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 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 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 slightly a difference 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 a single 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 $\mathcal{I}$. 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 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-known (for example if the proportion of stego objects that will be encountered is known precisely, but this seems unrealistic), 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 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.

Tomas, it would be nice for us to justify not using cost-sensitive SVMs or weighted LR in the main experiments. Do you think we could simply use the experiments we did for the abstract? We could call them 'preliminary investigation' since they are on a smaller data set on decimated features, and put them at the start of the experimental results section. It's quite convincing evidence that we can move straight to the convex surrogate loss functions.

The experimental results I did with SVM picked the SVM performing the best on the validation set. This is of course favouring SVM because (a), it does not include times for crossvalidation and (b) does not count the mis-match between training and testing samples. I am changing the script to do everything properly at this point, but it is going to be rather slow, at least five times. This means that one iteration would take about 11 hours.

Okay. I still think we could also use the preliminary experiments as a way to justify disposing on unequal-prior logistic regression. I might try to draft a section and you can see what you think.

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

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

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 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 adapted to non-linear circumstances, although we will follow the state-of-art in steganalysis by regaining nonlinearity via an ensemble of linear classifiers.

Incorporating the above into (3), the problem becomes

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

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

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

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

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

†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 $t>0$.

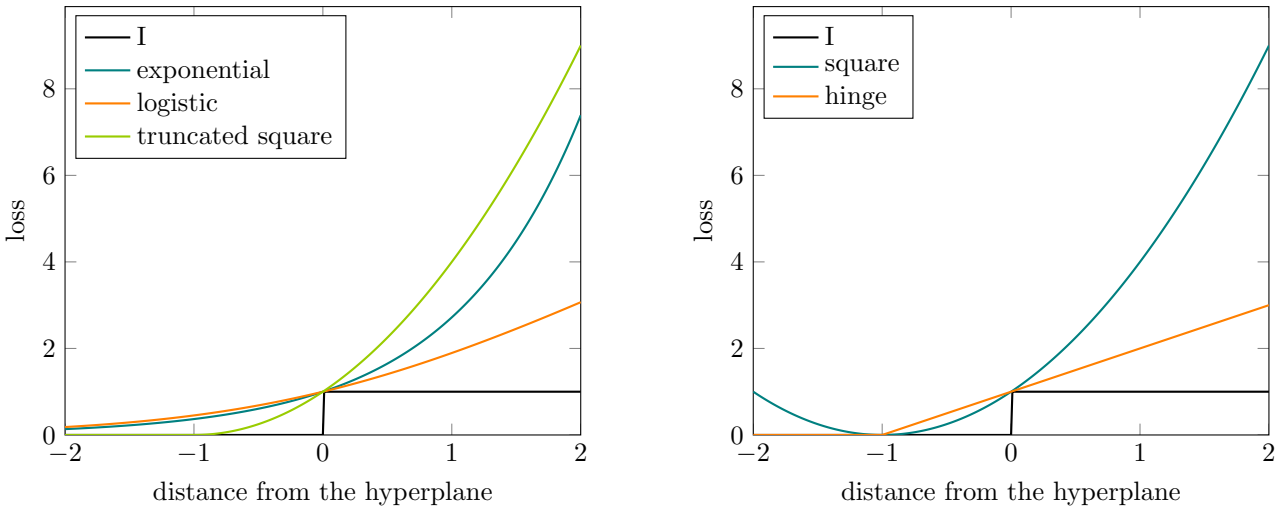


Figure 1: Convex surrogates for the 0-1 loss function.

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. 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 hyper-parameter λ, but in our experiments we will set $\lambda = 0$ to turn off regularization.

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

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

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.

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

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

The advantage of square loss over hinge loss is that it is smooth, so the optimization can be simpler: stochastic gradient methods are particularly effective.

