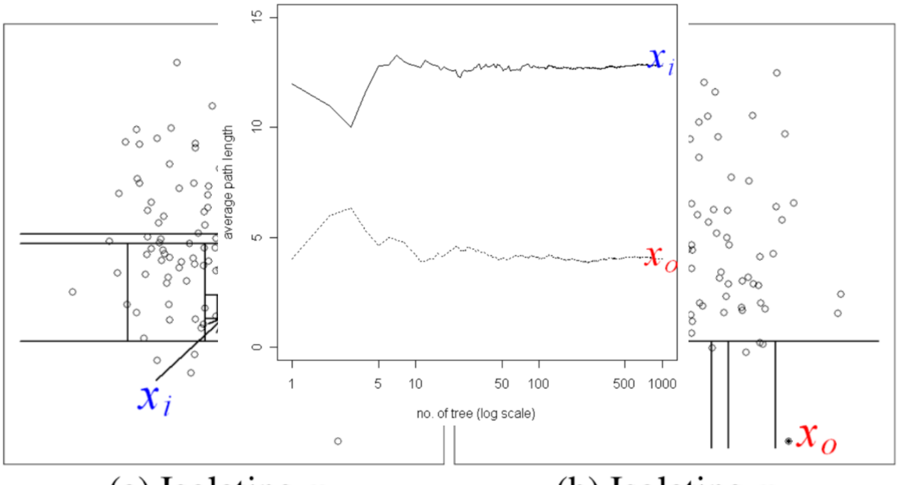
- Use average height to compute the anomaly score:
  - 0 (normality) 1 (abnormality)



- Score
  - Ensemble average path length to a data point
  - Normalized by the expected path length of balanced binary search tree

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Isolation Forest Scores



(a) Isolating x_i (b) Isolating x_o

12 partitions (not an anomaly) 4 partitions (anomaly)

iForest: Pros and Cons

- Pros
  - Very easy to construct (no distance/density function needed) avoiding hard decisions whether a data point is an anomaly or not
    - assigns an anomalous score to each data point
  - Achieve a sublinear time-complexity and a small memory-footprint
    - By exploiting subsampling
    - By eliminating major computational cost of distance calculation in all the distance-based and density-based AD methods
  - Can provide anomaly explanations [Siddiqui et al. 2015]
- Cons
  - Hyper-parameter tuning (e.g. number/height of trees, sample size)
    - Large datasets will need more isolation trees (how many?)
  - Requires a *high percentage of relevant features to identify anomalies* [Bandaragoda et al. 2018]
    - In presence of features that do not provide information over the anomaly, iForest increases height randomly by ignoring this fact

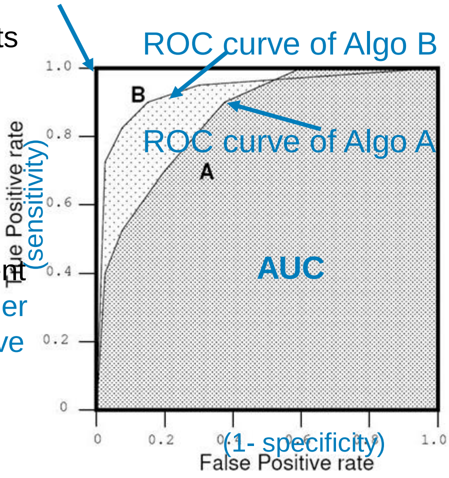
Performance Measures of AD Algorithms

Confusion Matrix	Actual Normal Data (n_n)	Actual Anomalous Data (n_a)
Predicted Non-anomalies	TN	FN
Predicted Anomalies	**FP**	**TP**

- *Accuracy Rate* (ACC) = (TP + TN) / (TN + FP + FN + TP)
- *False Alarm –Positive– Rate (FAR)* = FP/ (FP + TN)
- *True Positive –Detection– Rate (TPR)* = TP/ (TP + FN)
- *Receiver Operating Characteristic (ROC)= tradeoff between TPR&FAR*
- *Area Under the ROC Curve (AUC)*: can be computed by a slight modification of the algorithm for constructing ROC curves

ROC Spaces and AUC

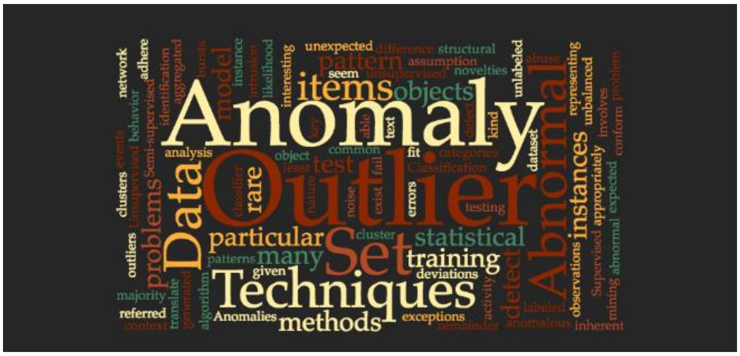
- ROC curves are two dimensional plots in which the true positive (TP) rate is platen on the Y axis and the false positive (FP) rate on the X axis
- Area Under the ROC Curve (AUC) has an important statistical property:
  - The AUC of a classifier is equivalent with the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance
  - When comparing classifiers, the bigger AUC the better!



