CHAPTER

Steganographer Identification

1

Hanzhou Wu[1,a,*]

[*]*Shanghai University, Shanghai 200444, China*
[a]*h.wu.phd@ieee.org; https://sites.google.com/view/hzwu*

ABSTRACT

Conventional steganalysis detects the presence of steganography within single objects. In the real-world, we may face a complex scenario that one or some of multiple users called *actors* are guilty of using steganography, which is typically defined as the *Steganographer Identification Problem (SIP)*. One might use the conventional steganalysis algorithms to separate stego objects from cover objects and then identify the guilty actors. However, the guilty actors may be lost due to a number of false alarms. To deal with the SIP, most of the state-of-the-arts use unsupervised learning based approaches. In their solutions, each actor holds multiple digital objects, from which a set of feature vectors can be extracted. The well-defined distances between these feature sets are determined to measure the similarity between the corresponding actors. By applying clustering or outlier detection, the most suspicious actor(s) will be judged as the steganographer(s). Though the SIP needs further study, the existing works have good ability to identify the steganographer(s) when non-adaptive steganographic embedding was applied. In this chapter, we will present foundational concepts and review advanced methodologies in SIP. This chapter is self-contained and intended as a tutorial introducing the SIP in the context of media steganography.

---

[1] This invited chapter draft is intended as a tutorial introducing the steganographer identification problem in the context of media steganography so that a reader could easily master the basics and key methodologies of SIP. Our focuses are the general technical frameworks, ensemble methods, and feature reduction techniques. No matter what the final decision of this chapter draft will be, if you find any descriptive or technical errors, please do not hesitate to contact me. Errors will be corrected in the revised version. Thank you so much.
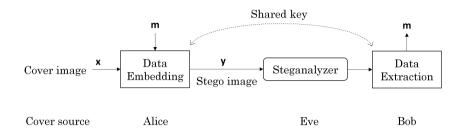
Figure 1.1   Model of steganographic communication channel.

steganalysis, clustering, anomaly detection, unsupervised learning, social networks.

1.1 INTRODUCTION

Steganography [1] hides from untrusted parties the fact that any secret is being communicated. It can be realized by embedding the secret message into an innocent object (e.g., digital image) called *cover*. The resultant object called *stego* will be sent to the trusted receiver who can perfectly retrieve the secret message from the stego according to a key. A most important requirement for any secure steganographic system is that it should be impossible for an eavesdropper to distinguish between ordinary objects and objects that contain secret information. From the perspective of the opponent, steganalysis reveals the presence of hidden messages. The objective of steganography and steganalysis can be described as follows. As shown in Figure 1.1, Alice wants to send message $\mathbf{m}$ into a randomly sampled genuine cover $\mathbf{x}$. She utilizes a key to drive the embedding procedure to generate the stego $\mathbf{y}$, which is sent through the *insecure* channel and inspected by Eve's detection algorithm, the steganalyzer. Eve has to gather evidence about the type of communication from a single output of the steganalyzer and decide whether or not to cut the communication channel. If Bob receives $\mathbf{y}$, he extracts $\mathbf{m}$ with the key.

There are two common principles for designing steganographic systems [2]. One is model-preserving steganography, aiming to preserve the selected model of the cover source during data embedding, e.g., the histogram. The other one treats steganography as a *rate-distortion* optimization task. For a payload, we hope to minimize the data embedding distortion. In other words, we want to embed as many message bits as possible for a distortion. Current advanced steganographic algorithms first assign costs of changing individual elements based on their local neighborhood and then hide the payload using coding techniques with (near) minimal total embedding cost [3].

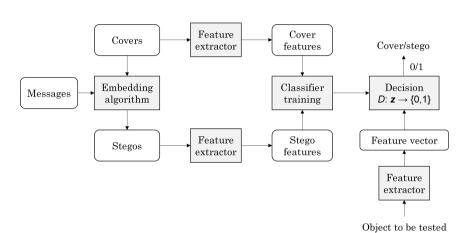Steganalysis has been greatly developed as well. As shown in Figure 1.2,

Figure 1.2   Steganalysis as a binary classification problem.

the contemporary approach to steganalysis involves three components. It extracts steganalytic features from objects; it supplies training sets of cover and stego objects; and it runs a machine learning algorithm (e.g., SVM) on the training data. This creates a decision function for novel objects, classifying them as cover or stego. Since steganography generally modifies noise-like components of the cover, advanced feature extractors aim to mine the statistical difference in high-frequency areas for detection.

In traditional steganalysis, each cover object is treated in isolation, which is implicit in the vast majority of the literature, e.g., they separate cover objects from stego objects, working on each object individually. However, such a restriction is unrealistic: In practice, we may face a complex scenario that multiple network actors send a set of media files while one or some of them are using steganography. To make matters even more complicated, an actor who is guilty of performing steganography will probably behave innocently some of the time, mixing the stego objects with genuine covers [4]. Therefore, any steganalyst will surely have to consider multiple actors, and each will transmit multiple objects, which is referred to as the *Steganographer Identification Problem (SIP)*. One might use traditional steganalysis to find stegos and then identify the guilty actors. However, the guilty actors may be lost due to a number of false alarms. Because, most traditional steganalysis algorithms involve a binary classification algorithm, which has to be trained on large sets of covers and stegos. In practice, the cover source of the actors will not be identical to that used for training by the steganalyst, so as well as being computationally demanding the training itself becomes a possible source of error if an innocent actor is falsely accused because of training model mismatch [5, 6].

It therefore requires us to design reliable detectors for the aforementioned complex scenario. Fortunately, though the SIP needs further investment, there
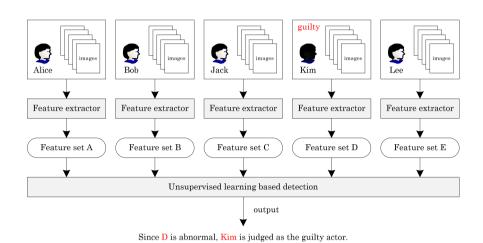
Figure 1.3   An example to identify the guilty actor by unsupervised learning.

are some effective works reported in the literature. Figure 1.3 shows an example to identify the guilty actor from multiple actors by unsupervised learning. Each actor holds multiple images, from which multiple feature sets can be extracted. Each feature set can be regarded as a "point". By using unsupervised learning based method such as clustering, the most suspicious point, i.e., feature set D, can be found out. Accordingly, Kim will be judged as the guilty actor. Following this line, in this chapter, we will introduce the foundational concepts and detail the methodologies of SIP. We hope this chapter could help readers to master both basic concepts and advanced technologies for SIP.

The structure of this chapter is organized as follows. In Section 1.2, we present the primary concepts and techniques. Then, in Section 1.3, we show two mainstream technical frameworks for SIP. In Section 1.4, we introduce the ensemble and dimensionality reduction techniques for improving identification performance. Finally, we conclude this chapter in Section 1.5.

1.2 PRIMARY CONCEPTS AND TECHNIQUES

1.2.1 JPEG COMPRESSION

JPEG is a common method of lossy compression for digital images. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. Unless mentioned, we will use JPEG images as the covers for steganography throughout this chapter because JPEG image is one of the most popular type of cover over social networks. We here briefly review the JPEG compression procedure in JPEG images.

The JPEG compression algorithm is designed specifically for the human eye. It exploits the biological properties of human sight: we are more sensitive to the illuminocity of color, rather than the chromatric value of an image, and we are not particularly sensitive to high-frequency content in images. The encoding process of the JPEG compression algorithm has five main basic steps:

1. The representation of the colors in the image is converted from RGB to YC_BC_R, having one luminance Y, and two chrominances C_B and C_R.

2. The resolution of the chroma data is reduced. This reflects the fact that eyes are less sensitive to fine color details than to fine brightness details.

3. The image is split into blocks of 8×8 pixels, and for each block, each of the Y, C_B, and C_R data undergoes the discrete cosine transform (DCT).

4. The amplitudes of the frequency components are quantized. The quality setting of the encoder affects to what extent the resolution of each frequency component is reduced. The magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components.

5. The resulting data for all 8×8 blocks is further compressed with a lossless algorithm, a variant of Huffman encoding.

The decoding process reverses these steps, except the quantization because it is irreversible. Y, C_B, and C_R can be determined as:

$$\begin{cases} Y = 0.299R + 0.587G + 0.114B, \\ C_B = 0.564(B - Y), \\ C_R = 0.713(R - Y). \end{cases} \tag{1.1}$$

For a pixel-block sized 8×8, the DCT can be described as:

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}, \tag{1.2}$$

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}, \tag{1.3}$$

where $C(z) = 1/\sqrt{2}$ for $z = 0$; otherwise, $C(z) = 1$. Here, $f(u,v)$ denotes the value at position (u,v) to be transformed, and, $F(u,v)$ is the DCT coefficient.

Once DCT is finished, JPEG moves on to quantization, in which the less important DCT coefficients are wiped out. For quantization, we need a quantization table $T \in \mathbb{Z}^{8\times8}$. After quantization, $F(u,v)$ will be updated as $\lfloor\frac{F(u,v)}{T(u,v)}\rfloor$.

1.2.2 **JPEG STEGANALYSIS FEATURES**

Each actor's JPEG images are represented by low dimensional feature vectors. We call them as JPEG steganalysis features. In this section, we review two JPEG feature extractors since they will be used latter. However, it is always

6   **CHAPTER 1** Steganographer Identification

free for us to design other efficient feature extractors for JPEG steganalysis.

1.2.2.1 PEV-274 Features

T. Pevný and J. Fridrich [7] merged the DCT features and Markov features to produce a 274-D feature vector called PEV-274 for JPEG steganalysis.