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

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

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

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

Turning off the regularization, we have recovered the exponential Chebyshev minizer (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

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

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 superlinearly (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 subspaces dimensions makes 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

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

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 $\mathrm{I}[\cdot]$ 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

$$t_i(\tau) = \arg\max_t \left\{ \frac{1}{|\mathcal{I}_c|} \sum_{i \in \mathcal{I}_c} \mathrm{I}[f_i(x) > t] \leq \tau \right\}.$$

This approach is connected with Neyman-Pearson classification, since as $|\mathcal{I}_c| \to \infty$, $1 - \tau$ tends to the false positive rate of each individual classifier. With this simplifications, 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. ... incomplete, depends on what other experimental results go in ...

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 is 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.

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 which is 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.

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.

The cover and stego JRM features, in double precision, require 900GB of disk space. The training and validation features alone require 150GB, which is hardly possible to load into memory in one go.

Do we want to say something about the compute machines we are using? Perhaps we'd better if we are including timing benchmarks.

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 a variety of image sizes.

Tomas, the most important question I have at the moment: how did you reach a figure of 250 000 images in the validation set? There are just under 450 000 covers and 450 000 stegos. Did you select 250 000 of them at random? If so, was it a different subset for each iteration?

The right sizes: In training + validation set they are in total 449395 covers. Every iteration we split the set into training and validation sets on level of actors, so approximately half them were used for training and half of them for testing. Sorry for confusing numbers.

And when you said in section 5.3 that the ROC curves were estimated from 250 000 images in the testing set, is this right? I thought the point of ROC curves was that they could only come from the validation set, because they involve the selection of a threshold?

Testing set in section 5.3 is the validation set. This confusion was caused by incremental writing of notes. This means that ROC curves and thresholds were estimated from $2 \times \frac{449395}{2}$ images. The large testing set was used only in Table 2.

I'm still confused, because the text says that the training set is only 2×40000. Are the 40000 selected at random from the $\frac{449395}{2}$ images in the training set? And by actor, or using as many diverse actors as possible?

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 (equal-cost) SVM, and the following convex surrogates for direct optimization of FP-50: square loss (7), almost equivalent to the QCM method (**??**); exponential loss (8), equivalent to the ECM method (**??**); and logistic loss (9). We did not test truncated square or hinge loss, after some initial experiments not reported here, because their non-smooth nature makes numerical optimization on large data infeasibly expensive.

The methodology was as follows. The training and validation subset, consisting of $2 \times 449\,395$ images (each cover has a corresponding stego) was partitioned into a training subset ($2 \times 40\,000$ images Tomas, how did you pick the 40000 images – by selecting actors, and using maximum diversity of actors?) and validation subset (the rest). The classifier was trained on the training set, either by standard FLD or SVM methods, by solving 7), or by numerical optimization of(**??**) using an iterative Newton methodTomas, can we be more precise about the numerical optimization algorithm?. The size of the training set was limited by feasibility of this optimization over such large dimension. Each experiment was repeated 10 times using different splits into training and validation.

This doesn't account for SVM optimization. Tomas, I'm afraid I still don't understand what experiments you are doing here. Can you explain again how you are finding the parameters for the SVM?

	FLD	SVM	Square loss	Exponential loss	Logistic loss
FP-50, training set	$1.21 \cdot 10^{-4}$	$6.06 \cdot 10^{-5}$	$2.02 \cdot 10^{-5}$	0	0
FP-50, validation set	$2.77 \cdot 10^{-4}$	$1.50 \cdot 10^{-4}$	$4.23 \cdot 10^{-4}$	$6.48 \cdot 10^{-4}$	$7.42 \cdot 10^{-4}$
Training time	$4.6 \cdot 10^{2}$s	$9.4 \cdot 10^{3}$s	$2.3 \cdot 10^{2}$s	$5.6 \cdot 10^{4}$s	$1.0 \cdot 10^{5}$s

Tomas, please note that I changed the order to match the earlier presentation
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.

We then measured the FP-50 metric on both the training and validation set Tomas, is this averaged over all 10 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), 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 ensmeble.

The last line in table 1 shows the time to train each classifier. Tomas, how do we account for time of hyperparameter optimization in the SVM? Unsurprisingly, training classifiers with algebraic solutions – FLD and square loss – is an order of magnitude faster than a linear SVM, and two orders of magnitude faster than the methods which require numerical optimization. Nonetheless, such time is tractable on a fast machine.