Summary

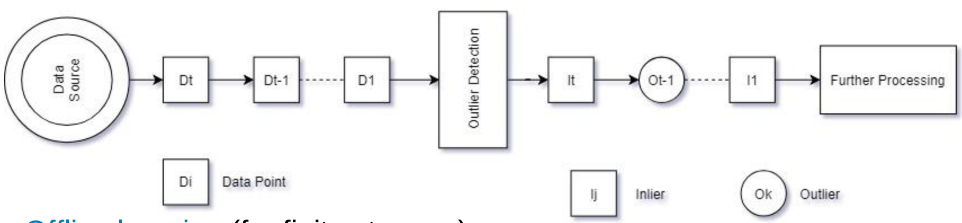
- AD algorithms usually report different outliers per dataset
  - they may miss obvious outliers especially in heterogeneous datasets that contain a mixture of numerical and categorical attributes
- Which anomaly detection method to use depends on
  - Data characteristics: *dimensionality* (Univariate, Multivariate) and *type* (categorical numerical)
  - Anomaly characteristics: *type* (Point, Collective), *semantics* (Set-based, Sequence-based), kind (Binary, Score)
  - Availability of labels: *supervised*, *unsupervised*, *semi-supervised*
  - Algorithmic properties: *computational cost* (exponential vs linear), *prior knowledge* (parametric, non-parametric), *distributional/ incremental computation potential*, …
- How we can automate the selection of the most suitable AD algorithm for a particular dataset with minimal human involvement and within limited computational budgets?

Anomaly Detection in Data Streams



Offline vs Online Anomaly Detection

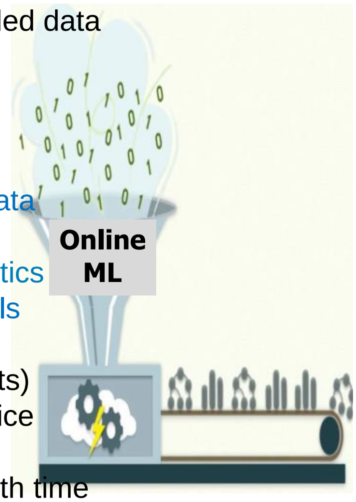
- A data stream is a possible infinite series of data points ..., o_{n-2}, o_{n-1}, o_n, ..., where data point o_n is received at time o_n.t.



- Offline learning (for finite streams):
  - All data is stored and can be accessed in multiple passes
  - Learned model is static
  - Potentially high memory and processing overhead
- Online learning (for infinite streams):
  - Single pass over the data which are discarded after being processed
  - Learned model is updated : one point at a time or in mini-batches
  - Low memory and processing overhead

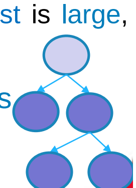
Online Anomaly Detection: Challenges

- Limited computational resources for unbounded data
  - window-based algorithms
- High speed data streams:
  - the rate of updating a detection model should be higher than the arrival rate of data
- Both normal and anomalous data characteristics may evolve in time and hence detector models need to quickly adjust to concept drifts/shifts:
  - adversarial situations (fraud, insider threats)
  - diverse set of potential causes (novel device failure modes)
  - user's notion of "anomaly" may change with time



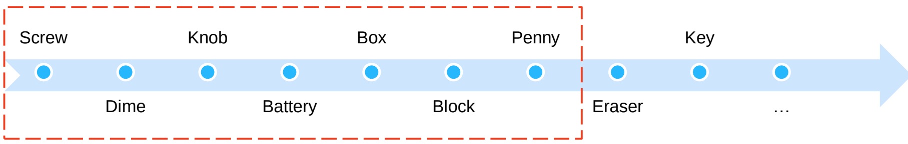
Robust Random Cut Forest (RRCF) [Guha et al, ICML 2016]

- Construct random trees as iForest but incrementally update their binary tree structure using sliding windows
  - Traverse a tree to Insert a new point to a leaf node, use FIFO to remove the oldest point if this procedure exceeds the max number of leaf nodes
  - Calculate the Displacement of the new point, as the number of leaf nodes that updated (moved/deleted) in a tree after the insertion
- Principled definitions of anomalies
  - Calculate the Collusive Displacement (CD) of a new point in a tree, as the maximal displacement across all nodes traversed in a path from the root
  - Outliers correspond to points that their average CD in the forest is large, instead of small average tree height
- Address iForest limitation in case of multiple irrelevant features
  - Does not uniformly selects the split feature



RRCF : Example

- Given the following input stream of points S



  - Start by collecting the first w = 7 points of S, as a Initial Training Sample x={x_1,x_2,…, x_j} where $x_i \in R^d$, in order to build the initial Tree
  - Assumption :
    - all objects within *x* are considered as inliers

