How to Find Relevant Training Data: A Paired Bootstrapping Approach to Blind Steganalysis

Pham Hai Dang Le #1, Matthias O. Franz #2

# *Institute for Optical Systems, HTWG Konstanz*
Brauneggerstr. 55
78462 Konstanz, Germany
1 dangle@htwg-konstanz.de
2 mfranz@htwg-konstanz.de

***Abstract*—Today, support vector machines (SVMs) seem to be the classifier of choice in blind steganalysis. This approach needs two steps: first, a training phase determines a separating hyperplane that distinguishes between cover and stego images; second, in a test phase the class membership of an unknown input image is detected using this hyperplane. As in all statistical classifiers, the number of training images is a critical factor: the more images that are used in the training phase, the better the steganalysis performance will be in the test phase, however at the price of a greatly increased training time of the SVM algorithm. Interestingly, only a few training data, the support vectors, determine the separating hyperplane of the SVM. In this paper, we introduce a paired bootstrapping approach specifically developed for the steganalysis scenario that selects likely candidates for support vectors. The resulting training set is considerably smaller, without a significant loss of steganalysis performance.**

I. Introduction

Most of current blind steganalyzers are based on a statistical classification algorithm. These algorithms require a large number of training images from which statistical regularities are extracted in a training phase before the trained classifier can be applied to unknown images in the test phase. Unfortunately, the complexity of the training process increases polynomially with the number of training images, which sets a practical limit on the training set size. Here, we consider the most widely used classifier in steganalysis: the support vector machine (SVM) [1]. In its most common form, the separating hyperplane of the SVM is found as the solution of a quadratic optimization problem with linear constraints. The resulting solutions are *sparse*, i.e., they depend only on a small subset of all training data, the so-called *support vectors* of the separating hyperplane. If an SVM is only trained on the support vectors, one obtains exactly the same hyperplane as if the full training set was used.

Unfortunately, the support vectors cannot be known in advance of the training process. Here, we propose a three-step procedure that at least produces likely candidates for support vectors and thus is capable of largely reducing training set size without compromising on steganalysis performance. This method is well-known in machine learning as *bootstrapping* or *shrinking* [2], [3]. We show that using this method naïvely, without adaptions to the steganalysis scenario, can even decrease the performance (Section III). Before, in Section II, we give the details of the two steganalyzers used in our study. In Section IV, we test our approach on the BOSSbase [4], [5] and on the BOWS-2 [6] image data set using ± 1 embedding (also called LSB matching) [7] and the HUGO algorithm [8] for generating the stego images. Finally, we conclude this paper with a brief summary in Section V.

WIFS'2012, December, 2-5, 2012, Tenerife, Spain. 978-1-4244-9080-6/10/$26.00 ©2012 IEEE.

II. Steganalyzers

The two steganalyzers investigated in our study use different features for characterizing the input images, but share the same classifier which takes these features as input (Figure 1). This final classification was computed with a 1-norm soft margin non-linear C-SVM using a Gaussian kernel. The choice of the parameter C of the SVM and the width σ of the Gaussian kernel was based on the paired cross-validation procedure described in [9].

A. Lyu and Farid type features

Our investigated features are similar to those of Lyu and Farid [10], but use an improved image modeling procedure [11]. In the first step, the input image is transformed from its pixel representation into its wavelet representation using QMF (quadrature mirror filter) wavelets of support size 9 [12].The number of subbands for an RGB image depends on the number of pyramid levels s in the wavelet representation. Because there are three orientation bands in wavelets, i.e. the diagonal, vertical, and horizontal orientations, the total number of subbands in a color image is $3(3s + 1)$.

The modeling step takes place in the wavelet domain. In order to model the dependency of a wavelet coefficient from its neighborhood, we have to define a neighborhood structure which is shown in Figure 2. Due to only including the neighboring coefficients from closest orientations on the same