5.3 Ensembles of classifiers

Andrew edited this far

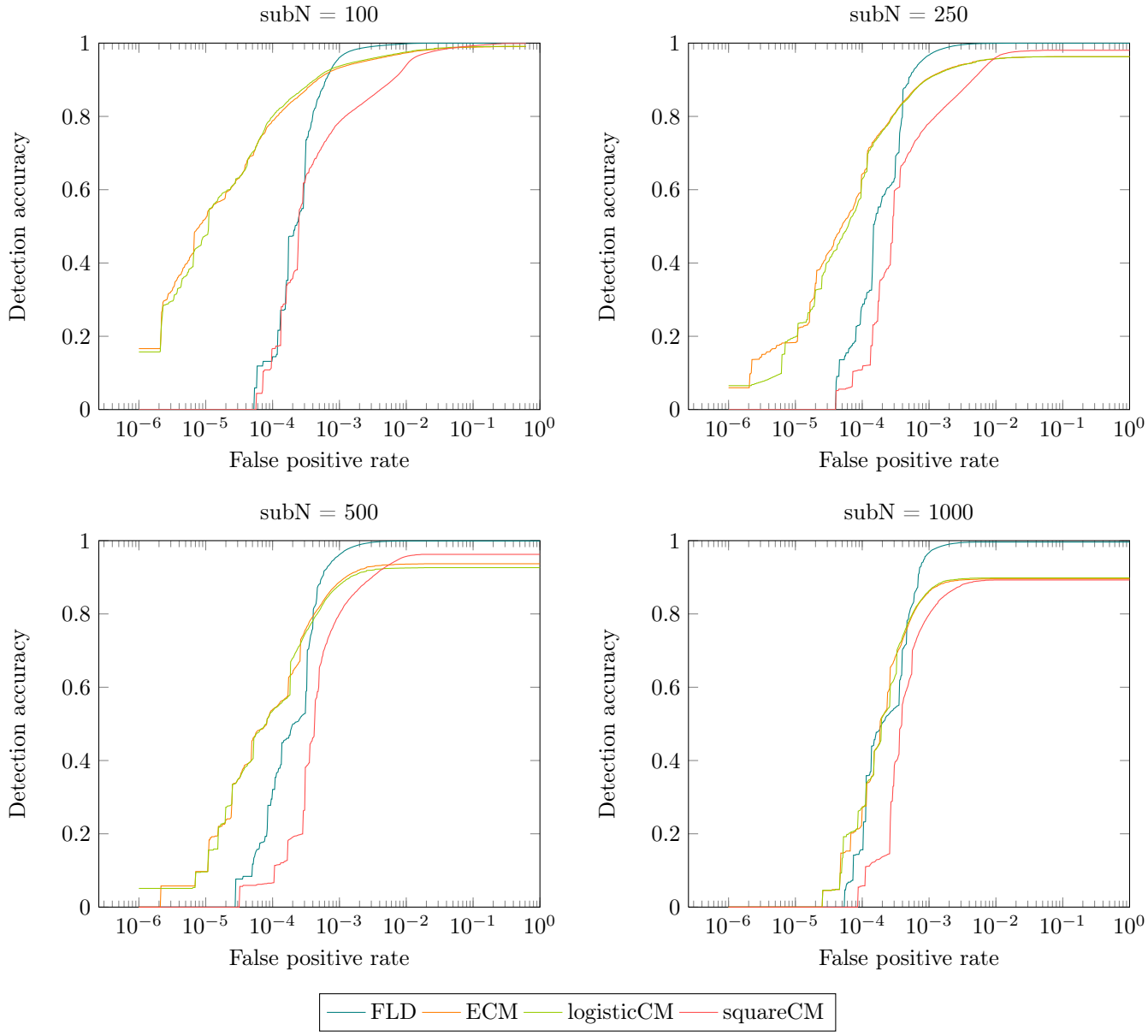
... include this ... We emphasize that the testing set was set was set completely aside and was not used in comparison of ensembles in Subsections **??** and **??** and for choosing their parameters, namely the dimension of the subspace. This separation of testing set prevents tuning of classifiers with respect to it.

