NTNU

Speeding-up k-means clustering for IDS

Slobodan Petrović

NISlab seminar, 22.09.2017

Contents

- Introduction
- The original k-means (MacQueen 1967, Lloyd 1982)
- The triangle inequality and k-means
- Some research questions regarding application in intrusion detection

Introduction

- Intrusion Detection Systems (IDS)
  - Automatically find intrusions in traffic/logs
  - Two main categories
    - Misuse detection (signature-based)
      - Known attack signatures in the knowledge base of the IDS
      - Traffic units (packets, sessions) compared to the signatures
      - If a *match* is found, an alarm/alert is raised
    - Anomaly detection
      - Normality defined somehow (for example, in a knowledge base)
      - If *non-match*, an alarm/alert is raised

Introduction

- In both misuse and anomaly detection
  - In the worst case, *search* through the whole knowledge base is necessary – difficult to run IDS in real time
    - Workaround
      - Improve the search algorithms – search faster
        - Example – use bit-parallel search
      - Reduce the data set to search in – *data reduction*
        - Group data records in the knowledge base according to some similarity criterion
        - Compare traffic/logs with group representatives, not with all signatures

Introduction

- We concentrate on *data reduction*
  - To group data records in the IDS knowledge base we need
    - A set of features of each record/data unit/*vector*
    - A *distance*/similarity measure to compare vectors
    - An efficient grouping *algorithm*
      - Grouping in this context – *classification*
        - Supervised
        - Unsupervised – *clustering* – finding structure in data

Introduction

- Unsupervised partitional algorithm – k-means
  - Given a set of feature vectors, partition it in k clusters (k is given in advance) such that
    - In each cluster, we have similar feature vectors
    - Clusters are well separated
  - Separation and similarity are defined with respect to the distance/similarity measure determined in advance
    - The distance must be a *metric*

Introduction

- A metric
  - If we have defined a distance measure d between the feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ then
    - $\forall \boldsymbol{a}, \boldsymbol{b}, d(\boldsymbol{a}, \boldsymbol{b}) \geq 0$
    - $\forall \boldsymbol{a}, \boldsymbol{b}, d(\boldsymbol{a}, \boldsymbol{b}) = 0 \Leftrightarrow \boldsymbol{a} = \boldsymbol{b}$
    - $\forall \boldsymbol{a}, \boldsymbol{b}, d(\boldsymbol{a}, \boldsymbol{b}) = d(\boldsymbol{b}, \boldsymbol{a})$
    - $\forall \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, d(\boldsymbol{a}, \boldsymbol{c}) \leq d(\boldsymbol{a}, \boldsymbol{b}) + d(\boldsymbol{b}, \boldsymbol{c})$ - *the triangle inequality*

Introduction

- Examples of distance measures that are metrics
  - Euclidean distance

    $d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$

    $\boldsymbol{a}$ and $\boldsymbol{b}$ are feature vectors of dimension n
  - Edit distance
    - Minimum number of elementary edit operations (insertions, deletions, and substitutions) needed to transform the feature vector $\boldsymbol{a}$ into the feature vector $\boldsymbol{b}$
    - The feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be of different dimensions

The original k-means

- MacQueen's algorithm (1967)
  - On-line clustering method
    - Start with k groups, each consisting of a single feature vector
    - A new feature vector (*point*) is added to the group, whose mean feature vector (*centroid*) is the closest
    - After adding the new point, a new centroid is computed as a mean of points in the group to which the new point was added
  - The term k-means comes from this mean calculation
  - Inefficient – too many means calculations (for each point)

The original k-means

- LLoyd's algorithm (Lloyd-Forgy) (1982)
  - Off-line (batch) clustering method
    - All the points are present in advance
    - Start with k centroids, chosen among the points
      - Several methods to do this – influences the convergence
    - Until the algorithm converges
      - Assign each point to the cluster, whose centroid is the closest
      - Compute the new centroid of each cluster as the mean of the points assigned to the current centroid
    - Convergence – no point changes the cluster in an iteration

