# On the Robustness of Clustering Algorithms to Adversarial Attacks

Ca' Foscari University of Venice
Department of Environmental Sciences, Informatics and Statistics


Computer Science Master's Thesis
Year 2018-2019

**Graduand** Antonio Emanuele Cinà
**Supervisor** Marcello Pelillo

**Assistant supervisor** Alessandro Torcinovich

# Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Marcello Pelillo, for having spent much time helping me during the development of my thesis, recommending papers which have been both useful and inspirational for expanding this work. His lessons have made him a great advisor and have taught me priceless notions on how to do research. Together with Alessandro Torcinovich, one of his Ph.D students, I feel we formed an excellent group in which we developed multiple curious and useful ideas both related to this work and also to future projects, which I will address during my Ph.D career.
I would also like to thank my friends for all the contributions that supported me while I was working on this challenging project.
Last but not least, I want to thank my parents, who have allowed me to pursue the best education I could ever have gotten and have given me the possibility to stay here in Venice during my academic career.
Parents and friends comforted me with much patience, encouragement and also kind smiles which I will never forget.

# Abstract

Machine learning is becoming more and more used by businesses and private users as an additional tool for aiding in decision making and automation processes. However, over the past few years, there has been an increased interest in research related to the security or robustness of learning models in presence of adversarial examples. It has been discovered that wisely crafted adversarial perturbations, unaffecting human judgment, can significantly affect the performance of the learning models. Adversarial machine learning studies how learning algorithms can be fooled by crafted adversarial examples. In many ways it is a recent research area, mainly focused on the analysis of supervised models, and only few works have been done in unsupervised settings. The adversarial analysis of this learning paradigm has become imperative as in recent years unsupervised learning has been increasingly adopted in multiple security and data analysis applications. In this thesis, we are going to show how an attacker can craft poisoning perturbations on the input data for reaching target goals. In particular, we are going to analyze the robustness of two fundamental applications of unsupervised learning, feature-based data clustering and image segmentation. We are going to show how an attacker can craft poisoning perturbations against the two applications. We choose three very well known clustering algorithms (K-Means, Spectral and Dominant Sets clustering) and multiple datasets for analyzing the robustness provided by them against adversarial examples, crafted with our designed algorithms.

**Keywords** Adversarial Machine Learning, Unsupervised Learning, Clustering Algorithms, Machine Learning, Security, Robustness.

# Contents

# Chapter 1

# Introduction

Nowadays we have enough knowledge about the ability of machine learning models to make good predictions in different domains, like image classification, speech recognition, market analysis and image segmentation. Due to their incredible results, learning systems are assuming a fundamental role in very sophisticated applications as tools for aiding in decision making. However, it has been observed that, despite their advanced capabilities, they are sensitive to adversarial perturbations in the input data, leading algorithms to make wrong predictions. A very significant observation is that sometimes these alterations are invisible to human eyes, leaving us some doubts about how these models manage data. The key problem is that these models have not been designed for working in scenarios where an attacker wants to subvert or compromise the results of the system. Essentially an attacker can carefully craft adversarial samples to inject inside data for subverting the normal behavior of the machine learning model. In [7, 18] the authors discussed the problems of adversarial perturbations for spam filtering systems. They show how linear classifiers can be easily fooled by simply injecting carefully crafted perturbations inside spam messages. Even other applications of machine learning models (like fraud detection, face recognition, video surveillance, traffic prediction, credit-risk assessment, etc.) could be subject to malicious activities. Due to the sensitivity of these domains, defense strategies and robustness analyses have been studied for limiting the vulnerability to adversarial examples [1, 2, 6, 9, 15]. Nevertheless, security can be seen as an arms race game between attacker and designer. The former wants to subvert the system, on the other hand, the latter makes the system available and wants to protect it developing opportune countermeasures. During the game each player evolves its strategy in contrast to the other. Indeed, the designer develops defensive countermeasures against threats and the attacker develops new attacks for breaking defensive strategies. The key problem of machine learning models is that they are completely data-driven and data contains noise by nature. Consider the scenario of a network of temperature sensors in a room and, according to the retrieved temperature, the system reacts with appropriate strategies. Data collected by sensors are commonly subject to noise, due to internal and/or external variables. However, sensors can also be manipulated by an attacker for obtaining certain results. From this observation, we can see how it is difficult to detect when data are subject to natural noise or when they are affected by malicious threats, named adversarial noise.
In the rest of this Chapter we are going to introduce the theory behind the adversarial machine learning field, limits of the current state of the art and how this thesis is organized.

## 1.1 Adversarial Machine Learning Theory

The sensitivity of machine learning models to adversarial noise has recently attracted a large interest in the computer science community. Different authors, such us Biggio and Roli, proposed a theoretical framework for studying the roles of attacker and designer in this arms race game. In their comprehensive review [2] they propose a framework for studying the security property of supervised models. In next works, Biggio et al. [4] extended the initial framework for the unsupervised paradigm. In particular, they provided a taxonomy about possible adversary's goals and knowledge.

### 1.1.1 Adversary's Goal

The adversary's goal defines the objective function that the attacker wants to maximize or minimize by injecting adversarial examples to the system. Essentially the attacker may be interested on affecting three security properties: integrity, availability, confidentiality.

**Integrity Violation** The attacker performs malicious activities that do not compromise significantly the system behavior. In the clustering scenario we refer to the violation of the resulting grouping. Indeed, with integrity violation the attackers aims to subvert the grouping for a portion of samples preserving the original grouping as much as possible for the rest of the sample. For example, given a dataset $X$ containing two groups of users: `authorized` or `guest`. The attacker may inject well crafted adversarial perturbations for moving samples from `guest` towards `authorized`.

**Availability Violation** The attacker performs malicious activities for limiting functionalities or access to the system. In the clustering scenario the attackers wants to subvert completely the clustering process. Considering the `authorized` and `guest` groups, the attacker injects noise such that at the end the two clusters are reversed or they are merged together.

**Confidentiality Violation** The attacker performs malicious activities in order to extract private information about the composition of the clusters. Considering the `authorized` and `guest` clusters, the attacker may extract fundamental features from `authorized` users by applying strategies of reverse-engineer of the clustering process.

For all the 3 cases the attacker may also define a certain specificity, that could be targeted or indiscriminate. In the first case a targeted subset of samples is subject to adversarial perturbations. On the other hand, any sample can be manipulated without any distinction with respect to the others. For example, consider a dataset composed by three clusters: `authorized`, `guest` and `rejected`. If the attacker moves explicitly samples from rejected towards authorized, then we define the attacker's goal as targeted. Conversely, if the attacker moves samples without a target direction, then we define the attacker's goal as indiscriminate. In the proposed example, this would mean that moving samples from `guest` towards `authorized` gives the same payoff as moving samples from `guest` towards `rejected`.

### 1.1.2 Adversary's Knowledge

The adversary's knowledge defines the capacity of the attacker to inject malicious threats. The more knowledge the attacker has about the system, the greater the attacker's capacity is. It is reasonable to think that if the attacker knows perfectly the system, then it is easier is for him/her to craft appropriate threats. Conversely, if the attacker has no knowledge about the system, it is more difficult to craft threats against a black box system.
Biggio et al. provided in [4] a taxonomy of attacker's knowledge:

- Knowledge of data $\mathcal{D}$: the attacker knows exactly the composition of the data samples taken into consideration. It could be unnecessary the complete knowledge but a portion of the entire collection might suffice. $\mathcal{D}$ is drawn from an unknown probability distribution $p$.

- Knowledge of the feature space: the attacker knows exactly the features composition of data samples or how they are obtained.

- Knowledge of the algorithm: the attacker knows which algorithm is used for clustering data samples and how similarity between them is computed if necessary.

- Knowledge of the algorithm's parameters: the attacker knows the parameters provided to the clustering algorithm (ex: the number of clusters $k$).