... we believe that the size as a sub-space in ensembles acts as a regularization indirectly controlling the complexity of the classifier. The size of the sub-space can be compared to the regularization parameter λ (inverse costs $\frac{1}{C}$) in SVMs, both parameters being optimized to prevent over-fitting.

We first compare ensembles of FLD and classifiers proposed in Section **??**, as ensembles of classifiers trained on subspaces are faster to train than classifiers operating on the full space. We compare ensembles of FLDs and Chebyshev minimizers with exponential (ECM), logistic (logisticCM), and square losses (squareCM), omitting classifiers with hinge and truncated square losses from the comparison due to difficult of their training caused by non-smoothness of the loss functions. 300 base classifiers within ensembles were trained on subspaces of size $\{100, 250, 500, 1000\}$ on training sets of size $2 \times 4 \cdot 10^4$ images. Stego images contained simulated payload of 0.5 bpnz embedded by nsF5 with matrix embedding turned off. Thresholds were optimized to minimize FP-50 criterion on the validation set as described in Subsection 4.1.

ROC curves shown in Figure 2, estimated on 2×250000 images from the testing set, demonstrates the advantage of ensembles of the proposed exponential Chebyshev minimizer or logistic Chebyshev minimizer, as their ROC curves have steeper slope on the onset. Notice how this advantage is out-weighted by the decreased detection accuracy on higher false positive rates in comparison to ensemble of FLDs. Interestingly Chebyshev minimizer with square loss function is inferior, which we attribute to the fact that the this loss function is symmetric and penalizes correctly classified samples (see Figure 1). UnlikeIn all cases ensembles with base classifiers operating on smaller dimensions are better, which is most probably due to higher diversity of classifiers within the ensemble.

Table 3 shows FP-50 of all evaluated ensembles on different sizes of training sets discussed in Subsection 5.4. The most important is the observation that FP-50 does not change too much with respect to sizes of the training



Tomas, can you make some change to the figures: reduce the width a little since this figure is about 30 points too wide, perhaps putting the y-axis label a little closer to the graph to get rid of unused white space; thicken the lines in the charts; change the x-axis to start at 10^{-6} or a little below, say $8 \cdot 10^{-7}$; re-order and re-label the caption so that it goes 'FLD', 'square loss', 'exponential loss', 'logistic loss'; change the label subN to d_{sub}; finally, could we have a thin horizontal black line at false negative=50%, to indicate our FP-50 benchmark?

Figure 2: ROC curves on validation set for ensembles of FLDs and Chebyshev minimizers with 300 weak classifiers trained in dimensions of random subspaces $\{100, 250, 500, 1000\}$. Color codes the type of the weak classifier within the ensemble. All classifiers were trained on $2 \times 4 \cdot 10^4$ samples with stego images containing simulated payload of 0.5 bpnz embedded by nsF5 with matrix embedding turned off.

	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.

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: FP-50 on validation set of ensembles with 300 base classifiers, where base classifiers were implemented as FLDs and Chebyshev minimizers with exponential (ECM), logistic (logisticCM), and square (squareCM). Base classifiers were trained on random subspaces of $\{100, 250, 500, 1000\}$ on training sets of size $2 \times \{1 \cdot 10^4, 2 \cdot 10^4, 4 \cdot 10^4\}$. Stego images contained simulated payload of 0.5 bpnz embedded by nsF5 with matrix embedding turned off.

set. This indicates that including more images from the same actor into the training set does not increase its diversity. Notice that FP-50 of ensemble of FLDs slightly increases as the size of the training set increases to $4 \cdot 10^4$. This is in a sharp contrast to a general mantra in machine learning that increasing the size of the training set should not degrade the performance. The cause can be either due to inherent property of square loss function used within FLD, or due to FLD optimizing P_E error. Since we have observed the same behavior if the error was measured by the usual P_E and thresholds within ensembles were set as originally proposed in,[15] we conjecture that this unwanted behavior is a property of the FLD classifiers caused by the symmetric square loss penalizing correctly classified samples.

