Introduction to Artificial Intelligence for security pros

Cylance

Contents:

- **Introduction:**
- **Clustering:** K-means and DBSCAN Algorithms
- **Classification:**Logistic Regression and Decision Tree Algorithms
- **Probability:**
- **Deep Learning:**

Introduction

- **Artificial Superintelligence:** Produce computers which are superior to humans in every way
- **Artificial General Intelligence:** Machine as intelligent as a human and equally capable of solving the broad range of problems that require learning and reasoning. (Turing test)
- **Artificial Narrow Intelligence:** Exploits a computer's superior ability to process vast quantities of data and detect parrerns and relationships that would otherwise be difficult or imposible to a human to detect.

Machine learning in the security domain

The context in wich we have to focus on protecting a specific service is important. For example, in a website's domains, the context is connection. Fortunately, these activities generate a vast ammount of logs by sensors and software. These logs can provide the contextual data we need to identify and ameliorate threats. These is the kind of precessing ML excels.

Clustering Using the K-Means and DBSCAN Algorithms

The purpose of cluster analysis is to segregate data into a set of discrete groups or clusters based on similarities among their keey features or attributes.

A variety of statistical, artificial intelligence, and machine learning techniques can be used to create the clusters.

In the network security domain, cluster analysis tipically proceeds through a well-defined series of data preparation and analysis operations.

Step 1: Data selection and sampling

Ideally, we want to collect all of the data generated by our network, but sometimes, this is neither possible or practical, so we apply statistical sampling techiques that allow us to create more manageable subset of the data for our analysis. The sample should reflect the characteristics of the dataset as closeley as possible, or the accuracy of our results may be compromised.

> For example, if we decide to analyze the internet activity for ten different computers, our sample should include representative log entries from all ten systems.

Step 2: Feature extraction

In this stage, we decide which data elements within our sample should be extracted and subject to analysis. *In ML, this is known as Features*

> In security, the relevant features might include the percentage of ports that are open, closed, or filtered, the application running on each of these ports, and the application version number.

It is good practice to include as many features as needed, excluding the ones you know are irrelevant, because these increase processor time

Step 3: Feature Encoding and Vectorization

Most ML algorithms require data to be encoded or represented in some mathematical fashon. One very common way data can be encoded is by mapping each sample and its set of features to a grid of rows and columns. Once structured in this way, each sample is refered to as a *vector* The entire set of rows and columns is refered to as a *matrix* The encoding process we use, depends on wether the data representing each feature is *continuous, categorical*, or od some other type.

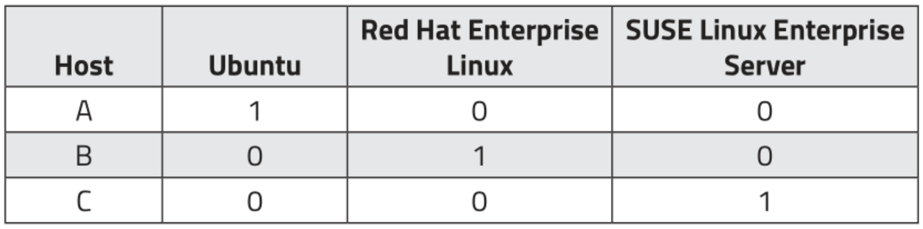
Continuous data can ocupy any nuber of infinite value within a range of values. For example, any vlaue between 0 ant 100, for cpu tempertarure.

Categorical data is represented by a small set of permissible values within a much more limited range. For example, software version or name. It is inherently useful in defining groups. For example, OS version or SW name, to group computers with similar characteristics. These categories, must be encoded as numbers before mathematical analysis. One example: OS

It is important to normalize data, so the algorithms give the same weight to diferent values and don't overblow gigher values.

Step 4: Computation and Graphin

Once we finish maping data, we can import it to a suitable statistical analysis or data mining application.



Host	Ubuntu	Red Hat Enterprise Linux	SUSE Linux Enterprise Server
A	1	0	0
B	0	1	0
C	0	0	1

Figure 1: OS

Clustering with K-Means

Clustering analysis introduces the concept of a “Feature space” that can contain thousands of dimensions, one each for every feature in our sample set. Clustering algorithms assign vectors to particular coordinates in this feature space and then measure the distance between any two vectors to determine weather they are sufficiently similar to be grouped together in the same cluster.