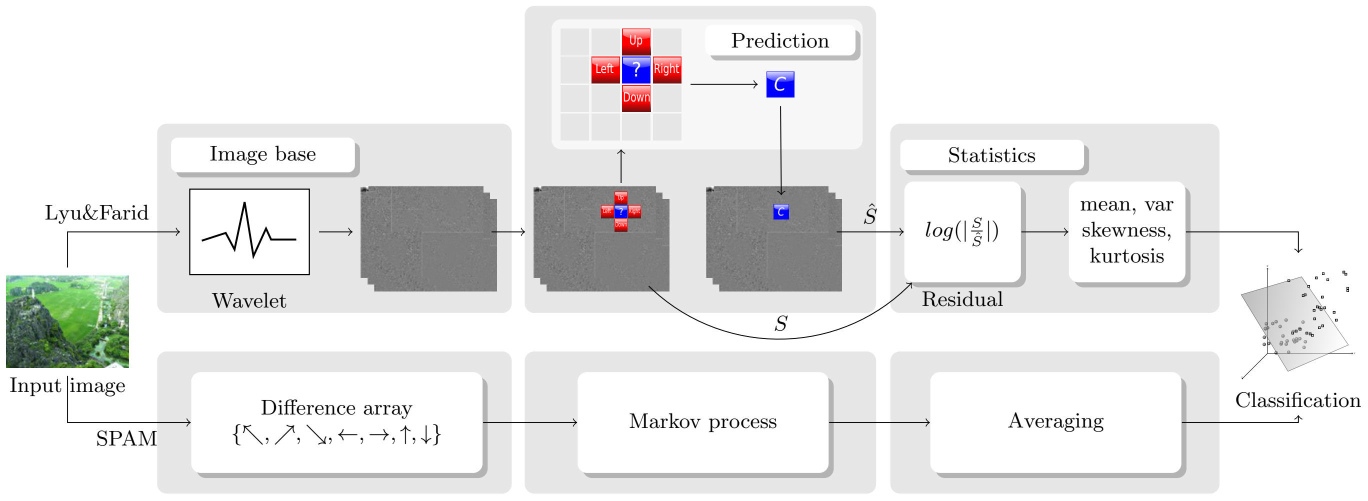
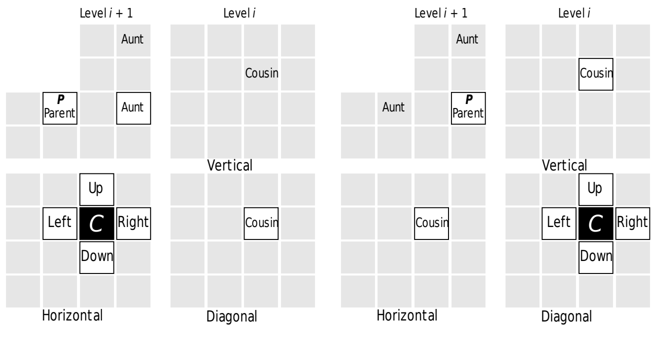

Lyu&Farid
Image base
Wavelet
Prediction
Up
Left
?
Right
Down
C
$\hat{S}$
Statistics
$log(|\frac{S}{\hat{S}}|)$
Residual
mean, var skewness, kurtosis
S
Classification
Input image
SPAM
Difference array $\{\nwarrow,\nearrow,\searrow,\leftarrow,\rightarrow,\uparrow,\downarrow\}$
Markov process
Averaging

Fig. 1. Steganalyzer architecture



(a) Horizontal (b) Diagonal

Fig. 2. The neighborhood structure for image modeling (color neighbors not included) is defined by white coefficients. The central coefficient to be predicted is C.

scale (hence including horizontal and vertical coefficients for predicting the diagonal subband, but only diagonal coefficients for both the horizontal and vertical subbands), and correspondingly only one (diagonal) or two neighbors (horizontal and vertical) from the coarser scales. Neighborhoods in the wavelet representation contain seven coefficients from the same color channel as well as the corresponding central coefficients from the other color channels (not shown in Figure 2).

The predictions are computed with linear regression applied to each subband separately, i. e., the magnitude of the central coefficient is obtained as a weighted sum of the magnitudes of its neighboring coefficients greater than a given threshold. It has been shown empirically that only the magnitudes of coefficients are correlated, and the correlation decreases for smaller magnitudes [13]. The weight sets over all subbands thus constitute the image model. In their original approach, Lyu and Farid use standard least-squares regression for this purpose. In our implementation, we use Gaussian Process (GP) regression [14], [15] after normalizing all subband coefficients to the interval $[0, 1]$. The GP regression needs an additional model selection step for estimating the noise content in the image; we use Geisser's surrogate predictive probability [16]. It is computed on a subset of the coefficients: The finest scales are subsampled by a factor of five and the coarser by a factor of three, each in both directions. Details on this regression technique can be found in [15]. Each estimator is trained and used for prediction on the same subband. Thus, the training and test set coincide for this application. From the predicted coefficients $\hat{S}$, small coefficients with amplitude below a threshold of $t = 1/255$ are set to zero. For reconstructing complete images, the algebraic signs are transferred from the original to the predicted subband coefficients.