Tomas, the thing that jumps out at me is that we have not included any time benchmarks. Given that exponential and logistic loss were so slow in the single classifier case, it seems likely to be a drawback of an ensemble too. Did you keep track of the time it takes to train an ensemble of the different classifiers? Or could you re-do one iteration (and discard the accuracy) just for the purposes of benchmarking the speed?

... To summarize ensembles of the proposed exponential Chebyshev minimizers have FP-50 by measure almost two order of magnitudes better than the state of the art ensemble of FLDs.

5.4 How many training images is sufficient?

Training on 2×250000 images is not straightforward, since feature matrix in double precision occupies about 83Gb of memory, which just fits into the memory of our computer, and the complexity of training with large number of samples becomes non-negligible, especially for exponential and logistic Chebyshev minimizers. But do we actually need to train on all images? If training images are 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 firstly raised in.[17]

In experiments in Section **??**, we have evaluated ensembles of base classifiers trained on three sizes of training sets $2 \times \{1 \cdot 10^4, 2 \cdot 10^4, 4 \cdot 10^4\}$. Subsets of training images were set such that the number of images from all actors were as equal as possible, which should maximize the diversity among them. We emphasize that while the number of training samples used to learn individual classifiers within the ensemble changes, the validation set always contained all images from all validation actors.

it looks like we are saying that 10 000 images is enough. is that really our conclusion?

I think that the conclusion should be that adding more images from the same actor does not decrease the error substantially. It is important to have images from different sources!!!

One way to test that would be to try 10 000 from a few actors, compared with 10 000 images from diverse actors, etc. Shall we postpone this for the journal version?

6. CONCLUSION AND FUTURE WORK

How many stego samples do I need for ECM?

Should we mention connection to learning with positive and unlabeled data?

Are symmetric loss functions any good? Its only advantage is fast training.

This is the conclusions text from the abstract:

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 greater then 1 million. Compare with contemporary practice, where the most frequently used data set (BOSSBase) has 10 000 images. Second, training on large numbers of samples and huge dimension requires either dimension reduction or a move to online algorithms.

In the full paper we intend to compare such approaches, and also to reduce false positives by addressing the cover mismatch problem,[2] as best as can be achieved. (Inspection of our results suggests that false positives do cluster in certain actors' images, but not to a large degree). We also consider training for unknown payload sizes, and the possibility of training complex classifiers (e.g. a boosting for cost-sensitivity[18]) on smaller training sets.

We also have to consider the possibility that very high-reliability steganalysis may be impossible with current features: if a signficant number of cover samples lie in the stego cluster, no generalizable classifier is likely to separate them. We plan some simple experiments to determine whether this is the case for JRM features. Dependable steganalysis may require different features as well as differently-optimized classifiers.

ACKNOWLEDGMENTS

We don't credit EOARD any more.

We should thank Yahoo! for the Flickr data set. Their main concern is that we cite them.