Let the luminance of a JPEG file be represented with a DCT coefficient array $d_{i,j}(k)$, $i, j = 1, ..., 8$, $k = 1, ..., n_B$, where $d_{i,j}(k)$ denotes the (i, j)-th quantized DCT coefficient in the k-th block (there are n_B blocks). Let $\mathbf{H}$ be the histogram of all $64 \times n_B$ luminance DCT coefficients, i.e., $\mathbf{H} = (H_L, ..., H_R)$, where $L = \min_{i,j,k} d_{i,j}(k)$ and $R = \max_{i,j,k} d_{i,j}(k)$. Let $\mathbf{h}^{i,j} = (h_L^{i,j}, ..., h_R^{i,j})$ be a histogram of coefficients for DCT mode (i, j). Let the dual histograms with 8×8 matrices be $\mathbf{g}_{i,j}^d = \sum_{k=1}^{n_B} \delta(d, d_{i,j}(k))$, where $i, j = 1, ..., 8$, $d = -5, ..., 5$, and $\delta(x, y) = 1$ if $x = y$ and 0 otherwise. The inter-block dependency among DCT coefficients can be captured. First, we find the variation $V =:$

$$\frac{\sum_{i,j=1}^{8} \sum_{k=1}^{|\mathbf{I}_r|-1} |d_{i,j}(\mathbf{I}_r(k)) - d_{i,j}(\mathbf{I}_r(k+1))|}{|\mathbf{I}_r| + |\mathbf{I}_c|} +$$

$$\frac{\sum_{i,j=1}^{8} \sum_{k=1}^{|\mathbf{I}_c|-1} |d_{i,j}(\mathbf{I}_c(k)) - d_{i,j}(\mathbf{I}_c(k+1))|}{|\mathbf{I}_r| + |\mathbf{I}_c|},$$

where $\mathbf{I}_r$ and $\mathbf{I}_c$ denote the vectors of block indices $1, ..., n_B$ while scanning the image by rows and by columns, respectively. Then, two scalars calculated from the decompressed JPEG image representing an integral measure of inter-block dependency over all DCT modes over the whole image are collected:

$$B_\alpha = \frac{\sum_{i=1}^{\lfloor (M-1)/8 \rfloor} \sum_{j=1}^{N} |\mathbf{c}_{8i,j} - \mathbf{c}_{8i+1,j}|^\alpha + \sum_{j=1}^{\lfloor (N-1)/8 \rfloor} \sum_{i=1}^{M} |\mathbf{c}_{i,8j} - \mathbf{c}_{i,8j+1}|^\alpha}{N\lfloor (M-1)/8 \rfloor + M\lfloor (N-1)/8 \rfloor},$$

where M and N are height and width, $\mathbf{c}_{i,j}$ is the grayscale value of the decompressed JPEG image, $\alpha = 1, 2$. We can also collect calibrated co-occurrence features, e.g., $\mathbf{C}_{0,0}(J_1) - \mathbf{C}_{0,0}(J_2)$, where $\mathbf{C}_{s,t} = \sum_{i,j=1}^{8} E_{i,j}$ and $E_{i,j}$ equals:

$$\frac{\sum_{k=1}^{|\mathbf{I}_r-1|} \delta(s, d_{i,j}(\mathbf{I}_r(k)))\delta(t, d_{i,j}(\mathbf{I}_r(k+1)))}{|\mathbf{I}_r| + |\mathbf{I}_c|} +$$

$$\frac{\sum_{k=1}^{|\mathbf{I}_c|-1} \delta(s, d_{i,j}(\mathbf{I}_c(k)))\delta(t, d_{i,j}(\mathbf{I}_c(k+1)))}{|\mathbf{I}_r| + |\mathbf{I}_c|}.$$

J_1 is the stego image, and J_2 is its calibrated version. Calibration estimates macroscopic properties of the cover from the stego. During calibration, J_1 is decompressed to the spatial domain, cropped by a few pixels in two directions, and compressed again with the same quantization matrix as J_1. The newly obtained J_2 has the most macroscopic features similar to the original cover. This is because the cropped image is visually similar to the original image. Moreover, the cropping operation brings the 8×8 DCT grid "out of sync"

with the previous compression, which effectively suppresses the influence of the previous JPEG compression and the embedding changes.

The above analysis allows us to construct 193-D DCT features. In detail, we first collect 66-D histogram features, i.e.,

$$\mathbf{H}_l(J_1) - \mathbf{H}_l(J_2), l \in \{-5, -4, ..., 5\}, \tag{1.4}$$

and for all $(i, j) \in \mathcal{L} = \{(1, 2), (2, 1), (3, 1), (2, 2), (1, 3)\}$,

$$\mathbf{h}_l^{i,j}(J_1) - \mathbf{h}_l^{i,j}(J_2), l \in \{-5, -4, ..., 5\}. \tag{1.5}$$

For dual histograms $\mathbf{g}^d, d \in \{-5, -4, ..., 5\}$, we determine the difference of the 9 lowest AC modes, i.e.,

$$\mathbf{g}_{i,j}^d(J_1) - \mathbf{g}_{i,j}^d(J_2), \tag{1.6}$$

where $(i, j) \in \{(2, 1), (3, 1), (4, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (1, 4)\}$. It allows us to collect 99-D DCT features. For the co-occurrence matrix, the central elements in the range $[-2, 2] \times [-2, 2]$ are used, yielding to 25-D features

$$\mathbf{C}_{s,t}(J_1) - \mathbf{C}_{s,t}(J_2), (s, t) \in [-2, 2] \times [-2, 2]. \tag{1.7}$$

By further collecting V, B_1, B_2, we have 193-D DCT features. The Markov feature set proposed in [8] models the differences between absolute values of neighboring DCT coefficients as a Markov process. The calculation starts by forming the matrix $F(u, v)$ of absolute values of DCT coefficients in the image. The DCT coefficients in $F(u, v)$ are arranged in the same way as pixels in the image by replacing each 8×8 block of pixels with the corresponding block of DCT coefficients Four difference arrays can be calculated along four directions: horizontal, vertical, diagonal, and minor diagonal, i.e.,

$$\begin{cases} F_h(u, v) = F(u, v) - F(u+1, v), \\ F_v(u, v) = F(u, v) - F(u, v+1), \\ F_d(u, v) = F(u, v) - F(u+1, v+1), \\ F_m(u, v) = F(u+1, v) - F(u, v+1). \end{cases} \tag{1.8}$$

From these difference arrays, four transition probability matrices $\mathbf{M}_h$, $\mathbf{M}_v$, $\mathbf{M}_d$, $\mathbf{M}_m$ are constructed as:

$$\begin{cases} \mathbf{M}_h(i, j) = \dfrac{\sum_{u,v} \delta(F_h(u, v) = i, F_h(u+1, v) = j)}{\sum_{u,v} \delta(F_h(u, v) = i)}, \\[2mm] \mathbf{M}_v(i, j) = \dfrac{\sum_{u,v} \delta(F_v(u, v) = i, F_v(u, v+1) = j)}{\sum_{u,v} \delta(F_v(u, v) = i)}, \\[2mm] \mathbf{M}_d(i, j) = \dfrac{\sum_{u,v} \delta(F_d(u, v) = i, F_d(u+1, v+1) = j)}{\sum_{u,v} \delta(F_d(u, v) = i)}, \\[2mm] \mathbf{M}_m(i, j) = \dfrac{\sum_{u,v} \delta(F_m(u+1, v) = i, F_m(u, v+1) = j)}{\sum_{u,v} \delta(F_m(u, v) = i)}. \end{cases} \tag{1.9}$$

8   **CHAPTER 1** Steganographer Identification

If the matrices are taken directly as features, the dimensionality would be too large. By using the central $[-4, 4]$ portion of the matrices, the dimension is then $4 \times 9^2 = 324$. Obviously, the above Markov features can be calibrated, i.e., $\mathbf{M}^{(c)} = \mathbf{M}(J_1) - \mathbf{M}(J_2)$. The dimension of the calibrated Markov feature set, $\{\mathbf{M}_h^{(c)}, \mathbf{M}_v^{(c)}, \mathbf{M}_d^{(c)}, \mathbf{M}_m^{(c)}\}$, remains the same as its original version. PEV-274 uses the average $\bar{\mathbf{M}} = (\mathbf{M}_h^{(c)} + \mathbf{M}_v^{(c)} + \mathbf{M}_d^{(c)} + \mathbf{M}_m^{(c)})/4$, which has a dimension of 81. Accordingly, for PEV-274, the dimension is $193 + 81 = 274$.

1.2.2.2 LI-250 Features

Li *et al.* [9] proposed high-order joint features for SIP. The motivation is that, most JPEG steganographic methods would change the correlation of neighboring coefficients due to the modification of DCT coefficients. Thus, Markov transition probability will be affected with the data embedding operation. However, Markov transition probabilities may be incapable of fully exploiting the changes in DCT coefficients and may not completely capture the neighboring joint relation. Thus, Li *et al.* developed a 250-D features, by employing joint density matrices, for pooled JPEG steganalysis. The high-order joint features are driven from high-order joint density matrices of DCT coefficients, which indicate the intra-block and inter-block dependencies of JPEG images. The features include two parts: one part is generated from mean joint density matrices of intra-block, while the other is derived from the mean joint density matrices of inter-block. The details of extracting the features are as follows.

