On dangers of cross-validation in steganalysis

Jan Kodovský
jan.kodovsky@binghamton.edu

Abstract

Modern steganalysis is a combination of a feature space design and a supervised binary classification. In this report, we assume that the feature space has been already constructed, i.e., the steganalyst has a set of training features and needs to train a binary classifier. Any machine learning tool can be used for this task and its parameters can be tuned through cross-validation, a standard automated model-selection procedure. However, classification problems arising in steganalysis have a very specific nature – individual training samples naturally form pairs of cover–stego feature vectors with opposite labels lying close to each other in the feature space. It is important to preserve these cover–stego pairs during cross-validation (prevent splitting each pair into different folds) otherwise the obtained error estimates may be misleading and lead to a suboptimal performance of the classifier.

In this report, we demonstrate the sketched problem with cross-validation on a specific example of image steganalysis in the JPEG domain. As a classifier, we selected the support vector machine (SVM), a popular choice in steganalysis. In particular, we show that the implicit k-fold cross-validation as implemented in LIBSVM [2], a widely used implementaion of SVM, is *not* suitable for steganalysis and may result in a suboptimal performance and a striking discrepancy between the predicted and the real testing error. Instead of the implicit k-fold cross-validation, a steganalysis-aware cover-stego pair preserving cross-validation should be used. We stress that this is a steganalysis-specific issue and does not indicate any implementation flaw in LIBSVM.

The issue with the standard cross-validation procedure in steganalysis has already been pointed out by Schwamberger and Franz in 2010 [14]. We believe, however, that the message may have been hidden to the reader in other experiments and conclusions presented in [14], as authors studied not only the cross-validation, but also different normalization techniques, and performed numerous experiments using different features and stego-algorithms. This technical report, on the other hand, is devoted solely to the problem of improper cross-validation. We go more in depth, provide explanation, and also study severity w.r.t. payload. Moreover, we point out a few examples of published work with results affected by the improper cross-validation.

1 Introduction

Due to the complexity of dependencies among DCT coefficients in natural images and the difficulty of modeling them, a feature-based steganalysis is nowadays the most common approach for detection of secret messages embedded in cover objects by a certain steganographic technique. In feature-based steganalysis, images are represented in a feature space using features carefully designed to be sensitive to embedding changes while suppressing the influence of content as much as possible. The detection problem is then transformed into a supervised classification task carried out in the feature space. In particular, the steganalyst trains a binary classifier on a sufficiently large set of training cover and stego features.

Support vector machine (SVM) [13] is a powerful and theoretically well founded classification tool capable of learning even a highly non-linear class boundaries. Thanks to the LIBSVM [2], a publicly available implementation of this non-trivial learning machinery with a user-friendly and easy-to-use interface, SVMs have become the most commonly chosen classifier for steganalysis up to date. Typically, a kernelized version of SVM with the Gaussian kernel is used. A standard technique for optimizing the hyper-parameters of the Gaussian SVM, the cost parameter C and the kernel width γ, is a search over a pre-defined two-dimensional grid of values combined with a k-fold cross-validation (CV) that gives a performance estimate (cross-validation error) on every point of the grid. The combination of parameters C and γ yielding the lowest CV error is then used for the final SVM training and testing. A more detailed description of the k-fold cross-validation procedure appears in Section 2.

Even though the implementation of k-fold cross-validation is part of the LIBSVM package, it should *not* be used for steganalysis. The reason for that is given by the very specific nature of the classification tasks arising in steganalysis and is explained in Section 3. Instead, a modified version of the cross-validation that

takes into account the specifics of steganalysis should be used. Our claims are experimentally illustrated on the JPEG-domain steganographic algorithm nsF5 [8].

2 Cross-validation

The k-fold cross-validation is a general procedure for estimating the prediction error of any supervised classifier. In this section, we briefly summarize its inner workings, explain how it can be used for optimizing the parameters of a SVM, and indicate a potential threat for steganalysis. A more detailed discussion on k-fold cross-validation can be found, for example, in [9].