The best scenario for the attacker, and consequently the worst case for the designer, is the one in which he/she has full knowledge (Perfect Knowledge) of the target system. Certainly, this is not always the case, bringing the attacker to a Limited knowledge, or worst, in a Zero knowledge. Even if the attacker has zero knowledge some strategies can be adopted for estimating the required components. For example, in absence of knowledge about $\mathcal{D}$ the attacker can collect a surrogate dataset $\hat{\mathcal{D}}$ with the hope that samples are retrieved from the same distribution $p$. For instance, if the target system uses clustering of malware samples, then the attacker can collect a surrogate dataset of malware $\hat{\mathcal{D}}$ and use them as an estimate of the true dataset $\mathcal{D}$. The attacker may also have no knowledge about the features representation used for projecting samples in the space. In certain conditions it could be easier to identify the right feature representation since machine learning is getting more and more standardized to provide unified services. For example, documents are likely represented in vectors of TF-IDF, or images are likely defined in matrices of intensities or RGB values.
A common argument in cyber-security is that in order to build a secure system, it should overestimate the attacker's capabilities rather than underestimating them.

## 1.2 Problem Description

Because of its strong implications, adversarial machine learning theory has been developed more and more in the latest years' research works. The greatest part of the research, including [3, 6, 11, 19], addresses the problem of adversarial examples against supervised learning. In these works authors try to give explanations to the adversarial phenomenon going deeper on how models work. Especially, Yin et al. give a first connection between adversarial learning and statistical learning theory,

with the usage of the Rademacher complexity. Papernot et al. introduce a key property of adversarial examples, which is their transferability between multiple models. Even defensive and adversary algorithms have been designed for crafting adversarial examples or for preventing the system against threats.
Conversely, from the best of our knowledge, only few works have been published against unsupervised learning paradigm applications. Biggio et al. develop some strategies for fooling single-linkage hierarchical clustering, and later on, a similar work [5] has been developed against complete-linkage hierarchical clustering. Despite this lack, the demanding techniques for cracking clustering models or for protecting them have recently turned to be fundamental. Clustering is finding a wide range of applications, due to the absence of labeled data, in very sensitive domains like image segmentation, face clustering, market or social analysis, information processing, etc.
The aim of this research is to investigate the robustness provided by some clustering algorithms in presence of very well-crafted adversarial examples. We provide three ways for crafting adversarial threats against clustering algorithms and we analyze how K-Means, Spectral and Dominant Sets clustering react against them.

## 1.3 Outline

For the remainder of this thesis, we organize our work as follows. In Chapter.2, we briefly introduce the background related to the unsupervised learning paradigm. More specifically we give a deeper introduction to the three clustering algorithm analyzed in the rest of this thesis (K-Means, Spectral and Dominant Sets clustering). We give knowledge about how these algorithms work, their formulation and properties. Following it in Chapter.3, we introduce the 3 designed algorithms against clustering. In the first part we introduce the concept and applications of image segmentation and how sensitive could be that field in presence of adversarial noise. We discuss the two designed algorithms that can be used by an attacker for fooling image classification. Conversely, in the second part of Chapter.3, we introduce the sensitivity of certain clustering applications in possible adversarial settings. We provide and explain how the three designed adversarial algorithms work in order to fool data clustering algorithms.
In Chapter.4 we show the experiments done during the development of this thesis and we analyze the robustness provided by the three algorithms. We show, using different visualization techniques, how the three algorithms react by changing the attacker's capacity.
Finally, in Chapter.5, we conclude this thesis with discussions, open issues and we highlight better our contribution thanks to this thesis. At last, we propose a list of open issues, possible future improvements and ideas, realized during the development of this work, in order to give future research directions on adversarial machine learning.

# Chapter 2

# Unsupervised Learning

Unsupervised learning is a paradigm of the machine learning field which is based on the training of knowledge without using a teacher. It includes a large set of techniques and algorithms used for learning from data without knowing the true classes. The main application of unsupervised learning consists on estimating how data are organized in the space, such that they can reconstruct the prior probability distribution of data. For doing that clustering algorithms are used with the goal of grouping a set of objects in such a way that objects in the same cluster are strongly similar (*internal criterion*) and objects from distinct clusters are strongly dissimilar (*external criterion*).
The classical clustering problem starts with a set of $n$ objects and an $n \times n$ affinity matrix $A$ of pairwise similarities that gives us an edge-weighted graph $G$. The goal of the clustering problem is to partition vertices of $G$ into maximally homogeneous groups (clusters). Usually the graph $G$ is an undirected graph, meaning that the affinity matrix $A$ is symmetric.

Figure 2.1: "Classical" clustering problem.

In literature we can find different clustering algorithms that are strongly used, and each of them manages data in different ways. Some of the clustering algorithm, that we are going to test against adversarial noise in chapter **??**, are: K-Means , Spectral and Dominant Sets clustering.

## 2.1 Images as Graphs

In some applications we can have that images correspond to our data for which we want to obtain groups partition. In this case the clustering algorithms could be used in order to reconstruct a simplified version of the input image, removing noisy information. For doing that the image is represented as an edge-weighted

undirected graph, where vertices correspond to individuals pixels and edge-weights reflect the similarity between pairs of vertices. Given an input image with $H \times W$ pixels we construct a similarity matrix $A$ such that the similarity between the pixels $i$ and $j$ is measured by:

$$A(i,j) = \exp\left(\frac{-||F(i) - F(j)||_2^2}{\sigma^2}\right)$$

- $F(i)$, is the normalized intensity of pixel $i$ (*intensity segmentation*).
- $F(i) = [v, vs\sin(h), vs\cos(h)](i)$ where $h, s, v$ are the HSV values of pixel $i$ (*color segmentation*).
- $F(i) = [|I * f_1|, \ldots, |I * f_k|](i)$ is a vector based on texture information at pixel $i$ (*texture segmentation*).

The constant $\sigma$ is introduced for obtaining a scaling effect on the affinity:

- Small $\sigma$: only nearby points are similar.
- Large $\sigma$: distant points tend to be similar.

An example of application of clustering algorithms for image segmentation is below provided:

Figure 2.2: Image of vegetables.

Figure 2.3: Clustering on pixels intensity.

Figure 2.4: Clustering on pixels color.

## 2.2 K-Means

K-Means is one of the simplest, famous and used iterative clustering algorithms. It aims to partition $n$ objects into $K$ maximal cohesive groups. The goal of the K-Means algorithm is to reach the following state: each observation belongs to the cluster with the nearest center. Its implementation can be shortly described in few lines:

- **Initialization:** Pick $K$ random points as cluster centers (centroids).

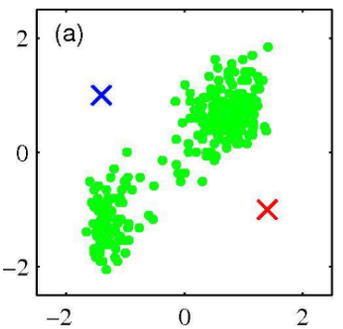



Figure 2.5: Initialization with $K = 2$.

- **Alternate:**
  1. Assign data points to closest cluster centroid.
  2. For each cluster C update the corresponding centroid to the average of points in C.

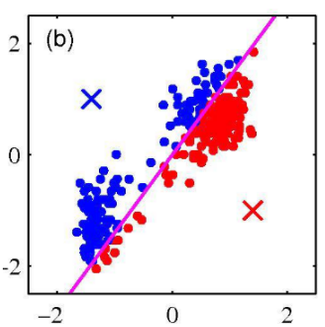



Figure 2.6: Iterative step 1.

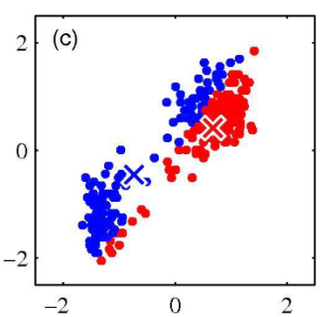



Figure 2.7: Iterative step 2.

- **Stop:** When no points' assignments change.

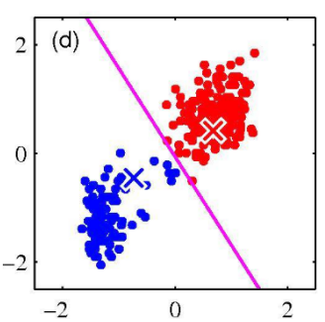



Figure 2.8: Repeat until convergence.