Next, the four lowest statistical moments, i.e. mean, standard deviation, skewness, and kurtosis, of the subband coefficients (called *marginal statistics* in [10]) and of the subband residuals (called *error statistics*) are computed, again for each color and subband separately. Finally, all these independently normalized statistics serve as input features for the support vector machine. In this study, we use $s = 3$ pyramid levels which results in a 80-dimensional feature vector.

B. SPAM features

The SPAM (Subtractive Pixel Adjacency Matrix) features [17] are more suitable than the Lyu and Farid type features for detecting ±1 embedding. The SPAM features make use of the dependencies between neighboring pixels by regarding their transition probabilities. For the 8-neighborhood $\{\nwarrow,\swarrow,\nearrow,\searrow,\leftarrow,\rightarrow,\uparrow,\downarrow\}$ of a pixel, the model determines the probabilities of the eight transitions. Let $X = (X_{ij}) \in \{0, ..., 255\}^{m_1 \times m_2}$ be an image. In the first step [17], [8], the model calculates the difference array $\mathbf{D}^{\bullet}$, for each direction $\bullet \in \{\nwarrow,\swarrow,\nearrow,\searrow,\leftarrow,\rightarrow,\uparrow,\downarrow\}$ separately. For instance, the horizontal left-to-right transition, $\mathbf{D}^{\rightarrow}_{ij}$ is calculated as

$$\mathbf{D}^{\rightarrow}_{ij} = X_{ij} - X_{ij+1} \ , \qquad (1)$$

where $1 \leq i \leq m_1, 1 \leq j \leq m_2 - 1$. The next step permits two options, either the first-order Markov process calculated by

$$\mathbf{M}^{\rightarrow}_{uv} = Pr(D^{\rightarrow}_{ij+1} = u | D^{\rightarrow}_{ij} = v) \ , \tag{2}$$

or the second-order Markov process described as

$$\mathbf{M}^{\rightarrow}_{uvw} = Pr(D^{\rightarrow}_{ij+2} = u | D^{\rightarrow}_{ij+1} = v, D^{\rightarrow}_{ij} = w) \ , \tag{3}$$

where $u, v, w \in \{-T, ..., T\}$, $1 \leq T \leq 255$. The SPAM features are obtained by averaging the eight matrices $\mathbf{M}^{\bullet}$ as follows

$$\begin{aligned} \mathbf{F}^{\cdot}_{1,...,k} &= \frac{1}{4} \quad [\mathbf{M}^{\leftarrow}_{\cdot} + \mathbf{M}^{\rightarrow}_{\cdot} + \mathbf{M}^{\uparrow}_{\cdot} + \mathbf{M}^{\downarrow}_{\cdot}] \ , \\ \mathbf{F}^{\cdot}_{k+1,...,2k} &= \frac{1}{4} \quad [\mathbf{M}^{\nwarrow}_{\cdot} + \mathbf{M}^{\swarrow}_{\cdot} + \mathbf{M}^{\nearrow}_{\cdot} + \mathbf{M}^{\searrow}_{\cdot}] \ , \end{aligned} \tag{4}$$

where $k = (2T+1)^3$ for the second-order features and $k = (2T+1)^2$ for the first-order features. In our implementation, the SPAM features were computed using the SPAM software provided on the BOSS website [5]. Originally, the authors in [17] propose to apply $T = 4$ for the first-order Markov process (162 features) and $T = 3$ for the second-order Markov process. Here, we use the second order features resulting in 686 values for the support vector machine.

III. Paired Bootstrapping

The purpose of our bootstrapping approach is to select candidate training examples that are likely to become support vectors. In this way, we obtain much smaller training sets without significantly losing steganalysis performance. An alternative way is boosting through AdaBoost, but this does not improve the performance as shown in [18]. In this paper, bootstrapping [2], [3] works in three steps. First, we train a classifier on a smaller subset of our training data; in the second step, we apply this classifier to the remaining training data, referred to as the *bootstrapping set*. Typically, the classifier will misclassify some of these training images. These data are likely to become the support vectors since they require a modification of the original hyperplane. The misclassified examples are subsequently added to the original small training set, and finally the classifier is re-trained on this augmented training set. Note that we disregard the model selection procedure and use the identical hyperparameters (C, σ) of the initial set for the bootstrapping set.