The original k-means

- Example – Anomaly-based IDS (1)
  - The IDS monitors 3 parameters of a host
    - CPU usage
    - Memory usage
    - Network usage

The original k-means

- Example – Anomaly-based IDS (2)
  - Observations of these parameters at 10 different time intervals $t_1, \ldots, t_{10}$

$$\mathbf{V}_1 = [0.2 \quad 0.4 \quad 0.1]; \mathbf{V}_2 = [0.9 \quad 0.7 \quad 0.3];$$
$$\mathbf{V}_3 = [0.9 \quad 0.7 \quad 0.2]; \mathbf{V}_4 = [0.8 \quad 0.6 \quad 0.4];$$
$$\mathbf{V}_5 = [0.8 \quad 0.5 \quad 0.4]; \mathbf{V}_6 = [0.8 \quad 0.5 \quad 0.3];$$
$$\mathbf{V}_7 = [0.2 \quad 0.2 \quad 0.2]; \mathbf{V}_8 = [0.2 \quad 0.3 \quad 0.2];$$
$$\mathbf{V}_9 = [0.2 \quad 0.2 \quad 0.1]; \mathbf{V}_{10} = [0.3 \quad 0.2 \quad 0.2]$$

The original k-means

- Example – Anomaly-based IDS (3)
  - $k = 2$
  - The initial centroids are $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$
  - Euclidean distance is used
  - Determine the clusters and the new centers of the clusters after the first iteration

The original k-means

- Example – Anomaly-based IDS (4)
  - The initial assignment of the given vectors (1)
    - $d(\mathbf{V}_3, \mathbf{V}_1) = \sqrt{(0.9 - 0.2)^2 + (0.7 - 0.4)^2 + (0.2 - 0.1)^2} = 0.768$
    - $d(\mathbf{V}_3, \mathbf{V}_2) = \sqrt{(0.9 - 0.9)^2 + (0.7 - 0.7)^2 + (0.2 - 0.3)^2} = 0.1$
    - $\Rightarrow \mathbf{V}_3 \rightarrow \mathbf{V}_2$

The original k-means

- Example – Anomaly-based IDS (5)
  - The initial assignment of the given vectors (2)
    - $d(\mathbf{V}_4, \mathbf{V}_1) = \sqrt{(0.8 - 0.2)^2 + (0.6 - 0.4)^2 + (0.4 - 0.1)^2} = 0.7$
    - $d(\mathbf{V}_4, \mathbf{V}_2) = \sqrt{(0.8 - 0.9)^2 + (0.6 - 0.7)^2 + (0.4 - 0.3)^2} = 0.173$
    - $\Rightarrow \mathbf{V}_4 \rightarrow \mathbf{V}_2$

The original k-means

- Example – Anomaly-based IDS (6)
  - The initial assignment of the given vectors (3)
    - $d(\mathbf{V}_5, \mathbf{V}_1) = \sqrt{(0.8 - 0.2)^2 + (0.5 - 0.4)^2 + (0.4 - 0.1)^2} = 0.678$
    - $d(\mathbf{V}_5, \mathbf{V}_2) = \sqrt{(0.8 - 0.9)^2 + (0.5 - 0.7)^2 + (0.4 - 0.3)^2} = 0.245$
    - $\Rightarrow \mathbf{V}_5 \rightarrow \mathbf{V}_2$

The original k-means

- Example – Anomaly-based IDS (7)
  - The initial assignment of the given vectors (4)
    - $d(\mathbf{V}_6, \mathbf{V}_1) = \sqrt{(0.8-0.2)^2 + (0.5-0.4)^2 + (0.3-0.1)^2} = 0.64$
    - $d(\mathbf{V}_6, \mathbf{V}_2) = \sqrt{(0.8-0.9)^2 + (0.5-0.7)^2 + (0.3-0.3)^2} = 0.224$
    - $\Rightarrow \mathbf{V}_6 \rightarrow \mathbf{V}_2$

