Steganalysis using Logistic Regression

Ivans Lubenko and Andrew D. Ker

Oxford University Computing Laboratory, Parks Road, Oxford OX1 3QD, England.

ABSTRACT

We advocate Logistic Regression (LR) as an alternative to the Support Vector Machine (SVM) classifiers commonly used in steganalysis. LR offers more information than traditional SVM methods – it estimates class probabilities as well as providing a simple classification – and can be adapted more easily and efficiently for multiclass problems. Like SVM, LR can be kernelised for nonlinear classification, and it shows comparable classification accuracy to SVM methods. This work is a case study, comparing accuracy and speed of SVM and LR classifiers in detection of LSB Matching and other related spatial-domain image steganography, through the state-of-art 686-dimensional SPAM feature set, in three image sets.

Keywords: Logistic Regression, Support Vector Machine, Steganalysis, Benchmarking, SPAM features

1. INTRODUCTION

Many modern steganalysis methods involve a classification engine. Typically, one reduces the objects under consideration to a set of *features*, statistics which aim to be more sensitive to embedded payload than variations in innocent content. Then a machine learning algorithm is chosen, trained on a large set of known innocent objects and objects containing hidden data, and used to test new objects for the presence of payload.

Although there has been much research into new steganalysis feature sets, particularly for steganalysis of digital images, there have been fewer investigations into the choice of classifier. The first proposals by Farid[1] and some other early work[2–4] used a Fisher Linear Discriminant classifier, and the independent work of Avcibas et al.[5] used an Analysis of Variance technique. Since then, almost all steganalysis research has settled on the Support Vector Machine[6–10] (SVM) as the classification engine, usually employing the kernel trick[11] to allow a nonlinear classification boundary.

This paper investigates an alternative machine learning paradigm: Logistic Regression[12] (LR). More than simply a classifier, LR attempts to model the probability that a feature vector belongs to each class. It has received little attention in steganalysis applications, but it has a number of advantages: as well as providing a level of certainty in its classification, it is easy to generalize for multiclass problems. Like SVMs, Logistic Regression can be kernelised.

In this paper we consider using Logistic Regression in a particular steganalysis problem: detection of spatial-domain embedding using a 686-dimensional feature vector known as SPAM.[10] We will test SPAM-based detectors using both linear and kernel versions of Logistic Regression against those using the more popular Support Vector Machine. As well as spatial-domain Least Significant Bit (LSB) Replacement and Matching, our experiments extend to less-studied operations Mod-5 matching and 2LSB replacement. If the problem is to identify the embedding method, this makes for a 5-way multiclass steganalysis problem (the fifth class being the case of innocent covers) where the advantages of LR will be apparent.

After outlining the application of machine learning techniques to steganalysis (Sect. 2) and describing the ideas behind Logistic Regression (Sect. 3), Sect. 4 contains a case study applying LR to our particular steganalysis application. We will see that LR is capable of producing an approximately equally good performance as SVMs, with the added information of class-belonging probabilities and, when benchmarked in a suitably fair manner, faster performance on the multiclass problems.

Further author information: (Send correspondence to ADK):
A. D. Ker: E-mail: adk@comlab.ox.ac.uk, Telephone: +44 1865 283530
I. Lubenko: E-mail: ivans.lubenko@comlab.ox.ac.uk

2. MACHINE LEARNING & STEGANALYSIS

Steganalysis is normally viewed as a classification problem, where the goal is to detect steganography by discriminating between cover objects and those containing hidden messages. Typically (but not always[13]) the approach is first to select a set of *features* – statistics which are more sensitive to hidden payload than to differences between different objects – and then train a classifier. Training uses many examples of known cover and stego objects, to find the statistical differences which form a decision function predicting whether a new object belongs to the class of cover or stego. Because statistical classifiers are so central to steganalysis, it is important that we use the best of them.

Most steganalysis techniques now use kernelised (nonlinear) Support Vector Machines as classifiers including, amongst many others, Refs. 6–10. Although a few other classifiers have been tested, SVMs have become firmly established as the standard tool of choice, perhaps partly because of the existence of a free, efficient, implementation.[14] There is as yet little research to establish the advantages of using SVMs over other machine learning techniques.

2.1 Binary and Multiclass Classification

In literature proposing steganalysis techniques, it is most common for the classification problem to be presented in binary form: the two classes are 1) no payload, and 2) a known-length payload embedded using a known algorithm. Implicitly, the source of the covers is also known and understood. Such a formulation can be justified in part by Kerckhoffs' Principle, the assumption that an enemy (here the steganalyst) should be allowed to know the system used by the embedder, not least because such details could be revealed by a traitor.

