Towards Dependable Steganalysis

Tomáš Pevný[a,b] and Andrew D. Ker[c]

[a]Cisco Systems, Inc., Cognitive Research Team in Prague, Czech Republic
[b]Agent Technology Center, Czech Technical University in Prague,
Karlovo naměstí 13, 121 35 Prague 2, Czech Republic.
[c]Oxford University Department of Computer Science, Parks Road, Oxford OX1 3QD, UK.

ABSTRACT

This paper considers the research goal of dependable steganalysis: where false positives occur once in a million or less, and this rate is known with high precision. Despite its importance for real-world application, there has been almost no study of steganalysis which produces very low false positives. We test existing and novel classifiers for their low false-positive performance, using millions of images from Flickr. Experiments on such a scale require considerable engineering. Standard steganalysis classifiers do not perform well in a low false-positive regime, and we make new proposals to penalise false positives more than false negatives.

1. INTRODUCTION

This paper considers the research goal of *dependable* steganalysis. By this we mean steganalysis that could potentially be suitable for forensic analysis, which requires two properties:

(a) The false positive rate of the detector should be known with high precision.

(b) The false positive rate of the detector should be very low (10^{-6}, or even 10^{-9}).

The aim is to move steganalysis more towards real-world applications.[1] Note that we focus on false positive rates because false negative rates can only be defined when there is a simple alternative hypothesis (for example that a steganographer uses a known embedding algorithm with a known length of payload, or payload with exactly known distribution) which is not likely to fit real-world scenarios. Furthermore, in a real world where true positives are very scarce, the false positives dominate the failure cases.

Both aims are challenging. (a) is difficult because steganalysis is highly dependent on context: the cover source,[2] scene content, and potentially unknown other factors all influence accuracy. There has been some research aimed in this direction[3] but empirical evidence suggests that false alarm rates are not robust in practice. (b) has been relatively little studied: practically every piece of steganalysis literature focuses on error rates under equal priors, or area under an ROC curve, metrics little affected by low false positive performance. One difficulty with (b) is that empirical tests of very low false alarm rates are simply impossible unless the evidence base is enormous.

This paper is an initial move towards (b). Using standard steganalysis features, we modify classifiers to optimize low false positive rates, and provide a very large real-world evidence base (millions of images) to evaluate the results. To do so, we examine the low false-positive region of the ROC directly, and also use a new metric.

In the following subsections we discuss the benchmarking metrics for steganalysis and establish notation. In Sect. 2 we discuss literature on classification that prioritizes one class over another. We propose new linear classifiers in Sect. 3, and adapt ensemble classification to the low false-positive regime in Sect. 4. We measure the performance of the new classifiers, single and in ensembles, in Sect. 5, and draw conclusions in Sect. 6.

Further author information:
T. Pevný: E-mail: pevnak@gmail.com, Telephone: +420 22435 7608
A. D. Ker: E-mail: adk@cs.ox.ac.uk, Telephone: +44 1865 283530

1.1 Benchmarks for steganalysis

Early steganalysis literature struggled to reduce the performance envelope of a detector (false positive and false negative rates as the payload size varies) to simple benchmarks. See Ref. 4 for a survey, which includes some of the popular options from the literature at the time, and a more recent discussion in Ref. 5. For at least the last five years, however, by far the most popular benchmark is the *minimal misclassification rate under equal priors* (assuming that the payload size is fixed). This can be defined by

$$P_{\mathrm{E}} = \tfrac{1}{2} \min(P_{\mathrm{FP}} + P_{\mathrm{FN}})$$

where P_{FP} and P_{FN} represent the false positive and false negative rate and the minimum ranges over the ROC curve. If, in application, the detector expects to see equal numbers of true positive and negative classes, and the costs of misclassification are symmetrical, then this is indeed the threshold that should be chosen and P_{E} represents the error rate of such a detector.

The P_{E} benchmark is useful for demonstrating advances in steganalysis feature design or classification, but as a measure of practical performance it seems rather far from reality. In almost any realistic problem domain, the vast majority of images transmitted will be covers, because most transmissions are not covert. Even if each false positive result *costs the detector the same* as a false negative (which itself is a dubious assumption), that does not imply an equal weighting between P_{FP} and P_{FN}. When positives are observed rarely, false negatives have fewer opportunities to happen, compared with false positives.

There is no perfect benchmark, and every problem application will have a slightly different preference for the classifier's performance envelope. However, *dependable* steganalysis requires low false positive rates, and in practice the cost of a false negative is likely to be relatively low (an enemy steganographer will probably act more than once). So we propose a metric which we call FP-50, the false positive rate when the false negative rate is 50%. This benchmark is not new, and indeed it was advocated in Ref. 4, but it has not been used much in steganalysis before now.

We also need to make more clear separation of training and testing sets. Following the gold standard of machine learning, we will use *three* sets of data: *training* data to learn a classifier, *validation* data to optimize hyperparameters and thresholds, and *testing* data to measure a final *single* $(P_{\mathrm{FP}}, P_{\mathrm{FN}})$ pair. Training and validation sets are selected repeatedly from the same pool, but testing is completely disjoint. If detection thresholds were set using results from the testing data (which is typical for the steganalysis literature, when drawing a ROC curve) this would be considered a form of cheating. We will still draw ROC curves, using a semi-log plot to display the low false-positive region directly, but they will be from the validation set. Our true benchmark is the final false positive/negative rate on the never-before-seen testing set.

1.2 Notation

We use the following notation throughout the paper. P^{c} (respectively, P^{s}) denotes the probability distribution of all cover images (respectively, stego images, with an implicit embedding method and payload or payload distribution). $\mathcal{I}^{\mathrm{c}}$ ($\mathcal{I}^{\mathrm{s}}$) is a finite set of cover (stego) images, typically for training a classifier. $\{x_i\}_{i\in\mathcal{I}}$ represents the matrix of features extracted from images in set $\mathcal{I}$, arranged in rows. The domain of the features is $\mathcal{X}$. μ_c and $\mathbf{C}_c$ (μ_s and $\mathbf{C}_s$) denote the empirical mean and covariance of features from $\mathcal{I}^{\mathrm{c}}$ ($\mathcal{I}^{\mathrm{s}}$).

Throughout, λ will be an optional regularisation parameter (in the experimental results we will always set it to zero). $\mathrm{I}[x]$ denotes the indicator function which is equal to 1 when x is true, 0 otherwise.

2. RELATED WORK

Any classification algorithm on continuous data can be adapted to favour false positives over false negatives, by moving a decision boundary. This is the traditional way to trace the receiver operating characteristic (ROC) curve, but it has no guarantee of optimality.