2.1 Overview

Let $\mathcal{F} \equiv \mathbb{R}^n$ be a feature space the classification takes place in. A training dataset $\mathcal{X} = \{\mathbf{x}_i \in \mathcal{F} | i = 1, \ldots, N\}$ is a collection of N features with known labels $y_i \in \mathcal{Y}$, $i = 1, \ldots, N$. For binary classification in steganalysis, $\mathcal{Y} \equiv \{-1, +1\}$, -1 standing for cover and $+1$ for stego classes. The k-fold cross-validation divides the whole training set $\mathcal{X}$ into k disjoint and approximately equally populated groups $\mathcal{X}_i$, $i = 1, \ldots, k$ called folds, where $\mathcal{X} = \cup_i \mathcal{X}_i$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for $i \neq j$. The first fold $\mathcal{X}_1$ is then set apart, the classifier is trained on the union of the remaining $k-1$ folds and its performance is evaluated in terms of the error rate on the samples from $\mathcal{X}_1$ which was not used during the training. This procedure is repeated k times, setting apart subsequently all the folds and using them as an evaluation feedback. All k error-rate estimates are then combined to form the final cross-validation error estimate E^{cv} which can be used as an estimate of the real testing error. The choice of k needs to be made in advance, and may be influenced by the available computational power and/or by the size of the training set. Lower k is less computationally expensive but may introduce a bias as the training set sizes used during the cross-validation are farther away from the size of the whole training set $\mathcal{X}$, which is used for the final classifier training.

The Gaussian SVM is a non-linear classification tool parametrized by the misclassification cost C and by the parameter γ determining the kernel width.[1] These two parameters influence the shape of the constructed decision boundary and consequently the performance of the classifier. A common way of optimizing the performance of a SVM w.r.t. these parameters is to minimize the error estimates E^{cv} obtained through k-fold cross-validation:

$$(C^{\text{opt}}, \gamma^{\text{opt}}) = \arg \min_{(C,\gamma) \in \mathcal{P}^C \times \mathcal{P}^\gamma} E^{\text{cv}}(C, \gamma), \qquad (1)$$

where $\mathcal{P}^C \times \mathcal{P}^\gamma$ is a pre-defined grid of parameter values. Once the best parameters $(C^{\text{opt}}, \gamma^{\text{opt}})$ are obtained, they are used to retrain the SVM on the entire training set $\mathcal{X}$. The resulting SVM is ready to be used for real predictions (or for the predictions on the testing set).

2.2 Application to steganalysis

As a consequence of the inability to accurately model natural images, by far the most common type of steganography nowadays is the steganography by cover *modification* – rather than trying to create a stego image that is consistent with a certain model, the goal is to minimize the embedding impact – to minimize an appropriately designed distortion function. This approach has notably advanced the security of steganographic schemes, especially after bringing advanced coding schemes into the field [7]. The adaptive algorithm HUGO [12] is a good example of this approach. Despite the recent BOSS competition [11] HUGO remains the most secure spatial domain steganographic algorithm today (August 2011).

An important implication of steganography by cover modification is that the stego image typically differs from its cover counterpart only in a small fraction of pixels (or DCT coefficients). Additionally, the changes may be in those areas of the image that are difficult to model. Consequently, the two features forming each cover-stego (C-S) pair are likely to be very close to each other and much further from features of other images. Even though the goal of steganalysis is to create such a feature space where the cover and stego features are separated as well as possible, this cover-stego pairing seems to be inevitable as steganography advances. It is exactly this formation of C-S pairs that is responsible for the inappropriateness of the standard k-fold cross-validation (as implemented in LIBSVM) for steganalysis.

