Manifold alignment approach to cover source mismatch in steganalysis

Daniel Lerch-Hostalot
Universitat Oberta de Catalunya,
Internet Interdisciplinary Institute (IN3),
Estudis d'Informàtica, Multimèdia i Telecomunicació,
Av. Carl Friedrich Gauss, 5,
08860 Castelldefels (Barcelona)
Email: dlerch@uoc.edu

David Megías
Universitat Oberta de Catalunya,
Internet Interdisciplinary Institute (IN3),
Estudis d'Informàtica, Multimèdia i Telecomunicació,
Av. Carl Friedrich Gauss, 5,
08860 Castelldefels (Barcelona)
Email: dmegias@uoc.edu

***Abstract*—Cover source mismatch (CSM) is an important open problem in steganalysis. This problem, known as domain adaptation in the field of machine learning, deals with the decrease in the classification accuracy when a classifier is moved from the laboratory into the real world. In this paper, we present an approach to CSM based on domain adaptation using manifold alignment algorithms. In this novel approach, we use manifold alignment to find a latent space where the two datasets (the one used for training and the one used for testing) have a common representation. We show that manifold alignment can significantly increase the accuracy of the classifier in cross-domain classification.**

***Index Terms*—Steganalysis, Cover source mismatch, Domain adaptation, Manifold alignment, Machine learning**

I. Introduction

Steganography is a collection of techniques for hiding data into apparently innocent objects with the aim of establing secret communications. Nowadays, these objects are usually digital media. One of the most common carriers are images because of their widespread use in the Internet.

On the other hand, steganalysis refers to different techniques aiming at detecting whether a cover source has been modified to hide data. The most successful techniques for carrying out steganalysis are based on *machine learning* [20], [7], [14]. The usual methodology implies preparing a training set formed by true covers (referred to as "the *cover* set") and covers that are embedded with hidden information (referred to as "the *stego* set"). This training set is used for training a classifier that is applied later on to test sets of unknown media and determine if they are cover or stego.

Steganalysis based on machine learning is usually successful in laboratory conditions, i.e. if it is assumed that we have access to a set of media of the same type as the set of media used by the steganographer. However, in the real world, the set of media used by the steganographer is not of the same type used for training the classifier [13]. In the case of images, for example, the test set may be taken with a different camera and with different properties of the cover, such as size, lighting conditions, and so on. Therefore, the domain of the training set is usually different from the domain of the testing set, and the classifier will not usually perform appropriately in the latter. In machine learning literature, this problem is known as the problem *domain adaptation* [17], [2], whereas, in steganalyis, this situation is referred to as *cover source mismatch* (CSM), being an important open problem in the field [13].

This paper stems from a family of algorithms known as *manifold alignment* [26] used in a subfield of machine learning called *manifold learning* [24], [22], [16]. This concept was first introduced in [9] adding a manifold constraint to the problem of correlating sets of high-dimensional vectors [10]. This initial idea has evolved in subsequent publications [29], [26], [27], [28] and it is becoming a powerful tool in the field of machine learning. Manifold alignment can be used as a framework for discovering a unifying representation of multiple datasets. In this paper, we show how to apply this framework to minimize the impact of CSM in steganalysis.

The rest of this paper is organized as follows. Section II presents the manifold alignment strategy and the proposed steganalysis algorithm applying this technique. Section III presents the experimental results obtained using the proposed method for seven different image databases in the cross-domain case. Finally, Section IV summarizes the conclusions of this work and suggests some directions for further research.

II. Manifold Alignment

This section presents a general overview of manifold alignment and specific algorithms to apply this technique in steganalysis.

A. Background

Manifold alignment [26] can be used as a framework for discovering a unifying representation of multiple datasets. In this paper, we show how it can be used in steganalysis with the aim of obtaining a common representation of the training and the testing datasets, being a solution to mitigate the problem of cover source mismatch.

The basic idea of manifold alignment is to use the similarities within each dataset and correspondences between datasets for mapping initially disparate datasets to a joint latent space, usually a low dimensional one. The new representation is extracted by modeling the local geometry of each dataset

and by constraining the dimensionality reduction with the correspondence among datasets.