The original k-means

- Example – Anomaly-based IDS (8)
  - The initial assignment of the given vectors (5)
    - $d(\mathbf{V}_7, \mathbf{V}_1) = \sqrt{(0.2 - 0.2)^2 + (0.2 - 0.4)^2 + (0.2 - 0.1)^2} = 0.224$
    - $d(\mathbf{V}_7, \mathbf{V}_2) = \sqrt{(0.2 - 0.9)^2 + (0.2 - 0.7)^2 + (0.2 - 0.3)^2} = 0.866$
    - $\Rightarrow \mathbf{V}_7 \rightarrow \mathbf{V}_1$

The original k-means

- Example – Anomaly-based IDS (9)
  - The initial assignment of the given vectors (6)
    - $d(\mathbf{V}_8, \mathbf{V}_1) = \sqrt{(0.2-0.2)^2 + (0.3-0.4)^2 + (0.2-0.1)^2} = 0.141$
    - $d(\mathbf{V}_8, \mathbf{V}_2) = \sqrt{(0.2-0.9)^2 + (0.3-0.7)^2 + (0.2-0.3)^2} = 0.812$
    - $\Rightarrow \mathbf{V}_8 \rightarrow \mathbf{V}_1$

The original k-means

- Example – Anomaly-based IDS (10)
  - The initial assignment of the given vectors (7)
    - $d(\mathbf{V}_9, \mathbf{V}_1) = \sqrt{(0.2-0.2)^2 + (0.2-0.4)^2 + (0.1-0.1)^2} = 0.2$
    - $d(\mathbf{V}_9, \mathbf{V}_2) = \sqrt{(0.2-0.9)^2 + (0.2-0.7)^2 + (0.1-0.3)^2} = 0.883$
    - $\Rightarrow \mathbf{V}_9 \rightarrow \mathbf{V}_1$

The original k-means

- Example – Anomaly-based IDS (11)
  - The initial assignment of the given vectors (8)
    - $d(\mathbf{V}_{10}, \mathbf{V}_1) = \sqrt{(0.3 - 0.2)^2 + (0.2 - 0.4)^2 + (0.2 - 0.1)^2} = 0.245$
    - $d(\mathbf{V}_{10}, \mathbf{V}_2) = \sqrt{(0.3 - 0.9)^2 + (0.2 - 0.7)^2 + (0.2 - 0.3)^2} = 0.787$
    - $\Rightarrow \mathbf{V}_{10} \rightarrow \mathbf{V}_1$

The original k-means

- Example – Anomaly-based IDS (12)
  - The clusters, after the first iteration:
    - Cluster 1: $\boldsymbol{V}_1, \boldsymbol{V}_7, \boldsymbol{V}_8, \boldsymbol{V}_9, \boldsymbol{V}_{10}$
    - Cluster 2: $\boldsymbol{V}_2, \boldsymbol{V}_3, \boldsymbol{V}_4, \boldsymbol{V}_5, \boldsymbol{V}_6$

The original k-means

- Example – Anomaly-based IDS (13)
  - Calculation of the new centroids (1)

$$\mathbf{V}_1 = [0.2 \quad 0.4 \quad 0.1]$$
$$\mathbf{V}_7 = [0.2 \quad 0.2 \quad 0.2]$$
$$\mathbf{V}_8 = [0.2 \quad 0.3 \quad 0.2]$$
$$\mathbf{V}_9 = [0.2 \quad 0.2 \quad 0.1]$$
$$\mathbf{V}_{10} = [0.3 \quad 0.2 \quad 0.2]$$
$$C_1' = [0.22 \quad 0.26 \quad 0.16]$$

The original k-means

- Example – Anomaly-based IDS (14)
  - Calculation of the new centroids (2)

$$\mathbf{V}_2 = [0.9 \quad 0.7 \quad 0.3]$$
$$\mathbf{V}_3 = [0.9 \quad 0.7 \quad 0.2]$$
$$\mathbf{V}_4 = [0.8 \quad 0.6 \quad 0.4]$$
$$\mathbf{V}_5 = [0.8 \quad 0.5 \quad 0.4]$$
$$\mathbf{V}_6 = [0.8 \quad 0.5 \quad 0.3]$$
$$C_2' = [0.84 \quad 0.6 \quad 0.32]$$