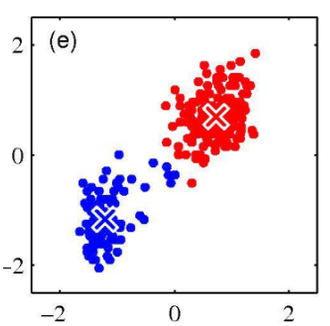



Figure 2.9: Final output.

Formally speaking we can define the K-Means algorithm over the set of points $X$ in the following way:

1. Initialize cluster centroids $\mu_1, \ldots, \mu_k$,.
2. Repeat until all points remain unchanged (convergence):

   2.1. $\forall i \in X \quad c^{(i)} = \arg\min_j ||x^{(i)} - \mu_j||^2$

   2.2. $\forall j \in C \quad \mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$

**Properties of K-Means.** K-Means benefits of the following properties:

- It is guaranteed to converge in a finite number of steps.
- It minimizes an objective function, which represents the compactness of the retrieved $K$ clusters:

$$\arg\min_{C} \sum_{i=0}^{K} \left\{ \sum_{j \in \text{elements of } C_i \text{ cluster}} ||x_j - \mu_i||^2 \right\}$$

  where $\mu_i$ is the centroid of cluster $i$.

- It is a polynomial algorithm: $O(Kn)$ for assigning each sample to the closest cluster and $O(n)$ for the update of the clusters center.

It is possible to say that K-Means is a very simple and efficient method but, on the other hand, it is strongly sensible to the initialization phase. If the initial centroids are not chosen correctly the algorithm converges to a local minimum of the error function. Another disadvantage of K-Means is that does not work well in presence non-convex shapes.

## 2.3 Eigenvector-based Clustering

The eigenvector-based clustering collects different techniques that use properties of eigenvalues and eigenvectors for solving the clustering problem. Let us represent a cluster using a vector $x$ whose $k$-th entry captures the participation of node $k$ in that cluster. If a node does not participate in cluster $x$, the corresponding entry is zero. We also impose the restriction that $x^T x = 1$. The goal of the clustering algorithm is to maximize:

$$\arg\max_{x} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j = x^T A x \tag{2.1}$$

which measures the cluster's cohesiveness and $x_i$, $x_j$ represent the measure of centrality to the cluster and it is defined as:

$$x_i = \begin{cases} \neq 0 \text{ if } i \in C \\ = 0 \text{ if } i \notin C \end{cases}$$

.

Coming back to the notion of eigenvalues of a matrix we can say that $\lambda$ is an eigenvalue of $A$ and $x_\lambda$ is the corresponding eigenvector if:

$$A x_\lambda = \lambda x_\lambda$$

From which we can derive that:

$$x_\lambda^T A x_\lambda = \lambda x_\lambda^T x_\lambda = \lambda$$

There are two important theorems that defines the nature of eigenvalues of a $n \times n$ matrix $A$:

1. If $A = A^T$ then $A$ is symmetric and has only "*real*" eigenvalues. It means that we can sort them, from the smallest one to the largest one.

2. If $A$ is symmetric then $\max_x x^T A x$ corresponds to the largest eigenvalue $\lambda$. Moreover, the corresponding eigenvector $x_\lambda$ is the argument which maximizes the cohesiveness.

Taking advantage of the two theorems we can say that considering $A$ as the affinity matrix, then clustering problem 2.1 corresponds to an **eigenvalue problem**, maximized by the eigenvector of $A$ with the largest eigenvalue.

#### Clustering by Eigenvectors Algorithm

We can define the algorithm for extracting clusters from data points using the eigenvectors strategy with the following steps:

1. Construct the affinity matrix $A$ from input $G$.
2. Compute the eigenvalues and eigenvectors of $A$.
3. Repeat
4. Take the largest unprocessed eigenvalue and the corresponding eigenvector.
5. Zero all the components corresponding to samples that have already been clustered.
6. Threshold the remaining components to detect which elements belong to this cluster.
7. If all elements have been accounted for, there are sufficient clusters.
8. Until there are sufficient clusters.

### 2.3.1 Clustering as Graph Partitioning

Let $G = (V, E, w)$ is an undirected weighted graph with $|V|$ nodes (samples) and $|E|$ edges. Note that it is undirected when the affinity matrix is symmetric. Given two graph partitions of vertices $A$ and $B$, with $B = V \setminus A$, we define $\text{cut}(A, B)$ in the following way:

$$cut(A, B) = \sum_{i \in A} \sum_{j \in B} w(i, j)$$

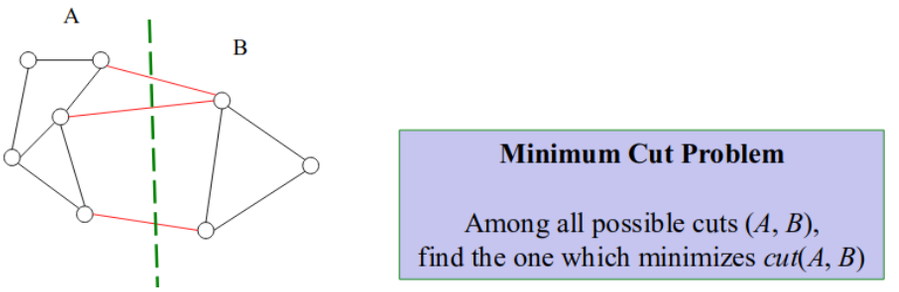



Figure 2.10: Minimum cut problem.

In the MinCut problem, we look for the partitioning that minimizes the cost of crossing from one $A$ to $B$, which is the sum of weights of the edges which cross the cut. The fundamental idea is to consider the clustering problem as a graph partitioning. Indeed, the MinCut problem can be considered a good way of solving the clustering problem in graph data. The MinCut clustering is advantageous because it is solvable in polynomial time but, on the other hand, it favors highly unbalanced clusters (often with isolated vertices), indeed, it only measures what happens between the clusters and not what happens within the clusters.

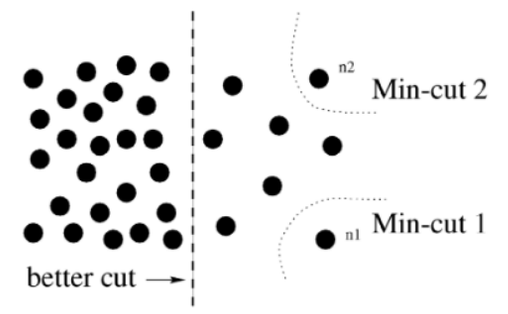



Figure 2.11: Minimum cut unbalance clusters.[16]

### 2.3.2 Normalized Cut

In order to overcome the problem of unbalanced clusters, a normalized version of the min cut problem, called **Normalized Cut**, is used and it is defined by:

$$Ncut(A,B) = \underbrace{cut(A,B)}_{\text{Between A and B}} \left( \underbrace{\frac{1}{vol(A)} + \frac{1}{vol(B)}}_{\text{Within A and B}} \right)$$

where $vol(A)$ is the volume of the set $A$ given by $vol(A) = \sum_{i \in A} d_i,\ A \subseteq V$ and $d_i = \sum_j w_{i,j}$ is the degree of nodes (sum of weights).
The Normalized Cut has the advantage of taking into consideration what happens within clusters, indeed, considering $vol(A)$ and $vol(B)$ it takes into account what's going on within $A$ and $B$.

### 2.3.3 Graph Laplacians

From an accurate analysis von Luxburg discovered that the main tools for spectral clustering are graph Laplacian matrices, defined in the spectral graph theory. In this section we are going to define different graph Laplacians and point out their most important properties since they will be later on used for solving the MinCut and NMinCut problem.

**The Unnormalized Graph Laplacian.** The **unnormalized graph Laplacian** matrix is defined as:

$$L = D - W$$

where:

- $D$ is a diagonal matrix containing information about the degree of each node in $G$.
- $W$ is the affinity matrix of $G$, containing $1s$ or $0s$ if nodes are adjacent. Diagonal elements are all set to 0.