The problem we want to solve can be formalized as follows: given two data sets $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_m\}$ in $\mathbb{R}^D$, along with correspondence information $x_i \in X \leftrightarrow y_j \in Y$, we need to map X and Y into a new space preserving the local geometry of each dataset and matching the instances (or samples) of both sets in correspondence.

The local geometry within each dataset is modeled with an adjacency matrix [8], usually constructed using a nearest neighbor graph. This means that we construct a graph using every sample of the dataset as a vertex and adding an edge between two vertices of the graph if they are in the set of the k-nearest neighbors. Therefore, we build the adjacency matrix of X as follows:

$$A_x(i,j) = \begin{cases} 1, & \text{if } x_i \text{ and } x_j \text{ are neighbors,} \\ 0, & \text{otherwise.} \end{cases}$$

When the two vertices are in the same dataset, we can improve the adjacency matrix using the geodesic distance [16]:

$$A_x(i,j) = \begin{cases} e^{-||x_i - x_j||}, & \text{if } x_i \text{ and } x_j \text{ are neighbors,} \\ 0, & \text{otherwise.} \end{cases}$$

The adjacency matrix for Y can be built in the same way:

$$A_y(i,j) = \begin{cases} e^{-||y_i - y_j||}, & \text{if } y_i \text{ and } y_j \text{ are neighbors,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The correspondence information between datasets is modeled through a matrix of correspondences. The matrix of correspondences is an adjacency matrix, but, since the two datasets lie in different manifolds, the geodesic distance between samples cannot be used. This matrix is constructed as follows:

$$C_{xy}(i,j) = \begin{cases} 1, & \text{if } x_i \text{ corresponds to } y_j, \\ 0, & \text{otherwise.} \end{cases}$$

The alignment algorithm needs both the correspondence and the local geometry as base information for finding the new representation. With the local geometry, we ensure that the local properties of the datasets are preserved and, with the correspondence information, we can obtain a common representation of both datasets in a different space.

Using the adjacency and the correspondence matrices, we can create the *joint adjacency matrix* [26] as follows:

$$W = \begin{pmatrix} A_x & C_{xy} \\ C_{xy}^{\mathrm{T}} & A_y \end{pmatrix}, \quad (2)$$

where $[\cdot]^{\mathrm{T}}$ denotes the transposition operator. With this joint adjacency matrix, we can compute the Laplacian matrix [8], defined as $L = D - W$, where D is the degree matrix, given by

$$D(i,i) = \sum_{j=1}^{n} W(i,j). \quad (3)$$

The projection into the new space of d dimensions is carried out optimizing a cost function. The new coordinates are given by the d smallest nonzero eigenvectors of $Lf = \lambda Df$ [26]. This optimization can be regarded as a way of obtaining a new graph in which the connected nodes with high weights come closer than the nodes with lower weights (see Wang *et al.* [26] for more details).

The above approach summarizes the basic idea of manifold alignment. However, in our application, we propose a variation in the construction of the adjacency matrix for X. Since X is part of the training set, we know if a sample of X is cover or stego. Therefore, we can apply a multiplier to the geodesic distance in each case, with the intention of increasing the weight when the pair is in the same class and to decrease it otherwise. The purpose of this operation is that, in the new representation, the samples belonging to the same class remain close to each other, whereas the samples of different classes draw away from each other. The new adjacency matrix is defined as follows:

$$A_x(i,j) = \begin{cases} m_1 e^{-||x_i - x_j||}, & \text{if } x_i \text{ and } x_j \text{ are neighbors and they are in the same class,} \\ m_2 e^{-||x_i - x_j||}, & \text{if } x_i \text{ and } x_j \text{ are neighbors and they are in different classes,} \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where m_1 and m_2 are the chosen multipliers.

B. Correspondence Information

In the process of alignment, the algorithm needs the correspondence information (the matrix C_{xy}) to find the new representation. However, the process of domain adaptation in steganalysis is unsupervised and, thus, we do not have information of these correspondences (since we do not know which images are stego or cover in the testing dataset). Since this correspondence information is required, we need to estimate it from the original data. We propose an approach that, to the best of our knowledge, is novel in the manifold alignment literature. The suggested procedure is as follows:

1) Use the first domain as a training dataset.
2) Use the second domain as a testing dataset.
3) Assuming that both datasets have some similarities, pre-classify the testing dataset using some machine learning algorithm.
4) Use the pre-classification results obtained in the previous step to estimate the correspondence information.