RRCF Tree Construction: Example

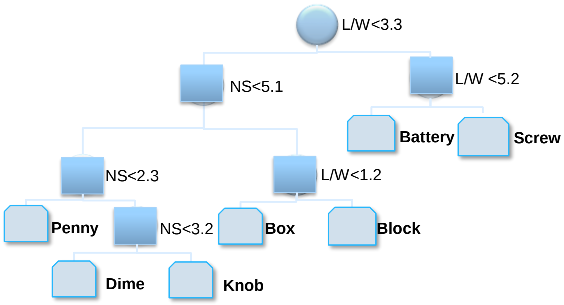
- Consider the value range per feature $l_i = \max_x x_i - \min_x x_i$
- Pick a spliting feature w.r.t. its normalized value range $\frac{l_i}{\sum_j l_j}$
- Choose uniformly a splitting value $X_i \sim \text{Uniform}[\min_x x_i , \max_x x_i]$

Object	L/W	NS	S
Screw	8	3	0
Dime	1	3	1
Knob	1	4	1
Battery	5	3	1
Box	1	6	1
Block	1.6	6	1
Penny	1	2	1

	L/W	NS	S
Minimum value	1	2	0
Maximum Value	8	6	1
Range	7	4	1

Length/Width (L/W), Num Surface (NS),Smooth (S)

RRCF Tree Construction: Example

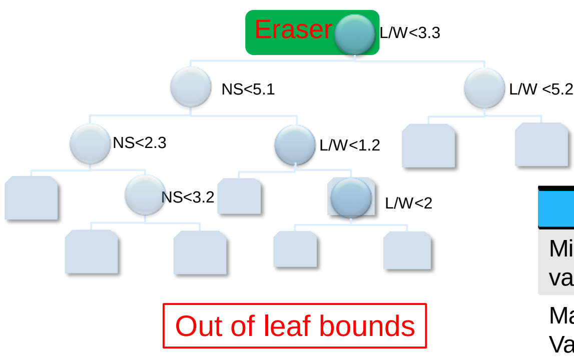


Object	L/W	NS	S
Dime	1	3	1
Knob	1	4	1
Penny	1	2	1
Range	0	2	0
Penny	1	2	1
Range	0.6	4	0

- Note that: Each (internal) child node has a sub-space (bound) feature value range of its parent node

RRCF Tree Update: Example

- Consider the new object *Eraser*, by sliding the window by one point

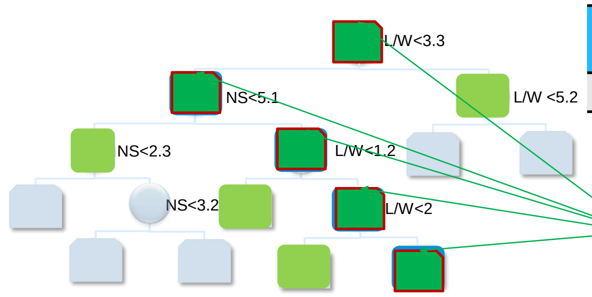


Object	L/W	NS	S
Eraser	2.75	6	1

	L/W	NS	S
Minimum value	1	6	1
Maximum Value	1.6	6	1
Value			

- A new tree instance has been computed
  - In theory update tree to the new instance <u>only if</u> low CoDisp(*Eraser*)

RRCF Tree Update: Example



Object	L/W	NS	S
Eraser	2.75	6	1

Leaves at sibling : 3

Update range

Leaves at current : 2

$$\mathrm{CoDisp(Eraser)} = \max(\mathrm{Displacement}) = 1 < \mathrm{Threshold} = 1.5$$

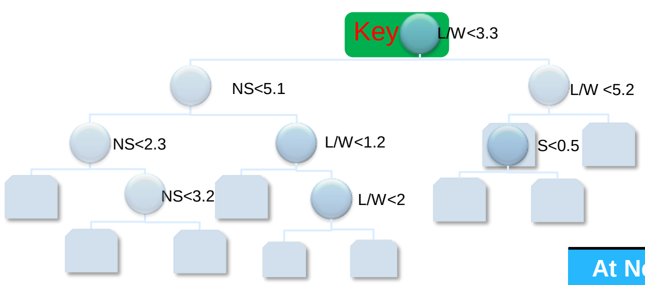
$$\mathrm{Disp} = \frac{\text{leaves of sibling node}}{\text{leaves of current node}}$$

$\boxed{\bullet\mathrm{Disp}(\mathrm{Eraser}_4) = \frac{1}{1}}$ $\bullet\mathrm{Disp}(\mathrm{Eraser}_2) = \frac{3}{3}$

$\bullet\mathrm{Disp}(\mathrm{Eraser}_3) = \frac{1}{2}$ $\bullet\mathrm{Disp}(\mathrm{Eraser}_1) = \frac{2}{6}$