The proper foundation is *classification with imbalanced costs*,[6] which despite its importance has not been studied in steganalysis. Most practical work in the machine learning literature uses a Bayesian framework, with known costs of false positives and false negatives, together with prior probabilities of encountering positive and

negative cases in the data. An example of an algorithm for class-imbalanced problems is the cost-sensitive (or *weighted*) support vector machine (SVM), optimizing the following cost function

$$\arg\min_{\rho,w} \ \frac{\lambda}{2}\|w\|_2^2 + \frac{\eta}{|\mathcal{I}^{\mathrm{c}}|}\sum_{i\in\mathcal{I}^{\mathrm{c}}}\max\{0, w^{\mathrm{T}}x_i - \rho\} + \frac{1-\eta}{|\mathcal{I}^{\mathrm{s}}|}\sum_{i\in\mathcal{I}^{\mathrm{s}}}\max\{0, \rho - w^{\mathrm{T}}x_i\}, \tag{1}$$

where λ is the regularization parameter (here using L_2 regularization, though other options are possible) and η balances the costs of misclassification of cover and stego samples. ρ is the margin hyperparameter.

Another option is to turn traditional logistic regression into a maximum a posteriori problem with a prior, optimizing

$$\arg\min_{\rho,w} \ \frac{\lambda}{2}\|w\|_2^2 + \frac{\eta}{|\mathcal{I}^{\mathrm{c}}|}\sum_{i\in\mathcal{I}^{\mathrm{c}}}\log\left(1+\exp\left(w^{\mathrm{T}}x_i - \rho\right)\right) + \frac{1-\eta}{|\mathcal{I}^{\mathrm{s}}|}\sum_{i\in\mathcal{I}^{\mathrm{s}}}\log\left(1+\exp\left(\rho - w^{\mathrm{T}}x_i\right)\right). \tag{2}$$

In both cases we have an additional hyperparameter η, balancing the cost of false positives and negatives on the training set. It is difficult to justify a correct value of η unless the problem domain is very well-understood (if there is a known cost of error on each class, which is plausible, as well as a known proportion of each class that will be encountered in the wild, which is not), so it might be optimized using cross-validation along with the other hyperparameters. Thus it can only indirectly target a benchmark such as FP-50.

In fact, miniziming the FP-50 benchmark corresponds to a different problem: *Neyman-Pearson classification* aims to minimize the false negative rate such that the false positive rate is below some threshold. This is more applicable in security scenarios, as the number of false positives that the user is willing to tolerate can be usually determined. Surprisingly, few works in machine learning literature deal with Neyman-Pearson classification[7,8] and to the best of our knowledge there is no algorithm directly optimizing this error. This is usually swept under the carpet by claiming that the same classifier can be obtained with an appropriate cost, such as η above. This approach, where η and λ are optimized on training data to minimize FP-50, is used in experiments in Subsection 5.2.

3. PROPOSALS FOR LINEAR CLASSIFIERS

In this work, we will try to target the FP-50 benchmark more directly. We will first present methods for single linear classifiers, and move to ensembles of linear classifiers (in a way that also targets Neyman-Pearson classification) in the following section.

We present two approaches to this problem: one minimizing upper bounds on the FP-50 benchmark, the other using convex surrogates for it. It turns out that these approaches are, in some sense, equivalent.

3.1 Probabilistic approach

Consider the FP-50 benchmark. The optimal classifier minimizing it is

$$\arg\min_{f\in\mathcal{F}} \ \mathbb{E}_{x\sim P^{\mathrm{c}}}\Big[\mathrm{I}\big[f(x) > \operatorname{median}\{f(x)|x\sim P^{\mathrm{s}}\}\big]\Big], \tag{3}$$

where $\mathcal{F}$ is the set of all possible classifiers* of the form $\mathcal{X}\mapsto\mathbb{R}$. Optimizing (3) is impractical, since a) P^{c} and P^{s} are unknown, and b) median is not a differentiable function so the optimisation is NP-complete. We tackle these problems by using finite sets of training samples instead of the distributions, and replacing median with mean.

For further simplicity, we restrict $\mathcal{F}$ to be the set of linear classifiers. Due to the well-known kernel trick,[9] the linear classifiers below could be adapted to non-linear circumstances, but instead we will follow the present state-of-art in steganalysis by regaining nonlinearity via an ensemble of linear classifiers.

*$\mathcal{F}$ is called the *hypothesis space* in the jargon of machine learning literature.

Incorporating the above into (3), the problem becomes

$$\arg\min_{w\in\mathbb{R}^d} \mathbb{E}_{x\sim P^c}\Big[\mathrm{I}\big[w^{\mathrm{T}}x > w^{\mathrm{T}}\mu_s\big]\Big] = \arg\min_{w\in\mathbb{R}^d} \mathbb{P}_{x\sim P^c}\big[w^{\mathrm{T}}x > w^{\mathrm{T}}\mu_s\big], \tag{4}$$

where μ_s denotes the mean of stego features. By replacing median with mean, we are assuming that about 50% of stego features projected on w lie beyond their mean, which should be true for moderately symmetrical distributions. Reminiscent of Vapnik's technique,[10] we use probabilitic inequalities to find upper bounds for (4), which we can optimize efficiently. Different optimization problems arise from different inequalities.

Quadratic Chebyschev Minimizer. Applying the standard Chebychev inequality[†] to (4), we solve

$$\arg\min_{w\in\mathbb{R}^d} \frac{w^{\mathrm{T}}\mathbf{C}_c w}{\big((\mu_s-\mu_c)^{\mathrm{T}}w\big)^2}. \tag{5}$$

This problem has a close relationship with the Fisher linear discriminant (FLD): the only difference is that it disregards the shape (covariance) of the stego distribution, which follows because we target the mean of the stego distribution. It has an analytic solution which, it can be shown, corresponds to finding a single unregularized projection by the calibrated least squares (CLS) method that we proposed in Ref. 5. (We also tested other polynomial versions of Chebyschev's inequality, with no success.)

Exponential Chebyschev Minimizer. Applying the exponential version of Chebychev's inequality, also known as the MGF bound[‡] to (4), we solve

$$\arg\min_{w\in\mathbb{R}^d} \sum_{i\in\mathcal{I}^c} e^{t(x_i-\mu_s)^{\mathrm{T}}w}. \tag{6}$$

By varying the scalar t, this balances inequalities based on all moments of the cover distribution. In optimization it is superfluous, since it can be absorbed by w. Equation (6) does not have an analytic solution, but the objective function is strongly convex and fast-converging numerical optimizers can be used.

