

Assignment #2

Network Anomaly Detection

ID	Name
19015359	احمد محمود عبدالحى العمري
19015357	احمد محمود السعيد جاب الله
19015894	عبدالرحمن السيد جاد السيد

Downloading data set and understand the format:

- We are using “kddcup.data 10 percent corrected” data for training.
- We are using “corrected” data for testing.
- The data have 42 features. The columns {1, 2, 3, 41(label)} are categorical. We have changed it to numeric data to easily run K-Means and normalized cut.
- You can find the names of features [here](#).
- Training data is (494 021 x 42).
- Test data is (311 029 x 42)
- There are 40 labels in the data. Only one of them is “normal” which is the only label for good request. Normal label after mapping to numeric is 16.

```
['apache2.' 'back.' 'buffer_overflow.' 'ftp_write.' 'guess_passwd.'  
'httptunnel.' 'imap.' 'ipsweep.' 'land.' 'loadmodule.' 'mailbomb.'  
'mscan.' 'multihop.' 'named.' 'neptune.' 'nmap.' 'normal.' 'perl.' 'phf.'  
'pod.' 'portsweep.' 'processtable.' 'ps.' 'rootkit.' 'saint.' 'satan.'  
'sendmail.' 'smurf.' 'snmpgetattack.' 'snmpguess.' 'spy.' 'sqlattack.'  
'teardrop.' 'udpstorm.' 'warezclient.' 'warezmaster.' 'worm.' 'xlock.'  
'xsnoop.' 'xterm.']
```

Clustering Using K-Means:

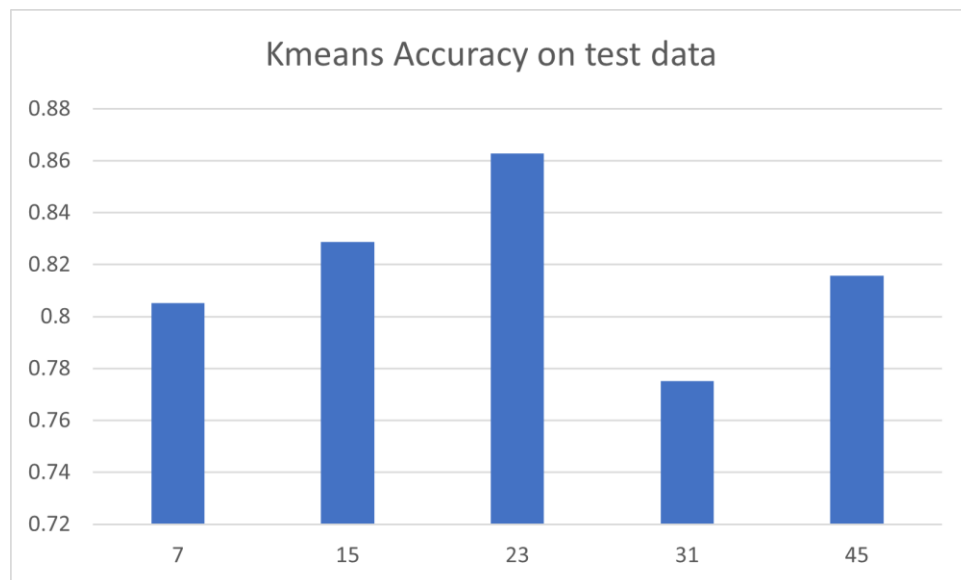
ALGORITHM 13.1. K-means Algorithm

K-MEANS (\mathbf{D}, k, ϵ):

- 1 $t = 0$
- 2 Randomly initialize k centroids: $\mu_1^t, \mu_2^t, \dots, \mu_k^t \in \mathbb{R}^d$
- 3 **repeat**
- 4 $t \leftarrow t + 1$
- 5 $C_j \leftarrow \emptyset$ for all $j = 1, \dots, k$
 // Cluster Assignment Step
- 6 **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do**
- 7 $j^* \leftarrow \operatorname{argmin}_i \{ \|\mathbf{x}_j - \mu_i^{t-1}\|^2 \}$ // Assign \mathbf{x}_j to closest centroid
- 8 $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$
- // Centroid Update Step
- 9 **foreach** $i = 1$ **to** k **do**
- 10 $\mu_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$
- 11 **until** $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$