However, if applied **naïvely** to steganalysis problems, bootstrapping will work in a limited fashion. The reason for this was already observed in the context of model selection [9]: in steganalysis, cover and stego image pairs are much more similar to each other than to any other image in the training data. If only one image in a stego-cover image pair is added to the augmented training set, the resulting separating hyperplane is not automatically forced to separate this pair, instead it is unconstrained from one side. This typically results in a largely reduced steganalysis performance, often approaching random guessing. Therefore, standard bootstrapping has to be modified for steganalysis: when a misclassification occurs, both the cover and the stego version of the misclassified image have to be added to the augmented training set.

In our approach, misclassifications can occur in two types: either we classify a stego image as a cover (*miss*), or a cover image as stego (*false alarm*). Furthermore, we call a correctly classified image either a *hit* (a stego image is classified as stego) or a *correct rejection* (a cover image is classified as cover). By choosing unbalanced bootstrapping sets, one can focus on reducing only one error type: if the bootstrapping set consists only of cover images, we reduce the false alarm rate, whereas if the bootstrapping set consists only of stego images, a reduced miss rate will result. The first option is especially attractive for steganalysis since in most realistic scenarios, there are many more cover images than stego images. In this case, we are mostly interested in minimizing the number of false alarms. In addition, cover images are much easier to obtain in large numbers than stego images. Note, however, that for each misclassified cover image one has to generate the corresponding stego image before it can be added to the augmented training set.

TABLE I
Ordering of the training and test data sets used for bootstrapping

No.	training set	bootstrapping set	testing set
1.	$\mathcal{X}_1$	$\{\mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4\}$	$\mathcal{X}_5$
2.	$\mathcal{X}_2$	$\{\mathcal{X}_3, \mathcal{X}_4, \mathcal{X}_5\}$	$\mathcal{X}_1$
3.	$\mathcal{X}_3$	$\{\mathcal{X}_1, \mathcal{X}_4, \mathcal{X}_5\}$	$\mathcal{X}_2$
4.	$\mathcal{X}_4$	$\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_5\}$	$\mathcal{X}_3$
5.	$\mathcal{X}_5$	$\{\mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4\}$	$\mathcal{X}_1$

IV. Experimental Results

As starting point, our paired bootstrapping approach was applied to the BOSSbase 0.92 data set [4], [5] containing 9,074 images. We randomly permuted the 9,074 images and used only the first 9,000. Next, the 9,000 images were separated into five disjoint sets $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4, \mathcal{X}_5$ (each containing 1,800 images).

For bootstrapping, one of the five sets acted as the initial training set for the first step, one as the test set and the remaining three were merged into the bootstrapping set. We used the five orderings of the five sets shown in Table I for testing both unbalanced approaches, i.e. the bootstrapping set consisted either only of cover images or only of stego images. In the following we compare the initial classifier (on the training set), the classifier after bootstrapping with cover images, the classifier after bootstrapping with stego images, and the classifier after training with the $Complete$ set, i.e. both the training set and the bootstrapping set (Table I). In the training set and the complete set, the data are always given in paired form, i.e., consisting of cover and stego pairs. The sets $\mathcal{B}^{Cover}$ and $\mathcal{B}^{Stego}$ are the union of the initial training set with the falsely classified cover or stego images found by bootstrapping. Note in the case of the set $\mathcal{B}^{\{Stego,\ Cover\}}_{\textbf{paired}}$ each misclassified cover image needs its corresponding stego image. For this purpose, one has to generate the stego image before it can be added to the augmented training set, whereas the augmented training set $\mathcal{B}^{\{Stego, Cover\}}_{single}$ contains only the misclassified images.

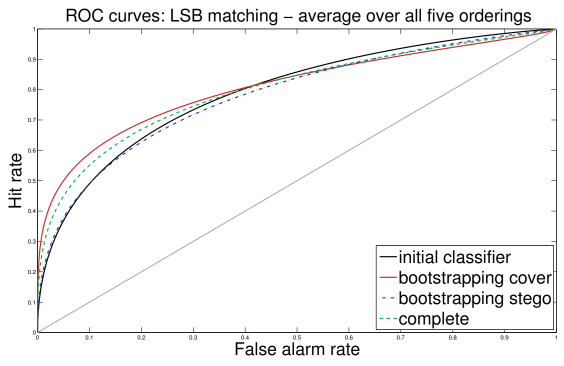


Fig. 3. LSB matching ($E_R = 40\,\%$) using Lyu and Farid features, BOSSbase: Average ROC curves (through merging the instances) of the five different orderings (Table I) of the training and test sets used for *paired* bootstrapping.

