Ensemble classification in steganalysis – Cross-validation and AdaBoost

Jan Kodovský
jan.kodovsky@binghamton.edu

Abstract

Two alternative designs to the ensemble classifier proposed in [13] are studied in this report. First, the out-of-bag error estimation is replaced with cross-validation. Second, we incorporate AdaBoost and modify the weights of the individual training samples as the training progresses. The final decision is formed as a weighted combination of individual predictions rather than through majority voting. We experimentally compare both alternatives with the original design and conclude that they bring no performance gain.

1 Introduction

Today's most accurate steganalysis methods for digital media are built as supervised classifiers trained on feature vectors extracted from the media. Even though the support vector machine (SVM) seems to be the most popular machine learning tool used in steganalysis, SVMs are quite restrictive when it comes to the number of training samples and/or the feature space dimensionality due to rapidly increasing complexity of training. In [12] we proposed an alternative classification tool to SVMs, ensemble classifiers. Modern feature spaces go high-dimensional and the complexity of ensemble classifiers scales much more favorably w.r.t. the feature space dimensionality than SVMs while delivering a comparable performance. More recently, in [13] we made a further step in the development of the ensemble-based steganalysis framework by removing the need for manual pre-determination of the two ensemble parameters – the number of base learners and the random subspace dimensionality. We proposed a fully automatized steganalysis tool that utilizes out-of-bag (OOB) error estimates as part of the model-selection feedback during the training process.

In this technical report, we discuss two alternative designs to the one proposed in [13]. First, we replace bootstrapping with k-fold cross-validation (CV) and utilize the combined error estimate coming from the k folds (instead of OOB). Second, the ensemble classifier may be boosted through the technique of AdaBoost [6]. We provide experimental results showing that neither of these two modifications yields a performance gain under several different steganalysis scenarios. Therefore, we recommend to use the simple and elegant design described in [13].

This report is organized as follows. In Section 2, we briefly describe the ensemble classifier and its components, focusing on parts relevant to this report. Section 3 presents an alternative design based on k-fold cross-validation and includes selected experiments showing similar results of both approaches. AdaBoost is applied in Section 4, also accompanied with comparative experiments from steganalysis. Conclusions are drawn in Section 5.

Our Matlab implementation of the fully automatized ensemble classifier is available at `http://dde.binghamton.edu/download/ensemble/`.

2 Ensemble classifier

2.1 Overview

The ensemble classifier proposed in [13] was designed to keep low complexity and overall simplicity. It consists of L independently trained base learners implemented as the Fisher Linear Discriminants (FLD) [5] each built on a (uniformly) randomly selected subspace of the original feature space. Furthermore, each learner is trained on a bootstrap sample drawn from the training set rather than on the whole training set. There are two reasons for that. First, different training sets increase mutual diversity of base learners, a crucial property for the success of any ensemble-based learning. Second, and more importantly, points that have not been used for training can be used for testing error estimation, which is an essential part of the whole framework.

Let us formalize the concepts; we use the same notation as in [13]. The original d-dimensional feature space is denoted $\mathcal{F} \equiv \mathbb{R}^d$, the symbols N^{trn} and N^{tst} denote the number of training and testing samples from each class, $\mathbf{x}_m, \bar{\mathbf{x}}_m \in \mathbb{R}^d$, $m = 1, \ldots, N^{\text{trn}}$, stand for the cover and

stego feature vectors computed from the training set and $\mathbf{y}_k, \bar{\mathbf{y}}_k \in \mathbb{R}^d$, $k = 1, \ldots, N^{\text{tst}}$, for features obtained from the testing cover and stego samples, respectively. The set of all training and testing samples will be denoted $\mathcal{X}^{\text{trn}} = \{\mathbf{x}_m, \bar{\mathbf{x}}_m\}_{m=1}^{N^{\text{trn}}}$ and $\mathcal{Y}^{\text{tst}} = \{\mathbf{y}_k, \bar{\mathbf{y}}_k\}_{k=1}^{N^{\text{tst}}}$. For $\mathcal{D} \subset \{1, \ldots, d\}$, $\mathbf{x}^{(\mathcal{D})}$ is a $|\mathcal{D}|$-dimensional feature vector consisting only of those features from $\mathbf{x}$ whose indices are in $\mathcal{D}$, preserving their original order.