3.2 Machine learning approach

Applying Chebyschev's inequalities can be alternatively viewed as approximating the ideal cost function assigning 1 to cover values projected beyond the stego mean, and zero otherwise. Machine learning algorithms use several surrogates of this function to make the problem (4) convex and solvable in polynomial time. Popular convex surrogates for $\mathrm{I}[y]$ include: hinge, $\max\{0, 1-y\}$; truncated square, $\max\{0, 1-y\}^2$; square, $(1-y)^2$; exponential, e^{-y}; logistic, $\log(1+e^{-y})$. They are depicted in Figure 1.

For some of these, including $\mathrm{I}[\cdot]$ itself, hinge, and truncated square, there may be an infinite number of solutions to optimizing (4). This is typically solved by using regularization (usually Tikhonov), which corresponds to a preference for simple solutions. Below, optimization problems with different loss functions are shown and their properties discussed. All formulations include L_2 regularization, controlled by a hyperparameter λ, but in our experiments we will set $\lambda = 0$ to turn off regularization. L_2 regularization was chosen as it has a smooth derivative, which is favourable for optimization; L_1 regularization promoting sparse solutions is also possible, but the optimization would be more difficult and it will require the use of special solvers.

Hinge loss. This has been popularized by its use in SVMs, and it requires regularisation. Putting hinge loss into (4) yields

$$\arg\min_{w\in\mathbb{R}^d} \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{|\mathcal{I}^c|}\sum_{i\in\mathcal{I}^c} \max\left\{0, 1-w^{\mathrm{T}}(\mu_s - x_i)\right\},$$

which is reminscent of the problem solved in one-class SVM.[11,12] Although the same results can be probably obtained by using weighted SVMs, the proposed formulation has one fewer hyperparameter and it directly optimizes the FP-50 criterion.

[†]The following version: $\mathrm{P}[Y > \epsilon] \le \mathbb{E}\big[(Y-\mu_Y)^2\big]/(\epsilon-\mu_Y)^2$, for $\epsilon > \mu_Y$.

[‡]The following version: $\mathrm{P}[Y > \epsilon] \le \mathbb{E}\big[e^{t(Y-\epsilon)}\big]$, for all $t > 0$.

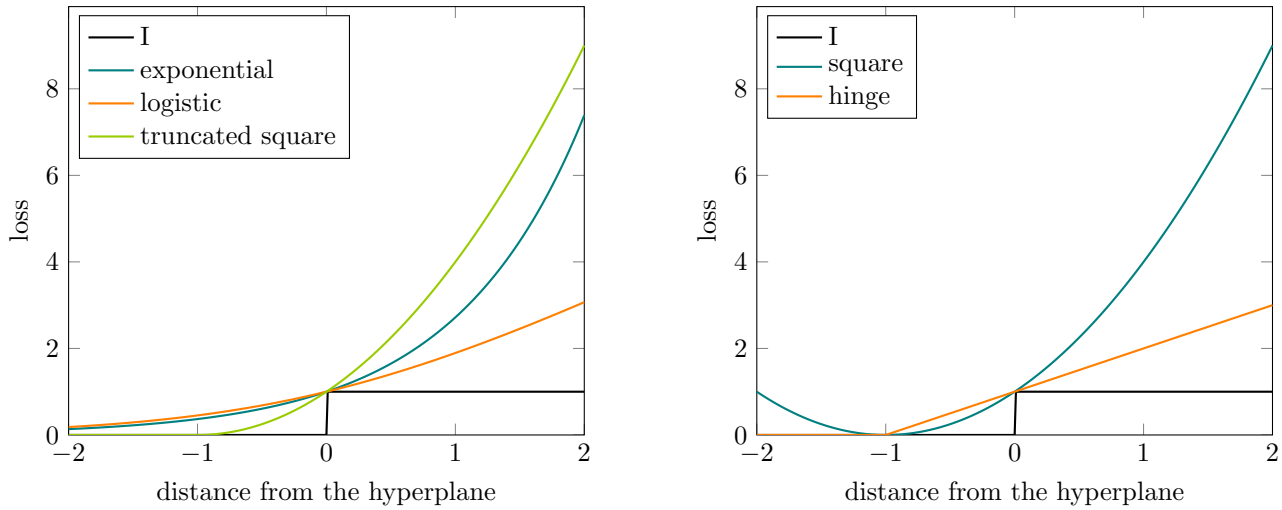

Figure 1: Convex surrogates for the 0-1 loss function.

Truncated square loss. This is also sometimes used with SVMs. Putting truncated squared loss into (4) (again requiring regularization to avoid infinitely many solutions) yields

$$\arg\min_{w\in\mathbb{R}^d} \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{|\mathcal{I}^c|}\sum_{i\in\mathcal{I}^c} \max\left\{0, 1 - w^{\mathrm{T}}(\mu_s - x_i)\right\}^2.$$

The advantage of square loss over hinge loss is that it is smooth, which simplifies the optimization by allowing particularly effective stochastic gradient methods.

Square loss. This is advantageous because the optimization has the analytic solution

$$\arg\min_{w\in\mathbb{R}^d} \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{|\mathcal{I}^c|}\sum_{i\in\mathcal{I}^c} \left(1 - w^{\mathrm{T}}(\mu_s - x_i)\right)^2 = (\mathbf{C} + \lambda\mathbf{I})^{-1}(\mu_s - \mu_c), \tag{7}$$

where where $\mathbf{C}$ denotes covariance matrix of *cover* samples centered at the mean of *stego* samples, $\mathbf{C} = \frac{1}{|\mathcal{I}^c|}\sum_{i\in\mathcal{I}^c}(\mu_s - x_i)(\mu_s - x_i)^{\mathrm{T}}$.

Note that this is almost identical to the previously-described CLS method (5), if $\lambda = 0$, but with the samples centered differently.

Exponential loss. Intuitively, an exponential penalty on misclassified covers is aligned with our goal of reducing false positives to very low rates. Exponential loss is used in Adaboost,[13] and it does not neccesarily require regularization, although this can be added to force the classifier toward simpler or sparser solutions. Optimization problem (4) with an exponential loss surrogate becomes

$$\arg\min_{w\in\mathbb{R}^d} \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{|\mathcal{I}^c|}\sum_{i\in\mathcal{I}^c} e^{-w^{\mathrm{T}}(\mu_s - x_i)}. \tag{8}$$

Turning off the regularization, we have recovered the Exponential Chebyshev Minimizer (6).