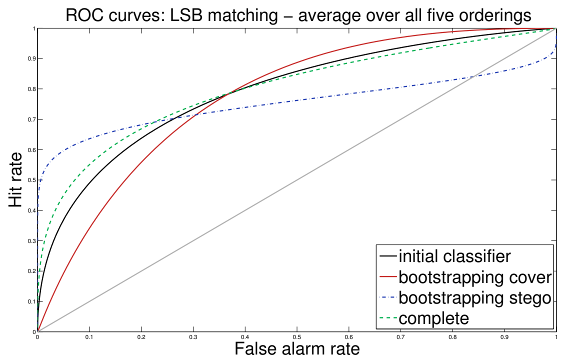


Fig. 4. Identical to Figure 3 but using *single* bootstrapping (LSB matching ($E_R = 40\,\%$) using Lyu and Farid features, BOSSbase).

We expect that bootstrapping will have different effects depending on the performance of the initial classifier. For this reason, we use the SPAM features for detecting the HUGO algorithm although there exist more suitable features [19]. Therefore, we chose four combinations of features, steganographic algorithms, and embedding rates according to the performance of the base classifier:

1) HUGO with SPAM features with an error probability P_E of the base classifier of $42.98\,\%$;
2) ± 1 embedding with embedding rate $E_R = 40\,\%$; Lyu and Farid features and $P_E = 28.01\,\%$;
3) ± 1 embedding with $E_R = 20\,\%$; SPAM features and $P_E = 14.47\,\%$;
4) ± 1 embedding with $E_R = 40\,\%$; SPAM features and $P_E = 6.39\,\%$.

Thus, steganograms ($E_R = 40\,\%$ and $E_R = 20\,\%$) were generated using ± 1 embedding (also called LSB matching) [7] from the BOSSbase 0.92 data set [4], [5]. Here, the embedding messages are random values. Furthermore, we used the stego images of BOSSbase 0.92 [5] with an embedding rate $E_R = 40\,\%$. Although the images in the BOSSbase are from multiple camera sources, we apply bootstrapping on a second database to avoid possible bias towards a specific database (BOWS-2 image database [6]). We use 8,000 of the 10,000 images and

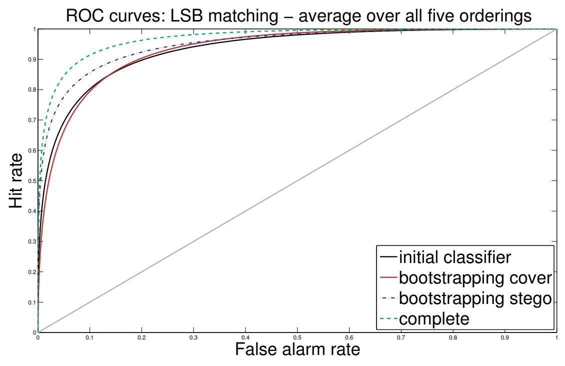


Fig. 5. *Paired* bootstrapping on LSB matching ($E_R = 20\,\%$) using SPAM features.

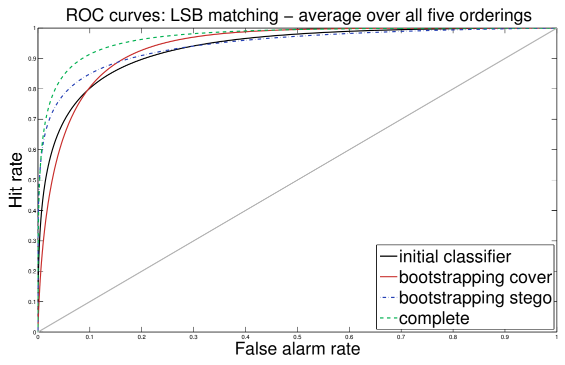


Fig. 6. *Single* bootstrapping on LSB matching ($E_R = 20\,\%$) using SPAM features, BOSSbase.

separated them into four disjoint sets (each containing 2,000 images) in the same manner as in the first experiment. In addition, we report run-times to demonstrate the speed-up due to bootstrapping.