Caveats

- This version of K-Means works with continuous data only
- The data can be meaningfully grouped into a set of similarly sized clusters

Clustering session

1. Dataset sampled, vectorized, normalized and imported into scikit-learn
2. Data analyst sets K hyperparameter (Number of clusters to generate)
3. K-means randomly selects three vectors and assigsn coordinate in frature set (Centroids)
4. K-means assigns each vector to it’s nearest centroid
5. K-means examnes the average distance from each vector to it’s centroid. If the centroid’s current location matches this computed average, it remains stationary. Otherwise, the centroid is moved to another coordinate wich matches the computed average
6. K-means repeats step 4 for all of the vectors and reassigns them to clusters based on the new centroid location
7. K-means iterates through step 5-6 until
    1. The centroid stops mooving and it’s membership remains fixed: *Convergence*
    2. The algorithm completes the maximum number of iterations specified in advance by the analyst

Once the clustering is completed, the analyst can:

- Evaluate accuracy using a variety of validation techiques
- Convert results into a mathematical model, to assess the cluster membership of new samples

- Analyze the cluster results further using aditional statustical and machine learning techiques

K-Menas pitfalls and limitations

- The analyst must make an informed guess as to how many clusters should be created
  - Have to repeat the clustering operation until the optimum cluster size has been reached
- The clustering results vary drastically depending on where the first centroids are created
- Euclidean distance breakes down to a measure of similarity in very high dimensional feature spaces. This is known as the "Curse of dimensionality"

Clustering with DBSCAN

Density Based Spatial Clustering of Applications with Noise

- DBSCAN identifies clusters by evaluating the density of points within a given region of feature space.
- DBSCAN constructs clusters in regions where vectors are most denesley packed and considers points in sparse regions to be noise.
- DBSCAN discovers for itself how many clusters to create
- Is able to create cluster of virtually any shape and size
- DBSCAN presents analyst with 2 hyperparameters:
  - Epsilon (Eps): Radius of the circular region surrounding each point, to evaluate it's membership
  - Minimum Points (MinPts): Minimum number of points that must appear within an Epsilon neighborhood for the points inside to be included in a cluster.
- DBSCAN performs clustering by examining each point in the dataset and then assigning it to one of three categories:
  - *A core point:* Point that has more than the specified number of MinPts within it's Epsilon neighborhood
  - *A border point:* Point that falls within a core point's neighborhood but has insufficient neighbors of it's own to be a core point
  - *A noise point:* Neither a Core point or a Border point

DBSCAN clustering session

1. Dataset is sampled, vectorized, normalized, and then imported into scikit-learn
2. Analyst builds a DBSCAN object and specifies the initial Eps and MinPts values.
3. DBSCAN randomly selects one of the points on the feature space and count the number of points that lie within this point's Eps neighborhood.

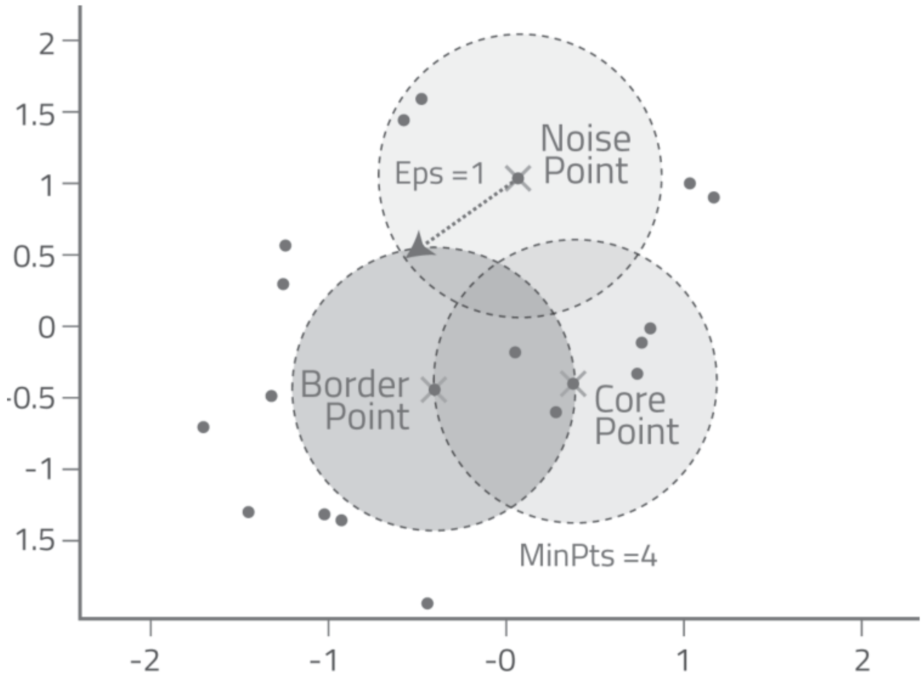


Figure 2: DBSCAN Cluster Process

If this number => than MinPts, then the point is classified as a core point and DBSCAN adds point A and it's neighbors to a cluster

4. DBSCAN moves from Point A to one of it's neighbors, eg Point B, and then classifies it as either a core or border point. If Point B classifies as a core point, then point B and it's neighbors are added to the cluster and assigned the same cluster ID. This process continues until DBSCAN has visited all of the neighbors and detected all of that cluster's core and border points.
5. DBSCAN moves to a point that it has not visited before and repeats steps 3 - 4 until all of the neighbor and noise points have been categorized. When this process concludes, al of the clusters have been identified and issued cluster ID's.