More precisely, the correspondence information is constructed as follows. The correspondence matrix shall provide a relationship between one sample in the training domain and one sample in the testing domain. We establish this correspondence based on the probability of classification of each sample obtained from the machine learning classifier. To do this, we create a correspondence between the sample with

the highest probability of being stego in the training domain and the sample with the highest probability of being stego in the testing domain. Then, we create another correspondence between the second sample with the highest probabilities of each set, and so on. Finally, we have a correspondence between each sample of each domain, and this information is used in the alignment process for finding the joint space.

Consequently, the correspondence matrix can be computed as

$$C_{xy}(i,j) = \begin{cases} 1, & \text{if } x_i \text{ and } y_j \text{ are in the same position of the pre-classified sets (ordered by probability),} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

In fact, the method described above for constructing the correspondence matrix applies only when the number of samples of each set and the number of samples belonging to each class (cover or stego) in each set are identical. Nevertheless, this is not a limitation of our method, since it is still possible to a construct the correspondence matrix linking only a subset of X to a subset of Y. For example, we may take only those samples for which we are more confident about their classification. However, we have not developed a final strategy for the selection of these subsets yet. Hence, in the sequel, we consider the homogeneous situation in which both X and Y have the same number of stego and cover images. The specific solution for the non-homogeneous case is left for the future research.

C. Alignment Algorithm

Using the building blocks described in the previous sections, we can define the algorithm for domain adaptation applying manifold alignment.

Given a dataset $X = \{x_1, x_2, ..., x_n\}$ of samples of the training set (vectors of image features), the associated set of labels $X_l = \{l_1, l_2, ..., l_n\}$, where each label indicates whether the corresponding image is cover or stego, and the target dataset $Y = \{y_1, y_2, ..., y_m\}$ of samples of the testing set (vectors of features), we proceed as follows:

1) **k-Nearest Neighbors Graphs.** Each sample of X becomes a vertex of a graph. A vertex (sample of the training dataset) x_i is connected with another vertex (sample) x_j by an edge if x_i is among the k-nearest neighbors of x_j. The same process is used to create the graph for the testing dataset.
2) **Adjacency Matrices.** Build the adjacency matrices A_x and A_y for the k-nearest neighbors graphs of X and Y using the geodesic distance as shown in Expressions (4) and (1).
3) **Pre-classification.** Classify the second domain (testing dataset) using the first domain (training dataset) as a training set. Use an algorithm that provides the probability of classification for each sample.
4) **Correspondence Information.** Establish the correspondence relationship, using the pre-classified sets ordered by probability, as per Expression (5).
5) **Joint Adjacency Matrix.** Use the adjacency matrices and the correspondence information for building the joint adjacency matrix as per Expression (2).
6) **Eigenmaps.** Compute the eigenvectors and eigenvalues for the generalized eigenvector problem: $Lf = \lambda Df$, where D is the diagonal weight matrix (degree matrix). The coefficients of this matrix are column (or row, since W is symmetric) sums of W, as shown in Expression (3), and L is the Laplacian matrix: $L = D - W$.
The projection into the new space is given by the d minimum nonzero eigenvectors $f_1, \ldots, f_d$ of the eigenvalue decomposition. Each of these vectors, f_j, has $n + m$ components, where n is the number of samples of the training dataset X and m is the number of samples of the testing dataset Y. The projection of X and Y, denoted as $\widetilde{X}$ and $\widetilde{Y}$, to the joint space is formed by taking the components of these vectors as follows:

$$\tilde{x}_j = \begin{bmatrix} (f_1)_j & (f_2)_j & \cdots & (f_d)_j \end{bmatrix}^{\mathrm{T}},$$

for $j = 1, \ldots, n$, and

$$\tilde{y}_j = \begin{bmatrix} (f_1)_{n+j} & (f_2)_{n+j} & \cdots & (f_d)_{n+j} \end{bmatrix}^{\mathrm{T}},$$

for $j = 1, \ldots, m$, where $(f_j)_k$ denotes the k-th component of the vector f_j.
Note that projected samples, $\tilde{x}_j$ and $\tilde{y}_j$, have dimension d, which must be lower than or equal to the dimension of the samples of the original datasets (D).