We report steganalysis performance at P_E, i.e. the minimum overall error point of the *receiver operating characteristic* (ROC) curve, and as well as the area under the ROC curve (AUC). Note that the test result values in Table II are obtained through first combining the decision values from all runs into one set [20] and then generating the ROC curves according to [21]. Table III gives the percentages of additionally found data that became support vectors. Figures 3 – 6 compare paired and naïve (single) bootstrapping of all three feature types in terms of the ROC curves of the steganalyzers. In Table II, the highest bootstrapping performance in terms of the AUC is highlighted (with a light gray background). The following points summarize our findings:

1) In terms of the AUC, the paired bootstrapping approach mostly leads to an improvement compared to the single bootstrapping approach (Table II).
2) Paired bootstrapping leads in our experiments to a comparable performance to the complete training set. In the case of Lyu and Farid features (and SPAM features with $E_R = 60\,\%$), paired bootstrapping with cover (and with stego) images outperforms even the results for the

TABLE II

STEGANALYSIS PERFORMANCE OF THE INITIAL CLASSIFIER (ON THE TRAINING SET) AND AFTER BOOTSTRAPPING WITH COVER ($\mathcal{B}^{Cover}$) AND STEGO ($\mathcal{B}^{Stego}$) IMAGES. P_E (IN %) IS THE MINIMUM OVERALL ERROR, AND AUC (IN %) IS THE AREA UNDER THE CURVE. E_R STANDS FOR THE EMBEDDING RATE FOR THE STEGANOGRAPHY ALGORITHMS.

BOSSbase

			Training set		$\mathcal{B}^{Cover}_{paired}$		$\mathcal{B}^{Stego}_{paired}$		$\mathcal{B}^{Cover}_{single}$		$\mathcal{B}^{Stego}_{single}$		$Complete$	
Alg.	E_R	Feat.	P_E	AUC	P_E	AUC	P_E	AUC	P_E	AUC	P_E	AUC	P_E	AUC
HUGO	0.4	SPAM	42.98	59.87	42.43	60.60	39.92	64.10	42.46	60.43	39.39	63.96	38.65	65.82
±1 emb.	0.4	Lyu&Farid	28.01	79.23	25.13	80.76	28.62	78.12	29.22	77.36	22.68	75.35	26.52	79.97
	0.2	SPAM	14.47	93.28	14.39	93.32	12.18	94.92	13.30	93.84	12.53	94.29	9.27	96.94
	0.4	SPAM	6.39	98.42	6.74	98.27	5.93	98.60	6.55	98.27	5.98	98.44	4.43	99.19

BOWS-2

			Training set		$\mathcal{B}^{Cover}_{paired}$		$\mathcal{B}^{Stego}_{paired}$		$\mathcal{B}^{Cover}_{single}$		$\mathcal{B}^{Stego}_{single}$		$Complete$	
Alg.	E_R	Feat.	P_E	AUC	P_E	AUC	P_E	AUC	P_E	AUC	P_E	AUC	P_E	AUC
±1 emb.	0.2	SPAM	9.23	97.47	8.58	98.32	3.78	99.63	9.85	98.13	8.28	98.60	3.28	99.83
	0.4	SPAM	4.65	99.50	4.70	99.50	1.25	99.98	4.80	99.49	4.55	99.82	1.18	99.98
	0.6	SPAM	4.35	99.72	4.35	99.72	0.85	99.99	4.35	99.72	4.02	99.89	1.03	99.98

TABLE III

BOSSBASE: PERCENTAGE OF ADDITIONAL TRAINING EXAMPLES THAT BECAME SUPPORT VECTORS IN THE CASE OF PAIRED BOOTSTRAPPING APPROACH. THE FIRST COLUMN GIVES THE PERCENTAGE OF THE COVER IMAGES WHEN THE BOOTSTRAPPING SET CONSISTED ONLY OF COVER IMAGES, THE SECOND COLUMN THE PERCENTAGE OF THE ADDITIONALLY GENERATED STEGO IMAGES. THE THIRD COLUMN GIVES THE PERCENTAGE OF THE COVER IMAGES WHEN THE BOOTSTRAPPING SET CONSISTED ONLY OF STEGO IMAGES, THE SECOND COLUMN THE PERCENTAGE OF STEGO IMAGES. THE LAST TWO COLUMNS SHOW THE NUMBER OF ADDITIONAL IMAGES FOUND BY BOOTSTRAPPING.