In the following we provide an example of matrices $D$ and $W$ obtained considering the graph shown in Fig. 2.14.

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Figure 2.12: Degree matrix $D$.

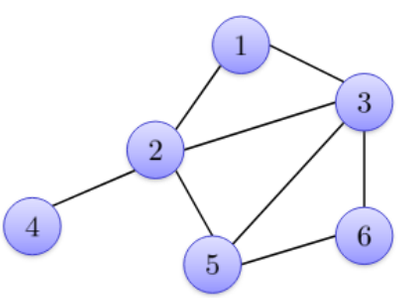
$$W = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.13: Affinity matrix $W$.

The elements of $L$ are given by:

$$L_{i,j} = \begin{cases} \mathrm{d}\,(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

where $d(v_i)$ is the degree of the vertex $i$.




$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 4 & 0 & -1 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

Assume the weights of edges are 1

Figure 2.14: Laplacian matrix L associated to the graph in the left.

In [17] are reported the properties satisfied by the matrix $L$, that are:

1. For all vectors $f$ in $\mathbb{R}^n$, we have:

$$f^\top L f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(f_i - f_j\right)^2$$

This is proved by the definition of $d_i$:

$$\begin{aligned} f^\top L f = f^\top D f - f^\top W f &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right) \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(f_i - f_j\right)^2 \end{aligned}$$

2. $L$ is symmetric (by assumption) and positive semi-definite. The symmetry of $L$ follows directly from the symmetry of $W$ and $D$. The positive semi-definiteness is a direct consequence of the first property, which shows that $f^T L f \geq 0$

3. The smallest eigenvalue of $L$ is 0, the corresponding eigenvector is the constant 1 vector.

4. $L$ has $n$ non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

First relation between spectrum and clusters:

- The multiplicity of eigenvalue $\lambda_1 = 0$ corresponds to the number of connected components of the graph.
- The eigenspace is spanned by the characteristic function of these components (so all eigenvectors are piecewise constant).

**Normalized Graph Laplacians.** In literature it is also defined the normalized form of a Laplacian graph. In particular, there exists two definitions that are closely related:

$$\begin{aligned} L_{\text{sym}} &= D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \\ L_{\text{rw}} &= D^{-1} L = I - D^{-1} W \end{aligned}$$

The first matrix $L_{sym}$ is a symmetric matrix, and the second one $L_{rw}$ as a normalized form of a Laplacian graph which is closely connected to a random walk [17].

**Definition 2.3.1** (Properties for Laplacian matrices and normalized ones)**.** In relation to Laplacian matrices, it is possible to notice that, let $L$ be the Laplacian of a graph $G = (V, E)$. Then, $L \geq 0$, indeed:
$\forall x = (x_1, \ldots, x_n)$,

$$\begin{aligned} x^T L x &= x^T \sum_{e \in E} L_e x \\ &= \sum_{e \in E} x^T L_e x \\ &= \sum_{i,j \in E} \left(x_i - x_j\right)^2 \geq 0 \end{aligned}$$

In relation instead to the normalized Laplacian Matrix we have that:

$$\forall x \in \mathbb{R}^n \quad x^T L x = \sum_{i,j} \left( \frac{x(i)}{\sqrt{d(i)}} - \frac{x(j)}{\sqrt{d(j)}} \right)^2 \geq 0$$

### 2.3.4 Solving Ncut

Any cut $(A, B)$ can be represented by a binary indicator vector $x$:

$$x_i = \begin{cases} +1 \text{ if } i \in A \\ -1 \text{ if } i \in B \end{cases}$$

It can be shown that:

$$min_x \; Ncut(x) = min_y \underbrace{\frac{y'(D-W)y}{y'Dy}}_{\text{Rayleigh quotient}} \tag{2.2}$$

subject to the constraint that $y'D1 = \sum_i y_i d_i = 0$ (with $y_i \in \{1, -b\}$ (relaxation introducing also real values), indeed $y$ is an indicator vector with 1 in the $i$-th position if the $i$-th feature point belongs to $A$, negative constant $(-b)$ otherwise).

**Theorem 2.3.2** (Solving Ncut proof)**.**

$$\lambda_2 = \min_x \frac{x^T L x}{x^T x} = \min_x \frac{x^T D^{-1/2} L D^{-1/2} x}{x^T x} \qquad \textit{Remember } L_{sym} = D^{-1/2} L D^{-1/2}$$

*Considering the change of variables obtained by setting* $y = D^{-1/2}x$ *and* $x = D^{1/2}y$*:*

$$\lambda_2 = \min_y \frac{y^T L y}{(D^{1/2}y)^T (D^{1/2}y)} = \min_y \frac{y^T L y}{y^T D y}$$

Issues rise up because solving Problem 2.2 is not computationally efficient since it is an **NP-Hard** problem. The huge Ncut time complexity brings us to take into consideration an approximation of it. If we relax the constraint that $y$ must be a discrete-valued vector and allow it to take on real values, then the original problem

$$\min_y \frac{y'(D-W)y}{y'Dy}$$

is equivalent to:

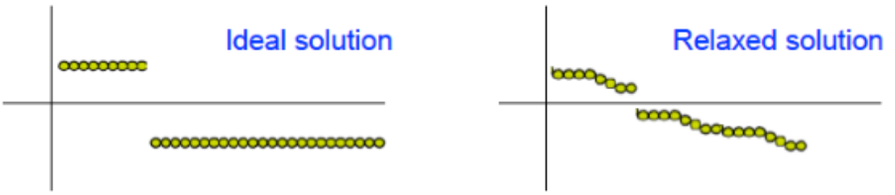
$$\min_y y'(D-W)y \quad \text{ subject to } y'Dy = 1$$

This amount to solve a *generalized* eigenvalue problem, but now the optimal solution is provided by the second smallest eigenvalue since we want to minimize the cut. Note that we pick the second smaller eigenvalues since we have seen the smallest one is always zero and corresponds to the trivial partitioning $A = V$ and $B = \emptyset$.

$$\underbrace{(D-W)}_{Laplacian} y = \lambda D y$$

We started from an NP-Hard problem and through relaxation we reached a feasible solution. However, we have not the warranty that the relaxed solution is in one to one correspondence.

**The effect of relaxation.** Through the relaxation we loose some precision in the final solution.



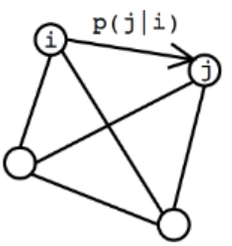

Note that the original problem returns binary values $(-1, 1)$, indicating the clustering membership. The relaxed version, on the right, returns continuous values of it can be the case that some points are not so clear to assign (close to the margin between the two). For that reason relaxed solution not always is in one-to-one correspondence with the original problem.

### 2.3.5 Random Walk Interpretation

The Ncut problem can be formalized also in terms of random walk, as highlighted in [17], since we want to find a cut that reduces the probability of jumping between nodes of different clusters. It can be defined by a Markov chain where each data point is a state, connected to all other states with some probability. With our affinity $W$ and degree $D$, the stochastic matrix is:

$$P = D^{-1}W$$

which is the row-normalized version of $W$, so each entry $P(i, j)$ is a probability of "*walking*" to state $j$ from state $i$[8].




Probability of a walk through states $(s_1, \ldots, s_m)$ is given by:

$$P(s_1, \ldots, s_2) = P(s_1) \prod_{i=2}^{m} P(s_i, s_{i-1})$$

Suppose we divide the states into two groups, and we want to minimize the probability of jumping between the two groups. We can formulate this as an eigenvector problem:

$$Py = \lambda y$$

where the component of vector $y$ will give the segmentation.
We can precise also that:

- $P$ is a stochastic matrix.
- The largest eigenvalue is 1, and its eigenvector is the all-one vector 1. Not very informative about segmentation.

- The second largest eigenvector is orthogonal to the first, and its components indicate the strongly connected sets of states.
- Meila and Shi (2001) showed that minimizing the probability of jumping between two groups in the Markov chain is equivalent to minimizing Ncut.