In practice, such a clearcut binary problem would not arise. There is a wide range of embedding methods freely available on the web and it is unlikely that a steganalyst would know which was being used (without knowing whether it was being used or not). Even if the embedding algorithm were known, the size of embedded payload would probably not be known, certainly not exactly. The ability to identify the embedding algorithm and the payload size (*quantitative steganalysis*) are the first steps towards forensic steganalysis.[15]

Identification of the embedding algorithm makes for a multiclass problem; identification of the payload size can be approximated as a multiclass problem, but is probably best formulated as one of regression. Multiclass steganalysis has attracted interest from researchers in efforts to create universal detectors,[8] and there is a little work on regression for quantiative steganalysis,[16] but these have generally been modifications of the binary classifiers and based on SVMs or related machines. There are also two pieces of literature on novelty detection, which creates a one-class problem,[17, 18] both using a modification of standard SVM techniques.

Unfortunately, there is no guarantee that good binary classifiers generalise well to multiclass situations[15] and there is little evidence of research into different multinomial classification algorithms for steganalysis.

It is known that SVMs can be adapted to perform multinomial classification, but unfortunately they are not well-suited to it[19] because of the very expensive optimisation that is required for training. The SVM is a binary classifier by construction and therefore a multinomial SVM classifier is typically implemented using many binary SVM classifiers. For example, the popular SVM software LIBSVM uses pairwise training (also called "all-vs-all"), which transforms a multiclass classification problem with k classes into a set of $k(k-1)/2$ binary problems comparing each class against every other. One binary classifier is trained for each pair of classes and a final prediction is obtained by combining the outputs of the trained classifiers through a simple vote. Ambiguity arises when there is a tie in votes, which can be resolved by random choice. The downside of this multinomial SVM classification is the increased computational requirements for training: it requires solving $O(k^2)$ minimisation problems (although the size of training sets can perhaps be kept lower than would be needed for training a full multinomial classifier). An alternative is to train binary classifiers for the cover class against each stego class separately, but absent some LR-style output of "certainty" it then becomes difficult to combine the answers into a single decision function.

3. LOGISTIC REGRESSION

Logistic Regression (LR) is a well established machine learning tool, which is not limited to classification. As the name suggests, it is a form of regression (it gives a continuous output), estimating the probabilities that the item under consideration comes from a particular class. It can, of course, be used for classification by returning the class with highest estimated probability: in this situation, Logistic Regression usually has equivalent accuracy to the Support Vector Machine.[20] The availability of class-belonging probabilities also makes it very attractive for statistical inference, but we leave the exploration of this for future work.

There is an additional advantage, which is fast multiclass classification. In contrast to SVM, LR does not require pairwise or one-against-all training of multiple models and is capable of estimating class probabilities for k classes, and hence multinomial classification, with optimisation complexity of $O(k)$. Thus it should have an obvious speed advantage over SVMs in multiclass situations, as we shall investigate later.

Given some classes $C_1, \ldots, C_k$, let us write $\mathrm{P}(C_i|x)$ to mean the probability that observation x belongs to class C_i. In Logistic Regression the linear decision function $y(x) = w^T x + b$ is transformed to model the classifier's outputs $y(x)$ as the class-belonging probabilities $\mathrm{P}(C_i|x)$, for some class C_i. First let us assume two classes, C_1 and C_2 and note that $\mathrm{P}(C_1|x) = 1 - \mathrm{P}(C_2|x)$, so we only need to model the probabilities for one of the two classes, say C_1. Probabilities such as $\mathrm{P}(C_1|x)$ are bounded, so cannot reasonably be approximated by a linear function,[12] so instead of taking $y(x) = \mathrm{P}(C_1|x)$, we define:

$$y(x) = \ln\left(\frac{\mathrm{P}(C_1|x)}{\mathrm{P}(C_2|x)}\right), \tag{1}$$

i.e. the logit function, which describes the log of the ratio of probabilities for the two classes given the input instance x. Then we can derive $\mathrm{P}(C_1|x)$ from the inverse of the logit function:

$$\mathrm{P}(C_1|x) = \frac{1}{1+e^{-y(x)}} = \frac{1}{1+e^{-(w^T x + b)}} \tag{2}$$

which gives us a functional form for the model of output classification probabilities using the linear discriminant $y(x) = w^T x + b$.