Logistic loss. This is used in logistic regression, which can provide a probability estimate of class membership (more than just a binary classification) under the right conditions. The optimization becomes

$$\arg\min_{w\in\mathbb{R}^d} \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{|\mathcal{I}^c|}\sum_{i\in\mathcal{I}^c} \log\left(1 + e^{-w^{\mathrm{T}}(\mu_s - x_i)}\right). \tag{9}$$

Similarly to exponential, logistic loss does not necessarily require regularization. Its shape is similar to that of hinge loss, but it has the considerable advantges of being Lipschitz, strongly convex, and infinitely many times differentiable. These are favorable properties for optimization, particularly online optimization.

4. PROPOSALS FOR ENSEMBLE CLASSIFIERS

The classifier used in state-of-the-art steganalysis is an ensemble of FLDs. It has come to dominate the literature because of its simplicity, speed of training, and results with *rich models*.[14] Our linear classifiers based on convex surrogates to (4) can also be used in an ensemble, but the ensemble parameters must be adapted to target the low false-positive regime.

The ensembles used in Ref. 15 consist of a set of base learners diversified by random subspace sampling, which means that each base learner classifier operates on a randomly selected subspace of the original feature space. The subspace sampling makes the training faster, as the complexity of training linear classifiers depends super-linearly (in the case of FLD cubically) on the dimension of the input space. Another side-effect of subspace sampling, to our knowledge not discussed so far in the literature, is that the size of the subspace acts as a regularization parameter controlling the complexity of individual classifiers within the ensemble and preventing over-fitting. Naturally, smaller subspace dimensions make individual classifiers less over-fitted.

4.1 Fusing classifiers to optimize low false-positives

We stick broadly to the ensemble framework used in Ref. 15 – binary classifiers voting with equal weight – but adjust various thresholds used in it.

To formalize the problem, we assume the ensemble consists of the set of classifiers $\left\{f_i | f_i : \mathbb{R}^d \mapsto \mathbb{R}\right\}_{i=1}^{l}$. The output of the ensemble is equal to

$$F(x) = \operatorname{sign}\left[\frac{1}{l}\sum_{i=1}^{l} \mathrm{I}\big[f_i(x) > t_i\big] - t_e\right],$$

where $\{t_i\}_{i=1}^{l}$ are thresholds for the individual classifiers and t_e is the threshold of the ensemble, the number of positive votes required for a positive classification. Determination of these is part of the training, because $F : \mathbb{R}^d \mapsto \{-1, +1\}$.

Kodovsky[15] optimizes thresholds t_i, separately for each classifier, to optimize P_{E} on the training set, and fixes $t_e = 0.5$. The hyperparameters of the ensemble – the number l of base learners and the subsampling dimension d_{sub} – are optimized using cross-validation targeting the overall P_{E} metric.

When we switch to the FP-50 criterion, we are once again unable to optimize $\{t_i\}_{i=1}^{l}$ and t_e collectively, because the problem is $l + 1$ dimensional and the I[·] function is discontinuous. We propose to parameterize all classifier thresholds t_i by a fixed quantile of the distribution $f_i(x)$, $x \sim P^c$. Formally, each threshold t_i is determined by a hyperparameter $\tau \in [0, 1]$ by

$$t_j(\tau) = \arg\max_{t} \left\{ \frac{1}{|\mathcal{I}_{\mathrm{c}}|} \sum_{i \in \mathcal{I}_{\mathrm{c}}} \mathrm{I}[f_j(x_i) > t] \leq \tau \right\}.$$

This approach is connected with Neyman-Pearson classification, since as $|\mathcal{I}_{\mathrm{c}}| \to \infty$, $1 - \tau$ tends to the false positive rate of each individual classifier. With this simplification, optimization is only with respect to τ and t_e.

In our experiments the parameters τ and t_e are found by direct optimization of t_e for each value of $\tau \in \left\{10^i | i \in \{-6, -5.9, \ldots, -0.1, 0\}\right\}$, where the objective is the FP-50 metric on the validation set. We also tested the original ensemble that optimizes P_{E} for each base learner.

5. EXPERIMENTAL RESULTS

Any experiments that want to explore the low-false positive performance of a detector need a huge corpus: to measure robustly a false positive rate of 10^{-6} would need a realistic minimum of about 5×10^6 examples from the negative class (preferably more). In our case this means millions of cover images. Furthermore, the data set needs to be, in some sense, representative in its diversity and difficulty of classification. Working with data of such size needs careful engineering.

In Subsection 5.1 we will explain the database that we collected, and our methodology for testing it. In Subsections 5.2 and 5.3 we will report the results of single linear classifiers and ensembles, respectively. A few further experiments are described in Subsection 5.4.

5.1 Experimental setup

In June 2014, Yahoo! Inc. made available to researchers a data set of 100 million creative-commons licensed images (including a few videos) from the popular photo-sharing website Flickr.[16] We were granted access to this database, and from it collected a set of images useful for testing low false-positive steganalysis. We began by selecting only compressed colour images where the full-size original uploaded image (as opposed to Flickr-downsampled versions) was available, and where the camera model was included in the EXIF data. Then we selected only images which were JPEGs compressed with quality factor 80 (steganalysis of images with varying quality factors is a difficult and largely unsolved problem; quality factor 80 was chosen as the most common choice that did not have very large file sizes). The images were partitioned depending on which *actor* they belong to, where an actor is defined to be a username and camera model combination. Thus each actor's images were uploaded by the same user and taken with the same camera model. Finally, we discarded actors with fewer than 10 images.

This yielded a total of 4 511 523 images from 47 807 actors, a total of approximately 9 128 115 megapixels of data taking 1307 GB on disk. The actor with the most images had 16 886, but only about 1% of the actors had more than 1000 images; the median number of images per actor was 30.

We partitioned this image set into two. The *training and validation* subset consists of 10% of the actors (in fact every 10th actor, ranking actors by size, so that a representative subset was taken): this totals 449 395 images from 4781 actors. The rest is the fixed *testing* subset which contains 4 062 128 images.