**Theorem 2.3.3** (Random Walk Proposition)**.** *$(\lambda, y)$ is a solution to $Py = \lambda y$ if and only if* [1]*:*

- *$1 - \lambda$ is an eigenvalue of $(D - W)y = \lambda Dy$*
- *$y$ is an eigenvector of $(D - W)y = \lambda Dy$*

***Proof:***

$$\begin{aligned} Py = \lambda y &\Leftrightarrow -Py = -\lambda y \\ &\Leftrightarrow y - Py = y - \lambda y \\ &\Leftrightarrow (I - P)y = (1 - \lambda)Iy \\ &\Leftrightarrow \left(D^{-1}D - D^{-1}W\right) y = (1 - \lambda)D^{-1}Dy \\ &\Leftrightarrow D^{-1}(D - W)y = D^{-1}(1 - \lambda)Dy \\ &\Leftrightarrow (D - W)y = (1 - \lambda)Dy \end{aligned}$$

The problem is to find a cut $(A, B)$ in a graph $G$ such that a random walk does not have many opportunities to jump between the two clusters.
This is equivalent to the Ncut problem due to the following relation:

$$Ncut(A, B) = P(A|B) + P(B|A)$$

### 2.3.6 2-way Ncut clustering algorithm

In section 2.3.4 we have seen how to solve the Normalized Cut clustering problem, and here we want to discuss its implementation for extracting just two clusters:

1. Compute the affinity matrix $W$, compute the degree matrix $D$. $D$ is diagonal and $D_{i,i} = \sum_{j \in V} W_{i,j}$
2. Solve the generalized eigenvalue problem $(D - W)y = \lambda Dy$
3. Use the eigenvector associated to the second smallest eigenvalue to bipartition the graph into two parts.

Sometimes there's not a clear threshold to split based on the second vector since it takes continuous values. In which way it is possible to choose the splitting point?

- Pick a constant value (0 or 0.5).
- Pick the median value as the splitting point.
- Look for the splitting point that has minimum Ncut value:
  1. Choose $n$ possible splitting points.
  2. Compute Ncut value.
  3. Pick the minimum.

[1] Adapted from Y. Weiss

### 2.3.7 K-way Ncut clustering algorithm

In the case we want to extract more than 2 clusters we can adopt two possible strategies:

**Approach # 1.** It is a recursive two-way cuts:

1. Given a weighted graph $G = (V, E, w)$, summarize the information into matrices $W$ and $D$.
2. Solve $(D - W)y = \lambda Dy$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph by finding the splitting point such that Ncut is minimized.
4. Decide if the current partition should be subdivided by checking the stability of the cut, and make sure Ncut is below the prespecified value.
5. Recursively repartition the segmented parts if necessary.

Note that the approach is computationally wasteful, only the second eigenvector is used, whereas the next few small eigenvectors also contain useful partitioning information.

**Approach #2.** Using the first $k$ eigenvectors:

1. Construct a similarity graph and compute the unnormalized graph Laplacian $L$.
2. Compute the $k$ smallest **generalized** eigenvectors $u_1, u_2, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.
3. Let $U = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{n \times k}$.
4. Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$th row of $U$.

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{bmatrix} = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix}$$

5. Thinking of $y_i$'s as points in $\mathbb{R}^k$, cluster them with $k$-means algorithms.

### 2.3.8 Spectral Clustering vs $K$-Means

First of all, let us define the spectral clustering algorithm [10], its goal is to cluster objects that are connected but not necessarily compact or clustered within convex boundaries. The algorithm has in input the similarity matrix $S \in \mathbb{R}^{n \times n}$ and the $k$ number of clusters to construct. It returns in output the $k$ clusters. It follows these steps:

1. Construct a similarity graph and compute the normalized graph Laplacian $L_{sym}$.
2. Embed data points in a low-dimensional space (spectral embedding), in which the clusters are more obvious, computing the $k$ smallest eigenvectors $v_1, \ldots, v_k$ of $L_{sym}$.
3. Let $V = [v_1, \ldots, v_k] \in \mathbb{R}^{n \times k}$.

4. Form the matrix $U \in \mathbb{R}^{n \times k}$ from $V$ by normalizing the row sums to have norm 1, that is:

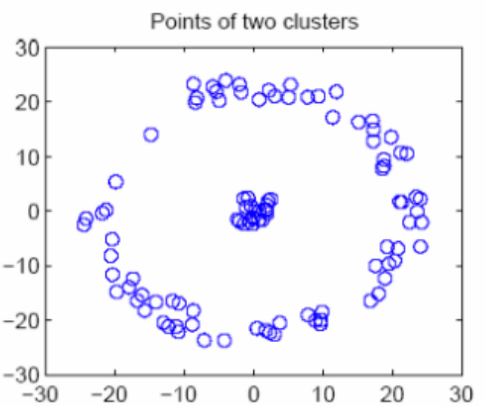
$$u_{ij} = \frac{v_{ij}}{\left(\sum_k v_{ik}^2\right)^{1/2}}$$

5. For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$th row of $U$.

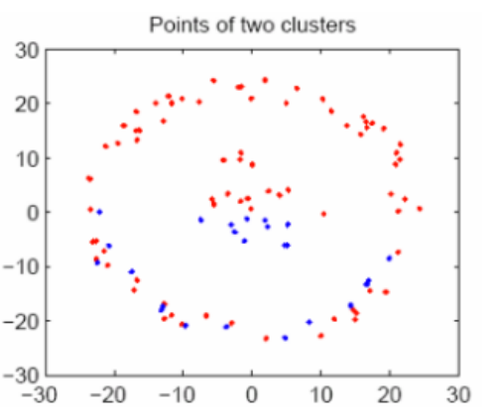
6. Cluster the points $y_i$ with $i = 1, \ldots, n$ with the $k$-means algorithm into clusters $C_1, \ldots, C_k$.

Applying $k$-means to Laplacian eigenvectors allows us to find cluster with non-convex boundaries.