To fit this probabilities model to our data we need to estimate parameters w, which is often done by maximum likelihood. Hence we compute the likelihood function in terms of the probability of training (observed) data as a function of w.[12] If we assume the given training data $x_1, \ldots, x_n$ form a random sample from a sequence of n Bernoulli trials, then the likelihood of observations $y_1, \ldots, y_n$ is:

$$L = \prod_{j=1}^{n} \mathrm{P}(C_1|x_j)^{y_j}(1 - \mathrm{P}(C_1|x_j))^{1-y_j} \tag{3}$$

The value of w which maximises this expression (or, more, conveniently, its logarithm) is its maximum likelihood estimate and is most likely to fit the training data. The maximisation problem is solved through optimisation, for example using a (pseudo-) Newton algorithm. When features are collinear (as is the case with the SPAM features we will use), the likelihood function is ill conditioned and its maximization subject to overfitting and instability.[21] To avoid this, an extra regularisation term can be added to the likelihood and parametrized by a hyperparameter λ (ridge):

$$\log L = \sum_{j=1}^{n} y_j \log \mathrm{P}(C_1|x_j) + (1-y_j)\log(1 - \mathrm{P}(C_1|x_j)) - \lambda\|w\| \tag{4}$$

where $\|w\| = \sqrt{\sum w_j^2}$ is the norm of the weight vector. When λ is zero the algorithm will find the estimate as per normal Newton method, and values of λ greater than zero will penalize large components in the estimate w, producing a more stable algorithm.[21] The suitable value for hyperparameter λ can only be found by trying a number of values and evaluating the performance of the trained machine. This stage of the learning process is called (hyper)parameter optimisation, and is often performed using grid search or another exhaustive technique.

Linear LR is highly scalable because the only attributes that define the decision function are the M weights (w) for an M-dimensional feature space, which means it can be applied to very large data sets (unlike SVMs, where the decision function is defined in terms of the support vectors). However, unlike in SVMs, the optimization is non-sparse, which means that every item of training data affects the weights, resulting in longer training in the binary case. For multinomial Logistic Regression, with k classes, the decision function takes the following form:

$$\mathrm{P}(C_i|x) = \frac{\exp(w_i^T x)}{\sum_j \exp(w_j^T x)} \tag{5}$$

This is called the softmax function and each inner product $w_j^T x$ represents relative log-odds of class C_j given observation x. The weight vectors w_j can be optimised individually ($O(k)$ optimisations) or as one large vector thus making some of the components correlated, which creates one larger optimisation problem but of smaller overall complexity. Even without sparsity this should be faster than the $O(k^2)$ figure for the pairwise SVM.

In Kernel Logistic Regression the weight vector w is replaced by its dual form representation $w = \sum x_i a_i$, as in SVMs.[22] Then the input vector x in (3) can be mapped to a higher-dimensional feature space using a transformation function (kernel) K, like[*]:

$$y(x) = \frac{1}{1 + e^{\sum a_i K(x, x_i)}} \tag{6}$$

This allows LR to learn non-linear decision boundaries, making it equally powerful to the kernelised SVM. There exists a variety of kernel functions (including polynomial, hyperbolic, and Gaussian) and they can be used with both algorithms. In steganalysis the most popular kernel is the radial basis function (RBF), and we employ the RBF kernel in our experiments too:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{7}$$

The kernel hyperparameter σ affects the complexity of the resulting classifier, and there is little one can do but test a wide range of parameters exhaustively: optimizing both λ and σ results in a two-dimensional grid search.

4. EXPERIMENTS

In the following experiments we test whether steganalysis using Logistic Regression has any value. To achieve this we need to compare it with state of the art alternative which is the SVM. We evaluate the performance of linear and kernel SVM and LR classifiers on both binary and multiclass problems. The comparison will be performed in terms of:

Detection accuracy: the proportion of correctly classified examples (measured on a testing set which is disjoint from the training data).

Speed: the time taken on the final training of the model, after the hyperparameters have been optimised.

The former seems fairly clear, though we note that the testing set will contain an equal number of examples of cover and stego objects (not necessarily a realistic situation for the deployment of steganalysis). For the latter there were a number of possible options: one can choose to benchmark training or testing, and if the former to include, or not include, the time taken to optimise the hyperparameters. We chose to measure training times because testing times are so fast as to be practically insignificant, and discarded the optimisation time because it is usually roughly proportional to the final training time. Our experimental design is based on statistical methods common to machine learning research as outlined in Ref. 23, and similar to that described in Ref. 24.

[*]An equivalent modification can be made for the multinomial kernel LR; see Ref. 22 for details.

4.1 Components

In order to evaluate the performance of the two algorithms in question we need four components: cover images, several embedding schemes, a method for feature extraction, and software to perform the classification. We now describe each in detail.