REFERENCES

1. A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, S. Filler, J. Fridrich, and T. Pevný, "Moving steganography and steganalysis from the laboratory into the real world," in *Proc. 1st ACM Workshop on Information Hiding and Multimedia Security*, pp. 45–58, ACM, 2013.
2. A. D. Ker and T. Pevný, "A mishmash of methods for mitigating the model mismatch mess," in *Media Watermarking, Security, and Forensics 2014*, *Proc. SPIE* **9028**, pp. 1601–1615, SPIE, 2014.
3. R. Cogranne, C. Zitzmann, F. Retraint, I. Nikiforov, L. Fillatre, and P. Cornu, "Statistical detection of LSB matching using hypothesis testing theory," in *Proc. 14th International Conference on Information Hiding*, *LNCS* **7692**, pp. 46–62, Springer-Verlag, 2013.
4. A. D. Ker, "Benchmarking steganalysis," in *Multimedia Forensics and Security*, C.-T. Li, ed., pp. 266–290, IGI Global, 2009.
5. T. Pevný and A. D. Ker, "The challenges of rich features in universal steganalysis," in *Media Watermarking, Security, and Forensics 2013*, *Proc. SPIE* **8665**, pp. 0M01–0M15, 2013.
6. F. R. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *Journal of Machine Learing Research* **7**, pp. 1713–1741, Dec. 2006.
7. P. Rigollet and X. Tong, "Neyman-pearson classification, convexity and stochastic constraints," *The Journal of Machine Learning Research* **12**, pp. 2831–2855, 2011.
8. M. Davenport, R. Baraniuk, and C. Scott, "Controlling false alarms with support vector machines," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, **5**, p. V, May 2006.
9. B. Scholkopf and A. J. Smola, "Learning with kernels," *MIT Press* **11**, pp. 110–146, 2002.
10. V. Vapnik, *Statistical learning theory*, Wiley, 1998.
11. B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection.," in *NIPS*, **12**, pp. 582–588, 1999.
12. C.-C. Chang and C.-J. Lin, "Training nu-support vector regression: theory and algorithms," *Neural Computation* **14**(8), pp. 1959–1978, 2002.
13. Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, pp. 23–37, Springer, 1995.
14. J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *Information Forensics and Security, IEEE Transactions on* **7**(3), pp. 868–882, 2012.
15. J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *Information Forensics and Security, IEEE Transactions on* **7**(2), pp. 432–444, 2012.
16. Yahoo! Webscope, 2014. Yahoo! Webscope dataset YFCC-100M. `http://webscope.sandbox.yahoo.com`.
17. Y. Miche, P. Bas, A. Lendasse, C. Jutten, and O. Simula, "Advantages of using feature selection techniques on steganalysis schemes," in *Computational and Ambient Intelligence*, F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, eds., *Lecture Notes in Computer Science* **4507**, pp. 606–613, Springer Berlin Heidelberg, 2007.
18. H. Masnadi-Shirazi and N. Vasconcelos, "Cost-sensitive boosting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **33**, pp. 294–309, Feb 2011.
19. H. Masnadi-Shirazi and N. Vasconcelos, "Risk minimization, probability elicitation, and cost-sensitive svms.," in *ICML*, J. Fürnkranz and T. Joachims, eds., pp. 759–766, Omnipress, 2010.

APPENDIX A. OLD ABSTRACT TEXT (TO BE REMOVED)

Classifiers with Imbalanced Costs

Prioritizing false positives is an example of *classification with imbalanced costs*,[6] but it has not been studied in steganalysis. We use only linear classifiers, since training non-linear versions has impossible time complexity for very large data sets.

Fisher Linear Discriminant (FLD). This returns a projection vector w which maximizes the distance between cover and stego means while minimizing their projected variance,

$$\arg\max_{w} \ \frac{\big((\mu_c - \mu_s)^{\mathrm{T}} w\big)^2}{w^{\mathrm{T}}(\mathbf{C}_c + \mathbf{C}_s)w}, \tag{10}$$

where μ_c / μ_s denotes the mean of cover / stego samples and $\mathbf{C}_c$ / $\mathbf{C}_s$ their covariances. FLD does not directly allow for imbalanced costs, and all one can do is adjust the classification threshold of projected feature vectors, so that false positives are rare.

Cost-sensitive Support Vector Machine. There is a simple and popular modification to standard SVMs to allow imbalanced costs,[6] solving

$$\arg\min_{w,\rho} \ \frac{\lambda}{2}\|w\|_2^2 + \frac{\pi_c}{|I_c|}\sum_{i\in I_c}\max\{0, x_i^{\mathrm{T}} w - \rho\} + \frac{\pi_s}{|I_s|}\sum_{i\in I_s}\max\{0, \rho - x_i^{\mathrm{T}} w\},$$

where I_c / I_s indexes the cover / stego samples respectively and π_c / π_s are the penalties. (It is assumed that cover / stego objects have negative / positive labels, respectively). Fixing $\pi_s = 1 - \pi_c$, and $\pi_c \in (0, 1)$ the cost-sensitive SVM has two hyperparameters π_c and λ, which we can optimize on training data to minimize the FP-50 criterion. (A more sophisticated approach[19] was not used, because the optimization is more complex.)