D. Steganalysis

Finally, in this section, we consider the problem of steganalyzing a target set of images from the real world [13]. In this case, we do not know which images are cover and which images are stego in the target set and we do not have information about the conditions in which these images were taken. In such scenario, we assume that the source and the destination domains are different and they need to be adapted. As already remarked, the only assumption we make is about an equal number of stego and cover images in each dataset. This is not required by our method, but we limit the experiments to this case only due to the fact that we do not have a final strategy for the non-homogeneous problem.

In these conditions, we propose the following procedure:

1) **Training Set Feature Extraction.** Choose a training database, embed secret information in some of the images (stego) and leave the other as cover. Perform feature extraction and obtain a set X of samples (vectors of features) and the corresponding set X_l of labels (stego or cover).
2) **Target Set Feature Extraction.** Extract the features from the target database (containing some cover and some stego images), obtaining a set Y of samples (vectors of features).
3) **Domain Adaptation.** Adapt the training and target domains using the algorithm detailed in Section II-C, yielding the adapted sets $\widetilde{X}$ and $\widetilde{Y}$.

4) **Training.** Train the classifier using the adapted set of "projected features" $\widetilde{X}$ and the set of labels X_l.
5) **Classification.** Using the classifier trained in the previous step, classify the adapted testing dataset, using their "projected features" $\widetilde{Y}$, into cover and stego images.

III. Experimental Results

This section presents the experiments conducted to show the effectiveness of the proposed approach for the cover source mismatch problem.

A. Image Databases

In order to test the proposed approach, we selected seven distinct image databases:

- The NRCS database consists of images from the National Resource Conservation System [19] with a fixed size of 2100×1500 pixels.
- The BOSS database consists of images from Break Our Steganographic System! [6] with a fixed size of 512×512 pixels.
- The ESO database consists of images from the European Southern Observatory [5] with variable sizes about 1200×1200 pixels.
- The Interactions (INTE) database consists of images from Interactions.org [12] with variable sizes about 600×400 pixels.
- The NOAA database consists of images from the National Oceanic and Atmospheric Administration (NOAA) [18] with variable sizes about 2000×1500 pixels.
- The Albion (ALBN) database consists of images from the Plant Image Database of the Albion College [1] with a fixed size of 1024×685 pixels.
- The Calphotos (CALP) database consists of images from the Regents of the University of California [25] with variable sizes about 700×500 pixels.

For each database, we generated four sets of 500 images each. Theses sets consist of 250 cover images and 250 stego ones. In two of these sets, the stego images were generated with an embedding rate of 0.5 bits per pixel (bpp), whereas an embedding rate of 0.25 bpp was used for the other two sets. The steganographic system used for embedding is LSB matching [23]. A set of the group with embedding rate of 0.5 bpp was used for training and the other one for testing. The same goes for the two sets with embedding rate of 0.25 bpp.

B. Classification

The experiments show how the classification accuracy is improved after domain adaptation. For comparison purposes, we need the results before and after domain adaptation.

To perform the experiments in a neutral manner, we chose a unique feature extractor and a unique classifier for all cases. As a feature extractor, we used Patterns of Pixel Differences (PPD) [14] for its low CPU time and lower number of features compared to other methods. For obtaining the PPD features, we set the threshold for the pixel differences to $S = 4$, which yields 256 features per image. As a classifier, we chose the implementation LibSVM [3] of Support Vector Machines (SVM) for its efficiency and accuracy.

We used an SVM with a Gaussian Kernel. This classifier must be adjusted to provide optimal results. In particular, the values of the parameters C and γ must be selected. These values should be chosen to give the classifier the ability to generalize. The process has been performed as described in [11]. For all the experiments of the paper, we have used cross-validation on the training set using the following multiplicative grid for C and γ:

$$\begin{aligned} C &\in \left\{2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, \ldots, 2^{15}\right\}, \\ \gamma &\in \left\{2^{-15}, 2^{-13}, 2^{-11}, \ldots, 2^{-1}, 2^{1}, 2^{3}\right\}. \end{aligned} \tag{6}$$

In the standard classification process (with no domain adaptation) we selected the training set and trained the SVM. After that, we tested the classifier with the testing set. This process is typically used in machine learning. On the other hand, in the classification with domain adaptation, we first took the training and the testing datasets (samples of features) and projected them into the joint space using the algorithm proposed in Section II-C. After that, we followed the rest of the steganalysis process as detailed in Section II-D.