Cover Images

There will be three cover image sets in these experiments:

1. The images called "set B" in Ref. 25, 2000 greyscale images from a personal digital camera converted from RAW format and cropped to 400 × 300 pixels.
2. The initial image database from the Break Our Steganography System (BOSS) contest consisting of 7520 greyscale images, 512 × 512 pixels, from various sources.[26]
3. The 10000 grayscale images, 512 × 512 pixels, used during the Break Our Watermarking System (BOWS2) contest.[27] These images have been rescaled and cropped.

Embedding

We selected four spatial domain steganography techniques: LSB Matching (LSBM), LSB Replacement (LSBR), Mod-5 Matching (Mod5) and 2LSB Replacement (2LSB). The embedded message is broken down into (binary, quaternary, or quinary) symbols $m_1, \ldots, m_n$; using a pre-chosen seed as a stego key, these algorithms choose a random ordered sublist of the image's pixels for embedding, of length n. Taking one pixel x_i at a time, they embed the message, symbol-by-symbol, inserting m_i into x_i in one of the following ways. We assume that there is a set of possible pixel values V, typically $\{0, 1, \ldots, 255\}$.

For LSBR and LSBM, the payload symbols are binary. Under LSBR each payload bit replaces the least significant bit of the relevant pixel:

$$x'_i = 2\lfloor x_i/2 \rfloor + m_i \tag{8}$$

while under LSBM the pixel is altered so that its value matches the payload, but possibly affecting other bit planes:

$$x'_i \in \{x \,|\, x \bmod 2 = m_i,\ |x_i - x| \leq 1\} \cap V, \tag{9}$$

picking uniformly at random when there are two options in the set. LSBR has long been known to contain insecurities[13] which are not shared by LSBM, but machine-learning based steganalysis has not generally been able to exploit them well.

2LSB and Mod5 are 2 and 2.32 bit, respectively, versions of LSBR and LSBM. The former replaces two bit planes:

$$x'_i = 4\lfloor x_i/4 \rfloor + m_i \tag{10}$$

while in the latter it is the remainder (mod 5) that carries a quinary payload symbol (a number from 0 to 4): "the embedding function alters the cover pixel to the nearest value with the correct remainder."[28]

$$x'_i = \underset{\substack{x \in V \\ x \bmod 5 = m_i}}{\operatorname{argmin}} |x_i - x|. \tag{11}$$

Whilst LSB Matching and LSB Replacement are very well studied, 2LSB embedding and Mod-5 Matching have attracted a disproportionately small number of attacks; their security is yet to be determined properly by modern steganalysis algorithms. Some results in Ref. 29 suggest that multiple-bit per pixel embedding, which therefore alters fewer pixels, might have a security advantage.

The four embedding schemes were used to create stego sets. We applied the embedding changes to 50% (LSB Replacement and LSB Matching), 25% (2LSB Replacement) and 21.5% (Mod-5 Matching) of the pixels in the cover images, corresponding to a payload of 0.5 bits per pixel in each case. This is an intermediate payload size which, for LSB Matching, is reasonably but not perfectly detectable by SPAM features.[10] Embedding was performed in random locations with uniformly random payload, which is the common practice, as for example in Refs. 4 and 25.

The stego sets, plus the cover set, were then combined into three five-class multinomial classification problems, one for each of the three image sets B, BOSS, and BOWS. Also, four binary problems were created by combining a single embedding algorithm with the cover set (set B only).

Features

We employ the state-of-the-art 686-dimensional SPAM feature set,[10] which currently gives the most sensitive detector of spatial domain LSB Matching when coupled with a kernelised SVM. It is a collection of second order transition probabilities, modelling successive pixel differences along eight horizontal, vertical, and diagonal directions. We compute the empirical probability of the difference D_k between pixels i and $i-1$ given preceding differences D_{k-1} and D_{k-2}:

$$\mathrm{P}(D_k = u | D_{k-1} = v, D_{k-2} = w), \tag{12}$$

Following Ref. 10, only pixel differences within the range ± 3 are considered, and they are averaged over the "grid" and "diagonal" directions, for a total of 686 features.

Researchers have shown that SPAM features are successful on other non-LSB schemes,[9] however it is not a universal detector as its detection capabilities of HUGO[30] are limited. SPAM's performance on 2LSB Replacement and Mod-5 Matching has not been previously reported.