LIBSVM, created as a general classification tool, does not take into account that the pairs of training samples that lie very close to each other (in $\mathcal{F}$) have opposite labels. Consequently, at the point when individual folds $\mathcal{X}_i$ are formed, these pairs are often not preserved – the cover and the corresponding stego features belong to different folds. Mathematically, only about $1/k$ of the pairs[2] is preserved. For example for the frequently

[1] For two features $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{F}$, the Gaussian kernel is defined as $\exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$; smaller γ thus implies a wider kernel.

[2] For k folds, the probability that the cover's stego counterpart is in the same fold is $\frac{N-k}{k(N-1)}$, which simplifies roughly to $\frac{1}{k}$ when $k \ll N$.

used five-fold cross-validation, only about 20% of the C-S pairs are together in the same fold. This means that when the performance of SVM is evaluated on the fold $\mathcal{X}_i$, the majority (roughly 80%) of all the examples in $\mathcal{X}_i$ have the second feature from their pair in the set the SVM was trained on. This is certainly undesirable and may negatively influence the results of the cross-validation procedure. In the next section, we will demonstrate how serious this issue is on an example of steganalysis of a real steganographic scheme.

3 Experiment

In this section, we demonstrate the effect of improperly formed cross-validation sets on a JPEG domain steganographic algorithm nsF5 [8]. For the purpose of this experiment, we downloaded the BOSSbase image database [1] (v1.00) consisting of 10,000 spatial-domain images and JPEG-compressed them using Matlab's `imwrite` with quality factor 75. We used the simulator of the optimally coded nsF5 algorithm[3] to create stego images carrying a range of different payloads from 0.01 to 0.20 bits per nonzero AC DCT coefficient (bpac). We used the 548-dimensional feature space $\mathcal{F}$ formed by Cartesian-calibrated Pevný features [10] [4] for steganalysis.

For every single payload, we trained a separate SVM on a randomly selected half of the image database, and tested the performance on the other half. Note the formulation "half of the image database" used in the previous sentence, implying that the C-S pairs are preserved during this division. The preservation of the C-S pairs *at this stage* seems to be self-evident and is usually correctly performed in research publications on steganalysis. However, while the preservation of the C-S pairs during the cross-validation is equally important, this issue may be overlooked when using SVM packages, such as LIBSVM.

3.1 Two different results

The Gaussian SVM was trained using five-fold cross-validation on the following grid of parameters C and γ:

$$(C,\gamma) \in \left\{ \left(10^{\alpha}, \frac{1}{d}2^{\beta}\right) \middle| \alpha = -3,\dots,4, \beta = -3,\dots,3 \right\}, \quad (2)$$

where $d = 548$ is the feature space dimensionality. Two different strategies for creating the folds $\mathcal{X}_i$ were used:

1. Implicit cross-validation (as implemented in LIBSVM),
2. Manually created folds preserving C-S pairs.

We note that in [14], the first strategy was called the standard cross-validation, while the second one "paired cross-validation."

Our experiment was conducted only over a single split of the image database into training and testing parts as the purpose is to compare the two fold-forming strategies rather than to report a statistically reliable detection performance. The performance is reported in terms of the testing error E^{tst} obtained from the testing set when the threshold was set to minimize the overall error (i.e., both types of error, false alarms and missed detections, were treated equally). The resulting errors for both fold-forming strategies are plotted in Figure 1 together with the CV estimates E^{cv} coming from the corresponding (best) points of the grid (2).

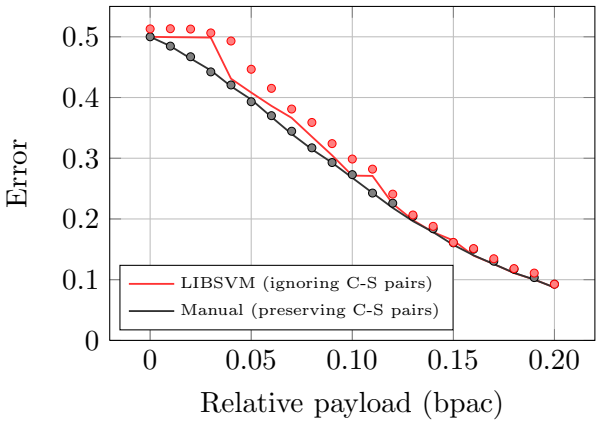

Figure 1: Real testing errors E^{tst} (solid line) and the corresponding CV estimates E^{cv} (dots) across different payloads of nsF5 and for two different strategies of forming cross-validation folds.