RRCF Anomalies: Example

- Consider the new object *Key*, by sliding the window by one point

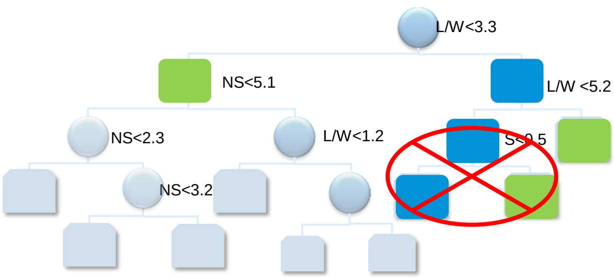


Object	L/W	NS	S
Key	4.25	3	0

Out of leaf bounds

At New Leaf	L/W	NS	S
Minimum value	4.25	3	0
Maximum Value	5.2	3	0
Range	0.95	0	0

RRCF Anomalies: Example



Object	L/W	NS	S
Key	4.25	3	0

$$\text{Displacement}(\text{Data Point}_{\text{depth}}) = \frac{\text{leaves of sibling node}}{\text{leaves of current node}}$$

Key -> Outlier

- $\text{Disp}(\text{Key}_3) = \frac{1}{1}$
- $\text{Disp}(\text{Key}_2) = \frac{1}{2}$
- $\text{Disp}(\text{Key}_1) = \frac{6}{3}$

Should discard changes and Keep Original Tree

$\text{CoDisp}(\text{Eraser}) = \max(\text{Displacement}) = 2 >$
$\text{Threshold} = 1.5$

RRCF Complexity

RRCF Building Complexity

$$O(t\,(n))$$

- Worst case : perfect binary tree, where each data point is a leaf
- Where,
  - t is the number of trees
  - n is the maximum number of data points in a tree
  - $(2n-1)$ is the total number of nodes of a perfect binary tree with n leaves
  - $log_2(n)$ is the minimal height of a perfect binary tree with n leaves
  - Tree height is increased by 1 after processing a new data point

RRCF Updating Complexity

$$O\left(t\,(\log_2(n))\right)$$

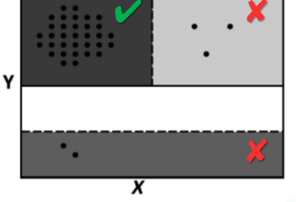
- Worst case : perfect binary tree, where each new data point is a new leaf at the maximal height updating the entire subtree

RRCF Summary

- RRCF chooses a splitting feature proportionally to its value range and not uniformly
  - Formally proves that tree structure updates are equivalent as if the data point were available during training
  - Tackles the presence of irrelevant dimensions for outlier isolation
- RRCF determines distance based anomalies without focusing on the specifics of the distance function
  - Define anomalies using high average of collusive displacement instead of low average tree height
  - Tackles finer differences (masking) between outliers and inliers
- RRCF hyper-parameters include the CoDisp threshold
  - As CoDisp is not normalized, we should first normalize the maximal tree displacements before computing their average in the forest
  - It is open how to tune threshold, in the normalized range [0, 1], to accurately separate normal from abnormal points
- RRCF is implemented in the AWS Data Analytics Engine

Half Space Trees (HST) [Tan et al, IJCAI 2011; Chen et al, MLJ 2015]

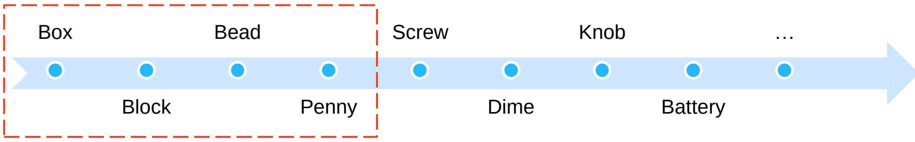
- Randomly construct small binary trees as iForest, but
  - For each feature define its workspace
  - A HST bisects data space using randomly selected features and the middle value of its workspace
  - A node in a HST captures the number of points within a particular subspace, called Mass profile
- Incrementally update the Mass profiles of the tree/s, for every new tumbling window, to learn the density of the stream
  - Transfer the non-zero mass of the last window to the model computed in the previous window
- Anomalies can be inferred using the current window points
  - Points are ranked in their ascending score order
  - The lower the score of a point, the more probable to be an anomaly



Example: A HST on a 2D Stream

HST Stream Snapshot: Example

- Given a stream of objects, lets say 8 objects



- Process the stream using tumbling windows of length *w* = 4
  - Start by collecting the first *w* objects, as a Reference window

Object	L/W	NS	S
Box	1	6	1
Block	1.6	6	1
Bead	1	2	1
Penny	1	3	1

HST Workspace: Example

- Compute the Workspace of each feature, using the reference window

A. Work range: Minimum, Maximum, Split and Range values of features

Object	L/W	NS	S
Box	1	6	1
Block	1.6	6	1
Bead	1	2	1
Penny	1	3	1

(A)