Software

Two software packages were used in these experiments:

LIBSVM is a popular C++-based library for SVMs, developed at National Taiwan University.[14] LIBSVM is based on Platt's highly optimised SMO algorithm[31] and is distributed under the GNU General Public License.

minFunc toolkit is a collection of MATLAB functions for solving optimisation problems.[32] The collection also includes multiple implementations of Logistic Regression and Support Vector Machines, including kernel and multiclass versions, and Smooth Support Vector Machines.[33]

We were unable to find a highly optimized implementation of LR which included both kernel and multinomial options. So there is a difficulty in creating a fair comparison between the SVM and LR methods. The former is widely-used and the available implementation is highly optimized, making use of the SMO algorithm and tightly coded in a low level programming language. A quasi-SMO technique does exist for LR,[34] but we did not find a suitable implementation of it, and although the inner optimization loop of **minFunc** is coded in C, the rest of the MATLAB interpreted code is no match for compiled C++ code.

To address this issue we use the Smooth SVM (SSVM) as a comparator. The SSVM is a reformulation of the SVM which leads to an unconstrained optimization: on one hand, its performance should be (and is) nearly identical to that of SVM, on the other it can be implemented using **minFunc** to provide a comparison with LR using the exact same minimizer. We will therefore check that SVM and SSVM have the same accuracy, and then compare LR and SSVM for speed.

Most of the experiments were conducted on dual-CPU quad-core machines with Intel Xeon E5540 processors at 2.53GHz and 24 GB of RAM, which were generously provided by the Oxford Supercomputing Centre. Cross-validation folds could be run in parallel using MATLAB's Parallel Computing Toolbox.

4.2 Experimental Setup

The experiments have the following design. Half of each data set was reserved for training and the other half for testing, to supply an estimate of classification accuracy on unseen examples. Hyperparameter selection was performed on the training set, using a simple grid-search, aiming for values such that the resulting accuracy is close to maximal for both algorithms. Under grid search the classifiers' regularisation parameters λ (ridge in LR) or C (cost in SVM), and for kernelised classifiers the RBF kernel parameter σ, are optimised using a set of values across a specified search range using geometric steps. Here the grid consists of sixteen values for λ and C,

$$\{i \cdot 10^j \mid i \in \{1, 5\},\, j \in \{-5, -4, \ldots, 2\}\} \tag{13}$$

(with closer examination around the best values) and twelve values for σ:

$$\{i \cdot 10^k \mid i \in \{1, 5\},\, k \in \{-3, -2, \ldots, 2\}\} \tag{14}$$

for a total search space of 192 samples. Each combination of parameters was evaluated on the training set using ten-fold cross-validation.

Having established the optimal hyperparameters, the entire training set was used for final training of the SVM or LR machine. Then the accuracy was measured on the testing set. All features were normalised to the range $[0, 1]$ on the training set, with the linear scaling factors saved for use on the testing set.

The testing set was also split into ten parts, to obtain ten measures of accuracy on smaller subsets, and these values used for a simple Student t-test comparing the accuracy of the different machines. Because there is no overlap between these subsets, there is no need for the "adjusted t-test" used in Ref. 25. Furthermore, the subsets were stratified such that each accuracy sample is fair. Stratification ensures the homogeneity of the sampled subsets, which improves the accuracy of classifiers performance estimation when trained and tested on small or difficult sets.

We use the percentage of correctly classified instances as the measure of accuracy, which is popular in this research field but must be interpreted with care. The problem arises when the training and test sets have different distributions of classes. In this case such a score function may lead to an incorrect estimate of the classifier's performance. Under our testing conditions this measure is sufficient. Note that a classifier's accuracy on a two-class problem is expected to be higher than on a multiclass problem, as there are fewer possibilities for error.

4.3 Results

Tables 1 and 2 show the results for the linear and kernelised classifiers, respectively. We first note that the accuracy rate of 97.67%, for the BOWS images and using kernelised SVM, is exactly in line with the error rate of 0.024 reported in Ref. 10. The tables confirm that the accuracy of SSVM is practically identical to that of SVM, as expected. They also show that, in most cases, the accuracy of LR is statistically indistinguishable from that of SVM: although there may be variations of around 1% in the absolute accuracy, the t-tests indicate that these fall within the margins of error given the test set sample sizes we used. The only exceptions are the five-class problem on the BOSS image set, where a marginally significant t-score is observed in favour of kernelised SVM over kernelised LR (this should probably be discounted, as the bar for significance must be raised when multiple tests are conducted) and very significantly poor performance for linear LR in the multinomial case. The latter is not observed in kernelised LR and we suspect that it might be caused by a bug, though one was not discovered: if others replicate these results, linear LR should be discounted for multinomial classification, but kernelised LR remains a good choice.