In our experiments we are assuming that the ground truth of these images is that they are covers, not used for steganography. There is no reasonable way to verify this assumption, but we can take comfort from the fact that, if it is wrong, only a small proportion of the database would be affected. Furthermore, if some of our assumed-cover images are actually stego images, our empirical estimates of false positive rates will be conservative. There is also an interesting connection with our presentation in Subsection 3.1: if by any chance some assumed-cover images are actually stego objects, Chebyshev minimizers should be be unaffected.[17]

The steganographic algorithm used in all experiments was a simulator of nsF5 embedding with matrix embedding turned off, which means that the number of embedding changes is equal to half the payload. The algorithm was chosen because of its historic importance in steganography, because we know that it can be detected by current steganalysis features, and its speed: essential for the number of images processed here. Since our goal was to measure very reliable classifiers, the payload size simulated was 0.5 bits per nonzero coefficient (bpnc). Our chosen steganographic features were the 22510-dimensional *JPEG rich model* (JRM).[14] The reference implementation takes approximately 15 seconds per megapixel to extract, on our main computing machine, but we re-implemented a highly optimized version in C which takes around 0.5 seconds per megapixel. Our benchmarks are taken from a workstation with two 6-core Xeon processors (Westmere-EP series) running at 3.47Ghz, with 192GB of memory.

We extracted JRM features for every cover image, for every stego image in the training and validation subset, and for 10% of the stego images of each actor in the testing subset (thus 407 417 stego images in the testing subset; the entire testing set is approximately 4.5 million images). It is not necessary, for our benchmarks, to have millions of stego images in the testing set, because we do not need to examine very low false negative rates. Extracting these features from all 5.4 million cover and stego images took around 120 core-days, spread across a small cluster.

	FLD	SVM	Square loss	Exponential loss	Logistic loss
FP-50, training set	$1.11 \cdot 10^{-4}$	$2.18 \cdot 10^{-5}$	$1.45 \cdot 10^{-5}$	0	0
FP-50, validation set	$2.52 \cdot 10^{-4}$	$1.99 \cdot 10^{-4}$	$5.61 \cdot 10^{-4}$	$9.87 \cdot 10^{-4}$	$1.02 \cdot 10^{-3}$
Training time	$4.8 \cdot 10^{2}$s	$3.3 \cdot 10^{4}$s	$2.4 \cdot 10^{2}$s	$5.5 \cdot 10^{4}$s	$1.0 \cdot 10^{5}$s

Table 1: Training and validation false positive rates (when false negative is 50%) of classifiers trained on the whole input space. The last line shows the time needed to train a single classifier on $2 \times 40\,000$ samples.

The cover and stego JRM features, in double precision, require 900GB of disk space. The training and validation features alone are 150GB, which is barely possible to load into memory in one go even on our largest server, and in fact we will only train on up to $2 \times 40\,000$ samples in the experiments for this paper.

The advantage of this data set is that it is from the real world, and it contains all the difficulties that a steganalyst should expect in pratice. In particular, there is cover source mismatch (note that the training/validation set and testing set are from disjoint actors, and when we split the training and validation set apart we will again segregate actors) and a variety of image sizes. Some of the images have been resampled prior to uploading, and it seems likely that image processing operations will have been applied to many of them.

5.2 Single linear classifiers

We begin by training linear classifiers proposed in Subsection 3.2 on the entire 22510-dimensional feature space. We tested standard FLD, linear weighted SVM optimizing (1), and the following convex surrogates for direct optimization of FP-50: square loss (7), almost equivalent to the QCM method (5); exponential loss (8), equivalent to the ECM method (6); and logistic loss (9). We did not test truncated square or hinge loss, after some initial experiments not reported here, because a) their non-smooth nature makes their numerical optimization on large data expensive, and b) they demand regularization, so λ would need to be optimised by an expensive grid-search.

The methodology was as follows. The training and validation subset, consisting of images from 4781 actors, was partitioned into two subsets of actors so that the number of images in each was approximately half the total (approximately 225 000 cover images in each half, plus the same number of corresponding stego images); this breaks the training and validation subsets apart. From the training subset, $2 \times 40\,000$ images (each cover with its corresponding stego) were selected using maximum diversity: the fewest images per actor necessary to reach this total. The rest of the training images were discarded. The classifier was trained on the training set, either by standard FLD or weighted SVM methods,[6,18] by solving (7), or by numerical optimization of (8) or (9) using an iterative Newton method (we used the implementation at Ref. 19). The size of the used part of the training set was limited by feasibility of this optimization over such large dimension. The trained classifier was tested on all approximately $2 \times 225\,000$ images in the validation set. Each experiment was repeated ten times using different splits into training and validation.

Out of these classifiers, only weighted SVM has hyperparameters controlling the solution: the regularization parameter λ and the class imbalance η. These were optimized by minimizing FP-50 on a full grid $\lambda \in \left\{2 \cdot 10^i | i \in \{-3, -4, -5\}\right\}$ and $\eta \in \left\{10^i | i \in \{-5, -4, -3, -2\}\right\}$. For every combination, the weighted SVM was trained on a randomly selected 75% of training samples and the FP-50 criteria was estimated from the remaining 25%. This was repeated five times with independent splits. The hyperparameters with the least FP-50 were used to train the final SVM on all training data.

We then measured the FP-50 metric on both the training and validation set, averaging error rates over the ten iterations. The results are displayed in Table 1. The results show that the loss functions which include an exponential penalty for false positives – exponential and logistic loss – have zero false positives on the training data (out of 40 000 cover samples and 10 repetitions), when the false negative rate is 50%, but also expose their weakness: they overfit the training data, and their accuracy on the validation set is slightly worse than the other loss functions. It is unsurprising that an exponential penality encourages overfitting, and it means

that we need some kind of regularisation, which we will supply indirectly in the following subsection with a dimension-subsampling ensemble.

At first sight, Table 1 suggests that the clear winner should be the weighted SVM. But its optimization involves tuning hyperparameters, which would be prohibitively expensive in an ensemble. Moreover, we see the zero false positive results, observed for exponential and logistic loss on the training set, as a positive sign: less a problem of overfitting than a problem of insufficiently diverse training data. Also the lack of hyperparameters simplifies their use.

The last line in Table 1 shows the time to train each classifier (for SVM this includes the search for hyperparameters). Unsurprisingly, training classifiers with algebraic solutions – FLD and square loss – is two orders of magnitude faster than a linear SVM, and two to three orders of magnitude faster than the methods which require numerical optimization. Nonetheless, such time is tractable on a fast machine. Furthermore, the time to *apply* the trained classifier is the same for all methods, since each classifier simply produces a projection direction.

5.3 Ensembles of classifiers

We now plug the classifiers into the ensemble framework descibed in Sect. 4, although we now exclude the SVM base learner as being too slow for ensemble settings.