Workrange	L/W	NS	S
Minimum	1	2	1
Maximum	1.6	6	1
Split	1.55	5.38	1
Range	1.10	6.76	0

B. Work space: Max, Min values of the features' *work range*

(B)

Max = Split + Range

Min = Split – Range

Workspace	L/W	NS	S
Max	2.66	12.14	1
Min	0.45	-1.38	1

HST Construction: Example

- Construct a HST using the features' work space

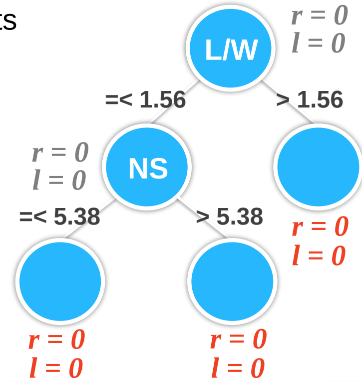
Workspace	L/W	NS	S
Max	2.66	12.14	1
Min	0.45	-1.38	1

A. Randomly select a feature F
B. Split data space using the mid value SP_F
C. Initialize the mass profile

- Control tree growth, using the objects of the reference window

Object	L/W	NS	S
Box	1	6	1
Block	$\boxed{1.6}$	6	1
Bead	1	2	1
Penny	1	3	1

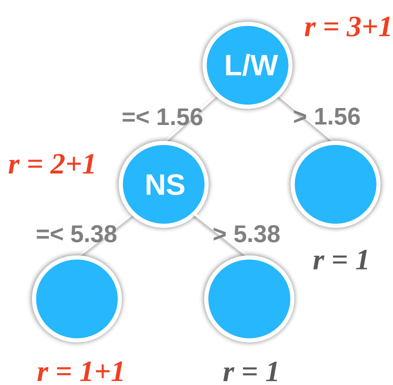
A. Activated: Max Height Limit = 2
B. Activated: Object Size Limit = 1



HST Mass Update: Example

- Update the Mass_r Profile of the HST, using the objects of the reference window
  - Increase by one the r mass of visited nodes

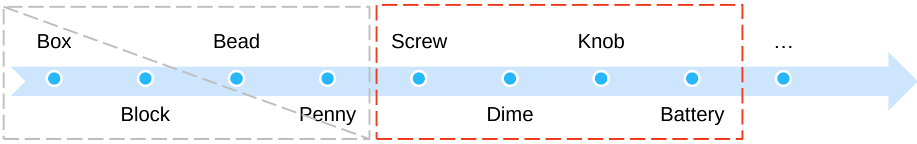
Object	L/W	NS	S
Box	1	6	1
Block	1.6	6	1
Bead	1	2	1
Penny	1	3	1



Fall 2019

HST Stream Snapshot: Example

- Process of the reference window has been completed



  – Continue by collecting the next *w* objects, as a Latest window

Object	L/W	NS	S
Screw	8	3	0
Dime	1	3	1
knob	1	4	1
Battery	5	3	1

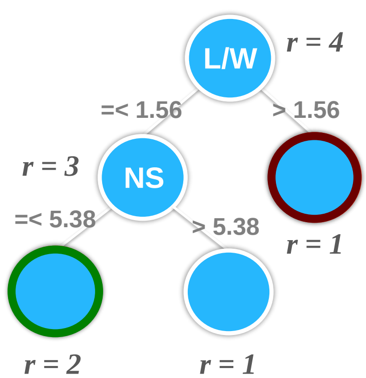
Fall 2019

HST Anomaly Detection: Example

Detect anomalies in the latest window, using the current HST Model

Object	L/W	NS	S
Screw	8	3	0
Dime	1	3	1
knob	1	4	1
Battery	5	3	1

Object	Score	Rank	
Screw	2	1st	**1**
Dime	8	2nd	**0**
knob	8	2nd	**0**
Battery	2	1st	**1**

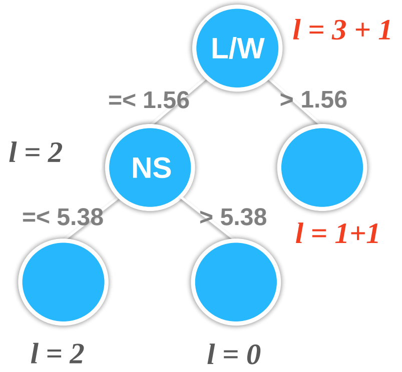


Predict$_{Obj}$ = Score$_{obj}$ < mean(score) : 1 ? 0

HST Mass Update: Example

- Update $Mass_l$ Profile of the HST, using the objects of the latest window
  - Increase by one the mass of all visited nodes in the path to a leaf