In almost all respects, then, our results suggest no evidence to dispute the hypothesis that LR, used as a classifier, is of comparable accuracy to SVM in this domain. It remains possible, of course, that larger-scale experiments would eventually reveal a significant difference in performance (though there is as-yet no particular pattern apparent).

With regards to the classifiers' speed, LIBSVM is vastly superior to both LR and Smooth SVM (timing was omitted, but is orders of magnitude faster). However, on a fair comparison between LR and SSVM using the

Table 1. Results for linear classifiers, showing accuracy of detection, a t-score comparing SVM and LR classifiers ($|t| > 2.26$ for significance at 5% level; $|t| > 3.25$ for significance at 1% level), and the time taken by SSVM and LR for the final training and testing experiment.

Image Set	Classes	Linear classifier accuracy			t-score	Time (sec)	
		SSVM	SVM	LR	SVM > LR	SSVM	LR
Set B	Cover/LSBM	94.84%	94.89%	95.19%	-0.20	5.84	3.33
Set B	Cover/Mod5	96.30%	96.25%	96.75%	-0.33	4.01	3.88
Set B	Cover/LSBR	97.35%	97.35%	97.00%	0.28	3.82	4.05
Set B	Cover/2LSB	98.20%	97.95%	97.90%	0.04	1.77	2.26
Set B	5 classes	80.99%	80.67%	50.92%	13.28	337.33	14.43
BOSS	Cover/LSBM	87.43%	87.16%	87.49%	-0.33	19.92	21.87
BOSS	5 classes	87.27%	86.83%	31.76%	64.44	860.61	318.60
BOWS	Cover/LSBM	97.66%	97.77%	97.65%	0.30	23.53	19.47
BOWS	5 classes	96.59%	96.86%	49.63%	73.49	1051.12	386.63

Table 2. Results for kernel classifiers, showing accuracy of detection, a t-score comparing SVM and LR classifiers ($|t| > 2.26$ for significance at 5% level; $|t| > 3.25$ for significance at 1% level), and the time taken by SSVM and LR for the final training and testing experiment.

Image Set	Classes	Kernelised classifier accuracy			t-score	Time (sec)	
		SSVM	SVM	LR	SVM > LR	SSVM	LR
Set B	Cover/LSBM	96.15%	96.20%	95.85%	0.43	106	145
Set B	Cover/Mod5	97.70%	97.55%	97.45%	0.11	95	160
Set B	Cover/LSBR	96.90%	97.15%	97.45%	-0.42	101	113
Set B	Cover/2LSB	98.35%	98.25%	98.45%	-0.23	85	127
Set B	5 classes	80.89%	83.01%	80.05%	1.68	2504	1394
BOSS	Cover/LSBM	86.60%	87.83%	85.43%	1.74	1708	2247
BOSS	5 classes	83.81%	85.05%	82.30%	2.90	43003	23446
BOWS	Cover/LSBM	98.01%	97.67%	97.89%	-0.72	2727	3452
BOWS	5 classes	96.59%	95.95%	96.32%	-1.44	84198	42311

same minimization procedure, LR is sometimes moderately (up to approx. 50%) slower than SSVM on the binary problems but more than twice as fast on the multiclass problems. This confirms the theoretical advantage of multinomial LR over multinomial SVM because a single machine, albeit one of more complexity than in the binary case, can be trained as described in section 3. This reinforces our confidence in the suitability of LR for multiclass steganalysis.

4.4 Comparative Security of LSB Steganography Schemes

The above results can also be used to test the comparative security of the four embedding schemes that we studied. In particular, we are interested to see whether LSB Replacement and LSB Matching are more secure

Table 3. Results of the t-test evaluating security (with respect to SPAM features) of LSB Replacement against 2LSB replacement and LSB Matching against Mod-5 Matching. $|t| > 2.26$ for significance at 5% level, $|t| > 3.25$ for significance at 1% level.

Image Set	Classes	Linear classifier accuracy		Kernelised classifier accuracy	
		SVM	LR	SVM	LR
Set B	Cover/LSBM	94.89%	95.19%	96.20%	95.85%
Set B	Cover/Mod5	96.25%	96.75%	97.55%	97.45%
t-score	LSBM > Mod5	0.90	1.11	1.78	5.46
Set B	Cover/LSBR	97.35%	97.00%	97.15%	97.45%
Set B	Cover/2LSB	97.95%	97.90%	98.25%	98.45%
t-score	LSBR > 2LSB	0.50	1.18	1.51	1.04

than their two- (or approximately two-) bit versions, 2LSB Replacement and Mod-5 Matching, respectively. This has been studied in Ref. 29 (for the case of bit replacement only, against structural detectors) and using information-theoretic measures in Ref. 28.