Denoting the quantized DCT coefficients in the blocks of JPEG image as $c_{m,n}(u,v)$, where $m \in [1, M]$ and $n \in [1, N]$ are the indices of blocks, and u and v indicate the positions of coefficients in each block. The intra-block joint density matrices along horizontal, vertical, and diagonal directions, denoted by $F_{ia}^{(h)}(x,y,z)$, $F_{ia}^{(v)}(x,y,z)$, $F_{ia}^{(d)}(x,y,z)$, are calculated by:

$$\frac{\sum_{m,n,u,v} \delta(|c_{m,n}(u,v)| = x, |c_{m,n}(u,v+1)| = y, |c_{m,n}(u,v+2)| = z)}{48MN},$$

$$\frac{\sum_{m,n,u,v} \delta(|c_{m,n}(u,v)| = x, |c_{m,n}(u+1,v)| = y, |c_{m,n}(u+2,v)| = z)}{48MN},$$

$$\frac{\sum_{m,n,u,v} \delta(|c_{m,n}(u,v)| = x, |c_{m,n}(u+1,v+1)| = y, |c_{m,n}(u+2,v+2)| = z)}{36MN}.$$

The intra-block joint features $F_{ia}(x,y,z)$ are then calculated by:

$$F_{ia}(x,y,z) = \frac{1}{3}\{F_{ia}^{(h)}(x,y,z) + F_{ia}^{(v)}(x,y,z) + F_{ia}^{(d)}(x,y,z)\}, \qquad (1.10)$$

where $x, y, z \in [0, 4]$. Thus, the dimension of intra-block joint features is 125. Similarly, the inter-block second-order joint density matrices along the three directions, denoted by $F_{ir}^{(h)}(x,y,z)$, $F_{ir}^{(v)}(x,y,z)$, $F_{ir}^{(d)}(x,y,z)$, can be calcu-

lated respectively by:

$$\frac{\sum_{m,n,u,v} \delta(|c_{m,n}(u,v)| = x, |c_{m,n+1}(u,v)| = y, |c_{m,n+2}(u,v)| = z)}{64M(N-2)},$$

$$\frac{\sum_{m,n,u,v} \delta(|c_{m,n}(u,v)| = x, |c_{m+1,n}(u,v)| = y, |c_{m+2,n}(u,v)| = z)}{64(M-2)N},$$

$$\frac{\sum_{m,n,u,v} \delta(|c_{m,n}(u,v)| = x, |c_{m+1,n+1}(u,v)| = y, |c_{m+2,n+2}(u,v)| = z)}{64(M-2)(N-2)}.$$

The inter-block joint features $F_{ir}(x,y,z)$ are calculated by

$$F_{ir}(x,y,z) = \frac{1}{3}\{F_{ir}^{(h)}(x,y,z) + F_{ir}^{(v)}(x,y,z) + F_{ir}^{(d)}(x,y,z)\}, \qquad (1.11)$$

where $x,y,z \in [0,4]$. Thus, the dimension of inter-block joint features is 125. By combining the intra-block features and the inter-block features, we can get 250-D high-order joint features, which will be used for SIP later.

1.2.3 BATCH STEGANOGRAPHY AND POOLED STEGANALYSIS

In SIP, each actor holds multiple digital objects. Normal actors take no action to their own digital objects. While, for a guilty actor, i.e., the steganographer, he distributes a secret payload into the objects held by himself. It involves the concept of *batch steganography* [4], which can be described as follows: given a total of n cover objects, the steganographer hides data in $m \leq n$ of them and leaves the other covers alone. Obviously, to secure steganography, how to best spread payload between multiple covers is a critical problem to the steganographer. For a steganalysis expert, how to pool evidence from multiple objects of suspicion is a key topic, which is called *pooled steganalysis* [4]. The methods for SIP can be regarded as pooled steganalysis.

Recalling the goal of batch steganography: the steganographer wants to spread a message of length L among images $(I_1, I_2, ..., I_n)$ with secure capacities $(c_1, c_2, ..., c_n)$ using a steganographic embedding algorithm for individual images. He needs to determine the message fragment lengths $(l_1, l_2, ..., l_n)$, with $L = \sum_{i=1}^{n} l_i$, to be embedded into the images. There are four common strategies reported in the literature [10]: max-greedy, max-random, linear and even. In the max-greedy strategy, the steganographer embeds the message into the fewest possible number of covers. Assuming that the images are ordered by capacity $c_1 > c_2 > ... > c_n$, it leads to the following message lengths: $l_i = c_i, \forall i \in [1, m-1]$, $l_m = L - \sum_{i=1}^{m-1} l_i$, and $l_i = 0, \forall i \in [m+1, n]$. The max-random strategy is the same as max-greedy, except that the images to be embedded are chosen in a random order. In the linear strategy, the

message is distributed into all images proportionately to their capacity, i.e., $l_i = \frac{c_i L}{\sum_{j=1}^{n} c_j}$. For the even strategy, the secret message is distributed evenly, i.e., $l_i = L/n$. The existing works [4, 11, 12] point that, for the steganographer, the optimal behaviour is likely to be extreme concentration of payload into as few covers as possible, or the opposite in which payload is spread as thinly as possible. However, these theoretical results could not be confirmed without practical pooled steganalyzers to test against.

The aforementioned strategies require the steganographer to estimate the capacities. Ker *et al.* [10] presented a method to estimating the maximum message length for each available image for practical steganography. They first query the implementation of an algorithm to provide an initial estimate of the maximum message length. This is done either by embedding a very short message into the given message, or asking for information about a given image. Once having an initial estimate, they try to embed a randomly string of this length. If the embedding fails, the estimate of the capacity is decreased by ten bytes and the procedure is repeated. Otherwise, they deem the current estimate of the capacity as the final. It is pointed that, the capacity for an image is actually not easy to be well-defined in terms of security. One may exploit steganalysis for estimating the secure capacity, rather than simply estimating the maximum embeddable payload as the capacity.

1.2.4 AGGLOMERATIVE CLUSTERING

The hierarchical cluster analysis [13] seeks to build a hierarchy of clusters, allowing us to partition a set of objects such that "similar" objects can be clustered. As a type of hierarchical cluster analysis, agglomerative clustering is such a "bottom-up" approach that each object starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. As shown in **Algorithm 1**, the basic agglomerative clustering maintains an "active set" of clusters (initially, all objects are belonging to the active set) and at each stage the nearest two clusters are merged. When two clusters are merged, they are removed from the active set and their union is added to the active set. It iterates until there is only one cluster in the active set. The tree is formed by keeping track of which clusters were merged. All that is needed is a method to compute a distance between two clusters, for which there are a number of options. We introduce several distance measures below.

Let us write $d(x, y)$ for the distance between two objects x and y, and $D(X, Y)$ for the distance between two clusters X and Y. The single linkage uses the distance between the nearest points in the two clusters:

$$D_{\text{SL}} = \min_{x \in X, y \in Y} d(x, y). \tag{1.12}$$

Algorithm 1 Basic Agglomerative Clustering

1: Collect objects $\{x_i\}_{i=1}^n$, and set distance measure $D(\cdot, \cdot)$.
2: $\mathcal{A} \leftarrow \emptyset$ ▷ Active set starts out empty.
3: **for** each $i \in [1, n]$ **do**
4:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{\{x_i\}\}$ ▷ Add each object as its own cluster.
5: **end for**
6: $\mathcal{T} \leftarrow \mathcal{A}$ ▷ Store the tree as a sequence of merges.
7: **while** $|\mathcal{A}| > 1$ **do**
8:     $\mathcal{G}_1^*, \mathcal{G}_2^* \leftarrow \arg\min_{\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{A}} D(\mathcal{G}_1, \mathcal{G}_2)$ ▷ Choose pair in $\mathcal{A}$ with best distance.
9:     $\mathcal{A} \leftarrow \mathcal{A} \setminus \{\mathcal{G}_1^*\} \setminus \{\mathcal{G}_2^*\}$ ▷ Remove both from active set.
10:    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{G}_1^* \cup \mathcal{G}_2^*\}$ ▷ Add union to active set.
11:    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{G}_1^* \cup \mathcal{G}_2^*\}$ ▷ Add union to tree.
12: **end while**
13: **return** Tree $\mathcal{T}$

The complete linkage uses the furthest points:

$$D_{\text{CL}} = \max_{x \in X, y \in Y} d(x, y). \tag{1.13}$$

The single linkage can cause long chains of clusters, whereas complete linkage prefers compact clusters; other agglomerative clustering algorithms are intermediate, including centroid clustering

$$D_{\text{CEN}} = \frac{1}{|X| \cdot |Y|} \sum_{x \in X} \sum_{y \in Y} d(x, y), \tag{1.14}$$

and average linkage

$$D_{\text{AL}} = \frac{1}{|X \cup Y|^2 - |X \cup Y|} \sum_{u, v \in X \cup Y, u \neq v} d(u, v). \tag{1.15}$$

The input to the basic agglomerative clustering is a distance matrix between objects. The output can be displayed in a dendrogram, a tree of the successive cluster agglomerations using "height" to indicate the distance between clusters being merged.

1.2.5 LOCAL OUTLIER FACTOR

The local outlier factor (LOF) [14] is an outlier detection method by measuring the local deviation of a sample point with respect to its neighbours, which requires a distance measure $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. Assuming that, we have a set of feature points C (vectors). We expect to estimate the degree of outlying of the point $p \in C$. The LOF uses a parameter $k \in \mathbb{N}$, with $1 < k < |C|$, specifying the number of nearest neighbors.

In LOF, the k-distance of p, denoted by $d_k(p)$, is defined as a distance between p and $o \in C \setminus \{p\}$, such that:

- for at least k points $o' \in C \setminus \{p\}$, it holds $d(p, o') \le d_k(p)$, and
- for at most $k - 1$ points $o' \in C \setminus \{p\}$, it holds $d(p, o') < d_k(p)$.