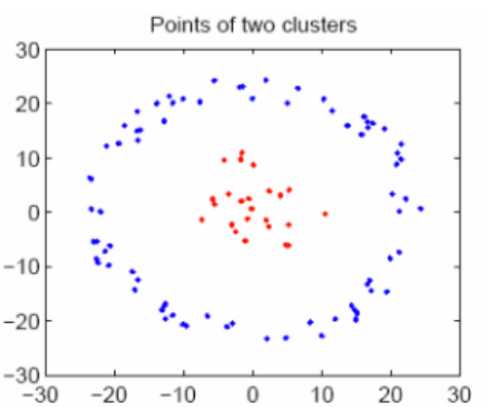

Both perform same



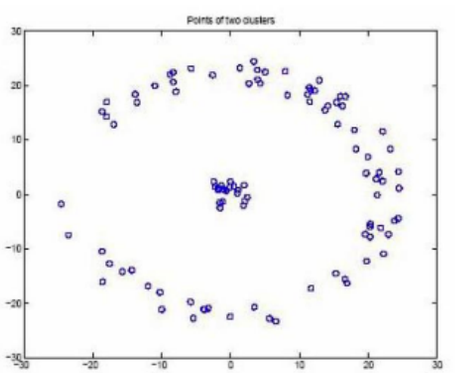

Spectral clustering is superior

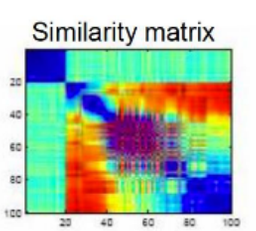


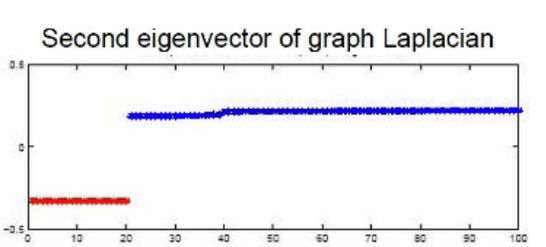

k-means output



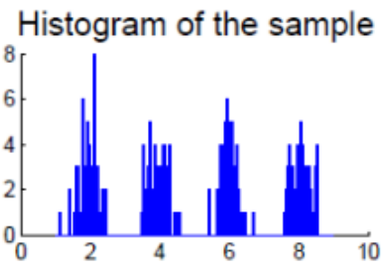

Spectral clustering output

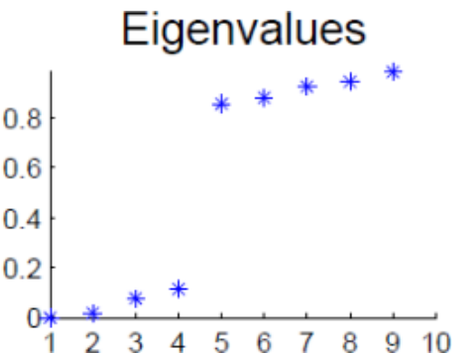


One of the possible problems that could appear on the usage of the Spectral Clustering algorithm consists on choosing the best $k$. We want to find a $k$ such that all eigenvalues $\lambda_1, \ldots, \lambda_k$ are very small, but $\lambda_{k+1}$ is relatively large. In this way, the choosing of $k$ maximizes the eigengap (difference between consecutive eigenvalues) $\delta_k = |\lambda_k - \lambda_{k-1}|$.



## 2.4 Dominant Sets

In the previous sections we have seen that data can be represented using weighted graphs, also called similarity graphs, in which data are represented by nodes in the graph and the edges represent the similarity relation between nodes. This representation allows us to codify also very complex structured entities. In literature some authors argue to the fact that a cluster can be seen as a **maximal clique**[2] of a graph, indeed the concept of clique is related to the internal cluster criteria, instead maximal clique responds to the external criteria. But the standard definition of click does not consider weighted graphs. For this reason, dominant set is introduced by Pavan and Pelillo as an extension of the maximal clique problem. We are going to see that the notion of dominant set provides measures of cohesiveness of a cluster and vertex participation of different clusters.

### 2.4.1 Cluster in Graph Theory

Data to be clustered could be coded as an undirected weighted graph with no self-loops: $G = (V, E, \omega)$, where $V = \{1, \ldots, n\}$ is the vertex set, $E \subseteq V \times V$ is the edges set and $w : E \to \mathbb{R}^*_+$ is the positive weight function. Vertices represent data points, edges neighborhood relationships and edge-weights similarity relations. $G$ is then represented with an adjacency matrix $A$, such that $a_{ij} = \omega(i, j)$. Since there are not self-loops we have that $\omega(i, i) = 0$ (main diagonal equal to 0).
One of the key problem of clustering is that there is not a unique and well define definition of cluster, but in literature researches agree that a cluster should satisfy two conditions:

- **High internal homogeneity**, also named *Internal criterion*. It means that all the objects inside a cluster should be highly similar(or low distance) to each other.
- **High external in-homogeneity**, also named *External criterion*. It means that objects coming from different clusters have low similarity (or high distance).

The idea of the criterion is that clusters are groups of objects which are strongly similar to each other if they become to the same cluster, otherwise they have a highly dissimilarity. Informally speaking a cluster is a set of entities which are alike, and entities from different clusters are not alike.
Let $S \subseteq V$ be a nonempty subset of vertices and $i \in S$. The average weighted degree of $i$ with regard to $S$ is defined as:

$$\text{awdeg}_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij} \tag{2.3}$$

This quantity over here represents the average similarity between entity $i$ and the rest of the entities in $S$. In other words how much similar $i$ is in average with all the objects in $S$. It can be observed that $\text{awdeg}_{\{i\}}(i) = 0 \ \forall i \in V$, since we have no self-loops.
We now introduce a new quantity $\phi$ such that if $j \notin S$:

$$\phi_S(i, j) = a_{ij} - \text{awdeg}_S(i) \tag{2.4}$$

[2] A **clique** is a subset of mutually adjacent vertices.
A **maximal clique** is a clique that is not contained in a larger one.

Note that $\phi_{\{i\}}(i,j) = a_{ij}\ \forall i,j \in V$ with $i \neq j$. $\phi_S(i,j)$ measures the relative similarity between $i$ and $j$ with respect to the average similarity between $i$ and its neighbors in $S$. This measures can be either positive or negative.

**Definition 2.4.1** (Pavan and Pelillo, Node's weight)**.** Let $S \subseteq V$ be a nonempty subset of vertices and $i \in S$. The weight of $i$ with regard to $S$ is:

$$w_S(i) = \begin{cases} 1 & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S\setminus\{i\}}(j,i) w_{S\setminus\{i\}}(j) & \text{otherwise} \end{cases} \tag{2.5}$$

Further, the total weight of $S$ is defined to be $W(S) = \sum_{i \in S} w_S(i)$.

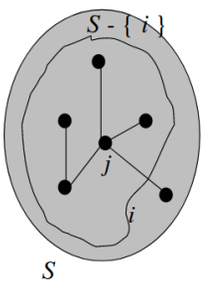



Figure 2.15: Weight of i respect to elements in S.

Note that $w_{\{i,j\}}(i) = w_{\{i,j\}}(j) = a_{ij}\ \forall i,j \in V \wedge i \neq j$. Then, $w_S(i)$ is calculated simply as a function of the weights on the edges of the sub-graph induced by $S$.

Intuitively, $w_S(i)$ gives a measure of the similarity between $i$ and $S \setminus \{i\}$ with respect to the overall similarity among the vertices of $S \setminus \{i\}$. In other words, how similar (important) $i$ is with respect to the entities in $S$. An important property of this definition is that it induces a sort of natural ranking among vertices of the graph.

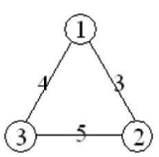



Figure 2.16: Similarity graph example.

Considering the graph proposed in Figure 2.16, we can derive a ranking between nodes:

$$w_{\{1,2,3\}}(1) < w_{\{1,2,3\}}(2) < w_{\{1,2,3\}}(3)$$

**Definition 2.4.2** (Pavan and Pelillo, Dominant Set)**.** A nonempty subset of vertices $S \subset V$ such that $W(T) > 0$ for any nonempty $T \subseteq S$, is said to be a **dominant set** if:

- $w_S(i) > 0\ \forall i \in S$ (*internal homogeneity*)
- $w_{S\cup\{i\}}(i) < 0\ \forall i \notin S$ (*external homogeneity*)

These conditions correspond to cluster properties (**internal homogeneity** and **external in-homogeneity**). Informally we can say that the first condition requires that all the nodes in the cluster $S$ are important (high weight, similar). The second one assumes that if we consider a new point in the cluster $S$, the cluster cohesiveness will be lower, meaning that the current cluster is already maximal.
By definition, dominant sets are expected to capture compact structures. Moreover, this definition is equivalent to the one of maximal clique problem when applied to unweighted graphs.

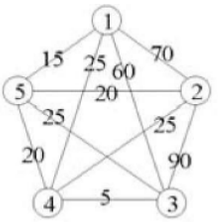



Figure 2.17: The set {1,2,3} is dominant.

### 2.4.2 Link to Standard Quadratic Optimization

Clusters are commonly represented as an $n$-dimensional vector expressing the participation of each node to a cluster. Large numbers denote a strong participation, while zero values no participation. In section 2.3 we have seen that the goal of clustering algorithm is to maximize the cohesiveness of the retrieved clusters. Formally speaking the goal can be expressed using the following optimization problem:

$$\begin{array}{ll} \text{maximize} & f(x) \\ \text{subject to} & x \in \Delta \end{array} \tag{2.6}$$

where $A$ is a symmetric real-valued matrix with null diagonal and

$$\Delta = \{x \in \mathbb{R}^n : x \geq 0 \wedge e^\top x = 1\} \tag{2.7}$$

is the standard simple of $\mathbb{R}^n$. This yields the following standard quadratic problem in which local solution corresponds to a maximally cohesive cluster.
The entity $x$ is a strict local solution of problem 2.6 if there exists a neighborhood $U \subset \Delta$ of $x$ such that $f(x) > f(z)\ \forall z \in U \setminus \{x\}$. Then we define the support $\sigma(x)$ of $x \in \Delta$ as the index set of the positive components in $x$.

$$\sigma(x) = \{i \in V : x_i > 0\}$$

In other words $\sigma(x)$ is the set of vertices in $V$ that belongs to the extracted cluster.