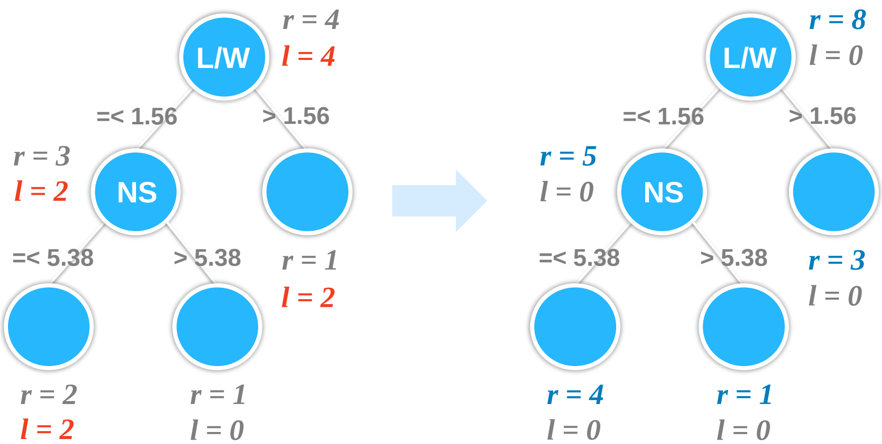
Object	L/W	NS	S
Screw	8	3	0
Dime	1	3	1
knob	1	4	1
Battery	5	3	1



Fall 2019

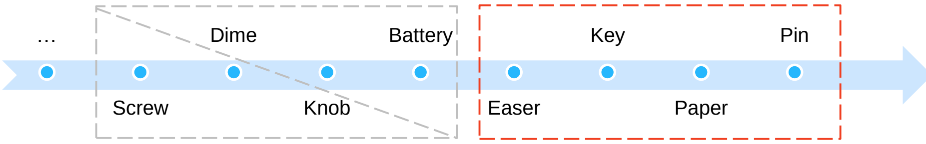
HST Mass Update: Example

- Aggregate the HST Mass Profile, before continue to the next window
  - Transfer the non-zero $Mass_l$ to the $Mass_r$ profile



HST Stream Snapshot: Example

- Process of the current latest window has been completed



  – Continue by collecting the final w objects as a Latest window

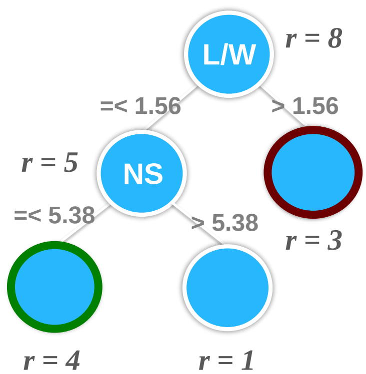
Object	L/W	NS	S
Easer	1.6	6	1
Key	4.25	3	0
Paper	10	2	1
Pin	1.14	3	0

HST Anomaly Detection: Example

- Detect anomalies in the latest window, using the current HST Model

Object	L/W	NS	S
Easer	1.6	6	1
Key	4.25	3	0
Paper	10	2	1
Pin	1.14	3	0

Object	Score	Rank	
Easer	6	1st	1
Key	6	1st	1
Paper	6	1st	1
Pin	16	2nd	0



$\text{Predict}_{Obj} = \text{Score}_{obj} < \text{mean(score)} : 1 \; ? \; 0$

HST Complexity

HST Building Complexity

$$O(t * 2^{h+1})$$

- Worst case : perfect binary tree, where each data point is a leaf
- Where,
  - *t* is the number of trees
  - *h* is the max height of a tree
  - *w* is the number of points in a window
  - *$(2^{h+1}-1)$* is the number of all nodes of a perfect binary tree of max height *h*

HST Updating Complexity

$$O(t * h * w)$$

- Worst case : perfect binary tree, where each new data point traverses requires to update the mass profiles of all sub trees

Complexities are constant (amortized) when the ***h***, ***t*** and ***w*** are set