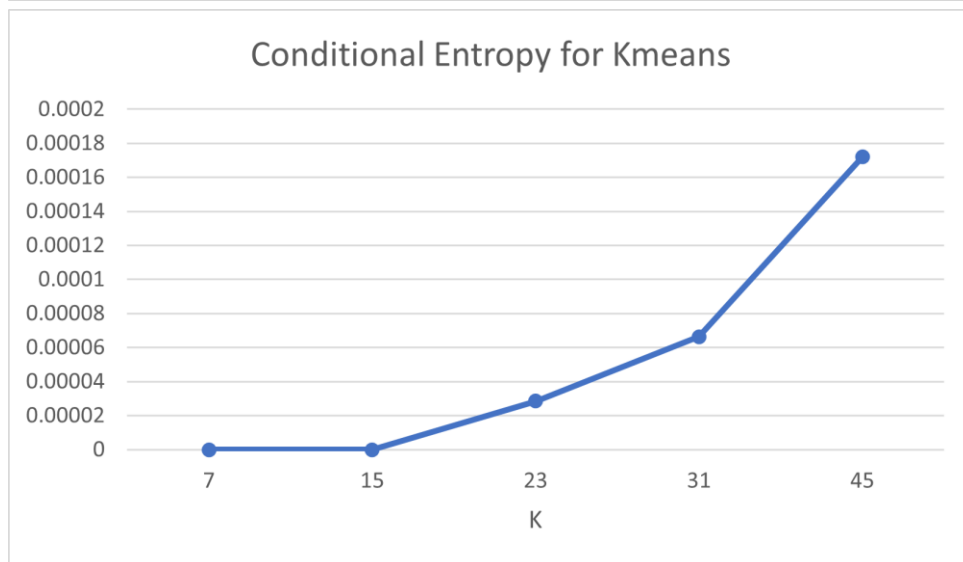
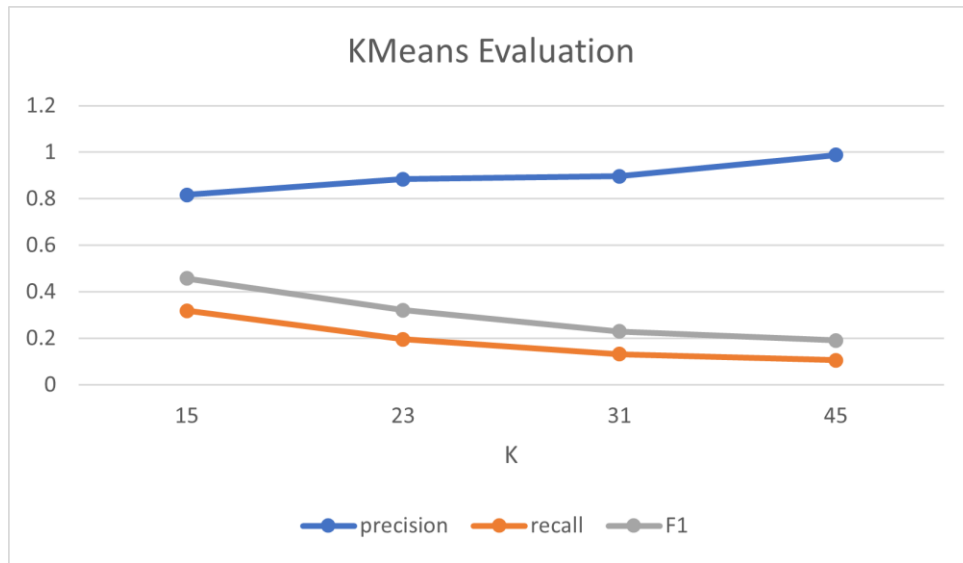
- We are dropping the label before the training, so the data is in shape (494 021 x 41).
- We are setting the random seed to 42.
- We chose k initial random centroids from the data.
- The code is finished if the error become less than epsilon (default is 0.01) or get to the max number of iterations (default is 300)
- We run the k means on $k = \{7, 15, 23, 31, 45\}$ and we evaluate the results of each cluster after that.
- To test clustering:
 - 1) we map each cluster to the major class in that cluster,
 - 2) for each row of test data we assign cluster of the nearest centroid,
 - 3) we map the test data to assigned class using its cluster,
 - 4) we calculate the accuracy using the true class of test data.
- The resulting accuracy:

<i>K</i>	7	15	23	31	45
<i>Accuracy</i>	0.8053	0.8287	0.8628	0.7752	0.8157



- The resulting evaluations:

<i>K</i>	7	15	23	31	45
<i>Precision</i>	0.5731	0.8159	0.8835	0.8968	0.9877
<i>Recall</i>	0.5903	0.3179	0.1957	0.1312	0.1051
<i>F1</i>	0.5816	0.4576	0.3205	0.2290	0.1901
<i>Conditional Entropy</i>	0.0	0.0	2.844e-05	6.638e-05	1.719e-04



Normalized Cut

- We ran the normalized cut on $k = 11$ on 0.15% from the data.
- We used RBF-kernel to build the similarity matrix.
- We used laplacian function to build L matrix.
- We used random seed = 42.

ALGORITHM 16.1. Spectral Clustering Algorithm



SPECTRAL CLUSTERING (\mathbf{D}, k):

- 1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
 - 2 **if** *ratio cut* **then** $\mathbf{B} \leftarrow \mathbf{L}$
 - 3 **else if** *normalized cut* **then** $\mathbf{B} \leftarrow \mathbf{L}^s$ or \mathbf{L}^a
 - 4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $i = n, \dots, n - k + 1$, where $\lambda_n \leq \lambda_{n-1} \leq \dots \leq \lambda_{n-k+1}$
 - 5 $\mathbf{U} \leftarrow (\mathbf{u}_n \quad \mathbf{u}_{n-1} \quad \dots \quad \mathbf{u}_{n-k+1})$
 - 6 $\mathbf{Y} \leftarrow$ normalize rows of \mathbf{U} using Eq. (16.19)
 - 7 $\mathcal{C} \leftarrow \{C_1, \dots, C_k\}$ via K-means on \mathbf{Y}
-

- Comparing normalized cut with K-means on the same data and K and DBSCAN:

	<i>Normalized-Cut</i>	<i>K-Means</i>	<i>DBSCAN</i>
# detected anomalies	5541	195433	-
Accuracy	0.8920	0.8451	<u>0.9889</u>
Precision	0.8837	0.8337	<u>0.9897</u>
Recall	<u>0.8996</u>	0.2862	0.8271
F1 Score	0.8916	0.4261	<u>0.9011</u>
Entropy	<u>1.566e-06</u>	1.540e-03	4.2512e-4

Evaluation

		Classes 				
		S₁	S₂	S₃	...	S_s
Clusters 	K₁	a ₁₁	a ₁₂	a ₁₃	...	a _{1s}
	K₂	a ₂₁	a ₂₂	a ₂₃	...	a _{2s}
	K₃	a ₃₁	a ₃₂	a ₃₃	...	a _{3s}
	...					
	K_K	a _{K1}	a _{K2}	a _{K3}	...	a _{Ks}

1. Precision:

$$\text{Precision: } P(k) = \text{Tp}/(\text{Tp}+\text{Fp}) = \frac{\max_s \{aks\}}{\sum_s aks}$$

For average precision= sum(Tp) / sum(Tp + Fp)

2. Recall:

$$P(k) = \text{Tp}/(\text{Tp}+\text{Fn}) = \frac{\max_s \{aks\}}{\sum_k aks}$$

For average precision= sum(Tp) / sum(Tp + Fn)