Each base learner B_l, $l = 1, \ldots, L$, is a mapping $\mathbb{R}^{d_{\text{sub}}} \to \{0, 1\}$, '0' standing for cover and '1' for stego, trained on $\mathcal{X}_l = \{\mathbf{x}_m^{(\mathcal{D}_l)}, \bar{\mathbf{x}}_m^{(\mathcal{D}_l)}\}_{m \in \mathcal{N}_l^{\text{b}}}$, where $D_l \subset \{1, \ldots, d\}$, $|D_l| = d_{\text{sub}} \ll d$, is the lth random subspace of $\mathcal{F}$ and $\mathcal{N}_l^{\text{b}}$ is the lth bootstrap sample of the set of indices $\{1, \ldots, N^{\text{trn}}\}$. As only about 63% of the training samples lie in $\mathcal{X}_l$, the remaining 37% can be used for error estimation as follows. Each $\mathbf{x} \notin \mathcal{X}_l$ is provided a single vote from the base learner B_l, and thus after n base learners are trained, each training sample $\mathbf{x} \in \mathcal{X}^{\text{trn}}$ collects on average $0.37 \cdot n$ predictions. These are fused using the majority voting strategy into a single prediction $B^{(n)}(\mathbf{x}) \in \{0, 1\}$. Collectively, the predictions $B^{(n)}(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}^{\text{trn}}$ form the so-called "out-of-bag" (OOB) error estimate:

$$E_{\text{OOB}}^{(n)} = \frac{1}{2N^{\text{trn}}} \sum_{m=1}^{N^{\text{trn}}} \left(B^{(n)}(\mathbf{x}_m) + 1 - B^{(n)}(\bar{\mathbf{x}}_m) \right), \quad (1)$$

which is known to be an unbiased estimate of the testing error.

The error estimates $E_{\text{OOB}}^{(n)}$, $n = 1, 2, \ldots$ provide an essential feedback for automatization of the whole procedure, as the framework is parametrized by L, the number of base learners, and by d_{sub}, the random subspace dimensionality. For a fixed value of d_{sub}, keeping track of the progress of $E_{\text{OOB}}^{(n)}$ allows one to determine L, i.e., to stop generating more random subspaces (and base learners) once $E_{\text{OOB}}^{(n)}$ converges. This can be done by defining a stopping criterion. Once converged, $E_{\text{OOB}} \equiv E_{\text{OOB}}^{(L)}$ serves as the final out-of-bag error estimate, an unbiased estimate of a real testing error (with that fixed value of d_{sub}). The sketched procedure can be repeated for different values of d_{sub}, allowing us to search for the "optimal" value of d_{sub}. A detailed description of this fully automated system appears in [13].

2.2 Relationship to prior art

Ensemble classification, the process of boosting the accuracy of a set of "weaker" base learners by aggregation, is not a new idea in the machine learning community [6, 18]. The novelty lies in its application to high-dimensional steganalysis and in the fully automated design capable of determining the optimal subspace dimensionality (and the sufficient number of base learners). Our design combines the element of bagging (bootstrap aggregating), a well-established technique for reducing the variance of classifiers [2], with a random subspace forming strategy, known under different names – decision forest [9], attribute bagging [4], CERP (Classification by Ensembles from Random Partitions) [1], or the recently proposed RSE (Random Subsample Ensemble) [20]. All these are ensemble-based classifiers sampling the feature space prior base learner training to either increase diversity among classifiers or reduce the original high dimension into manageable values.

The proposed ensemble classifier could be viewed as an instance of a *random forest* proposed by Breiman [3]. Random forest is an extension of bagging – it also trains individual base learners on bootstrap samples of the training set, but they are additionally somehow randomized, i.e., dependent on a random vector that is drawn independently and from the same distribution for all base learners. For example, if the base learner is a decision tree (as in [3]), its splitting variables can be chosen randomly. The final prediction is formed as a majority vote. The additional randomization in random forests, as compared to bagging, increases the diversity of individual base learners and usually speeds-up the training – when only a small (randomly chosen) subset of variables is chosen for tree splitting, the best split is found much faster than if all variables were considered. The price for this randomization is a worse performance of individual base learners. However, when combined together through majority voting, one can obtain comparable (or even better) results to bagging or AdaBoost [6].