			$\mathcal{B}^{Cover}_{\textbf{paired}}$		$\mathcal{B}^{Stego}_{\textbf{paired}}$		Additional imgs.	
Alg.	E_R	Feat.	% cover imgs.	% stego imgs.	% cover imgs.	% stego imgs.	$\mathcal{B}^{Cover}$	$\mathcal{B}^{Stego}$
HUGO	0.4	SPAM	99.3	97.4	92.6	94.3	1419 (±1237)	3461(±1617)
±1 emb.	0.4	Lyu&Farid	97.3	79.8	86.4	99.8	1567 (±198)	1374 (±160)
	0.2	SPAM	98.4	86.9	77.8	96.4	706 (±51)	711 (±42)
	0.4	SPAM	98.7	79.3	59.1	77.2	421 (±44)	377 (±22)

complete set. Bootstrapping on HUGO shows minor improvement.

3) According to Table II, bootstrapping on stego images seems to be more suitable for SPAM features. In contrast, bootstrapping on cover images appears preferable for Lyu and Farid features.
4) Table III shows that — as originally intended by our approach — most new image feature pairs in the augmented data set indeed become new support vectors. This effect is less pronouced in the case where the initial detection performance is already high (±1 embedding Table II and Table III).
5) When using cover images for bootstrapping, the cover image features of the additionally included bootstrapping pairs are more likely to become support vectors than the corresponding stego image features. The converse is true for bootstrapping using stego images (see Table III).
6) With increasing P_E value (the minimum overall error point of the ROC curve), the number of additional images decreases (Tables II and III).
7) The single bootstrapping approach — instead of the paired bootstrapping approach — improves the performance in most cases to a smaller degree (see ±1 embedding, Table II, Figure 4, and Figure 6). In the case of LSB matching $E_R = 40\,\%$ (BOWS-2 database), single bootstrapping on cover images decreases the detection performance.
8) Figures 4 and 6 show that single bootstrapping leads only to improvements at small false alarm rates, whereas performance deteriorates at higher false alarm rates in the ROC curves.
9) Table IV demonstrates that training set and the augmented training set need only between a third and a half of the time in comparison to the full training set (training set and bootstrapping set, Table I).

In the case of Lyu and Farid features, the small false alarm rate as compared to the much higher miss rate indicates that the classifier mainly tries to model the set of cover images by fitting its decision surface as closely as possible around it. As a result, misses of stego images are predominantly *inside* the cover image set (as seen from the point of view of the Lyu and Farid feature space). If one tries to improve steganalysis

TABLE IV
RUN-TIME RESULTS OF THE TRAINING AND TEST DATA SETS USED FOR BOOTSTRAPPING ON THE BOWS-2 DATABASE.

E_R		training set	augmented training set	full training set
0.2	min	02 hrs 13 min	02 hrs 16 min	07 hrs 59 min
	max	03 hrs 23 min	03 hrs 26 min	11 hrs 15 min
0.4	min	01 hrs 11 min	01 hrs 16 min	04 hrs 30 min
	max	01 hrs 45 min	01 hrs 50 min	06 hrs 15 min
0.6	min	00 hrs 47 min	00 hrs 50 min	03 hrs 18 min
	max	01 hrs 22 min	01 hrs 25 min	04 hrs 59 min

performance by including more misses into the training data, one risks destroying the enclosing capability of the decision surface on which the success of this steganalyzer is based. On the other hand, including more false alarms leads to a finer modeling of the cover image set enclosure. One reason why this seems to be less risky could be that the cover images form a more compact set in the space of the Lyu and Farid features than stego images. The converse observation seems to be valid for SPAM features: here, the stego images seem to form a more compact set than the cover images. Unfortunately, we currently do not have experimental results that explicitly confirm this idea.

V. CONCLUSION

Our investigation of paired bootstrapping shows that one can obtain a increased steganalysis performance by including only a small number of additional training images. In this paper, applying single bootstrapping naïvely, as is well-known in the literature, works only at small false alarm rates. We observed that almost all image pairs selected by the paired bootstrapping process result in additional support vectors.