C. Tuning Parameters

As discussed in the previous sections, we need to tune some parameters to apply domain adaptation. Since we are working in an unsupervised environment, we have no information about the testing domain other than some grade of similarity with the training domain. Therefore, it is not convenient to tune the parameters using a specific domain, because they might not work with other domains. Hence, we decided to make some reasonable assumptions to select their values.

The first parameter to tune is k, required in the k-nearest neighbors algorithm used for obtaining the graph in the Alignment Algorithm (Section II-C). A common option is to choose the square root of the total number of samples: $k = \sqrt{n+m}$ as suggested in [4].

In addition, we need to tune the weights m_1 and m_2. The weight m_1 affects the pairs which lay on the same class, and we want it as large as possible (in order to decrease its distance in the projected space). On the ohter hand, m_2 affects the pairs in different classes, and we want it as small as possible (so as to increase its distance in the projected space). We tested different values experimentally, and the results always improved when m_1 increased and m_2 decreased, until we reached a value at which no improvement occured and it remained unchanged. After this process, we finally selected $m_1 = 1000$ and $m_2 = 0.001$.

For the pre-classification step of the Alignment Algorithm, we also used an SVM with a multiplicative grid and cross-validation, as described in [11], and the specific values used for the experiments are given in Expression (6). In addition, in this step, we need the membership probabilities for each sample. For this reason, we used an SVM with this capability, more precisely, LibSVM [3] with Platt scaling [21], [15].

Another relevant parameter is the number of dimensions of the latent (joint) space (d). Unfortunately, there is no standard strategy for choosing this value. We made tests with different values of d between 1 and 250 (the original dimension was 256). The classification accuracy (the percentage of correctly classified images) was almost identical for all $d < 100$ and it decreased for larger values of d. According to this result, we selected $d = 2$ in all the other experiments, since $d = 2$ yielded somewhat better results than $d = 1$. It is worth pointing out that just two dimensions were enough to achieve remarkable classification results in the latent space. This illustrates that the data in the latent space are very rich to discriminate between cover and stego images.

D. Results

We can see the results for all the experiments in Table I. Each row in the table corresponds to a training database (first domain) and a testing database (second domain). Firstly, we show the classification accuracy without domain adaptation (column "Acc"), where we can see the effects of CSM. In the next column ("Acc DA"), the classification accuracy results **using** domain adaptation through manifold alignment are shown. Here, we can notice how the accuracy changes with manifold alignment. The results are provided for the two selected embedding rates: 0.5 bpp and 0.25 bpp. The table is divided into several blocks, each of which corresponding to the same training database. The first row of each block shows the results obtained using training and testing databases in the same domain (the same image database). This is not a case of domain adaptation, because the domain would not need to be adapted. However, it is an interesting case, since it allows to analyze how manifold alignment affects the accuracy results when adaptation is not required. In addition, since we are assuming an unsupervised environment, we do not know whether the testing domain needs to be adapted or not. The classification accuracy improves in almost all cases with two different domains (improvement is highlighted in boldface). However, the adaptation of two domains that not need to be adapted usually results in a decreased accuracy. In this case, the decrease in accuracy is typically small, with the only exception of the Albion and Calphotos databases.

As shown in the last row of the Table I, on average, the classification accuracy results with domain adaption improve from 75.00% to 82.00% in the case of 0.5 bpp embedding, and from 67.00% to 72.00% for 0.25 bpp embedding. The increase in classification accuracy is even more remarkable if we consider only those cases that do not need adaptation, that is, when the training and the testing domain differ. In that case, the classification accuracy increases from 72.00% to 81.00% for 0.5 bpp and from 63.00% to 70.00% for 0.25 bpp. Hence, a method to detect when the domains really need to be adapted will improve the effectiveness of the proposed approach.