Our first experiments, similarly to the last section, use only the training and validation sets. As before, we take $2 \times 40\,000$ training images. This time we train a fixed number (300) of FLD base learners, each using a randomly-selected subset of $d_{sub} \in \{100, 250, 500, 1000\}$ features, and optimize the individual base learner thresholds t_i and the ensemble voting threshold t_e according to the method of Subsection 4.1 by measuring their FP-50 benchmark on all approximately $2 \times 225\,000$ images in the validation set. We computed the ROC curves across the data in the validation set for the optimal parameter combination.

We repeated with 10 different splits of training and validation actors, and then the entire procedure (with the same training data and random subspaces) with base learners optimizing square loss, exponential loss, and logistic loss. For each false positive rate on the validation data we averaged the corresponding false negative rate over the ten iterations of the experiment. The ROC curves are displayed in Figure 2, where we have used a logarithmic scale on the x-axis to highlight the low false-positive region. They show that the loss functions with an exponential penalty – exponential and logistic loss – carry good true positive detection power into the low false-positive regime much better than the quadratic penalties in FLD and the square loss surrogate (their ROC curves have steeper slope on the onset). Observe that this advantage is balanced by decreased detection accuracy on higher false positive rates, compared with the traditional ensemble of FLDs, but that aligns with our aim.

Another reason why the square loss surrogate and FLD classifier may be inferior is because their losses are symmetric (see Figure 1). This means that they penalize correctly classified samples, or more precisely that the optimal projection direction w is still influenced by outliers even if they already lie on the correctly-classified side of the boundary. We can see that this is so for FLDs, because Fisher's presentation models the two classes as Gaussian, and outliers in the correctly-classified direction influence the empirical covariance matrix in such a way that the classifier expects also outliers in the incorrectly-classified direction, whether they exist or not.

The exponential and logistic loss surrogates work best if the dimension of the random subspace in the ensemble is kept low: we believe d_{sub} acts as an indirect regularizer, limiting the complexity of the base learners and reducing their propensity to overfit, and overcoming a key weakness in their use as a full-space classifier. Perhaps this, rather than the potential for nonlinear classification, explains why the ensemble of FLDs has been so successful with large-dimensional "rich features"; in any case, the advantage is even more evidence for classifiers with exponential penalties.

Thanks to the reduced dimension, the exponential loss base learner is no longer orders-of-magnitude more expensive to train than the FLD: in our experiments with $d_{sub} = 100$, training the ensemble of 300 learners on $2 \times 40\,000$ samples took 136 seconds with the FLD base learner and 286 for the exponential loss base learner. Both cases required a further 190 seconds to optimize the thresholds.

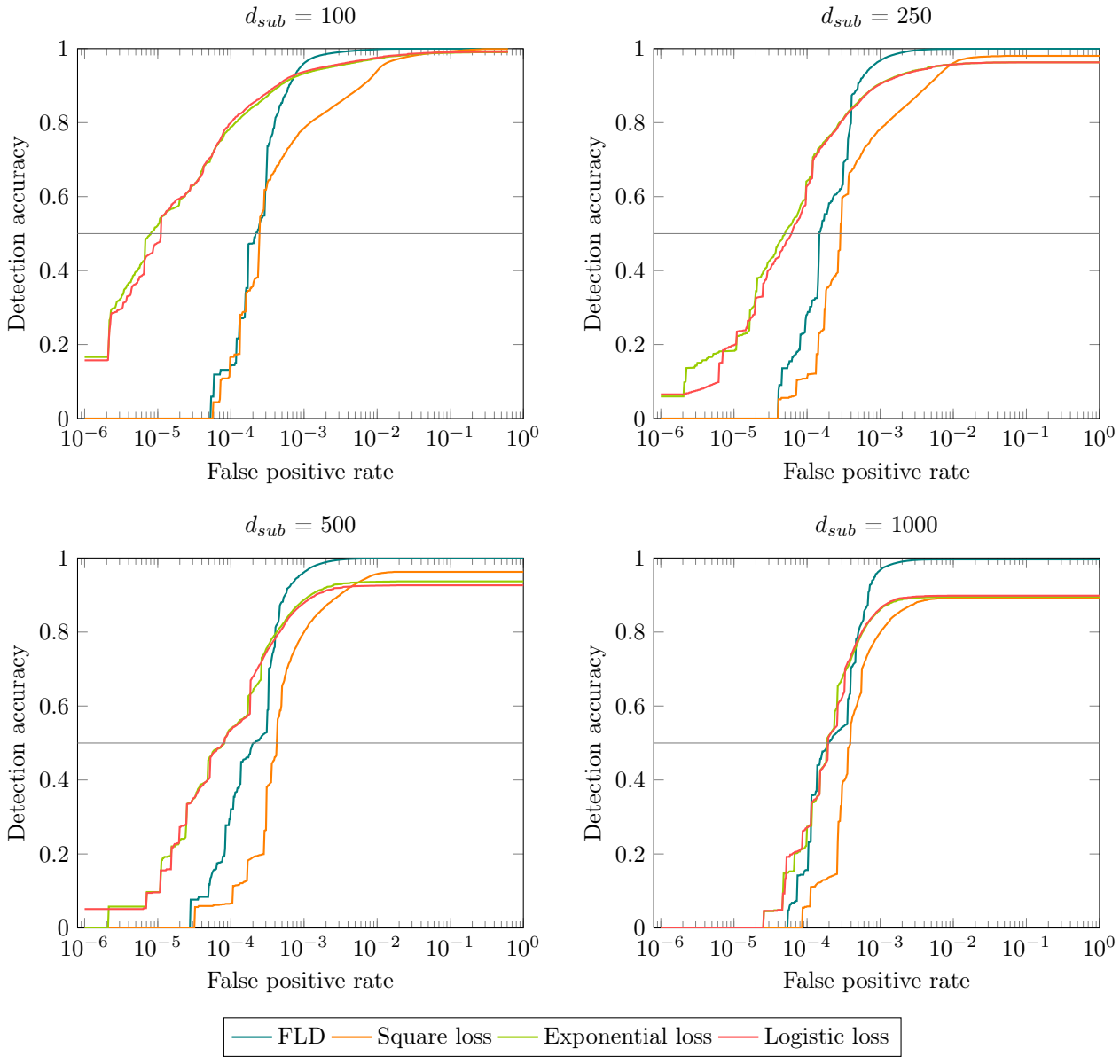

Figure 2: ROC curves, on the validation set, for ensembles of FLDs and other base learners. 300 weak classifiers were trained in random d_{sub} dimensions of the feature space. All classifiers were trained on $2 \times 40\,000$ samples. Note the logarithmic scale of the x-axis.