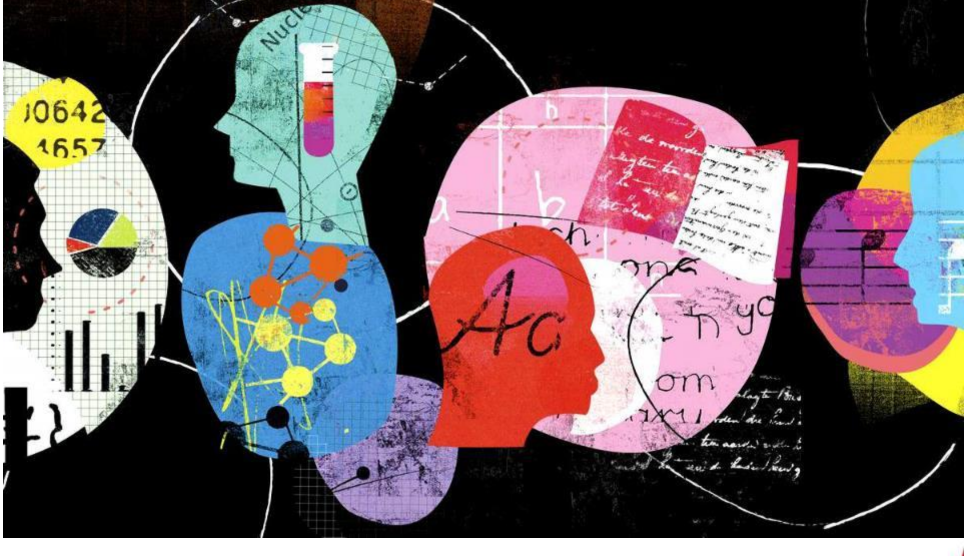
HST Summary

- HST workspace estimates the feature value ranges of future data in a stream, resulting to more accurate mass updates in stable distributions
- HST structure initialized using the first *reference* window, and remains fixed for all streaming data
  - Incrementally updating only the mass profiles of the HST model
- HST ranks anomalies using a scoring function that considers only the mass profiles of the leaf nodes
- HST fails to support concept drift when the statistical characteristics of features evolve over time
  - incorrectly selected features may lead to inaccurate mass updates
- One of the HST hyper-parameters is the size of tumbling windows
  - How we can tune window size to tackle concept drift/shift?

Qualitative Comparison

Algorithm	Novelty	Splitting Feature	Splitting Value	Window Type	Anomaly Score	Model Update	Anomaly Types
IF	Avoids computation of distances/ densities	Uniform	Uniform	✗	Tree height	batch	Global & Local
RRCF	Addresses Masking & Irrelevant Features	Proportional to the feature range	Uniform	Slide	Collusive Disp. (sensitive to neighborhood)	Incremental tree updates	Global & Local; *Sample* based range subspace
HST	High Speed Data Streams	Uniform	Mid Value	Tumble	Mass Profiles (sensitive to neighborhood)	Mini-batch update of mass profiles	Global & Local; fixed range subspace

Questions?



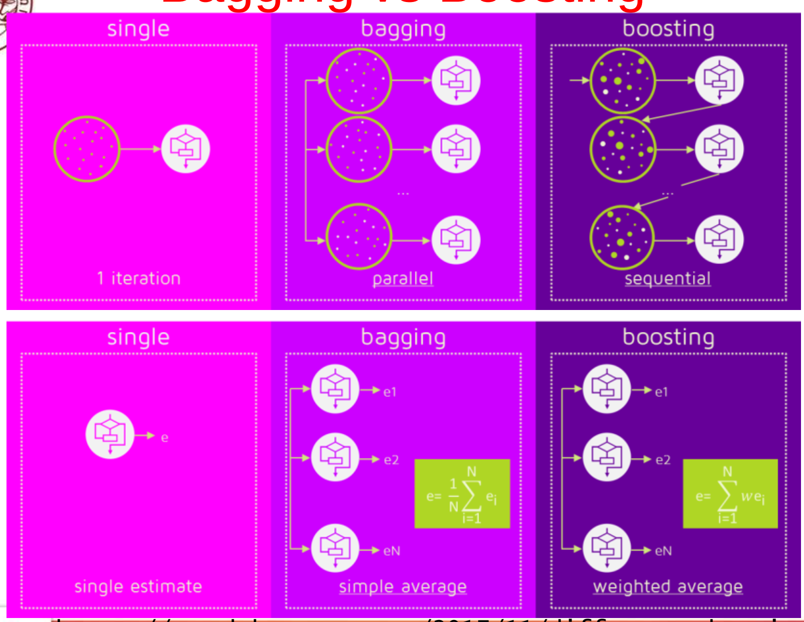
Ensemble Learning Techniques

Can we turn a weak learner into a strong learner?

	Bagging (Bootstrap Aggregating)	Boosting	Stacking
Data Partitioning	Random samples are drawn with replacement	every new subsets contains the samples that were (likely to be) misclassified by previous models	Various
Goal	Minimize Variance	Increase predictive power	Both
Exploiting methods	Random Subspace	Gradient descent	Logistic Regression
Fusion of models	(Weighted) Average	(Weighted) Majority Vote	Meta model to estimate the weights