TABLE I: Classification accuracy ("Acc") with and without domain adaptation ("DA") for LSB matching with embedding bit rates of 0.5 bpp and 0.25 bpp

Embedding bit rate		**0.5 bpp**		**0.25 bpp**	
Train	**Test**	**Acc (%)**	**Acc DA (%)**	**Acc (%)**	**Acc DA (%)**
CALP	CALP	89.6	**91.4**	**89.6**	86.0
CALP	ALBN	53.0	**66.2**	**50.2**	49.8
CALP	BOSS	74.4	**82.8**	53.6	**75.0**
CALP	ESO	95.4	**97.8**	52.6	**87.4**
CALP	INTE	91.4	**98.8**	51.2	**93.8**
CALP	NOAA	90.4	**99.8**	50.0	**99.4**
CALP	NRCS	51.0	**62.6**	50.0	**52.8**
BOSS	BOSS	**88.2**	85.6	**84.4**	76.6
BOSS	ALBN	57.2	**63.4**	**57.0**	53.2
BOSS	CALP	84.6	**90.6**	55.6	**75.2**
BOSS	ESO	64.4	**94.2**	81.4	**87.4**
BOSS	INTE	60.6	**98.0**	83.4	**87.4**
BOSS	NOAA	53.0	**97.6**	71.8	**97.8**
BOSS	NRCS	62.8	**68.6**	**56.0**	54.6
NRCS	NRCS	**89.2**	74.8	**74.4**	60.4
NRCS	ALBN	**70.8**	67.0	**61.4**	58.4
NRCS	BOSS	73.2	**79.2**	**72.2**	70.6
NRCS	CALP	67.4	**87.2**	61.2	**68.0**
NRCS	ESO	78.6	**94.2**	62.8	**92.4**
NRCS	INTE	82.0	**98.4**	62.2	**91.0**
NRCS	NOAA	66.8	**99.2**	55.0	**98.4**
ALBN	ALBN	**76.0**	57.8	**58.4**	53.6
ALBN	BOSS	76.4	**84.4**	**52.2**	42.0
ALBN	CALP	51.2	**82.0**	**50.0**	48.6
ALBN	ESO	76.0	**81.2**	**50.0**	47.4
ALBN	INTE	73.0	**88.4**	50.0	**51.6**
ALBN	NOAA	58.0	**98.6**	**50.0**	49.8
ALBN	NRCS	**72.0**	66.4	54.8	54.8
ESO	ESO	**98.0**	97.2	97.2	**97.4**
ESO	ALBN	50.0	**54.8**	50.0	**51.2**
ESO	BOSS	72.8	**74.4**	55.4	**64.8**
ESO	CALP	72.4	**74.4**	**64.4**	64.2
ESO	INTE	**100.0**	99.6	**99.8**	98.2
ESO	NOAA	**100.0**	99.8	100.0	100.0
ESO	NRCS	50.2	**66.0**	50.0	**54.2**
INTE	INTE	**99.8**	99.6	**99.2**	97.6
INTE	ALBN	**56.4**	52.8	50.6	**58.4**
INTE	BOSS	74.4	**77.0**	**69.4**	65.6
INTE	CALP	72.6	**77.6**	**62.2**	59.0
INTE	ESO	**97.0**	96.8	**95.8**	95.4
INTE	NOAA	100.0	100.0	100.0	100.0
INTE	NRCS	56.6	**59.4**	51.0	**55.4**
NOAA	NOAA	**100.0**	99.8	100.0	100.0
NOAA	NRCS	51.0	**54.0**	50.0	**51.0**
NOAA	ALBN	**51.0**	45.6	**50.0**	46.4
NOAA	BOSS	64.8	**66.0**	**65.8**	57.8
NOAA	CALP	**62.6**	61.2	54.2	**54.8**
NOAA	ESO	95.6	**96.0**	**95.2**	95.0
NOAA	INTE	99.4	**100.0**	**98.2**	97.4
AVERAGE		75.0	**82.0**	67.0	**72.0**

IV. Conclusion

In some applications of classification, like steganalysis, a problem appears when the classifier is moved into the real world. In steganalysis, this problem is known as cover source mismatch and is caused by the existence of significant differences between the training and the testing datasets. In this paper, we present a novel approach for dealing with CSM based on manifold alignment techniques. As shown in the paper, the manifold alignment algorithm seems to be a convenient approach for dealing with the CSM problem, since the classification accuracy results improve when using this technique if the training and testing datasets differ. The results have been obtained for seven different databases showing a relevant improvement compared to the non-adapted case.