There are two patterns to be observed in Figure 1. First, ignoring the C-S pairs delivers suboptimal performance over manually created folds that preserve the pairs. This is barely noticeable for large payloads, but becomes stronger with increasing payload, culminating in a "jump" from roughly 43% to undetectability (50%) between 0.03 and 0.04 bpac. The second pattern is a growing overshoot of the error estimate E^{cv} over the real testing error E^{tst} as the payload decreases. This difference is about 7% at 0.04 bpac and then vanishes

[3] The simulator of nsF5 is available at `http://dde.binghamton.edu/download/nsf5simulator/`.

[4] Feature extractor is available at `http://dde.binghamton.edu/download/ccmerged/`.

when both errors jump to random guessing. Notice that, for very small payloads, the values of E^{cv} are slightly above the random guessing value of 0.5.

3.2 Explanation

In the five-fold cross-validation that ignores the C-S pairs, roughly 80% of the validation samples have their C-S counterpart in the set on which the SVM was trained. Figure 2 is a 2D illustration of what happens when SVM predicts the class labels on such examples. In case of a simple decision boundary (bottom portion of the figure), the classifier trained on C-S pairs yields a similar predictor as the classifier trained on C-S pairs with missing features. If these missing features are then used for the estimation of the testing error E^{tst}, they are classified correctly and the obtained error estimate E^{cv} is indeed a good estimate of the real testing error (for which *all* training C-S pairs are used in the training phase!). On the other hand, when the classes are less distinguishable (small payloads), the decision boundary learnt from all C-S pairs and the one learnt from a set consisting of only a single feature from every C-S pair may be quite different, as is illustrated by the top part of Figure 2. Most of the missing pairs are then classified incorrectly during validation as they are assigned the label of their counterpart that appeared in the classifier training. Consequently, the CV error E^{cv} is higher (and often *much* higher as will be shown later in this section) than the testing error E^{tst}.

The reasoning above explains the growing overshoot of E^{cv} over E^{tst} (when the incorrect implementation of cross-validation is used) as the payload decreases, i.e., when the two classes are less distinguishable and the decision boundary is more complex. But this would not be sufficient by itself for the explanation of the different *testing* performance between both types of cross-validation as the *whole* training set is used for the final SVM training when all the training C-S pairs are preserved in both cases.

If the overshoot was roughly the same for all the points in the grid (2), the resulting optimal parameters $(C^{\text{opt}}, \gamma^{\text{opt}})$ and thus the testing errors E^{tst} would be the same for both cross-validation strategies, and the only consequence of the incorrect cross-validation would be the inability to accurately estimate the testing error as the overshoot differs from payload to payload. Unfortunately, that is not the case – the complexity of the constructed decision boundary does not depend only on the class distinguishability, but also on the SVM hyper-parameters (C, γ). The larger is the misclassification cost C and the narrower is the Gaussian kernel (the larger is γ), the more complex is the learnt class boundary. Therefore, the overshoot is much

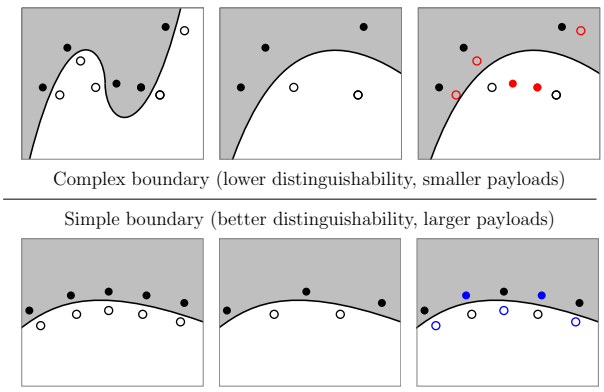

Figure 2: Illustration of what happens when C-S pairs are separated. Top: the case of a complex decision boundary; Bottom: the case of a simple decision boundary. Left: learnt boundary when all C-S pairs are included; Middle: learnt boundary when one feature from every C-S pair is missing; Right: Correctly (blue) and incorrectly (red) classified points when the missing features are used for testing.