In the case of the Lyu and Farid features in conjunction with SVM classifiers, best results are obtained on LSB matching steganograms when bootstrapping is performed only on cover images. Additional stego images, however, can sometimes have the opposite effect, so it is not advisable to use them for bootstrapping with this type of steganalyzer. In the case of SPAM features, bootstrapping on stego images led to a better performance than on cover images. Although bootstrapping on stego images led to a a better performance for detecting HUGO than on cover images, bootstrapping on cover images for HUGO still will improve steganalysis performance, although at a slower rate. Paired bootstrapping on cover images in terms of Lyu and Farid features and bootstrapping on stego images in terms of SPAM features lead to improvements at small false alarm rates. This, however, is not a severe disadvantage in realistic scenarios, since there are many more cover images than stego images, so that reducing false alarms has a large impact on the overall error rate. Moreover, bootstrapping sets consisting entirely of cover images are much easier to obtain than stego images.

ACKNOWLEDGMENT

P. H. D. Le was supported by the German Federal Ministry of Research and Education under the research grant number 17N0208, PT-AIF. We would like to thank Andrew P. Smith.

REFERENCES

[1] B. Schölkopf and A. J. Smola, *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[2] B. E. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT*, 1992, pp. 144–152.

[3] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, Cambridge, MA, USA, 1999.

[4] P. Bas, T. Filler, and T. Pevný, ""Break our steganographic system": The ins and outs of organizing BOSS," in *Information Hiding, 13th International Conference*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, S. Craver, and A. D. Ker, Eds. Springer-Verlag, May 18–20 2011, pp. 59–70.

[5] T. Filler, T. Pevný, and P. Bas, "BOSS (Break Our Steganography System)," 2010, software available at http://www.agents.cz/boss/.

[6] European Network of Excellence ECRYPT, "BOWS-2 (Break Our Watermarking System)," 2008, software available at http://www.agents.cz/boss/ (accessed 2010).

[7] A. D. Ker and I. Lubenko, "Feature reduction and payload location with WAM steganalysis," in *Media Forensics and Security*, 2009, p. 72540.

[8] T. Pevný, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *Information Hiding - 12th International Conference*, ser. Lecture Notes in Computer Science, R. Böhme, P. W. L. Fong, and R. Safavi-Naini, Eds., vol. 6387. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 161–177. [Online]. Available: http://portal.acm.org/citation.cfm?id=1929304.1929317

[9] V. Schwamberger and M. O. Franz, "Simple algorithmic modifications for improving blind steganalysis performance," in *Proceedings of the 12th ACM workshop on Multimedia and security*, ser. MM&Sec '10. New York, NY, USA: ACM, 2010, pp. 225–230. [Online]. Available: http://doi.acm.org/10.1145/1854229.1854268

[10] S. Lyu and H. Farid, "Steganalysis using higher-order image statistics," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 1, pp. 111–119, March 2006.

[11] P. H. D. Le and M. O. Franz, "Single band statistics and steganalysis performance," in *IIH-MSP*. IEEE Computer Society, 2010, pp. 188–191.

[12] E. P. Simoncelli and E. H. Adelson, "Subband transforms," in *Subband Image Coding*, J. W. Woods, Ed. Norwell, MA, USA: Kluwer Academic Publishers, 1990.

[13] R. W. Buccigrossi and E. P. Simoncelli, "Image compression via joint statistical characterization in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 8, no. 12, pp. 1688–1701, December 1999.

[14] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., vol. 3176. Berlin, Germany: Springer-Verlag, October 2004, pp. 63–71.

[15] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, January 2006.

[16] S. Geisser and W. F. Eddy, "A predictive approach to model selection," *Journal of the American Statistical Association*, vol. 74, no. 365, pp. 153–160, March 1979.

[17] 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.

[18] J. Kodovskỳ, "Ensemble classification in steganalysis-crossvalidation and AdaBoost," Binghamton University, Tech. Rep., 2011.

[19] J. J. Fridrich, J. Kodovský, V. Holub, and M. Goljan, "Breaking hugo - the process discovery," in *Information Hiding - 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*, ser. Lecture Notes in Computer Science, T. Filler, T. Pevný, S. Craver, and A. D. Ker, Eds., vol. 6958. Springer, 2011, pp. 85–101.

[20] T. Fawcett, "An introduction to ROC analysis," *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2005.10.010

[21] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The binormal assumption on precision-recall curves," in *Proceedings of the 20th International Conference on Pattern Recognition*. IEEE Computer Society, 2010, pp. 4263–4266.