The original k-means

- LLoyd's algorithm is faster than MacQueen's
  - But we must have all the points in advance
  - Still, inefficient
    - We have to compute too many distances between the points and the centroids
    - After a small number of iterations, for most points, these distances do not change
      - We should avoid the distance calculations that are not needed
        - This can be exploited to accelerate the Lloyd's algorithm

The triangle inequality and k-means

- Recall the definition of a metric
  - The *triangle inequality* must hold
  - We can exploit this inequality to speed-up the k-means
  - Several ways to do this
    - Compare-means and sort-means (Phillips, 2002)
    - Elkan's method (2003)
    - Hamerly's method (2010)
    - Drake's and Hamerly's method (2012)
    - Annular and Heap method (2015), etc.

The triangle inequality and k-means

- Compare-means and sort-means (1)
  - Let $\boldsymbol{x}$ be a point and let $\boldsymbol{c}$ and $\boldsymbol{c}'$ be centroids
  - Let $\boldsymbol{x}$ be assigned to $\boldsymbol{c}$ in some iteration of k-means
    - To do this, we computed $d(\boldsymbol{x}, \boldsymbol{c})$
    - Since $\boldsymbol{c}$ and $\boldsymbol{c}'$ are known, we can also compute $d(\boldsymbol{c}, \boldsymbol{c}')$
    - We would like to avoid the computation of $d(\boldsymbol{x}, \boldsymbol{c}')$ for the next iteration of k-means
    - How can we do that?

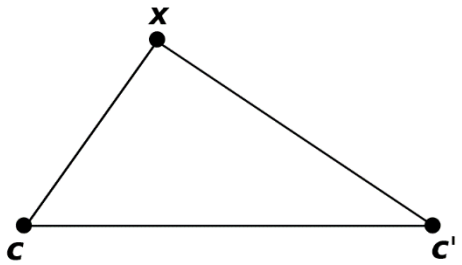
The triangle inequality and k-means

- Compare-means and sort-means (2)
  - The main idea
    - If $\boldsymbol{c}'$ is *far away* from $\boldsymbol{c}$, then $\boldsymbol{c}$ must be closer to $\boldsymbol{x}$ than $\boldsymbol{c}'$
    - By the triangle inequality

$$d(\boldsymbol{c}, \boldsymbol{c}') \leq d(\boldsymbol{x}, \boldsymbol{c}) + d(\boldsymbol{x}, \boldsymbol{c}')$$

    - Then

$$d(\boldsymbol{c}, \boldsymbol{c}') - d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{x}, \boldsymbol{c}')$$



The triangle inequality and k-means

- Compare-means and sort-means (3)
  - How far away must $\boldsymbol{c}$ and $\boldsymbol{c}'$ be to guarantee that $\boldsymbol{x}$ is closer to $\boldsymbol{c}$ than to $\boldsymbol{c}'$, i.e. that $d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{x}, \boldsymbol{c}')$?
    - Lemma 1
      - If $2d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{c}, \boldsymbol{c}')$ then we can substitute $d(\boldsymbol{c}, \boldsymbol{c}')$ with $2d(\boldsymbol{x}, \boldsymbol{c})$, so we get

        $$2d(\boldsymbol{x}, \boldsymbol{c}) - d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{x}, \boldsymbol{c}')$$

      - Consequently

        $$d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{x}, \boldsymbol{c}')$$

The triangle inequality and k-means

- Compare-means and sort-means (4)
  - Thus, we do not have to compute $d(\boldsymbol{x}, \boldsymbol{c}')$
    - We just check whether $2d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{c}, \boldsymbol{c}')$
      - If yes, the point $\boldsymbol{x}$ does not change the cluster, otherwise it does
    - This method is called *compare-means*