The k-neighborhood of p is defined as $N_k(p) = \{o \in C | d(p, o) \le d_k(p)\}$. Note that, the number of elements in the k-th neighborhood, i.e., $|N_k(p)|$, can be greater than k. The reachability distance of p w.r.t. o is defined as $r_k(p, o) = \max\{d_k(o), d(p, o)\}$. The local reachability density of p is defined as an inverse of the average reachability distance of p to all points o in its k-neighborhood, i.e.,

$$\mathrm{LRD}_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} r_k(p, o)}. \tag{1.16}$$

The LOF method uses reachability distance to estimate density of the probability at point p. It also estimates the average density of the probability of all points in the k-neighborhood, $o \in N_k(p)$, and compares these quantities. The level of outlying is therefore adaptive with respect to the closest neighborhood. Thus, the LOF value of p is finally defined as:

$$\mathrm{LOF}_k(p) = \frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \frac{\mathrm{LRD}_k(o)}{\mathrm{LRD}_k(p)}. \tag{1.17}$$

The $\mathrm{LOF}_k(p)$ captures degree to which p can be called an outlier. A high value indicates that p is an outlier, since its reachability density, $\mathrm{LRD}_k(p)$, is smaller in comparison to the local reachability density of its neighbors.

1.2.6 MAXIMUM MEAN DISCREPANCY

In SIP, we need a measure of distance between two actors. Since each actor has a set of feature vectors, we have to design a distance measure for two sets of feature vectors. The maximum mean discrepancy (MMD) [15, 16, 17] has been empirically shown to be quite effective for distance measurement. It can be used for SIP. Given observations $X = \{\mathbf{x}_i\}_{i=1}^{|X|}$ and $Y = \{\mathbf{y}_i\}_{i=1}^{|Y|}$, which are i.i.d. drawn from $p(\mathbf{x})$ and $q(\mathbf{y})$ defined on $\mathbb{R}^d$, let $\mathcal{F}$ be a class of functions $f : \mathbb{R}^d \mapsto \mathbb{R}$, the MMD and its empirical estimate are:

$$\mathrm{MMD}[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q(\mathbf{y})} f(\mathbf{y}), \tag{1.18}$$

$$\mathrm{MMD}[\mathcal{F}, X, Y] = \sup_{f \in \mathcal{F}} \frac{1}{|X|} \sum_{\mathbf{x} \in X} f(\mathbf{x}) - \frac{1}{|Y|} \sum_{\mathbf{y} \in Y} f(\mathbf{y}). \tag{1.19}$$

Usually, $\mathcal{F}$ is selected as a unit ball in a universal RKHS $\mathcal{H}$ defined on compact metric space $\mathbb{R}^d$ with kernel $k(\cdot, \cdot)$ and feature mapping $\phi(\cdot)$. The

Gaussian and Laplacian kernels are universal. It has been proven that,

$$\text{MMD}^2[\mathcal{F}, p, q] = \left\| \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \phi(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q(\mathbf{y})} \phi(\mathbf{y}) \right\|_{\mathcal{H}}^2. \tag{1.20}$$

An *unbiased* estimate of MMD is:

$$\text{MMD}[\mathcal{F}, X, Y] = \left(\frac{1}{|X|^2 - |X|} \sum_{i \neq j} h[i, j] \right)^{1/2}, \tag{1.21}$$

where $|X| = |Y|$ is assumed and

$$h[i, j] = k(\mathbf{x}_i, \mathbf{x}_j) + k(\mathbf{y}_i, \mathbf{y}_j) - k(\mathbf{x}_i, \mathbf{y}_j) - k(\mathbf{x}_j, \mathbf{y}_i). \tag{1.22}$$

For any two sets, we can use the unbiased estimate of MMD to measure their distance. However, it is noted that, when a set has only one feature vector, we cannot use MMD since its value always equals zero. In this case, one may use Euclidean metric, i.e., $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2$, or other metrics. A kernel function is required when to use MMD. There are choices for k. For example, the linear kernel which is simply a scalar product

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}, \tag{1.23}$$

and the Gaussian kernel is defined as

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma ||\mathbf{x} - \mathbf{y}||^2), \tag{1.24}$$

where γ is a parameter. Typically, γ is set to η^2, where η is the median of the L_2-distance between features in the set of images being considered: this means that the exponents are, on average, close to -1.

1.3 GENERAL FRAMEWORKS

In this section, we review two general frameworks used in SIP. It can be said that, most of the present state-of-the-arts base on the two frameworks. Both are proposed by Ker *et al.*, which can be found in references [5, 6, 18].

1.3.1 CLUSTERING-BASED DETECTION

Suppose that, multiple actors each will transmit multiple objects, all of which have been intercepted. Each actor may use multiple sources of objects. For simplicity, we assume that, each actor has just one source of objects, but that these sources could be different from each other. For each actor, we compute the feature vectors from the objects held by this actor. We can think of this as many clouds of points in the feature space, one cloud for each actor. A guilty actor is one who using steganography in (some of) their transmitted objects. We aim to identify guilty actors if their clouds of feature vectors stand out, in

some way, from the normal actors'.

Applying steganalysis to each object individually might be valuable, but it is likely that any guilty party could be lost in a crowd of false positives. If we use a trained model for the objects to be tested, it will produce the mismatching problem in practice. It is possible to perform clustering on the individual objects, which may provide little information. It inspires us to consider each actor's objects as a whole to characterize their source. In this way, we are to cluster the actors, hoping to separate an innocent majority from a guilty minority. Here, clustering the actors is equivalent to clustering the clouds mentioned above. Each cloud consists of a set of feature vectors.

Thus, we combine the hierarchical clustering with the MMD distance measure for SIP. The distances between actors are defined as the MMD distance between the sets of feature vectors. If the features have been well-chosen so that the difference between actors' sources is less than the difference between guilty and normal actors, the final agglomeration will be between two clusters, the normal and the guilty. Thus, we can extract a list of suspected guilty actors from the cluster dendrogram. As well as depending on the features, the accuracy of this method will also depend on the proportion of objects which guilty actors embed in, and the amount they embed. Such detector assumes that, the majority of actors is normal, and tries to identify guilty actors as an outlier cluster.

Pre-processing the directly extracted features is necessary to guarantee the accuracy of the detection method since the raw features have different scales. Feature normalization is a good choice and other preprocessing methods may be suitable as well such as principal component transformation. By normalization, we mean linear scaling of features such that each column of the data matrix $\widehat{\mathrm{X}}$ has zero mean and unit variance. This means that

$$\frac{1}{nl} \sum_{i=1}^{nl} \widehat{\mathrm{X}}_{ir} = 0, \forall r, \tag{1.25}$$

and

$$\frac{1}{nl} \sum_{i=1}^{nl} \widehat{\mathrm{X}}_{ir}^2 = 1, \forall r, \tag{1.26}$$

where n is the number of actors, and l for the number of objects held by each actor. The preprocessing enables the distance measure to be more meaningful and not significantly affected by noisy components.

Mathematically, let $A = \{a_1, a_2, ..., a_n\}$ and $S(a_i) = \{\mathbf{I}_1^{(i)}, \mathbf{I}_2^{(i)}, ..., \mathbf{I}_m^{(i)}\}$ respectively represent the actors and the objects held by actor a_i. Based on the aforementioned analysis, we describe the detailed steps for the (hierarchical) clustering based framework as follows.

1. For each actor a_i, $1 \leq i \leq n$, with a well-designed feature extractor $\mathcal{E}_{\mathrm{fea}}$,

MDS plot

MDS plot

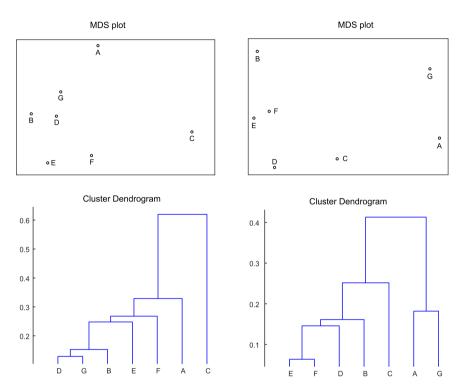Cluster Dendrogram

Cluster Dendrogram

Figure 1.4   Examples for MDS representation and cluster dendrogram.

we extract the feature vectors from the objects in $S(a_i)$, which is denoted by $F(a_i) = \{\mathbf{f}_1^{(i)}, \mathbf{f}_2^{(i)}, ..., \mathbf{f}_m^{(i)}\}$.

2. For each actor a_i, $1 \leq i \leq n$, normalizing $F(a_i)$ by the above method.

3. For any two different actors a_i and a_j, $1 \leq i < j \leq n$, determine the MMD distance between normalized $F(a_i)$ and normalized $F(a_j)$.

4. Apply hierarchical clustering, and collect two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ at the final stage of merging. The actors belonging to the cluster with a smaller size are considered as the guilty actors.