	False positive rate		False negative rate	
	FLD	Exp. Loss	FLD	Exp. Loss
Optimizing P_{E}	$9.07 \cdot 10^{-3}$	$1.88 \cdot 10^{-3}$	$1.33 \cdot 10^{-3}$	$1.89 \cdot 10^{-2}$
Optimizing FP-50	$3.26 \cdot 10^{-4}$	$5.56 \cdot 10^{-5}$	$4.58 \cdot 10^{-1}$	$5.12 \cdot 10^{-1}$

Table 2: Error rates on the testing set, after optimization of ensemble parameters on the training and validation set.

Ideally we would optimize both the number of base learners (currently fixed at 300) and the subspace dimension. But the training is too slow to perform a grid search, so we postpone such an investigation for further work.

Finally, we apply the trained and threshold-optimized ensembles (selecting $d_{sub} = 100$) to the testing set of 4 062 128 cover images and 407 417 stego images. The size of this set gives us confidence in measuring false positive rates into the region of 10^{-6}. We emphasise this procedure, which did not tune *any* hyperparameters with respect to the testing set, is the gold standard for machine learning. We also tuned another ensemble, this time optimizing the P_{E} benchmark for each base learner and fixing $t_e = 0.5$, to compare with the prior art.

We display results, only for the FLD and exponential loss base learners, in Table 2. The classification thresholds were set for 50% false negatives on the validation set, and achieved close to that on the testing set. The traditional ensemble of FLDs, optimizing for P_{E}, can only manage a false positive rate of approximately 1 in 110 (and a false negative rate much lower than our target of 50%). Using our new ensemble optimization of thresholds for FP-50, the FLD ensemble reduces its false positive rate to approximately 1 in 3000 (and a false negative rate close to 50%), but the same ensemble using exponential loss base learners (equivalent to ECM) achieves a false positive rate of approximately 1 in 18 000. We can have high confidence in these false positive rates because the testing set is representative of the real world, and very large.

5.4 How many training images is sufficient?

We were not able to train on the potential training set of $2 \times 225\,000$ images because of the time and memory requirements: our current numerical optimzation method requires calculations over all data points on each iteration, and the matrix of features, in double precision, would require about 83GB of memory. (This does suggest further research using online base learners.) We had to limit ourselves to a diverse selection of 40 000 image from the training data.

But do we actually need even 40 000 images? If the training images were not sufficiently diverse, training on more images would not be reflected by the lower error rates. We note that this question, largely neglected in the literature, was first raised in Ref. 20 and was also studied in Ref. 21. In experiments for Subsect. 5.3, we also evaluated ensembles trained on two smaller sizes of training data: $2 \times 10\,000$ and $2 \times 20\,000$ samples, as well as $2 \times 40,000$. As before, the images were selected from the training subset to diversify amongst actors as much as possible. We emphasize the validation set was unchanged in these experiments, as were the selection of random subspaces for the base learners. The results appear in Table 3.

The most important observation is that FP-50 does not change much with the size of the training set. This indicates that including more images from the same actors does not increase the diversity of the training set. This also suggests that the cover mismatch phenomenon[2] is at work, and some misclassifications are more likely due to some actors having non-typical image sources, than than individual outlier images.

Notice that FP-50 of the ensemble of FLDs slightly increases as the size of the training set increases to 40 000, in contrast to the general mantra of machine learning that increasing the size of the training set should not degrade performance. This seems likely to be due to the inherent loss function in FLDs, since we observed the same phenomenon with the traditional FLD ensemble from Ref. 15, perhaps making them more prone to outliers.

d_{sub}	Training set	FLD	Square loss	Exponential loss	Logistic loss
100	$2 \times 10\,000$	$1.69 \cdot 10^{-4}$	$1.63 \cdot 10^{-4}$	$7.55 \cdot 10^{-6}$	$8.91 \cdot 10^{-6}$
	$2 \times 20\,000$	$1.69 \cdot 10^{-4}$	$1.63 \cdot 10^{-4}$	$7.55 \cdot 10^{-6}$	$8.91 \cdot 10^{-6}$
	$2 \times 40\,000$	$1.78 \cdot 10^{-4}$	$1.70 \cdot 10^{-4}$	$7.56 \cdot 10^{-6}$	$8.02 \cdot 10^{-6}$
250	$2 \times 10\,000$	$1.72 \cdot 10^{-4}$	$2.43 \cdot 10^{-4}$	$4.57 \cdot 10^{-5}$	$4.61 \cdot 10^{-5}$
	$2 \times 20\,000$	$1.72 \cdot 10^{-4}$	$2.43 \cdot 10^{-4}$	$4.57 \cdot 10^{-5}$	$4.61 \cdot 10^{-5}$
	$2 \times 40\,000$	$1.80 \cdot 10^{-4}$	$2.12 \cdot 10^{-4}$	$4.40 \cdot 10^{-5}$	$4.49 \cdot 10^{-5}$
500	$2 \times 10\,000$	$1.86 \cdot 10^{-4}$	$2.88 \cdot 10^{-4}$	$8.53 \cdot 10^{-5}$	$8.18 \cdot 10^{-5}$
	$2 \times 20\,000$	$1.86 \cdot 10^{-4}$	$2.88 \cdot 10^{-4}$	$8.53 \cdot 10^{-5}$	$8.18 \cdot 10^{-5}$
	$2 \times 40\,000$	$1.91 \cdot 10^{-4}$	$3.02 \cdot 10^{-4}$	$7.27 \cdot 10^{-5}$	$7.19 \cdot 10^{-5}$
1000	$2 \times 10\,000$	$2.45 \cdot 10^{-4}$	$3.10 \cdot 10^{-4}$	$1.51 \cdot 10^{-4}$	$1.51 \cdot 10^{-4}$
	$2 \times 20\,000$	$2.45 \cdot 10^{-4}$	$3.10 \cdot 10^{-4}$	$1.51 \cdot 10^{-4}$	$1.51 \cdot 10^{-4}$
	$2 \times 40\,000$	$2.50 \cdot 10^{-4}$	$3.14 \cdot 10^{-4}$	$1.45 \cdot 10^{-4}$	$1.43 \cdot 10^{-4}$

Table 3: The FP-50 benchmark on the validation set. 300 weak classifiers were trained in random d_{sub} dimensions of the feature space, using $2 \times 10\,000$, $2 \times 20\,000$, or $2 \times 40\,000$ maximally-diverse samples from the training set.