Unequal prior Logistic Regression. Plain logistic regression models the probability that a sample x is stego by

$$\frac{1}{1 + e^{x^{\mathrm{T}} w}}$$

and fits w to the training data using maximum likelihood estimation. There are many ways to introduce a prior, the simplest being to use the maximum a posteriori method instead. This leaves the prior probability of cover π_c and ridge regulariser λ as hyperparameters that can be optimized on the training set.

We now consider new approaches to finding a linear classifier (projection direction w) to optimize FP-50. Doing so directly is practically impossible because its derivative has many discontinuities. But with two approximations we reach a tractable optimization which leads to new linear classifiers. First, if we assume that the distribution of stego objects has a moderately symmetrical shape, we can approximate the problem by

$$\arg\min_{w} \ \mathrm{P}\big[X_c^{\mathrm{T}} w > \mu_s^{\mathrm{T}} w\big] \tag{11}$$

since, projecting in most directions w, we can expect that about 50% of stego objects lie beyond their mean. Then we seek an upper bound on (**??**), reminiscent of Vapnik's technique.[10]

Quadratic Chebyschev Minimizer. Applying Chebychev's inequality[§] to (**??**) we solve

$$\arg\min_{w} \ \frac{w^{\mathrm{T}}\mathbf{C}_c w}{\big((\mu_s - \mu_c)^{\mathrm{T}} w\big)^2}. \tag{12}$$

This has a close relationship with the FLD (10), except that it disregards the shape of the stego distribution. It has an analytic solution which, it can be shown, corresponds to finding a single unregularized projection by the CLS method.[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 (**??**) we solve

$$\arg\min_{w,t} \ \sum_{i\in I_c} e^{t(x_i - \mu_s)^{\mathrm{T}} w}. \tag{13}$$

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

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

Classifier	FP-50	sensitivity $(1 - P_{\text{FN}})$ at P_{FP} 10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
0.5 bpnc payload						
Fisher Linear Discriminant	5.5×10^{-5}	0.0001	0.9896	0.9998	0.9999	0.9999
Logistic Regression	9.8×10^{-5}	0.0001	0.5246	0.9884	0.9995	0.9999
Linear Support Vector Machine	10.4×10^{-5}	0.0001	0.4116	0.9810	0.9993	0.9999
Quadratic Chebyschev Minimizer	4.9×10^{-5}	0.0002	0.9914	0.9998	0.9999	0.9999
Exponential Chebyschev Minimizer	4.9×10^{-5}	0.3161	0.6262	0.8994	0.9705	0.9859
0.25 bpnc payload						
Fisher Linear Discriminant	2.0×10^{-4}	0.0000	0.0147	0.9898	0.9968	0.9991
Logistic Regression	4.1×10^{-4}	0.0000	0.0046	0.7693	0.9882	0.9991
Linear Support Vector Machine	2.7×10^{-4}	0.0001	0.0066	0.8937	0.9906	0.9991
Quadratic Chebyschev Minimizer	2.1×10^{-4}	0.0000	0.0249	0.9923	0.9966	0.9990
Exponential Chebyschev Minimizer	2.2×10^{-4}	0.0016	0.2120	0.9400	0.9937	0.9992

Table 4: Low false-positive performance of various classifiers. FP-50 is the false positive rate when the sensitivity (true positive rate) is 50%. Embedding algorithm is nsF5. Results from the testing set.

By varying t, it balances inequalities based on all moments of the covers. In this case, since w is unconstrained the t scalar can be absorbed into it, or used for numerical stability. Equation (**??**) does not have an analytic solution, but it is straightforward to show that the objective is convex, to compute gradient and hessian, and to apply fast-converging numerical optimizers.

Another way to understand the Chebyschev methods is as an approximation to the ideal cost function, which assigns cost 1 to cover values projected beyond the stego mean, and zero otherwise. The first approximation assigns a quadratic cost, and the second an exponential cost, so that cover outliers in the false position direction are penalized heavily. This aligns well with our dependability aim.