higher for larger C and γ. As a result, during the incorrectly formed cross-validation the optimal parameters C^{opt} and/or γ^{opt} are being artificially shifted to smaller (and sub-optimal) values as payload decreases, and thus the final testing error is getting higher than it would be when the correct parameters $(C^{\text{opt}}, \gamma^{\text{opt}})$ were found.

We experimentally confirmed this behavior and demonstrate it in Figure 3 where we show the results of the grid-search for a fixed small payload 0.02 bpac. The top part of the figure shows the correctly performed grid-search and shows the lowest found $E^{\text{cv}} = 0.4671$ at the point $(C^{\text{opt}}, \gamma^{\text{opt}}) = (10^4, \frac{1}{d}2^{-2})$ which corresponds to the real testing error of nsF5 at this payload for our experimental setup (compare to Figure 1). On the other hand, when the incorrect cross-validation is performed, the very same point of the grid results in the error estimate $E^{\text{cv}} = 0.7084$, i.e., there is an overshoot by more than 20% (see the point marked with a circle in the bottom part of Figure 3). Instead, as the "best" point of the grid was declared as the point marked by a cross lying in the random guessing area in the left part of the grid. Note that the highest overshoot is indeed in the right top part of the grid with high values of the cost C and with a narrow kernel (large γ), i.e., in the areas where the SVM forms complex decision boundaries. Note that error rates over 50% are suspicious by themselves as the worst possible error should not exceed 50% (random guessing).

To complete our understanding of the inner workings of the incorrectly performed cross-validation, let us make a closer inspection of the point marked by a cir-

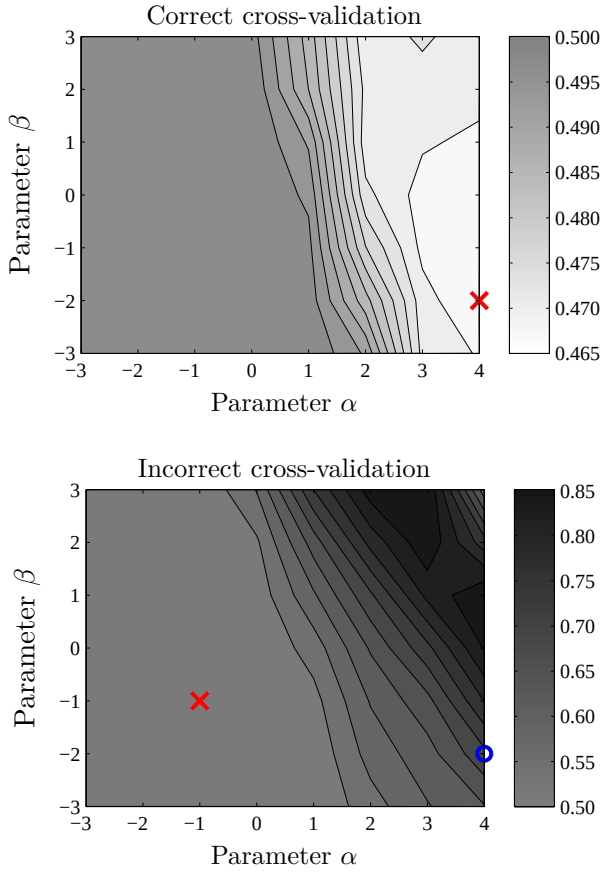

Figure 3: Contour graphs of the grid-search error estimates E^{cv} as functions of the parameters $C = 10^{\alpha}$ and $\gamma = \frac{1}{d}2^{\beta}$ at payload 0.02 bpac for both correctly (top) and incorrectly (bottom) performed cross-validation. Crosses mark the points with the lowest CV errors. Circle marks the point commented on in the text.