**Definition 2.4.3** (Pavan and Pelillo, Characteristic vector)**.** A non-empty subset $C \subseteq V$ and $C$ is a dominant set, admits a weighted **characteristic vector** $x^c \in \Delta$ if it has positive total weight $W(C)$, in which:

$$x_i^C = \begin{cases} \frac{W_c(i)}{W(C)} & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$$

The important notion provided by Definition 2.4.3 is that also dominant set solutions belong to the standard simplex, as imposed in problem 2.6. The advantage is that, empirically, strict local maximizers of the dominant sets procedure work well in extracting clusters.

### 2.4.3 Link to Game Theory

Game theory is a theoretical framework used for examining and analyzing models of strategic interaction between competing rational actors. The clustering problem, as suggested by Pavan and Pelillo, can be formulated in terms of a game, also called *clustering game*, with the following properties:

- **Symmetric game**, the payoff of playing any strategy does not depend by the player but only by the strategy itself.
- **Complete knowledge**, players have complete knowledge about the game, they know what are the strategies that can be played and the corresponding payoffs.
- **Non-cooperative game**, players take independent decisions about the strategy to play, they don't make a priori alliance.
- Players play only **pure strategies**, meaning that they do not behave "rationally" but they take decisions in a pre-programmed pattern.

In the clustering game we have two players that want to extract the best structure of cluster from data samples. The pure strategies available to the players are the data points themselves in $V$ and the similarity matrix $A$ is used as the *payoff matrix* for the clustering game. The values $A_{ij}$ and $A_{ji}$ are the revenues obtained by player 1 and player 2 considering that they have player strategies $(i, j) \in V \times V$. Remember that the main diagonal of the similarity matrix is zero, meaning that $A_{ii} = 0$. A *mixed strategy* $x = (x_1, \ldots, x_n)^T \in \Delta$ is a probability distribution over the set of pure strategies, which models a stochastic playing strategy of a player. If player 1 and 2 play mixed strategies $(x_1, x_2) \in \Delta \times \Delta$, then the expected payoffs for the players are: $\mathbf{x_1^T A x_2}$ and $\mathbf{x_2^T A x_1}$ respectively. The goal of the two players of course is to maximize as much as possible their resulting revenue. During the game each player extract an object $(i, j)$ and the resulting revenue is associated according to the payoff matrix $A$. Since we are considering $A$ equal to the similarity matrix we can say that in order to maximize their revenue the two players would coordinate their strategies so that the extracted samples belong to the same cluster. In other words, only by selecting objects belonging to the same cluster, each player is able to maximize his expected payoff. The desired condition is that the two players reach a **symmetric Nash equilibrium**, that is state in which the two players agree about the cluster membership. A **Nash Equilibrium** is a mixed-strategy profile $(x_1, x_2) \in \Delta \times \Delta$ such that no player can improve the expected payoff by changing his playing strategy, given the opponent's strategy being fixed. This concept can be expressed with the following expression:

$$y_1^T A x_2 \leq x_1^T A x_2 \qquad y_2^T A x_1 \leq x_2^T A x_1 \qquad \forall (y_1, y_2) \in (V \times V).$$

A Nash equilibrium is **symmetric** if $x_1 = x_2$, meaning that considering a symmetric Nash Equilibrium $x \in \Delta$ the two conditions hold in a unique one:

$$y^T A x \leq x^T A x$$

The symmetric Nash equilibrium condition satisfies the internal homogeneity criterion required by the dominant set definition. However, it does not include any kind of constraint that guarantees the maximality condition. In order to satisfy

this condition it is necessary to look for a different type of Nash Equilibrium, known as **Evolutionary Stable Strategy (ESS)**.

**Definition.** A symmetric Nash equilibrium $x \in \Delta$ is an ESS if it satisfies also:

$$y^T Ax = x^T Ax \implies x^T Ay > y^T Ay \qquad \forall y \in \Delta \setminus \{x\}$$

$$y^T Ax = x^T Ax \implies x^T Ay < x^T Ax \qquad \forall y \in \Delta \setminus \{x\}$$

Even if the strategy $y$ provides the same payoff of the strategy $x$, it is better to play $x$ since the payoff against itself is greater than the one provided by $y$. The two strategies $x$ and $y$ represents two Nash Equilibrium, but only $x$ is an ESS.
In conclusion we can say that the ESSs of the clustering game with affinity matrix $A$ are in **correspondence** with dominant sets of the same clustering problem instance. However, we can also conclude that ESS's are in one-to-one correspondence to (strict) local solutions of StQP.
It is possible to say that ESS's satisfy the main characteristics of a cluster:

- **Internal coherency:** High support for all samples within the group.
- **External incoherency:** Low support for external samples.

### 2.4.4 Extracting Dominant Sets

One of the major advantages of using dominant sets is that it can be written with few lines of code, and moreover we can define different clustering approaches:

- Extracting a dominant set, done using the replicator dynamics procedure.
- Partitioning of the data points, obtained using the *peel-off* strategy, which means that at each iteration we extract a dominant set and the corresponding vertices are remove from the graphs. This is done until all vertices have been clustered (Partitioning-based clustering).
- Extracting overlapping clusters, obtained enumerating dominant sets.

In our applications we are going to deal with the second one, assuming that each entity belongs to a cluster. This assumption is required since the subject of this thesis is based on comparing three algorithms, but the first two (K-Means and Spectral clustering) are essentially partitioning-based algorithm.

The **Replicator Dynamics** are deterministic game dynamics that have been developed in evolutionary game theory. It considers an ideal scenario whereby individuals are repeatedly drawn at random from a large, ideally infinite, population to play a two-player game. Players are not supposed to behave rationally, but they act according to an inherited behavioral pattern (pure strategy). An evolutionary selection process operates over time on the distribution of behaviors [14].

Let $x_i(t)$ the population share playing pure strategy $i$ at time $t$. The state of the population at time $t$ is: $x(t) = (x_1(t), \ldots, x_n(t)) \in \Delta$.

We define an evolution equation, derived from Darwin's principle of nature selection:

$$\dot{x_i} = x_i \; g_i(x)$$

where $g_i$ specifies the rate at which pure strategy $i$ replicates, $\dot{x_i}$ is grow rate of strategy $i$.

$$\frac{\dot{x_i}}{x_i} \propto \text{payoff of pure strategy } i \text{ - average population payoff}$$

The most general continuous form is given by the following equation:

$$\dot{x_i} = x_i[(Ax)_i - x^T Ax]$$

where $(Ax)_i$ is the $i$-th component of the vector and $x^T Ax$ is the average payoff for the population. If we have a result better than the average strategy there's an improvement.

**Theorem 2.4.4** (Nachbar,1990 TaylorandJonker,1978)**.** *A point $x \in \Delta$ is a Nash equilibrium if and only if $x$ is the limit point of a replicator dynamics trajectory starting from the interior of $\Delta$. Furthermore, if $x \in \Delta$ is an ESS, then it is an asymptotically stable equilibrium point for the replicator dynamics.*

Assuming that the payoff matrix $A$ is symmetric ($A = A^T$) then the game is said to be doubly symmetric. Thanks to this assumption we can derive some conclusions:

- *Fundamental Theorem of Natural Selection (Losert and Akin,1983)*
  For any doubly symmetric game, the average population payoff $f(x) = x^T Ax$ is strictly increasing along any non-constant trajectory of replicator dynamics, meaning that $\frac{df(x(t))}{dt} \geq 0 \; \forall t \geq 0$, with equality if and only if $x(t)$ is a stationary point.

- *Characterization of ESS's (Hofbauer and Sigmund, 1988)*
  For any doubly symmetric game with payoff matrix $A$, the following statements are equivalent:

  - $x \in \Delta^{ESS}$
  - $x \in \Delta$ is a strict local maximizer of $f(x) = x^T Ax$ over the standard simplex $\Delta$.
  - $x \in \Delta$ is asymptotically stable in the replicator dynamics.