Assuming that, we want to find $k \leq |\mathcal{C}_1| + |\mathcal{C}_2|$ suspicious actors, and $|\mathcal{C}_1| \leq |\mathcal{C}_2|$. If $k \leq |\mathcal{C}_1|$, we randomly choose k actors from $\mathcal{C}_1$ as the guilty actors. Otherwise, all actors in $\mathcal{C}_1$ and $k - |\mathcal{C}_1|$ randomly-selected actors in $\mathcal{C}_2$ are judged as guilty ones. For better understanding, we take Figure 1.4 for explanation. In Figure 1.4, the above means MDS representation of actors, and the below shows cluster dendrograms, from which the final two clusters are encircled in the MDS plot. The left example implies $\mathcal{C}_1 = \{C\}$ and $\mathcal{C}_2 = \{A, B, D, E, F, G\}$. For the right example, we have $\mathcal{C}_1 = \{A, G\}$ and $\mathcal{C}_2 = \{B, C, D, E, F\}$. Suppose that, there has only one guilty actor. Then, for the

16 **CHAPTER 1** Steganographer Identification

Algorithm 2 LOF based detection method for SIP

Input: $A = \{a_1, ..., a_n\}$, $S(a_i) = \{\mathbf{I}_1^{(i)}, ..., \mathbf{I}_m^{(i)}\}$, $i \in [1, n]$.
Output: A ranking list $\mathbf{r}$. $\triangleright r_k \in \mathbf{r}$ reveals the k-th most suspicious actor.
 1: Extract feature vectors for each actor
 2: Preprocess feature vectors by normalization $\triangleright$ Other methods also work.
 3: Apply LOF to the n points (i.e., sets of normalized feature vectors)
 4: Return a ranking list $\mathbf{r}$ based on the LOF values $\triangleright \mathbf{r} = (r_1, r_2, ..., r_n)$.

---

left example, C will be judged as the suspicious actor. For the right example, either A or G will be judged as the steganographer. If there are three guilty actors, for the right example, {A, G} and an actor randomly chosen from $\mathcal{C}_2$ can be selected out as the guilty actors.

1.3.2 OUTLIER-BASED DETECTION

The advantage of the hierarchical clustering based framework is that, by pooling objects from each actor together, the method improves the signal-to-noise ratio compared with working on individual objects. This leads to better accuracy in the identification of guilty actor. The rationale for using a clustering algorithm was that the group of guilty actors should form a distinct cluster, that would be well separated from the normal actors. In reality, differences in actors' cover sources means that there is already variation in feature vectors between difference actors. The steganographic embedding may exhibit itself by a shift in the feature vectors in some direction(s). Consequently, the guilty actors could be represented by outliers, and thus allows us to switch our attention from clustering to outlier detection.

A lot of outlier detection methods can be found in the literature. The local outlier factor (LOF) method would be a good choice as: (a) it requires only the pairwise distances between points, (b) it is not tied to any particular application domain, (c) it directly provides a measure of how much an outlier each point is, and (d) it relies on single hyper-parameter. We use LOF for detection here. However, we point out that, it is always free for us to apply any effective outlier detection method for SIP.

We have introduced the concept of LOF in the previous section. Here, we directly present the pseudo-code of using LOF for SIP, which is shown in **Algorithm 2**. It is seen that, the steps are quite similar to that of the hierarchical clustering based framework. The LOF method does not provide a threshold, above which the element should be considered as an outlier. Such method has an advantage. Namely, the LOF values do not depend on the absolute values of distances, i.e., it is scale invariant. It is worth mentioning that, by switching to outlier detection, we have overcome another shortcoming of the clustering approach, which was the lack of any direct measure of "being

Table 1.1   Confusion matrix for identifying one guilty actor: 0.25 bpnc, 25%.

Guilty actor	Identified actor						
	A	B	C	D	E	F	G
A	100	0	0	0	0	0	0
B	0	100	0	0	0	0	0
C	0	0	100	0	0	0	0
D	0	1	5	94	0	0	0
E	0	9	7	0	84	0	0
F	0	9	14	0	0	77	0
G	0	16	7	0	0	0	77

an outlier". If we know that we want to identify a single guilty, we can take the one with the largest LOF value; if we want to identify a short list of the k most suspicious actors, we can choose those with top-k LOF values.

1.3.3 PERFORMANCE EVALUATION AND ANALYSIS

We report experimental results in [5, 18] for performance analysis.

1.3.3.1 Clustering-based Detection

Suppose that, there are n actors, each of whom has m images, and we want to identify a single guilty actor from them. The actors are simulated by images taken from n different cameras, all are JPEG compressed with an identical quality factor (QF). nsF5 [19] is used as the steganographic algorithm, which is an improved version of F5 [20]. F5 is a steganographic algorithm that aims to preserve the shape of the histogram of quantized DCT coefficients, making it similar to that of the cover image. The message is embedded by changing the absolute values of DCT coefficients toward zero. F5 is also the first algorithm to use matrix embedding, which is a coding scheme that increases embedding efficiency measured by the number of bits embedded per embedding change. As its variant, nsF5 uses the same type of embedding changes as F5. To avoid introducing more zeros, however, nsF5 uses wet paper codes [21, 22] with improved efficiency.

Each image is represented by a 274-D feature vector called PEV-274, designed for JPEG steganalysis and previously shown to be effective against nsF5. However, it is admitted that, nsF5 is detectable by modern steganalysis features. The hierarchical clustering technique with MMD distance between actors is used. The expectation is that the final agglomeration should be between one guilty actor and a cluster of $n - 1$ normal actors. It is pointed that, there is no training in the paradigm, and it does not assume any knowledge of the actors' cameras or the embedding algorithm.

Table 1.2   Confusion matrix for identifying one guilty actor: 0.3 bpnc, 30%.

Guilty actor	Identified actor						
	A	B	C	D	E	F	G
A	100	0	0	0	0	0	0
B	0	100	0	0	0	0	0
C	0	0	100	0	0	0	0
D	0	0	0	100	0	0	0
E	0	0	0	0	100	0	0
F	0	0	1	0	0	99	0
G	0	0	0	0	0	0	100

There are $n = 7$ actors sending JPEG images. Each actor converts the RAW photos to JPEG QF=80 before transmitting them. Each actor uses images of a constant size (the default resolution of the camera), but the image size does vary from actor to actor. Tables 1.1 and 1.2 show the confusion matrices for the identification of one guilty actor, with each experiment repeated 100 times per guilty actor. Each actor transmits $m = 50$ JPEG images: Table 1.1, the guilty actor embeds 0.25 bits per non-zero coefficient (bpnc) in 25% of his images; Table 1.2, the guilty actor embeds 0.3 bpnc in 30% of his images. The single linkage and linear MMD are used here. The identification of the guilty party is performed by cutting the dendrogram at the final agglomeration, which has been described previously. It can be observed that, the overall accuracy are 90.3% and 99.9%, respectively. Performance falls of sharply for smaller payloads since the guilty actors fade into the normal cluster and they cannot be identified accurately.

Experiments are conducted with both linear and Gaussian MMD. Table 1.3 shows the experimental results due to different parameter settings. It can be found that, the clustering based on linear MMD worked better than that based on Gaussian MMD. Also, the choice of agglomeration algorithm did not make a substantial difference. As expected, there is a dependency on m: more evidence allows the detector to make more accurate detection from smaller payloads. When $m = 200$, the linear MMD combined with any linkage achieved perfectly accurate detection of the guilty parity with payloads as small as 0.3 bpnc embedded in only 30% of the images.

1.3.3.2 Outlier-based Detection

Compared to clustering-based detection, outlier-based detection allows us to consider up to thousands actors each transmitting up to hundreds images. In [18], in each experiment, a randomly selected subset of 4000 actors and 200 images for each actor was used. The images were obtained from a social network site, which automatically resized large images, to approximately 1Mpix,

Table 1.3   Overall accuracy due to different parameter settings.

			Guilty actors proportion of stego images @ payload per image			
MMD kernel	**Cluster linkage**	m	10% @ 0.1 bpnc	30% @ 0.3 bpnc	50% @ 0.5 bpnc	70% @ 0.7 bpnc
Linear	Single	200	21.3%	100.0%	100.0%	100.0%
Linear	Single	100	18.0%	100.0%	100.0%	100.0%
Linear	Single	50	17.7%	99.9%	100.0%	100.0%
Linear	Single	20	14.1%	96.7%	100.0%	100.0%
Linear	Single	10	15.9%	86.4%	99.9%	100.0%
Linear	Centroid	200	17.0%	100.0%	100.0%	100.0%
Linear	Centroid	100	16.6%	100.0%	100.0%	100.0%
Linear	Centroid	50	16.3%	100.0%	100.0%	100.0%
Linear	Centroid	20	13.4%	97.0%	100.0%	100.0%
Linear	Centroid	10	15.4%	87.1%	99.9%	100.0%
Linear	Average	200	17.0%	100.0%	100.0%	100.0%
Linear	Average	100	16.4%	100.0%	100.0%	100.0%
Linear	Average	50	15.7%	100.0%	100.0%	100.0%
Linear	Average	20	14.3%	97.0%	100.0%	100.0%
Linear	Average	10	16.6%	87.1%	99.9%	100.0%
Linear	Complete	200	15.4%	100.0%	100.0%	100.0%
Linear	Complete	100	14.4%	100.0%	100.0%	100.0%
Linear	Complete	50	13.9%	99.3%	100.0%	100.0%
Linear	Complete	20	15.6%	94.9%	100.0%	100.0%
Linear	Complete	10	16.6%	84.0%	99.9%	100.0%
Gaussian	Centroid	200	14.0%	92.0%	100.0%	100.0%
Gaussian	Centroid	100	13.7%	87.4%	100.0%	100.0%
Gaussian	Centroid	50	12.6%	76.1%	100.0%	100.0%
Gaussian	Centroid	20	12.9%	60.1%	99.7%	100.0%
Gaussian	Centroid	10	15.3%	40.9%	98.4%	100.0%

and then JPEG compressed with QF = 85, at the time of crawling. Exactly one guilty actor was simulated by nsF5. Four different embedding strategies, i.e., max-greedy, max-random, linear and even, were used to insert Lp bits to the images held by the guilty actor. L is the total number of non-zero coefficients in the images. p is the number of bits per non-zero coefficient. The PEV-274 features were extracted from the nm JPEG images, and then normalized. The MMD distance between each pair of actors can be then determined. Finally, the LOF detection method was applied to compute the LOF score for each actor, which allowed us to identify the most suspicious actor that will be judged as the guilty actor.
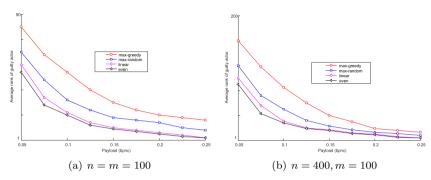
(a) $n = m = 100$ (b) $n = 400, m = 100$

Figure 1.5   Detection performance due to different parameters.