The triangle inequality and k-means

- Compare-means and sort-means (5)
  - *Sort-means* is a variant of compare-means
    - Compute a $k \times k$ matrix of centroid to centroid distances each time new centroids are calculated
    - Sort-ascending each row of the matrix
    - When finding the closest centroid to the point $\boldsymbol{x}$
      - Search centroids from the currently assigned centroid $\boldsymbol{c}$ to $\boldsymbol{x}$
      - If $d(\boldsymbol{c}, \boldsymbol{c}')$ to some centroid $\boldsymbol{c}'$ is $\geq 2d(\boldsymbol{x}, \boldsymbol{c})$ then stop searching
        - The point $\boldsymbol{x}$ remains in the current cluster
    - Faster than compare-means on average, but overhead (matrix)

The triangle inequality and k-means

- Elkan's method (1)
  - Let $\boldsymbol{x}$ be a point
  - Let $\boldsymbol{c}$ be the centroid $\boldsymbol{x}$ is assigned to at the beginning of the current iteration of k-means
  - Let $\boldsymbol{c}'$ be the new centroid obtained after re-computing the cluster mean at the end of the current iteration
    - We say $\boldsymbol{c}$ has *moved to* $\boldsymbol{c}'$

The triangle inequality and k-means

- Elkan's method (2)
  - Since $\boldsymbol{x}$ is assigned to $\boldsymbol{c}$ in the current iteration of k-means
    - We know $d(\boldsymbol{x}, \boldsymbol{c})$
    - From $\boldsymbol{c}$ (the centroid at the beginning of the iteration) and $\boldsymbol{c}'$ (the centroid at the end of the iteration), we can compute $d(\boldsymbol{c}, \boldsymbol{c}')$

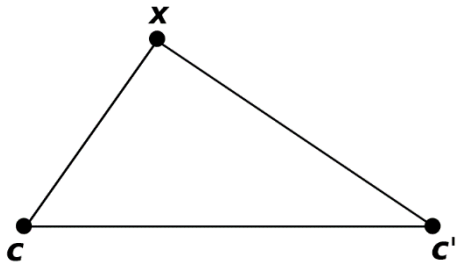
The triangle inequality and k-means

- Elkan's method (3)
  - To speed-up k-means, we must be able to compute a *lower bound* for $d(\boldsymbol{x}, \boldsymbol{c}')$ without explicitly computing $d(\boldsymbol{x}, \boldsymbol{c}')$
    - From the triangle inequality

      $$d(\boldsymbol{x}, \boldsymbol{c}) \leq d(\boldsymbol{x}, \boldsymbol{c}') + d(\boldsymbol{c}, \boldsymbol{c}')$$

    - So we have Lemma 2

      $$d(\boldsymbol{x}, \boldsymbol{c}') \geq d(\boldsymbol{x}, \boldsymbol{c}) - d(\boldsymbol{c}, \boldsymbol{c}')$$



The triangle inequality and k-means

- Elkan's method (4)
  - Uses both Lemma 1 (used also by Phillips) and Lemma 2
  - Let $\boldsymbol{x}$ be a point and let $\boldsymbol{c}$ and $\boldsymbol{c}'$ be any two centroids
  - Then, by Lemma 1

    $$\frac{1}{2} d(\boldsymbol{c}, \boldsymbol{c}') \geq d(\boldsymbol{x}, \boldsymbol{c}) \Rightarrow d(\boldsymbol{x}, \boldsymbol{c}') \geq d(\boldsymbol{x}, \boldsymbol{c})$$

    and we do not have to compute $d(\boldsymbol{x}, \boldsymbol{c}')$

The triangle inequality and k-means

- Elkan's method (5)
  - Using Lemma 1
    - Suppose we do not know the exact value of $d(\boldsymbol{x}, \boldsymbol{c})$
      - We know some *upper bound* u for $d(\boldsymbol{x}, \boldsymbol{c})$
    - If $u \leq \frac{1}{2} d(\boldsymbol{c}, \boldsymbol{c}')$, we do not have to compute $d(\boldsymbol{x}, \boldsymbol{c}')$
    - If we take the minimum over all $\boldsymbol{c}' \neq \boldsymbol{c}$ and get