The detection accuracy is again compared using t-tests, and the results displayed in Table 3. In all, neither linear nor kernel classifiers showed much evidence of the one-bit algorithms being more or less secure than their respective two-bit versions. Only in one case, against kernelised LR, did it appear that LSB Matching was more secure against SPAM features than Mod-5 Matching, with the t-score of 5.46 ($p < 0.001$); however, the same difference is not observed with SVM and is just as likely to be a slight weakness in the classifier as a security advantage in the embedding.

We draw two lessons from this data. First, that SPAM features are not significantly more or less sensitive to multi-bit embedding. Second, that performance gaps of 1% or even 1.5% (e.g. the first column of Table 3) do *not* represent statistically significant differences, when the testing set is only 1000 images (as here, half of image set B). On the other hand, the gap of 1.6% in the last colum of Table 3 *is* significant, because in this case the detection rate of both classifiers was much more stable. Authors should be cautious to claim improvements in security or detectors simply by quoting a slightly altered accuracy rate: it can only be determined by proper statistical tests.

5. CONCLUSIONS AND FURTHER WORK

We have compared the performance, in terms of accuracy and speed, of Support Vector Machines and Logistic Regression classification engines. Our experiments only extended to SPAM steganalysis features but there is no reason to assume that there would be any difference with different features. By using four different spatial-domain embedding schemes, we tested four binary and three five-class problems.

Our conclusion is that the accuracy of detection, in both binary and multiclass scenarios, is approximately equal. When comparing the speed of algorithms running on the same minimizer, LR has a small disadvantage compared with (Smooth) SVM in the binary case, and a substantial advantage in the five-class case. It is likely that this advantage would grow with the number of classes. However, the speed benchmark applies to an artificial situation and does not use the best implementation or the best SVM optimization algorithm: in order to make these conclusions more robust, it is necessary to find (or create) an optimized low-level implementation of LR which uses a quasi-SMO algorithm. Only then can true performance be compared.

Logistic Regression has value beyond its multiclass performance, since it outputs estimated class probabilities. There should be many applications of this information in steganalysis situations (we have in mind the particular problem of *pooled steganalysis*[35]) but we leave them to future work.

We stress the importance of testing any observed accuracy differences for statistical significance. In this work we used a Student t-test, by splitting the test set into ten subsets (implicitly assuming a Gaussian fit for accuracy across subsets), but other methods are available. It was notable that, for example in Subsect. 4.4, accuracy differences of around 1% were not at all significant and could equally be attributed to random chance in the selection of the testing set, as to a true difference in performance.

Finally, we comment on the size of our image sets. Set B has 2000 images, providing only 1000 for training, compared with a feature set of 686 dimensions. In normal circumstances, this would be grossly insufficient, yet we still observe good accuracy (and, though not displayed here, very similar performance on the training and testing set, hence no indication of overfitting). We attribute this to the high degree of correlation in the SPAM features, so that they do not fully occupy nearly as many as 686 dimensions. In any case, the limits of our computing systems' memory was reached with the BOSS and BOWS image sets. Nonetheless, larger image sets would be advantageous in terms of statistical power.

REFERENCES

[1] Farid, H., "Detecting hidden messages using higher-order statistical models," in [*Proc. 2002 International Conference on Image Processing*], **2**, 905–908 (2002).

[2] Harmsen, J. and Pearlman, W., "Higher-order statistical steganalysis of palette images," in [*Security and Watermarking of Multimedia Contents V*], *Proc. SPIE* **5020**, 131–142 (2003).

[3] Fridrich, J., "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes," in [*Proc. 6th Information Hiding Workshop*], *Springer LNCS* **3200**, 67–81 (2004).

[4] Goljan, M., Fridrich, J., and Holotyak, T., "New blind steganalysis and its implications," in [*Security, Steganography and Watermarking of Multimedia Contents VIII*], *Proc. SPIE* **6072**, 0101–0113 (2006).

[5] Avcibas, I., Memon, N., and Sankur, B., "Steganalysis of watermarking techniques using image quality metrics," in [*Security, Steganography, and Watermarking of Multimedia Contents III*], *Proc. SPIE* **4314**, 523–531 (2001).

[6] Farid, H. and Lyu, S., "Detecting hidden messages using higher-order statistics and support vector machines," in [*Proc. 5th Information Hiding Workshop*], *Springer LNCS* **2578**, 340–354 (2002).