For each combination of parameters, each experiment was repeated 500 times with a different selection of actors and guilty actor. The *average rank of the guilty actor* was used as a benchmark to reflect how well the guilty actor was identified. An average rank of one corresponds to perfect detection, i.e., the guilty actor is always ranked most suspicious. And, an average rank of $\frac{n+1}{2}$ corresponds to random guessing. We do not expect a universal, unsupervised detector to achieve perfect accuracy, but instead hope that, it provides intelligence by ranking a truly guilty actor among the top 10% of all actors. In addition, for the MMD distance, the centroid 'kernel' (a variant of linear kernel, please refer to the Appendix of [18]) was used. And, for the LOF detection, a parameter k was set to 10. Figure 1.5 show the detection performance due to different parameters. It can be observed that, the max-greedy strategy is the most secure for the embedder: the average rank of the guilty actor is higher in all payloads. The second most secure strategy is max-random. The linear strategy is next and even is the most insecure. The reason that the max-greedy/max-random strategy is more secure than linear/even has been explained in detail in [10]. In brief, this effect is caused by whitening the features in the pre-processing stage. Overall, with relatively large payloads, the LOF method achieves efficient detection results.

1.4 ENSEMBLE AND DIMENSIONALITY REDUCTION

1.4.1 CLUSTERING ENSEMBLE

The clustering ensemble method combines the detection results from multiple clustering sub-models. As shown in Figure 1.6, the clustering ensemble framework is comprised of three parts: preprocessing, feature extraction, and clustering ensemble. The preprocessing procedure requires us to analyze the characteristics of the steganographic algorithm so that those sensitive components
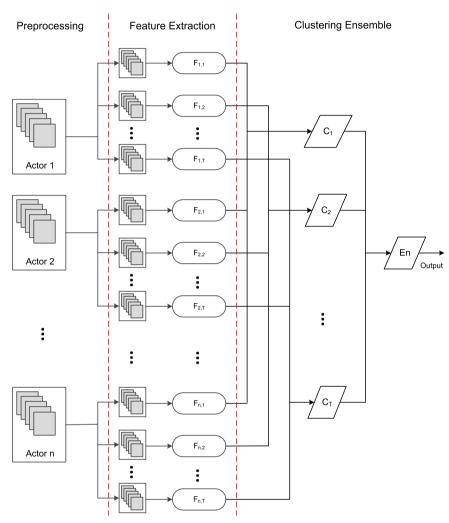
Figure 1.6   General framework for clustering ensemble.

concealing the steganographic traces can be separated out from the stego. The preprocessing could be cropping, high-pass filtering, and other operations that can improve the statistical discriminability between the steganographic noise and the cover content. It can be said that, the goal of preprocessing is to enlarge the "signal-to-noise" ratio.

After preprocessing, we extract steganalysis features from the preprocessed objects, which reduces the preprocessed objects to vectors with a small dimension. We use T clustering sub-models $\{C_1, C_2, ..., C_T\}$ for clustering ensemble. For each sub-model C_i, we can directly apply the hierarchical clustering

method, by which we can collect suspicious actor(s) that will be judged as the steganographer candidate(s). By merging the results from these sub-models, the index value(s) indicating the most suspicious actor(s) that will be chosen as the steganographer(s), can be determined.

For better understanding, we review Li *et al.*'s work [9], which is a related work exploiting clustering ensemble for SIP. In their work, the images held by actors are randomly cropped with smaller size to build actor subsets. Then, the LI-250 features mentioned in the aforementioned section are extracted from these actor subsets. After performing the agglomerative clustering in the (normalized) feature space, a most suspicious actor can be separated from the innocent ones. By repeating these steps, multiple decisions are made, and the final guilty actor can be identified by majority voting. It is mentioned that, in Li *et al.*'s work, the clustering operation is slightly different from the above basic agglomerative clustering algorithm, and they use Euclidean distance for two feature vectors. We detail the algorithm below.

Denote a set of n actors as $\{A_1, A_2, ..., A_n\}$, which include one guilty actor and $n-1$ normal (innocent) actors, each of whom transmits m JPEG images. Clustering ensemble is used to identify the steganographer according to the following steps.

1. Denote the image sets of n actors as $\{I_1, I_2, ..., I_n\}$. Each I_i $(1 \leq i \leq n)$ contains m JPEG images of size $M \times N$. Randomly crop each JPEG image in I_i $(1 \leq i \leq n)$ to size $m' \times n'$ $(m' < M, n' < N)$. The cropped images consist of n subsets $I_1', I_2', ..., I_n'$, where I_i' $(1 \leq i \leq n)$ includes m JPEG images of size $m' \times n'$.

2. For each image subset I_i' $(1 \leq i \leq n)$, extract the LI-250 features to form feature sets F_i $(1 \leq i \leq n)$, where $F_i = (f_i^{(1)}, f_i^{(2)}, ..., f_i^{(m)})$ and each $f_i^{(j)}$ represents a 250-D feature vector. Notice that, each F_i is associated with a second subscript j $(1 \leq j \leq T)$, i.e., $F_{i,j}$, to denote F_i used in the j-th clustering sub-model since we have T sub-models.

3. Normalize the feature sets $\{F_1, F_2, ..., F_n\}$ such that every column of feature matrix has zero mean and unit variance. Then, the distance between different pairs of feature sets can be calculated with

$$D(F_i, F_j) = \frac{1}{m^2} \sum_{p=1}^{m} \sum_{q=1}^{m} d(f_i^{(p)}, f_j^{(q)}), i, j \in [1, n], \qquad (1.27)$$

where $d(f_i^{(p)}, f_j^{(q)})$ represents the Euclidean distance between the two feature vectors.

4. Perform hierarchical clustering with the above distance measure. First, two actors with the minimum distance are combined into a new cluster, denoted by $\mathcal{X}$, and the remaining actors are in a set denoted by $\mathcal{Y}$. We repeat to select an actor from $\mathcal{Y}$ and add the selected actor into $\mathcal{X}$ until only one actor is left in $\mathcal{Y}$. At each selection step, the actor $Y \in \mathcal{Y}$ who

has the smallest distance to $\mathcal{X}$ is selected, where the distance between an actor $Y \in \mathcal{Y}$ and a cluster $\mathcal{X}$ is defined as:

$$D(Y, \mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} D(Y, X). \qquad (1.28)$$

The last remaining actor in $\mathcal{Y}$ is considered a suspicious actor.

5. Repeat Steps 1-4 for T times. Each time, we identify a suspicious actor, denoted by C_i, $i = 1, 2, ..., T$. By majority voting with T sub-models, the actor who is identified as suspicious the most frequently is determined as the final culprit. When there is a tie, randomly select one to break the tie.

The detection performance is evaluated by the overall identification accuracy rate, calculated as the ratio between the number of correctly detected steganographic actors and the selected total number of steganographic actors. In Li *et al*'s method, the original images are cropped to images with a smaller size. Li *et al.* point that, there is generally an optimal range for m' ($= n'$), namely, when the value of m' is far away from the optimal range, the detection performance will become worse. This interesting phenomenon is explained as: When the size of the cropped images is too large, there is not much diversity in the randomly cropped images, and as such, each sub-clustering tends to generate same result. The benefit of clustering ensemble may disappear. When the size of the cropped images is too small, the statistical features of the images become unstable, leading to poor detection in each sub-clustering.

Experiments reported in [9] show that LI-250 outperforms pervious popular JPEG steganalysis features PEV-274 [7], LIU-144 [23], DCTR-8000 [24], and PHARM-12600 [25], which is explained as: on the one hand, comparing with PEV-274 and LIU-144, LI-250 could well reveal the changes of cover elements caused by the tested steganographic algorithms; on the other hand, although DCTR-8000 and PHARM-12600 are conclusively more sensitive for supervised binary classification, they have a high dimension and contain a number of weak features, which may lead to worse performance for SIP [6, 26].

Usually, for single hierarchical clustering, better performance can be obtained when the images are with large sizes [27]. However, good performance with large image size in the single hierarchical clustering method does not imply good performance of clustering ensemble using the same large image size for Li *et al.*'s method. That is, when the size of cropped images is too large, the diversity in the randomly cropped images is reduced, and thus multiple clustering rounds are likely to generate the same result. The benefit of clustering ensemble disappears. Therefore, it can be said that, the preprocessing operation may have significant impact on the detection performance.

Intuitively, the higher T, the higher the overall accuracy, but the longer the running time. In [9], the overall accuracy is tending towards stability with an increasing T, and does not change seriously when T is larger than a threshold. It is believed that, the cropped images have a lot of overlapped area

when T increases, while the diversity will be reduced implying that multiple sub-models may generate the same results.

It can be easily inferred that the clustering ensemble framework shown in Figure 1.6 can be applied to other detection methods (e.g., LOF-based), by making adjustments accordingly. We do not provide more discussion here.

1.4.2 DIMENSIONALITY REDUCTION

Different from clustering ensemble (which has good generalization in terms of technical framework), dimensionality reduction focuses on the process of reducing the number of random variables under consideration by obtaining a set of principal variables. There are two important approaches for dimensionality reduction: feature selection and feature projection.