For future research, some issues need to be addressed. To begin with, we need to develop a strategy to choose the optimal value of d (although the preliminary results indicate that a very low dimensional space is a convenient choice). Similarly, the effect of the parameter k used in the k-nearest neighbors algorithm should be analysed. We have used the value of k suggested in machine learning literature, but a deeper analysis is required. Finally, we have noticed that adaption is not advisable when the training and testing domains do not need to be adapted. This drawback may be overcome by exploring the degree of similarity between domains prior to deciding whether to use adaptation.

ACKNOWLEDGMENTS

This work was partly funded by the Spanish Government through grants TIN2011-27076-C03-02 "CO-PRIVACY" and TIN2014-57364-C2-2-R "SMARTGLACIS".

REFERENCES

[1] "Plant image database," Available: http://www4.albion.edu/plants/, Albion College, accessed on June 6, 2016 [Online].

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics Series. New York, Secaucus, NJ, USA: Springer, 2006.

[3] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, available: http://www.csie.ntu.edu.tw/~cjlin/libsvm, accessed on June 6, 2016.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2000.

[5] "ESO images," Available: http://www.eso.org/public/images/, European Southern Observatory, accessed on June 6, 2016 [Online].

[6] T. Filler, T. Pevný, and P. Bas, "Break our steganographic system (BOSS)," http://agents.fel.cvut.cz/stegodata/, 2010, accessed on June 6, 2016 [Online].

[7] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Information Forensics and Security, June 2012*, vol. 7, no. 3, pp. 868–882, 2012.

[8] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, Apr. 2001.

[9] J. H. Ham, D. D. Lee, and L. K. Saul, "Learning high dimensional correspondences from low dimensional manifolds," in *Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003, pp. 34–41.

[10] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, Dec. 1936.

[11] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Tech. Rep., 2003, accessed on June 6, 2016 [Online]. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers.html

[12] "ImageBank," Available: http://www.interactions.org/cms/?pid=1900, Interactions.org Particle Physics News and Resources, accessed on June 6, 2016 [Online].

[13] A. D. Ker, P. Bas, R. Boehme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevný, "Moving steganography and steganalysis from the laboratory into the real world," in *Proc. 1st IH&MMSec. Workshop*. Montpellier, France: ACM, Jun. 2013, pp. 45–58.

[14] D. Lerch-Hostalot and D. Megías., "LSB matching steganalysis based on patterns of pixel differences and random embedding," *Computers & Security*, vol. 32, pp. 192–206, 2013.

[15] H.-T. Lin, C.-J. Lin, and R. C. Weng, "A note on platt's probabilistic outputs for support vector machines," *Mach. Learn.*, vol. 68, no. 3, pp. 267–276, Oct. 2007.

[16] Y. Ma and Y. Fu, *Manifold Learning Theory and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2011.

[17] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[18] "NOAA photo library," Available: http://www.photolib.noaa.gov/, National Oceanic and Atmospheric Administration, accessed on June 6, 2016 [Online].

[19] "NRCS photo gallery," Available: http://photogallery.nrcs.usda.gov, National Resource Conservation System, accessed on June 6, 2016 [Online].

[20] T. Pevný, P. Bas, and J. J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix." *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215–224, 2010.

[21] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances In Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.

[22] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[23] T. Sharp, "An implementation of key-based digital signal steganography," in *In Information Hiding*, ser. Lecture Notes in Computer Science. Berlin-Heidelberg, Germany: Springer, 2001, vol. 2137, pp. 13–26.

[24] J. B. Tenenbaum, V. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[25] "CalPhotos," Available: http://calphotos.berkeley.edu/, University of California, Berkeley, accessed on June 6, 2016 [Online].

[26] C. Wang and S. Mahadevan, "A general framework for manifold alignment." in *AAAI Fall Symposium: Manifold Learning and Its Applications*, ser. AAAI Technical Report, vol. FS-09-04. AAAI, 2009.

[27] ——, "Manifold alignment without correspondence," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, ser. IJCAI'09. Morgan Kaufmann Publishers Inc., 2009, pp. 1273–1278.

[28] ——, "Manifold alignment preserving global geometry," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI'13. AAAI Press, 2013, pp. 1743–1749.

[29] L. Xiong, F. Wang, and C. Zhang, "Semi-definite manifold alignment," in *ECML*, ser. Lecture Notes in Computer Science, J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, and A. Skowron, Eds., vol. 4701. Springer, 2007, pp. 773–781.