Experimental Results

This work requires a huge corpus, which we have obtained by crawling and downloading creative commons licensed images from the photo-sharing website Flickr. Experiments reported here are based on an initial data set of 600 000 JPEG images (all with the same quantization table) uploaded by 600 different users (each having at least 50 images, some over 10 000). They were divided into training and testing sets of disjoint actors (mimicking a real world in which training data for the tested image source is not available). This size of corpus allows us to measure empirical false positives robustly down to rates of 10^{-5}. For the full paper we are gathering a larger data set, aiming for at least 1-5M images, and the resolution of the final experiments will depend on the success of this crawl and available processing power to conduct the experiments.

In experiments reported here, we used the simple nsF5 embedding algorithm because it is much faster than adaptive methods. Since we are aiming for very low false positive rates, we must scale back our ambitions and aim to detect payloads of 0.25 or 0.5 bits per nonzero coefficient (bpnc). We extracted 22510-dimensional features using a custom speed-optimized implementation of the JRM algorithm. The entire feature set is of the order of 100 GB. We then reduced the dimension to 600 using a bagged version of the CLS method,[5] to reduce the memory requirements, improve the efficiency of training algorithms, and to stablize the covariance matrices.

In this abstract we report only the first set of results, table 1 and figure 1 (where the ROC curves have a logarithmic false positive axis, to concentrate on the region of interest), comparing the methods in section 1. The cost-sensitive SVM and logistic regression classifiers are weaker in the low false-positive regime; the simple FLD performs quite well and the QCM method similarly (not surprising, given the similar optimization problems they induce). By the FP-50 metric there is only a small advantage in moving to ECM at the higher payload, and none at the lower payload. But further inspection of the ROC shows that the ECM linear classifier manages to hold on to some significant sensitivity (true positive rate) even when the false positive rate is reduced down into the

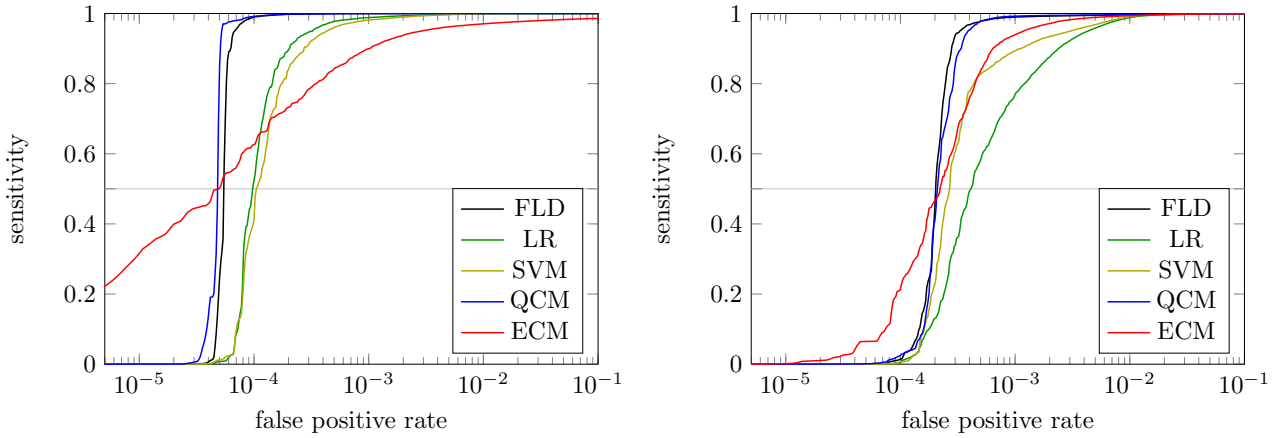


Figure 3: Semi-logarithmic ROC curves of classifiers. Payload 0.5bpnc (left) and 0.25bpnc (right).

10^{-5} or 10^{-4} region (for larger or smaller payload respectively), where the other methods are essentially unable to detect any true positives at all. Such a method is promising for dependable steganalysis, but would not be good for 'standard' steganalysis under comparable priors of cover and stego because its sensitivity is inferior to existing techniques at higher false positive rates.

The state-of-art in steganalysis uses an ensemble of FLDs. In this study we have tried only to optimize the base learners individually; in the full paper we propose to compare ensembles of such base learners as well, optimizing their parameters for the FP-50 metric.