1.4.2.1 Feature Selection

Feature selection attempts to select a feature subspace from the original full feature space. Feature selection is often used in domains where there are many features and comparatively few sample points. Feature selection simplifies the model, and thus results in a shorter running time. It can also avoid disasters due to the dimensional problem. For SIP, in mainstream works, the distances in high dimensional space between features of actors are determined to find the abnormal cluster or outlier corresponding to the steganographer. However, in the high dimensional space, the distances between feature points may become similar to each other, resulting in that, it is not easy to separate the abnormal points from the normal ones. To this end, Wu [28] uses a simple feature selection based ensemble algorithm for SIP. The method merges results from multiple detection sub-models, each of which feature space is randomly sampled from the raw full dimensional space. Unlike the basic framework, the images held by a single actor are divided into multiple disjoint sets, resulting in that, each actor is represented by multiple sets of feature vectors.

In detail, let $A = \{a_1, a_2, ..., a_n\}$ and $S(a_i) = \{\mathbf{I}_1^{(i)}, \mathbf{I}_2^{(i)}, ..., \mathbf{I}_m^{(i)}\}$ respectively represent actors and the images held by actor a_i. A detector computes the *preprocessed* features for each a_i, i.e., $F(a_i) = \{\mathbf{f}_1^{(i)}, \mathbf{f}_2^{(i)}, ..., \mathbf{f}_m^{(i)}\}$. All $F(a_i)$ $(1 \leq i \leq n)$ are divided to disjoint sets with an identical size, i.e.,

$$P(a_i) = \bigcup_{j=1}^{p} P_j(a_i), (1 \leq i \leq n), \tag{1.29}$$

where $m = p \cdot q$ and $P_j(a_i) = \{\mathbf{f}_{jq-q+1}^{(i)}, \mathbf{f}_{jq-q+2}^{(i)}, ..., \mathbf{f}_{jq}^{(i)}\}$.

Accordingly, each a_i can be represented by p set of feature vectors. We call the p sets as "p points". We can collect a total of $p \cdot n$ points, each of which belongs to one of the n actors. It is naturally assumed that, distances between an abnormal point and a normal point should be larger than that between two

normal points. In other words, normal points are densely distributed while abnormal ones are sparsely distributed. For any two points, we can use MMD or other metrics to measure their distance. However, when a point has only one feature vector, we cannot use MMD since its value always equals zero. In this case, one may adopt the Euclidean distance or other suitable metrics. Therefore, by anomaly detection (e.g., LOF detection), a ranking list for the pn points is determined according to their anomaly scores.

We use pn triples $\{(u_i, v_i, w_i)\}_{i=1}^{pn}$ to denote the sorted information, where $u_1 \geq u_2 \geq ... \geq u_{pn}$ represent the anomaly scores. v_i denotes the corresponding actor and w_i is the point index, namely, we have $P_{w_i}(v_i) \in P(v_i)$. For each actor a_i, we can determine a fusion score below:

$$s(a_i) = \sum_{j=1}^{pn} \frac{(pn + 1 - j) \cdot \delta(v_j, a_i)}{p}, (1 \leq i \leq n), \tag{1.30}$$

where $\delta(x, y) = 1$ if $x = y$, otherwise $\delta(x, y) = 0$. By sorting the fusion scores, we can generate the final ranking list, where the actor with the largest score will be the most suspicious.

In this way, we can construct a single anomaly detection system operated on the full feature space. Obviously, we can build T sub-models $\{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_T\}$, whose feature dimensions are $\{d_1, d_2, ..., d_T\}$. Each d_i $(1 \leq i \leq T)$, is chosen from $[H/2, H - 1]$, where H is the dimension of the raw full feature space. It is seen that, each sub-model is exactly the same as the single anomaly detection system, except that the used features are randomly sampled from the full feature space. For each $\mathcal{M}_i$, we can collect a ranking list, denoted by $\mathbf{r}_i = \{r_{i,1}, ..., r_{i,n}\}$, and $r_{i,j}$ means the actor with the j-th largest anomaly score, i.e., $r_{i,1}$ is the most suspicious and $r_{i,n}$ is the least suspicious. Accordingly, by further processing $\{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_T\}$, the final fusion score for a_i can be:

$$s_{\mathrm{F}}(a_i) = \sum_{j=1}^{T} \frac{n + 1 - \sum_{k=1}^{n}[k \cdot \delta(r_{j,k}, a_i)]}{T}. \tag{1.31}$$

By sorting $\{s_{\mathrm{F}}(a_1), ..., s_{\mathrm{F}}(a_n)\}$, the actor with the largest score will be the most suspicious, and the smallest score corresponds to the least suspicious.

It is observed that the method has the ability to improve the performance, which, however, is not significant and could be affected by parameter settings. For performance optimization, one may design advanced feature selection algorithms for choosing efficient feature components for detection.

1.4.2.2 Feature Projection

Feature projection should be distinguished from feature selection. While feature selection returns a subset of the original features by somehow way, feature projection creates new features from functions of the original features. It is excepted that, the projections of the original high dimensional features exhibit

maximal information about the class label. Same as the feature selection, feature projection also reduces the complexity to help avoid over-fitting. For feature projection, the feature transformation can be either linear or non-linear. E.g., the linear principle components analysis (PCA) [29] performs a linear mapping of the raw features to a lower-dimensional space such that the variance of the features in the lower-dimensional representation is maximized. Kernel PCA [30], as a non-linear extension of PCA, uses a kernel to enable the originally linear operations of PCA to be performed in a RKHS.

Pevný and Ker [26] have provided profound study on feature projection for SIP. They limit themselves to linear feature projections because it is believed that the steganalysis problem is essentially linear "in some sense", e.g., experiments in [31] demonstrate that accurate estimators of payload size can be created as linear functions of feature vectors. As pointed by [26], we cannot expect that all stego objects move in the same direction, certainly not when created with direrent stego algorithms, nor that cover objects begin from nearby points if they arise from different sources. The boundary between cover and stego objects may be nonlinear. Accordingly, a good set of condensed features will be a collection of different linear projections, which between them capture the behaviour of different cover objects, sources, and embedding algorithms.

Therefore, to achieve better performance, they exploit principal component transformation (PCT), maximum covariance transformation, ordinary least square regression, and calibrated least squares regression for feature projection. The PCT can be defined as an iterative algorithm, where in k-th iteration one seeks a projection vector $w_k \in \mathbb{R}^d$ best explaining the raw features and being orthogonal to all projections $\{w_i\}_{i=1}^{k-1}$. It is formulated as:

$$w_k = \arg\max_{||w||=1} \quad w^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}w, \tag{1.32}$$

subject to $w^{\mathrm{T}}w_i = 0, \forall i \in [1, k-1]$. w_k is k-th eigenvector of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, assuming that the vectors are sorted such that the eigenvalue sequence is non-increasing.

A weakness of PCT is that, it does not take into account the objective function. Pevný and Ker define the object function as the presence of steganography signalled by the steganographic change rate. It is to some extent resolved by finding a direction maximizing the covariance between the projected data $\mathbf{X}^s w$ and the dependent variable $\mathbf{Y}^s$. Also, w_k should be orthogonal to $\{w_i\}_{i=1}^{k-1}$. It is called the Maximum CoVariance (MCV) transformation method, which can be formulated as:

$$w_k = \arg\max_{||w||=1} \quad \mathbf{Y}^{s\mathrm{T}}\mathbf{X}^s w, \tag{1.33}$$

subject to $w^{\mathrm{T}}w_i = 0, \forall i \in [1, k-1]$. The analytical solution is $w_k = \mathbf{Y}^{s\mathrm{T}}\mathbf{X}_k^s$, where $\mathbf{X}_k^s = \mathbf{X}_{k-1}^s(\mathbf{I} - w_{k-1}w_{k-1}^{\mathrm{T}})$, and $\mathbf{X}_1^s = \mathbf{X}^s$.

Ordinary Least Square regression (OLS) finds a single direction w minimiz-

ing the total square error between $\mathbf{X}^s w$ and $\mathbf{Y}^s$. Such optimization problem can be formulated as:

$$w = \arg\min_{w \in \mathbb{R}^d} \quad 2\mathbf{Y}^{s\mathrm{T}}\mathbf{X}^s w - w^{\mathrm{T}}\mathbf{X}^{s\mathrm{T}}\mathbf{X}^s w, \tag{1.34}$$

the analytical solution of which is:

$$w = (\mathbf{X}^{s\mathrm{T}}\mathbf{X}^s)^{-1}\mathbf{X}^{s\mathrm{T}}\mathbf{Y}^s, \tag{1.35}$$

which assumes that, $\mathbf{X}^{s\mathrm{T}}\mathbf{X}^s$ is regular and the inversion is numerically stable. In practice, this is not always true, and steganalysis features may lead to a nearly singular matrix. To alleviate this, a small diagonal matrix is added to prevent non-singularity and increase the stability of the solution:

$$w = (\mathbf{X}^{s\mathrm{T}}\mathbf{X}^s + \lambda\mathbf{I})^{-1}\mathbf{X}^{s\mathrm{T}}\mathbf{Y}^s. \tag{1.36}$$

Good steganographic projections should be sensitive to embedding changes yet insensitive to the object content, meaning that, the covariance between the projection of stego features and their embedding change rate should be high, while the variance of projection of cover features should be low. This objective is not optimized in any of the above methods. To this end, Pevný and Ker propose a so-called *calibrated least squares* (CLS) regression method, which finds the projections by iteratively solving:

$$w = \arg\min_{w \in \mathbb{R}^d} \quad 2\mathbf{Y}^{s\mathrm{T}}\mathbf{X}^s w - w^{\mathrm{T}}\mathbf{X}^{c\mathrm{T}}\mathbf{X}^c w - \lambda||w||^2, \tag{1.37}$$

subject to $w^{\mathrm{T}}w_i = 0, \forall i \in [1, k-1]$. And, the analytical solution is:

$$w_k = (\mathbf{X}_k^{c\mathrm{T}}\mathbf{X}_k^c + \lambda\mathbf{I})^{-1}\mathbf{X}_k^{s\mathrm{T}}\mathbf{Y}^s, \tag{1.38}$$

where $\mathbf{X}_k^s = \mathbf{X}_{k-1}^s(\mathbf{I} - w_{k-1}w_{k-1}^{\mathrm{T}})$, $\mathbf{X}_1^s = \mathbf{X}^s$, and $\mathbf{X}_k^c = \mathbf{X}_{k-1}^c(\mathbf{I} - w_{k-1}w_{k-1}^{\mathrm{T}})$, $\mathbf{X}_1^c = \mathbf{X}^c$.