A well-known discretization of replicator dynamics, which assumes non-overlapping generations, is the following (assuming a non-negative $A$):

$$x_i(t+1) = x_i(t)\frac{A(x(t))_i}{x(t)^T Ax(t)}$$

which inherits most of the dynamical properties of its continuous-time counterpart.

```
distance=inf;
while distance>epsilon
    old_x=x;
    x = x.*(A*x);
    x = x./sum(x);
    distance=pdist([x,old_x]');
end
```

Figure 2.18: MATLAB implementation of discrete-time replicator dynamics

The components of the converged vector give us a measure of the participation of the corresponding vertices in the cluster, while the value of the objective function provides of the cohesiveness of the cluster.

### 2.4.5 Dominant Sets Hierarchy

A useful extension of the Dominant Sets formulation is introduced in the optimization problem. The new formulation is now defined:

$$\begin{array}{ll} \text{maximize} & f_\alpha(x) = x'(A - \alpha I)x \\ \text{subject to} & x \in \Delta \end{array} \tag{2.8}$$

where $\alpha \geq 0$ is a parameter and $I$ is the identity matrix.

The parameter $\alpha$ affects the number of clusters found by the algorithm. With an huge value of $\alpha$ each point defines a cluster since we require a strong cohesiveness. Instead decreasing the value of $\alpha$ the number of cluster increases.
The objective function $f_\alpha$ has now two kinds of solutions:

- solutions which correspond to dominant sets for original matrix $A$ ($\alpha = 0$).
- solutions which don't correspond to any dominant set for the original matrix $A$, although they are dominant for the scaled matrix $A + \alpha(ee' - I)$. In other words, it allows to find subsets of points that are not sufficiently coherent to be dominant with respect to $A$, and hence they should be split.

This algorithm has the basic idea of starting with a sufficiently large $\alpha$ and adaptively decrease it during the clustering process following these steps:

1. Let $\alpha$ be a large positive value (ex: $\alpha > |V| - 1$).
2. Find a partition of the data into $\alpha$-clusters.
3. For all the $\alpha$-clusters that are not 0-clusters recursively repeat step 2 with decreasing $\alpha$.

### 2.4.6 Properties

- **Well separation between structure and noise.** In such situations it is often more important to cluster a small subset of the data very well, rather than optimizing a clustering criterion over all the data points, particularly in application scenarios where a large amount of noisy data is encountered.
- **Overlapping clustering**. In some cases we can have that two distinct clusters share some points, but partitional approaches impose that each element cannot be long to more than one cluster.
- Dominant sets can be found by mining **local solutions**, so it is not necessary to look for global solutions.
- Deal very well in presence of noise.
- Strong connection with theoretical results.
- Makes no assumptions on the structure of the affinity matrix, being it able to work with asymmetric and even negative similarity functions.
- Does not require a priori knowledge on the number of clusters (since it extracts them sequentially).
- Leaves clutter elements unassigned (useful, e.g., in figure/ground separation or one-class clustering problems).
- Limiting the number of dynamic's iterations it is possible to detect quasi-click structures.

# Chapter 3

# Conclusions and Future Work

In this work, we have presented how adversarial machine learning is increasingly attracting researchers and privates due to the strong implications that it can have in real life. We have shown how in sophisticated domains, such as cyber-security, adversarial perturbations can bring to dangerous conclusions. We have in particular analyzed the impact of adversarial machine learning for the unsupervised paradigm. After having shortly reviewed image segmentation in terms of clustering, we have noted how fooling an image segmentation algorithm can bring sophisticated systems to make dangerous decisions.
We have testified a lack of adversarial analysis in clustering algorithms and we have addressed this lack by performing extensive experimentation on three clustering algorithms, discovering a strong sensitivity to the adversarial perturbations. In particular, we have developed three methods for crafting adversarial examples against K-Means, Spectral and Dominant Sets clustering. The first two methods have been designed for fooling image segmentation algorithms, and we have discussed how they work along with the differences between them. The last one, instead, has been designed for fooling feature-based data clustering. We have shown how the three clustering algorithms behave, for both applications, in presence of adversarial noise. In particular, we have seen how Spectral Clustering seems to be strongly sensitive to small perturbations with respect to the other two clustering algorithms. Moreover, we have seen how K-Means and Dominant Sets preserve a similar behavior against adversarial noise, but the latter has the advantage of working better in absence of adversarial noise.
During the development of this work we have realized some ideas that could be interesting to address in the future. The first one is related to the optimization of the adversarial algorithms. At the moment, the major component that results to be computationally costly is the noise evaluation, which requires multiple iterations of the clustering algorithms. A possible solution could be to develop the proposed clustering algorithms in order to work with GPUs, to speed up the computations. In our implementation the designed code is strongly scalable, meaning that it is possible to run the entire algorithm over parallel architectures without much effort. Right now, due to the absence of GPU-supported implementations of clustering algorithms, we can not run the adversarial algorithms on GPUs. Another idea for speeding up the adversarial target algorithm **??**, is to introduce a new heuristic that avoids the injection of noise if the target samples are locally close to the desired cluster. Indeed, at the moment target samples are always moved towards the destination cluster, meaning that the algorithm injects small perturbations even if samples are already locally close. From multiple observations, we have

seen that the optimizer leaves local features more or less unchanged if samples are locally close. This consideration brings us to the conclusion that executing the optimization for those features can lead to waste computational resources, since only negligible perturbations are crafted.
The second topic of interest for future work is the connection between adversarial examples crafted against clustering and supervised classification models. A clustering algorithm can be seen as a way for estimating probability distributions of samples. If samples belong to the same group, then they are probably sampled from the same distribution. We believe that fooling clustering is tightly related to change the classes probability distributions. Supervised models are built with the goal of discriminating samples coming from different classes, therefore we think that adversarial examples crafted to fool clustering can even fool supervised algorithms. If we could verify the correctness of this idea then we might think to use the adversarial masks, like in Fig. **??**, generated offline against clustering algorithms, for fooling online classifiers. This strategy could be interesting for drastically reducing the computational time required for crafting adversarial examples. The latest idea is related to the design of the target clustering algorithm **??**. Indeed, it works moving samples from one cluster towards another one. We can imagine to generalize this framework allowing the attacker to move samples from a cluster towards multiple ones.

# Bibliography


[1] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 274–283, 2018.

[2] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 2154–2156, 2018.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML/PKDD*, 2013.

[4] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli. Is data clustering in adversarial settings secure? In *AISec'13, Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, Co-located with CCS 2013, Berlin, Germany, November 4, 2013*, pages 87–98, 2013.

[5] Battista Biggio, Samuel Rota Bulò, Ignazio Pillai, Michele Mura, Eyasu Zemene Mequanint, Marcello Pelillo, and Fabio Roli. Poisoning complete-linkage hierarchical clustering. In *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings*, pages 42–52, 2014. doi: 10.1007/978-3-662-44415-3_5.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[7] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *CEAS 2005 - Second Conference on Email and Anti-Spam, July 21-22, 2005, Stanford University, California, USA*, 2005.

[8] Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics, AISTATS 2001, Key West, Florida, USA, January 4-7, 2001*, 2001.

[9] Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 135–147, 2017.

[10] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 849–856, 2001.

[11] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[12] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I, June 2003. doi: 10.1109/CVPR.2003.1211348.

[13] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:167–172, 2007.

[14] Marcello Pelillo and Edwin R. Hancock, editors. *Energy Minimization Methods in Computer Vision and Pattern Recognition - 11th International Conference, EMMCVPR 2017, Venice, Italy, October 30 - November 1, 2017, Revised Selected Papers*, volume 10746 of *Lecture Notes in Computer Science*, 2018. Springer.

[15] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[16] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[17] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4): 395–416, 2007.

[18] Gregory L. Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA*, 2004.

[19] Dong Yin, Kannan Ramchandran, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. *CoRR*, abs/1810.11914, 2018.