6. CONCLUSION AND FUTURE WORK

Low probability of false alarm would be crucial for steganalyzers to be used in the real world. Yet there has been no serious study of how to achieve it. We admit that such a study is not simple: to estimate detection accuracy at a false positive rate of 10^{-6}, the number of validation samples has to be considerably greater then 1 million. Compare with contemporary practice, where the most frequently used data set (BOSSBase) has 10 000 images. We gathered 4.5 million images from Flickr, which has the realistic challenges of a real-world database, but even this would be insufficient if we wanted to examine false-positive rates below 10^{-6}.

We have advocated measuring the reliability of steganalyzers for real world applications by the false positive rate at 50% detection accuracy (FP-50 error) and proposed a family of classifiers optimizing it. Although our experiments demonstrated the advantage of the proposed classifiers over the current state of the art, we believe the questions and new directions revealed are more important still.

It appears that evaluating how a steganalyzer would perform in the real world is impossible if one possesses only a few thousand images from a few sources (e.g. BOSSBase). It should be done on millions of images, and it is unlikely that a database of such size can be acquired with a known ground truth. So it is timely to consider the literature on learning from *positive and unlabelled data*, as we move towards public databases. We note that the proposed family of classifiers is an embodiment of recently proposed methods for this problem.[17]

Our experiments also revealed that classifiers with symmetric loss functions, of which FLD is a popular example, have the undesirable property of penalizing correctly classified samples. This may be the cause of *increased* FP-50 error when the size of training data increases. Our experiments demonstrated that the FLD ensemble, which currently dominates the steganalysis literature, may benefit from indirect regularization controlled by the size of the random subspaces on which the base learners operate. This suggests a study comparing symmetric and asymmetric loss functions and regularizations. Such a study needs many sources and a large database, and to include cover source mismatch.

We conclude with a return to Section 1, and the definition of *dependable* steganalysis. Is it achievable? In this study we used a non-adaptive method of steganography that is well known to be detectable (nsF5), a very high payload (0.5 bits per nonzero coefficient), images typically much larger than the 512×512 in BOSSBase, and our best detectors managed a false positive rate of about 1 in 18 000. Even accounting for progress in steganalysis features and new classifiers optimizing the FP-50 criterion, a false positive rate of 1 in a million, or lower, seems

out of reach. If a steganalyst wants to make fewer errors, it seems the only option is *pooled steganalysis*,[22] where evidence is aggregated from many images instead of one alone.

ACKNOWLEDGMENTS

The work of T. Pevný was supported by the Grant Agency of Czech Republic under the project P103/12/P514. We thank Yahoo! for making available the 100M creative commons image set from Flickr.

REFERENCES

[1] Ker, A. D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, S., Fridrich, J., and Pevný, T., "Moving steganography and steganalysis from the laboratory into the real world," in [*Proc. 1st ACM Workshop on Information Hiding and Multimedia Security*], 45–58, ACM (2013).

[2] Ker, A. D. and Pevný, T., "A mishmash of methods for mitigating the model mismatch mess," in [*Media Watermarking, Security, and Forensics 2014*], *Proc. SPIE* **9028**, 1601–1615, SPIE (2014).

[3] Cogranne, R., Zitzmann, C., Retraint, F., Nikiforov, I., Fillatre, L., and Cornu, P., "Statistical detection of LSB matching using hypothesis testing theory," in [*Proc. 14th International Conference on Information Hiding*], *LNCS* **7692**, 46–62, Springer-Verlag (2013).

[4] Ker, A. D., "Benchmarking steganalysis," in [*Multimedia Forensics and Security*], Li, C.-T., ed., 266–290, IGI Global (2009).

[5] Pevný, T. and Ker, A. D., "The challenges of rich features in universal steganalysis," in [*Media Watermarking, Security, and Forensics 2013*], *Proc. SPIE* **8665**, 0M01–0M15 (2013).

[6] Bach, F. R., Heckerman, D., and Horvitz, E., "Considering cost asymmetry in learning classifiers," *Journal of Machine Learing Research* **7**, 1713–1741 (2006).

[7] Rigollet, P. and Tong, X., "Neyman-Pearson classification, convexity and stochastic constraints," *Journal of Machine Learning Research* **12**, 2831–2855 (2011).

[8] Davenport, M., Baraniuk, R., and Scott, C., "Controlling false alarms with support vector machines," in [*Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2006*], **5**, 589–592 (2006).

[9] Scholkopf, B. and Smola, A. J., [*Learning with Kernels*], MIT Press (2002).

[10] Vapnik, V., [*Statistical learning theory*], Wiley (1998).

[11] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., and Platt, J. C., "Support vector method for novelty detection," in [*Advances in Neural Information Processing Systems 12*], 582–588, MIT Press (2000).

[12] Chang, C.-C. and Lin, C.-J., "Training nu-support vector regression: theory and algorithms," *Neural Computation* **14**(8), 1959–1978 (2002).

[13] Freund, Y. and Schapire, R. E., "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences* **55**(1), 119–139 (1997).

[14] Fridrich, J. and Kodovský, J., "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security* **7**(3), 868–882 (2012).

[15] Kodovský, J., Fridrich, J., and Holub, V., "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information Forensics and Security* **7**(2), 432–444 (2012).

[16] Yahoo! Webscope (2014). Yahoo! Webscope dataset YFCC-100M. `http://webscope.sandbox.yahoo.com`.

[17] Geurts, P., "Learning from positive and unlabeled examples by enforcing statistical significance," in [*Proc. 14th International Conference on Artificial Intelligence and Statistics*], *Journal of Machine Learning Research* **15**, 305–314 (2011).

[18] Franc, V. and Sonnenburg, S., "Optimized cutting plane algorithm for support vector machines," in [*Proc. 25th International Conference on Machine Learning*], 320–327, ACM (2008).

[19] Schmidt, M., "minfunc," (2013). `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`, accessed January 2014.

[20] Miche, Y., Bas, P., Lendasse, A., Jutten, C., and Simula, O., "Advantages of using feature selection techniques on steganalysis schemes," in [*Computational and Ambient Intelligence, Proc. 9th International Work-Conference on Artificial Neural Networks*], *LNCS* **4507**, 606–613, Springer-Verlag (2007).

[21] Lubenko, I. and Ker, A. D., "Steganalysis with mismatched covers: Do simple classifiers help?," in [*Proc. 13th ACM Workshop on Multimedia and Security*], 11–18, ACM (2012).

[22] Ker, A. D., "Batch steganography and pooled steganalysis," in [*Proc. 8th Information Hiding Workshop*], *LNCS* **4437**, 265–281, Springer (2006).