Instead of PEV-274, Pevný and Ker use CF-7850 [32] and outlier detection [6] for experiments. Experimental results show that, in terms of identification accuracy (i.e., the average rank of the only one guilty actor), the unsupervised PCT method is worst. The MCV method is inferior to OLS in most situations, and CLS is significantly superior to the others. The explanation can be described as follows: PCT focuses on the explanation of the variance in the data, which is not exactly on par with our goal, as the variance in features can be dominated by cover image content. MCV finds directions maximally correlated with the explained variable (change rate). Although this is better than PCT, the image content acting as noise is not suppressed. OLS maximizes the covariance of a projection with the payload and minimizes the variance of the projection. Since the covariance and variance of projections are measured on the same data, it is not clear whether the variance comes from image content or from embedding changes, whereas, CLS removes this ambiguity.

1.5 CONCLUSION AND FURTHER RESEARCH

In this chapter, we have reviewed the problem of steganographer identification, in which we aim to identify a steganographer who sends many images (some of them may be innocent) in social networks or public environment of many other innocent users. The analyzer must deal with multiple users and multiple images per user, and particularly the differences between cover sources, as well as the differences between the covers and the stegos.

In despite of being posed for more than ten years, this problem was first addressed by Ker and Pevný [5] in 2011. They introduced two general technical frameworks for steganographer identification, i.e., clustering-based detection and outlier-based detection. It can be said that, most of the advanced works reported in the literature are based on them. Though, throughout this chapter, we consider only one guilty actor (i.e., the steganographer), we can actually identify a short list of k most suspicious actors.

It has been shown that, both ensemble and dimensionality reduction can improve the identification performance. As a common operation used in steganalysis, ensemble merges results from multiple sub-models to make a final decision. Dimensionality reduction, seeming to be more important, exploits salient features and eliminates irrelevant feature fluctuations by representing the discriminative information in a lower dimensional manifold [33].

The outlier-based analyzer is not compatible with rich features containing lots of weak features, because the weak features contain lots of noise (caused by cover content), resulting in inferior performance. Pevný and Ker [26] used supervised feature reduction methods to improve performance, meaning that, the complete outlier detector was no longer unsupervised. However, the features showed a high level of robustness to varying the embedding algorithm, meaning that the outlier detector retained its universal behaviour. Finding the balance between supervision (to focus information) and universality (for robustness) is an important challenge for pooled steganalysis for SIP, which needs further investment.

Building upon advances in machine learning, Pevný and Nikolaev [34] attempted to learn an optimal pooling function for pooled steganalysis. Although experiments show that learned combining functions are superior to the prior art, many interesting phenomenons were found, pointing to directions of further research, e.g., are there better strategies for the steganographer to distribute the message into multiple objects, or the linear strategy is the best he can do despite the steganalyzer knowing it? It is believed that, more questions will arise as the study of SIP moves ahead [35].

BIBLIOGRAPHY

[1] J. Fridrich. Steganography in digital media: principles, algorithms, and applications. *Cambridge University Press*, 2010.

[2] T. Filler, and J. Fridrich. Gibbs construction in steganography. *IEEE Transactions on Information Forensics and Security*, 5(4): 705-720, 2010.

[3] T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3): 920-935, 2011.

[4] A. Ker. Batch steganography and pooled steganalysis. *Proc. Information Hiding Workshop*, vol. 4437, pp. 265-281, 2006.

[5] A. Ker, and T. Pevný. A new paradigm for steganalysis via clustering. *Proc. SPIE, Media Watermarking, Security, and Forensics XIII*, vol. 7880, pp. 0U01-0U13, 2011.

[6] A. Ker, and T. Pevný. Identifying a steganographer in realistic and heterogeneous data sets. *Proc. SPIE, Media Watermarking, Security, and Forensics XIV*, vol. 8303, pp. 0N01-0N13, 2012.

[7] T. Pevný, and J. Fridrich. Merging markov and DCT features for multiclass JPEG steganalysis. *Proc. SPIE, Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6065, pp. 0301-0314, 2007.

[8] Y. Shi, C. Chen, and W. Chen. A markov process based approach to effective attacking JPEG steganography. *Proc. Information Hiding Workshop*, vol. 4437, pp. 249-264, 2006.

[9] F. Li, K. Wu, J. Lei, M. Wen, Z. Bi, and C. Gu. Steganalysis over large-scale social networks with high-order joint features and clustering ensembles. *IEEE Transactions on Information Forensics and Security*, 11(2): 344-357, 2016.

[10] A. Ker, and T. Pevný. Batch steganography in the real world. *Proc. Multimedia and Security Workshop*, pp. 1-10, 2012.

[11] A. Ker. Batch steganography and the threshold game. *Proc. SPIE, Security, Steganography, and Watermarking of of Multimedia Contents IX*, vol. 6505, pp. 0401-0413, 2007.

[12] A. Ker. A capacity result for batch steganography. *IEEE Signal Processing Letters*, 14(8): 525-528, 2007.

[13] L. Rokach, and O. Maimon. Clustering methods. *Data Mining and Knowledge Discovery Handbook*, pp. 321-352, 2005.

[14] M. Breunig, H. Kriegel, R. Ng, and J. Sander. LOF: identifying density-based local outliers. *ACM SIGMOD International Conference on Management of Data*, pp. 93-104, 2000.

[15] L. Song. Learning via hilbert space embedding of distributions. PhD Thesis, University of Sydney, 2008.

[16] A. Gretton, K. Borgwardt, M. Rasch, B. Scholkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, vol. 13, pp. 723-773, 2012.

[17] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Scholkopf. Kernel mean embedding of distributions: a review and beyond. *Foundations and Trends in Machine Learning*, vol. 10, no. 1-2, pp. 1-141, 2017.

[18] A. Ker, and T. Pevný. The steganographer is the outlier: realistic large-scale steganalysis. *IEEE Transactions on Information Forensics and Security*, 9(9): 1424-1435, 2014.

[19] J. Fridrich, T. Pevný, and J. Kodovsky. Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities. *Proc. ACM Multimedia and Security*

29

30 **CHAPTER 1** Steganographer Identification

Workshop, pp. 314, 2007.

[20] A. Westfeld. F5: A steganographic algorithm. *Proc. Information Hiding Workshop*, pp. 289-302, 2001.

[21] J. Fridrich, M. Goljan, and D. Soukal. Efficient wet paper codes. *Proc. Information Hiding Workshop*, pp. 204-218, 2005.

[22] J. Fridrich, M. Goljan, and D. Soukal. Wet paper codes with improved embedding efficiency. *IEEE Transactions on Information Forensics and Security*, 1(1): pp. 102-110, 2006.

[23] Q. Liu, and Z. Chen. Improved approaches with calibrated neighboring joint density to steganalysis and seam-carved forgery detection in JPEG images. *ACM Transactions on Intelligent Systems and Technology*, 5(4): 1-30, 2015. (Artile ID 63.)

[24] V. Holub, and J. Fridrich. Low-complexity features for JPEG steganalysis using undecimated DCT. *IEEE Transactions on Information Forensics and Security*, 10(2): 219-228, 2015.

[25] V. Holub, and J. Fridrich. Phase-aware projection model for steganalysis of JPEG images. *Proc. SPIE, Meida Watermarking, Security, and Forennsics*, vol. 9049, pp. 0T01-0T11, 2015.

[26] T. Pevný, and A. Ker. The challenges of rich features in universal steganalysis. *Proc. SPIE, Media Watermarking, Security, and Forensics*, vol. 8665, pp. 0M01-0M15, 2013.

[27] T. Filler, A. Ker, and J. Fridrich. The square root law of steganographic capacity for Markov covers. *Proc. SPIE, Media Watermarking, Security, and Forensics*, vol. 7254, pp. 0801-0811, 2009.

[28] H. Wu. Feature bagging for steganography identification. *arXiv:1810.11973*, 2018.

[29] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11): 559-572, 1901.

[30] B. Scholkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5): 1299-1319, 1998.

[31] T. Pevný, J. Fridrich, and A. Ker. From blind to quantitative steganalysis. *IEEE Transactions on Information Forensics and Security*, 7(2): 445-454, 2012.

[32] J. Kodovsky, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2): 432-444, 2012.

[33] D. Erdogmus, U. Ozertem, and T. Lan. Information theoretic feature selection and projection. In: Prasad B., Prasanna S.R.M. (eds) Speech, Audio, Image and Biomedical Signal Processing using Neural Networks, vol. 83, Springer, 2008.

[34] T. Pevný, and I. Nikolaev. Optimizing pooling function for pooled steganalysis. *IEEE International Workshop on Information Forensics and Security*, pp. 1-6, 2015.

[35] A. Ker, P. Bas, R. Bohme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevný. Moving steganography and steganalysis from the laboratory into the real world. *ACM Workshop on Information Hiding and Multimedia Security*, pp. 45-58, 2013.