cle in Figure 3 ($C = 10^4$ and $\gamma = \frac{1}{d}2^{-2}$). Each of the folds $\mathcal{X}_i, i = 1, \ldots, 5$ consists of two disjoint parts $\mathcal{X}_i = \mathcal{X}_i^{\mathrm{P}} \cup \mathcal{X}_i^{\mathrm{B}}$, where the set $\mathcal{X}_i^{\mathrm{P}}$ is the union of all the C-S pairs from $\mathcal{X}_i$ and the set $\mathcal{X}_i^{\mathrm{B}}$ contains those samples $\mathbf{x} \in \mathcal{X}_i$ whose C-S counterparts appear in a different fold and thus were used for the SVM training. Let $n_{\mathrm{P}} = |\mathcal{X}_i^{\mathrm{P}}|$, $n_{\mathrm{B}} = |\mathcal{X}_i^{\mathrm{B}}|$ be the sizes of sets $\mathcal{X}_i^{\mathrm{P}}$ and $\mathcal{X}_i^{\mathrm{B}}$. Let $E_{\mathrm{P}}^{\mathrm{cv}}$ and $E_{\mathrm{B}}^{\mathrm{cv}}$ be the validation errors obtained only from the sets $\mathcal{X}_i^{\mathrm{P}}$ and $\mathcal{X}_i^{\mathrm{B}}$, respectively. Then the cross-validation error obtained from the ith fold is calculated as $E^{\mathrm{cv}} = (n_{\mathrm{P}}E_{\mathrm{P}}^{\mathrm{cv}} + n_{\mathrm{B}}E_{\mathrm{B}}^{\mathrm{cv}})/(n_{\mathrm{P}} + n_{\mathrm{B}})$. The values of n_{P}, n_{B}, $E_{\mathrm{P}}^{\mathrm{cv}}$, $E_{\mathrm{B}}^{\mathrm{cv}}$ and E^{cv} for this particular point of the grid are shown in Table 1. We can see that while the CV error $E_{\mathrm{P}}^{\mathrm{cv}}$ always correctly estimates the real testing error (values around 47%), the error rate $E_{\mathrm{B}}^{\mathrm{cv}}$ lies above 75%, confirming that only the validation samples from $\mathcal{X}_i^{\mathrm{B}}$ (where C-S pairs are not preserved) are responsible for the error overshoot. As roughly 80% of the validation samples are from $\mathcal{X}_i^{\mathrm{B}}$, the resulting error $E^{\mathrm{cv}} \approx 70\%$. This is in agreement with the 2D illustration in Figure 2 (top) – the red points correspond to the artificially misclassified samples from $\mathcal{X}_i^{\mathrm{B}}$.

Fold	Size	n_{P}	n_{B}	$E_{\mathrm{P}}^{\mathrm{cv}}$	$E_{\mathrm{B}}^{\mathrm{cv}}$	E^{cv}
$\mathcal{X}_1$	2001	394	1607	.4772	.7561	.7011
$\mathcal{X}_2$	1980	398	1582	.4799	.7794	.7192
$\mathcal{X}_3$	2047	406	1641	.4754	.7782	.7181
$\mathcal{X}_4$	1990	430	1560	.4744	.7750	.7101
$\mathcal{X}_5$	1982	388	1594	.4665	.7501	.6937
Mean	2000	403.2	1596.8	.4747	.7676	.7084

Table 1: Results of the incorrect five-fold cross-validation at the grid point $(10^4, \frac{1}{d}2^{-2})$ – the point marked with a circle in Figure 3. All ymbols are defined in the text.