      $$u \leq \frac{1}{2} \min d(\boldsymbol{c}, \boldsymbol{c}')$$

      then $\boldsymbol{x}$ does not change the cluster

The triangle inequality and k-means

- Elkan's method (6)
  - Using Lemma 2 (1)
    - Let $\boldsymbol{x}$ be a point
    - Let $\boldsymbol{c}$ be the centroid $\boldsymbol{x}$ is assigned to at the beginning of the current iteration of k-means
    - Let $\boldsymbol{c}'$ be the new centroid obtained after re-computing the cluster mean at the end of the current iteration
    - Suppose we do not know the exact value of $d(\boldsymbol{x}, \boldsymbol{c})$
      - We know some *lower bound* l for $d(\boldsymbol{x}, \boldsymbol{c})$, i.e. $d(\boldsymbol{x}, \boldsymbol{c}) \geq l$

The triangle inequality and k-means

- Elkan's method (7)
  - Using Lemma 2 (2)
    - By Lemma 2

      $$d(\boldsymbol{x}, \boldsymbol{c}') \geq d(\boldsymbol{x}, \boldsymbol{c}) - d(\boldsymbol{c}, \boldsymbol{c}') \geq l - d(\boldsymbol{c}, \boldsymbol{c}') = l'$$

    - This means, we get the lower bound for $d(\boldsymbol{x}, \boldsymbol{c}')$ from the lower bound for $d(\boldsymbol{x}, \boldsymbol{c})$ so we do not have to compute $d(\boldsymbol{x}, \boldsymbol{c}')$
    - Informally, if l was a good approximation for $d(\boldsymbol{x}, \boldsymbol{c})$ and $\boldsymbol{c}$ did not move too much at the end of the iteration, then l' is a good approximation for $d(\boldsymbol{x}, \boldsymbol{c}')$

The triangle inequality and k-means

- Elkan's method (8)
  - So, the main idea of the algorithm is
    - Instead of using the exact values of the distances of the points to the cluster centers, use their upper and lower bounds
    - Sometimes, it will be necessary to compute exact values of $d(\boldsymbol{x}, \boldsymbol{c})$, but never of $d(\boldsymbol{x}, \boldsymbol{c}')$

The triangle inequality and k-means

- Elkan's method (9)
  - The method satisfies 3 properties
    - Can use any initialization method (i.e. initial assignment of cluster centers)
    - Given the same initial cluster centers, it produces the same clustering as the standard k-means
    - Can use any metric (i.e. the only requirement for the distance measure is that it is a metric – satisfies the triangle inequality)
  - 30 to 50 times faster than Lloyd's algorithm on same data

The triangle inequality and k-means

- Hamerly's method
  - An improvement of the Elkan's method
  - Uses only 1 lower bound instead of k lower bounds that Elkan's method uses
- Drake's and Hamerly's method
  - Can use $1 < b < k$ lower bounds

Some research questions

- What is the optimal set of features that characterize attack signatures in a knowledge base of an IDS?
- What is the optimal number of clusters for the reduced-size knowledge base?
  - Affects the efficiency of the IDS, but also the False Positive Rate (FPR)
- What if we use unequal length feature vectors to characterize the signatures?

Some research questions

- What is the best cluster representative? Does it have to be the centroid?
- How can we accelerate other clustering algorithms?
- How can we determine the quality of the obtained clusters?
- How can we implement accelerated k-means on distributed computing architectures (Hadoop, Spark, Flink)?

References

- J. MacQueen, Some Methods for Classification and Analysis of Multivariate Observations
- S. Lloyd, Least Squares Quantization in PCM
- S. Phillips, Acceleration of k-Means and Related Clustering Algorithms

References

- C. Elkan, Using the Triangle Inequality to Accelerate k-Means
- G. Hamerly, J. Drake, Accelerating Lloyd's Algorithm for k-Means Clustering