If the results are satisfactory, the clustering session ends. If not, the analyst can:

- Tune the Eps and MinPts hyperparameters
- Change the distance calculation algorithm. DBSCAN supports:
  - Euclidean Distance
  - Manhattan or City Block Distance
  - Cosine Similarity

DBSCAN pitfalls and limitations

- Extremely sensitive to small changes in MinPts and Eps settings.
- < computationally efficient as more dimensions are added
- Performs poorly with datasets that result in regions of varying densities due to the fixed values that must be assigned to MinPts and Eps.

Assessing Cluster Validity

- Run sample set through an external model and see if the resulting cluster assignments match our own
- Test our results with "Hold out data" (Data from our dataset not used to generate cluster model)
- Use statistical methods. (K-means can use Silhouette Coefficient, wich compares the average distance between points in a cluster to those in other clusters. The lower the result, the better)
- Compare results produced by diferent algorithms or by the same algorithm using diferent hyperparameter settings.

Cluster Analysis Applied to Real-World Threat Scenarios

Example with the Panama Papers breach:

If we map our network logs to clusters, including data like upload and download speeds, location of connection, emails processed versus emails sent, we could cluster the data and look for outliers, generated by the DBSCAN cluster algorithm. This might have been useful to detect strange behaviours and act rapidly

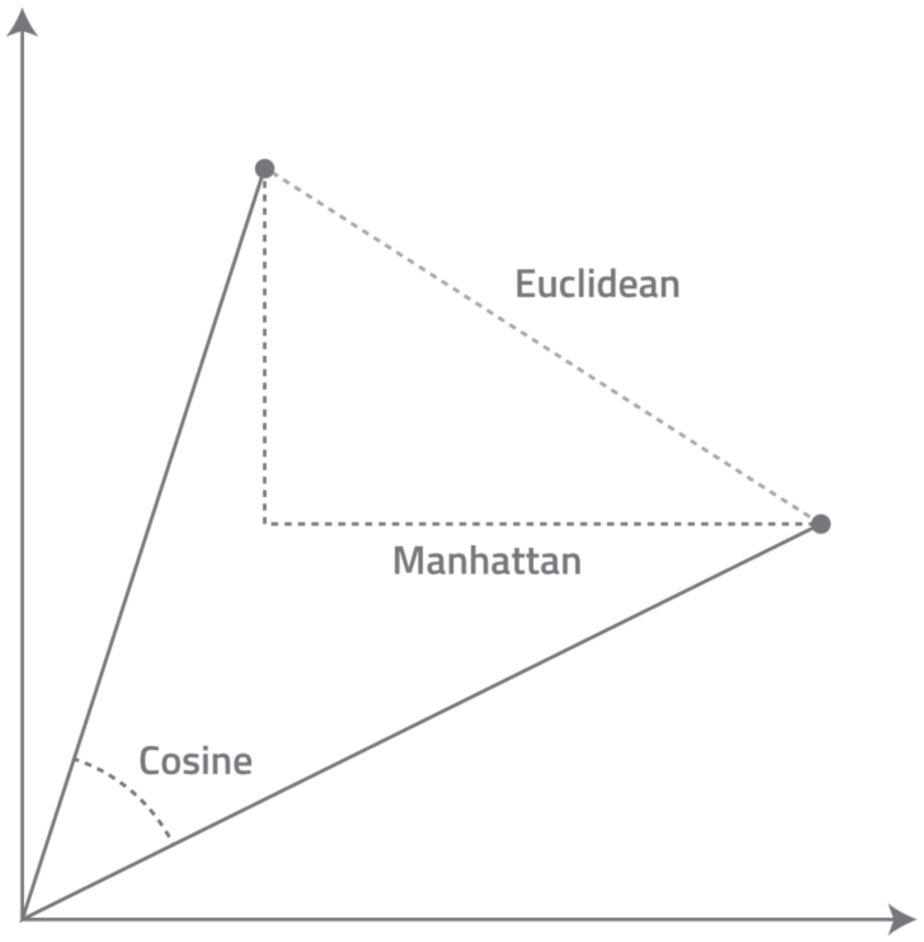


Figure 3: Euclidean, Manhattan and Cosine Distances

upon their discovery. This advantage lies in that it only needs data to detect outliers, no IPS / IDS needed or Antivirus

Clustering Session Utilizing HTTP Log Data (Practical course)

Clustering takeaways

Clustering provides a mathematically rigorous approach to detecting patterns and relationships among network, application, file, and user data that might be dificult or imposible to secure in any other way.

- Cluster analysis can be applied to virtually any kind of data once the relevant features have been extracted and normalized
- In cluster analysis, similarity between samples and their resulting cluster is determined measuring the distance between vectors.
- K-Means and DBSCAN are easy to use, efficient and broadly applicable. They are also vulnerable to "Curse of dimensionality" and may not be suitable when analyzing extremely high dimensional feature spaces.
- Clustering results must be statistically validated and carefully evaluated with respect to real world security threats.
- Clustering is particularly useful in data exploration and forensic analysis because it allows us to sift through vast ammounts of data to identify outliers.

Classification

Probability

Deep Learning