4 Conclusion

We showed that the implicit k-fold cross-validation procedure implemented in the popular machine learning package LIBSVM should not be used for steganalysis as it does not preserve the pairs of cover-stego images. Instead, manually created folds taking this specifics of steganalysis into account should be used, as incorrectly created folds result in misleading values of error estimates and consequently in a suboptimal performance. These negative impacts of the incorrect cross-validation manifest themselves stronger when the class distinguishability is lower, i.e., for smaller payloads. This makes the whole problem even more important as the secure payload of a given steganographic algorithm is defined as a maximal payload that can be embedded without being detected (the best possible detector is random guesser). For example, ignoring cover-stego pairs, one might conclude that the secure payload of nsF5 is around 0.03 bpac (see Figure 1). However, that would be an incorrect conclusion as we can detect nsF5 at this payload with error around 43% if we take the cover-stego pairing into account.

We would like to conclude this technical report by pointing out a few examples of published work with results that may have been affected by the improper cross-validation. All of the following publications exhibit a 'jump' in the reported performance to random guessing, similar to the one in Figure 1 in this report. The publications are: [6] (Figures 3 and 4), [7] (Figures 10 and 12), [3] (Figure 6), [5] (Figure 4), [4] (Figure 2). Note that the list contains only those publications where the results are reported graphically and where the focus is on smaller payloads as there the problem is 'visible'. But ALL the steganalysis works that used the implicit cross-validation of SVM have been affected.

5 Acknowledgments

The work on this report was supported by Air Force Office of Scientific Research under the research grant number FA9550-09-1-0147. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of AFOSR or the U.S. Government.

References

[1] P. Bas, T. Filler, and T. Pevný. Break our steganographic system – the ins and outs of organizing boss. In T. Filler, editor, *Information Hiding, 13th International Workshop*, Lecture Notes in Computer Science, Prague, Czech Republic, May 18–20, 2011. Springer-Verlag, New York.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

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

[4] T. Filler and J. Fridrich. Minimizing additive distortion functions with non-binary embedding operation in steganography. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, December 2010.

[5] T. Filler and J. Fridrich. Steganography using gibbs random fields. In J. Dittmann, S. Craver, and P. Campisi, editors, *Proceedings of the 12th ACM Multimedia & Security Workshop*, MM&Sec '10, pages 199–212, Rome, Italy, September 9–10, 2010. ACM.

[6] T. Filler and J. Fridrich. Design of adaptive steganographic schemes for digital images. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XIII*, volume 7880, pages OF 1–14, San Francisco, CA, January 23–26, 2011.

[7] T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 2010. Under preparation.

[8] J. Fridrich, T. Pevný, and J. Kodovský. Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities. In J. Dittmann and J. Fridrich, editors, *Proceedings of the 9th ACM Multimedia & Security Workshop*, pages 3–14, Dallas, TX, September 20–21, 2007.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer-Verlag, 2009.

[10] J. Kodovský and J. Fridrich. Calibration revisited. In J. Dittmann, S. Craver, and J. Fridrich, editors, *Proceedings of the 11th ACM Multimedia & Security Workshop*, pages 63–74, Princeton, NJ, September 7–8, 2009.

[11] T. Pevný, T. Filler, and P. Bas. Break Our Steganographic System, `http://boss.gipsa-lab.grenoble-inp.fr`.

[12] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In R. Böhme and R. Safavi-Naini, editors, *Information Hiding, 12th International Workshop*, volume 6387 of Lecture Notes in Computer Science, pages 161–177, Calgary, Canada, June 28–30, 2010. Springer-Verlag, New York.

[13] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.

[14] V. Schwamberger and M. O. Franz. Simple algorithmic modifications for improving blind steganalysis performance. In J. Dittmann, S. Craver, and P. Campisi, editors, *Proceedings of the 12th ACM Multimedia & Security Workshop*, MM&Sec '10, pages 225–230, Rome, Italy, September 9–10, 2010. ACM.