3. F1:

$$F1 = \frac{2 * P * R}{P + R}$$

4. Conditional Entropy:

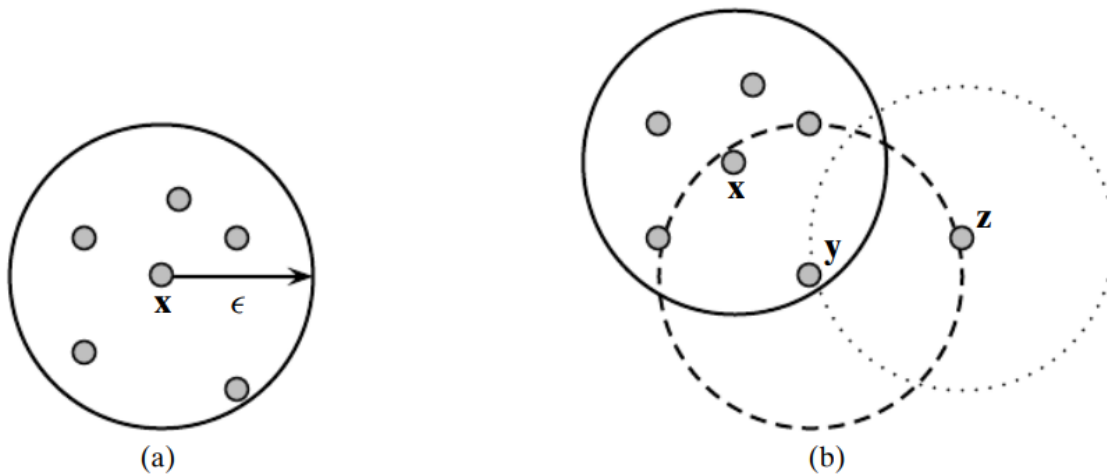
$$CE = - \sum_{i=1}^k P(ci|D) \cdot \log_2 P(ci|D)$$

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Is a popular clustering algorithm that groups together data points based on their proximity to each other and their density. The algorithm is particularly useful for discovering clusters of arbitrary shape and for handling noise and outliers in the data.

The basic idea behind DBSCAN is to group together data points that are close to each other and have a minimum number of other data points within a certain distance. In the algorithm, each data point is assigned one of three labels: "core", "border", or "noise".

A **core** point is a data point that has at least a minimum number of other data points within a specified radius, while a **border** point is a data point that is not a core point but is within the specified radius of at least one core point. **Noise** points are data points that are neither core nor border points.



(a) Neighborhood of a point. (b) Core, border, and noise points.

The DBSCAN algorithm works as follows:

1. Choose a random unvisited data point.
2. Determine whether the data point is a core point by checking if it has at least a minimum number of other data points within a specified radius.
3. If the data point is a core point, *create a new cluster* and add it to the cluster.
4. Find all the neighboring data points within the specified radius and add them to the cluster.
5. Recursively repeat steps 3 and 4 for all newly added data points until no more points can be added.
6. If the data point is not a core point, label it as a border point.
7. Move on to the next unvisited data point and repeat steps 2-6 until all data points have been visited.
8. Label any remaining unvisited data points as noise points.

ALGORITHM 15.1. Density-based Clustering Algorithm

```

DBSCAN (D,  $\epsilon$ , minpts):
1  Core  $\leftarrow \emptyset$ 
2  foreach  $\mathbf{x}_i \in \mathbf{D}$  do
3    Compute  $N_\epsilon(\mathbf{x}_i)$ 
4     $id(\mathbf{x}_i) \leftarrow \emptyset$ 
5    if  $|N_\epsilon(\mathbf{x}_i)| \geq minpts$  then Core  $\leftarrow$  Core  $\cup \{\mathbf{x}_i\}$ 
6     $k \leftarrow 0$ 
7    foreach  $\mathbf{x}_i \in$  Core, such that  $id(\mathbf{x}_i) = \emptyset$  do
8       $k \leftarrow k + 1$ 
9       $id(\mathbf{x}_i) \leftarrow k$ 
10     DENSITYCONNECTED ( $\mathbf{x}_i, k$ )
11      $\leftarrow \{C_i\}_{i=1}^k$ , where  $C_i \leftarrow \{\mathbf{x} \in \mathbf{D} \mid id(\mathbf{x}) = i\}$ 
12 Noise  $\leftarrow \{\mathbf{x} \in \mathbf{D} \mid id(\mathbf{x}) = \emptyset\}$ 
13 Border  $\leftarrow \mathbf{D} \setminus \{Core \cup Noise\}$ 
14 return Core, Border, Noise

DENSITYCONNECTED ( $\mathbf{x}, k$ ):
15 foreach  $\mathbf{y} \in N_\epsilon(\mathbf{x})$  do
16    $id(\mathbf{y}) \leftarrow k$ 
17   if  $\mathbf{y} \in Core$  then DENSITYCONNECTED ( $\mathbf{y}, k$ )

```