[7] Shi, Y., Chen, C., and Chen, W., "A Markov process based approach to effective attacking JPEG steganography," in [*Proc. 8th Information Hiding Workshop*], *Springer LNCS* **4437**, 249–264 (2006).

[8] Pevný, T. and Fridrich, J., "Multiclass detector of current steganographic methods for JPEG format," *IEEE Transactions on Information Forensics and Security* **3**(4), 635–650 (2008).

[9] Kodovsky, J., Pevný, T., and Fridrich, J., "Modern steganalysis can detect YASS," in [*Media Forensics and Security XII*], *Proc. SPIE* **7541**, 0201–0211 (2010).

[10] Pevný, T., Bas, P., and Fridrich, J., "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on Information Forensics and Security* **5**(2), 215–224 (2010).

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

[12] Hosmer, D. and Lemeshow, S., [*Applied Logistic Regression*], Wiley (2000).

[13] Ker, A., "A general framework for structural analysis of LSB replacement," in [*Proc. 7th Information Hiding Workshop*], *Springer LNCS* **3727**, 296–311 (2005).

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

[15] Fridrich, J., [*Steganography in Digital Media: Principles, Algorithms, and Applications*], Cambridge University Press (2009).

[16] Pevný, T., Fridrich, J., and Ker, A., "From blind to quantitative steganalysis," in [*Media Forensics and Security XI*], *Proc. SPIE* **7254**, 0C01–0C14 (2009).

[17] Lyu, S. and Farid, H., "Steganalysis using color wavelet statistics and one-class support vector machines," in [*Security, Steganography, and Watermarking of Multimedia Contents VI*], *Proc. SPIE* **5306**, 35–45 (2004).

[18] Pevný, T. and Fridrich, J., "Novelty detection in blind steganalysis," in [*Proc. 10th ACM workshop on Multimedia and Security*], *MM&Sec '08*, 167–176, ACM (2008).

[19] Hsu, C.-W. and Lin, C.-J., "A comparison of methods for multi-class support vector machines," *IEEE Transactions on Neural Networks* **13**, 415–425 (2002).

[20] Hastie, T., Tibshirani, R., and Friedman, J., [*The Elements of Statistical Learning: Data Mining, Inference, and Prediction*], Springer (2003).

[21] Schaefer, R., "Alternative estimators in logistic regression when the data are collinear," *Journal of Statistical Computation and Simulation* **25**(1-2), 75–91 (1986).

[22] Roth, V., "Probabilistic discriminative kernel classifiers for multi-class problems," in [*Proc. 23rd DAGM Symposium on Pattern Recognition*], *Springer LNCS* **2191**, 246–253 (2001).

[23] Witten, I. and Frank, E., [*Data Mining: Practical Machine Learning Tools and Techniques*], Morgan Kaufmann, second ed. (2005).

[24] Pevný, T., *Kernel Methods in Steganalysis*, PhD thesis, Binghamton University, SUNY (2008).

[25] Ker, A. and Lubenko, I., "Feature reduction and payload location with WAM steganalysis," in [*Media Forensics and Security XI*], *Proc. SPIE* **7254**, 0A01–0A13 (2009).

[26] Bas, P., Filler, T., and Pevný, T., "BOSS: Break Our Steganographic System." `http://boss.gipsa-lab.grenoble-inp.fr` (2010).

[27] Bas, P. and Furon, T., "BOWS2." `http://bows2.gipsa-lab.inpg.fr` (2007).

[28] Ker, A., "Estimating the information theoretic optimal stego noise," in [*Proc. 8th International Workshop on Digital Watermarking*], *Springer LNCS* **5703**, 184–198 (2009).

[29] Ker, A., "Steganalysis of embedding in two least significant bits," *IEEE Transactions on Information Forensics and Security* **2**, 46–54 (2007).

[30] Pevný, T., Filler, T., and Bas, P., "Using high-dimensional image models to perform highly undetectable steganography," in [*Proc. 12th Information Hiding Conference*], *Springer LNCS* **6387**, 161–177 (2010).

[31] Platt, J., "Using analytic QP and sparseness to speed training of support vector machines," in [*Proc. Advances in Neural Information Processing Systems II*], 557–563, MIT Press (1999).

[32] Schmidt, M., "minFunc software for unconstrained optimisation." Software available at `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html` (2005).

[33] Lee, Y.-J. and Mangasarian, O., "SSVM: A smooth support vector machine for classification," *Computational Optimization and Applications* **20**(1), 5–22 (2001).

[34] Keerthi, S., Duan, K., Shevade, S., and Poo, A., "A fast dual algorithm for kernel logistic regression," *Machine Learning* **61**, 151–165 (2005).

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