https://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning

Bagging vs Boosting



https://www.kdnuggets.com/2017/11/difference-bagging-boosting.html

Bibliography

Fall 2019

- Anomaly Detection

  - C. Aggarwal, S. Sathe. Outlier Ensembles: An Introduction (1st ed.). 2017 Springer.
  - C. Aggarwal. *Outlier Analysis*. 2013 Springer.
  - E. Schubert. Generalized and efficient outlier detection for spatial, temporal, & high-dimensional data mining. Ph.D. LMU München, 2013
  - M Facure. Semi-Supervised Anomaly Detection Survey https://www.kaggle.com/quietrags/semi-supervised-anomaly-detection-survey
  - R. Dominguesa M. Filipponea P. Michiardia J. Zouaouib A comparative evaluation of outlier detection algorithms: Experiments and analyses Pattern Recognition Volume 74, Feb. 2018
  - A. Zimek, E. Schubert, and H.-P. Kriegel, A survey on unsupervised outlier detection in high-dimensional numerical data, Statistical Analysis and Data Mining, 5(5), 2012
  - A. Muallem, S. Shetty, J. Pan, J. Zhao, B. Biswal Hoeffding Tree Algorithms for Anomaly Detection in Streaming Datasets: A Survey JIS. Vol.8 No.4, October 2017
  - A. Lavin, S. Ahmad. Evaluating Real-time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark, ICMLA, December 2015.
  - Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-Based Anomaly Detection. ACM Trans. Knowl. Discov. Data 6, 1, Article 3 March 2012, 39 pages.
  - Z. Ding, M. Fei, An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window, IFAC Proceedings Volumes 46 (2013) 12–17.
  - S. C. Tan, K. M. Ting, T. F. Liu. Fast anomaly detection for streaming data. In Proc. of the IJCAI 2011 Volume Two, AAAI Press 1511-1516.
  - K. M. Ting, J. R. Wells. Multi-dimensional Mass Estimation and Mass-based Clustering. In Proc. of the ICDM 2010, IEEE, Washington, DC, USA, 511-520.
  - T. Bandaragoda, K. Ting, D. Albrecht, F. Liu, Y. Zhu, J. R. Wells. Isolation-based anomaly detection using nearest-neighbor ensembles Int. Journal Comput. Intelligence, Wiley 2018

Bibliography

Fall 2019

- Anomaly Detection

  - J. Marcus Hughes, Anomaly Detection and Robust Random Cut Trees, 2019 Github
  - S. Guha, N. Mishra, G. Roy, O. Schrijvers. Robust random cut forest based anomaly detection on streams. In Proc. ICML 2016, Vol. 48. JMLR.org 2712-2721.
  - A. Siddiqui, A. Fern, T. Dietterich, W.-K. Wong Sequential Feature Explanations for Anomaly Detection. Workshop on Outlier Definition, Detection, and Description (ODDx3), August 2015
  - B. Chen, K.M. Ting, T. Washio, G.Haffari. Half-Space Mass: A maximally robust and efficient data depth method. Machine Learning. 100(2-3), 2015, 677:699.
  - K. M. Ting, G.-T. Zhou, F. T. L., J. S. C. Tan. Mass estimation and its applications. In Proc. KDD 2010. ACM, New York, NY, USA, 989-998.
  - K Wu, K Zhang, W Fan, A Edwards, PS Yu RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. IEEE ICDM 2014, pp 600–609.
  - J.R. Wells, K.M Ting.,T. Washio LiNearN: A New Approach to Nearest Neighbour Density Estimator. Pattern Recognition. Vol.47, 8, 2014, 2702-2720.
  - B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, J. Platt. Support vector method for novelty detection. In Proc. NIPS 1999, MIT Press, Cambridge.

- Distributed, Data Stream Processing

  - M. Dias de Assuno, A. da Silva Veith, R. Buyya. Distributed data stream processing and edge computing. J. Netw. Comput. Appl. 103, C, February 2018, 1-17.
  - A. Shukla,Y. Simmhan. Benchmarking Distributed Stream Processing Platforms for IoT Applications. In Proc. of the Technology Conference on Performance Evaluation and Benchmarking February 2017, LNCS.
  - N. Tatbul, S. Zdonik, J. Meehan, C. Aslantas, M. Stonebraker, K. Tufte, C. Giossi, H. Quach Handling Shared, Mutable State in Stream Processing with Correctness Guarantees. IEEE Data Eng. Bull. 38(4): 94-104 2015

Anomaly Detection Software Libraries

- Collection of anomaly detection examples : https://github.com/shubhomoydas/ad_examples
- Isolation Forest (iForest) implementation
  - in R: https://r-forge.r-project.org/projects/iforest/, https://rdrr.io/rforge/IsolationForest/, https://sourceforge.net/projects/iforest/, https://github.com/Zelazny7/isoform, https://github.com/zmzhang/IOS
  - in sklearn: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html
  - In python: https://github.com/mgckind/iso_forest
  - In spark: https://github.com/titicaca/spark-iforest
  - In Java: https://github.com/bnjmn/weka/blob/master/packages/internal/isolationForest/src/main/java/weka/classifiers/misc/IsolationForest.java
  - in Go: https://github.com/e-XpertSolutions/go-iforest
- One Class SVN (1CSVN) implementation
  - in R: https://gumroad.com/l/nbjri (download the supplemental zip file at http://univprofblog.html.xdomain.jp/code/R_scripts_functions.zip)
  - In Python: https://gum.co/oPLZ (download the supplemental zip file at http://univprofblog.html.xdomain.jp/code/supportingfunctions.zip)
  - In Java: https://github.com/jnioche/libsvm-java
  - In MS Azure: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/one-class-support-vector-machine
- Unsupervised Anomaly detection algorithms
  - In Rapid Miner: http://madm.dfki.de/rapidminer/anomalydetection
- Anomaly Detection using One-Class Neural Networks: https://github.com/raghavchalapathy/oc-nn

License