Note that, unlike in AdaBoost, individual base learners are treated equally in forming the final decision in random forests (the majority voting is used) – this is because all the base learners were generated using the same random procedure. In AdaBoost, on the other hand, the final decision is formed as a weighted combination of individual predictions, with weights corresponding to the individual accuracy of every base learner. Furthermore, the training weights in AdaBoost are being continuously updated so that every base learner focuses more on those samples that were more difficult to classify by previous base learners. More on AdaBoost appears in Section 4.

Back to our proposed scheme – it is a random forest with the FLD as a base learner instead of a random decision tree as in [3]. Our randomization is in the feature subspace generation, and is a crucial part of the system as using a full feature space would be computationally intractable due to its high dimension.

3 Cross-validation

The k-fold cross-validation is a general procedure for estimating the prediction error of any supervised classifier [8]. Thus, cross-validation (CV) can be incorporated into our ensemble framework and its error estimates can be used instead of OOB error estimates for the purpose of automatization of the search for d_{sub} and the stopping criterion for the number of base learners L. In this section, we describe this modified ensemble design in more details, and experimentally compare its performance to the original design using OOB error estimates. We will refer to the original design as the OOB-ensemble and to the new, cross-validation-based design, as the CV-ensemble.

3.1 Modification

The OOB-ensemble trains every base learner B_l, $l = 1, \ldots, L$, on a bootstrap sample drawn from the original training set, as described in Section 2.1. Every bootstrap sample contains roughly 2/3 of all the training samples, and the trained FLD assigns one vote to the last third of the samples that have not been used for its training.

In order to incorporate k-fold cross-validation, we need to divide the training set into k equally populated folds at the very beginning of the training process. Instead of training each base learner on a bootstrap sample and evaluating it on the out-of-bootstrap sample points, it will be trained on $k-1$ folds and evaluated on the remaining fold. This will be repeated k times, leaving subsequently all the folds out and using them for error estimation.

The OOB-ensemble and the CV-ensemble are thus quite similar, and differ in the following. After the first k base-learner trainings (the first base learner for each fold), the CV-ensemble returns exactly one vote for each training sample, as each of them was left out exactly once. On the other hand, the number of votes per training sample after the first k base-learner trainings of the OOB-ensemble follows the binomial distribution with the mean around $0.37 \cdot k$, as roughly 37% of the training samples are left-out in every round. So, as the ensemble training proceeds and after training the total of n base learners, the individual predictions of the CV error estimate are formed at every moment from n/k votes (for example $0.2 \cdot n$ votes in case of five-fold cross-validation), while in the case of the OOB error estimate, they are formed from roughly $0.37 \cdot n$ votes on average. Consequently, the OOB error estimate typically converges faster, for the price that the individual base learners are trained on a smaller number of unique samples, and the training points do not have identical number of votes, unlike in the case of CV.

Another difference is that in the CV approach, the folds are formed in the beginning and remain the same for the whole training process, while in the OOB approach, a new bootstrap sample is drawn every time, yielding higher diversity.

The complexity of both approaches is similar.

3.2 Experimental comparison

We implemented the CV-ensemble, and experimentally compared its performance to the OOB-ensemble. There are two important points regarding the CV implementation we would like to make.

First, the folds need to be created in a way that the cover-stego pairs are preserved, i.e., the pairs of features coming from cover and the corresponding stego images should not be separated into two different folds. This issue was first pointed out in [19] and then studied in more details in [10]. If the pairs were not preserved, the resulting CV error estimate might be heavily biased and the overall performance of the ensemble would be sub-optimal. Obviously, the same holds for the OOB-ensemble, the individual bootstrap samples need to be drawn "by pairs."

Second, as the training process of the CV-ensemble can be reinterpreted as training k parallel ensembles, the question is: should we use the same random subspaces for those k parallel ensembles? In other words, should we always use the same random subspace for every k-tuple of the subsequent base learners? Indeed, it would make sense to do it this way in order to give each of the training point votes from the same subspaces. However, it turns out that generating a new random subspace every time gives slightly better results. This is not surprising, as it yields better diversity. Thus, we generate new random subspaces every time.

Experiments were conducted on the CAMERA image database containing 6,500 JPEG images originally acquired in their RAW format taken by 22 digital cameras, resized so that the smaller size is 512 pixels with aspect ratio preserved, converted to grayscale, and finally compressed with JPEG quality factor 75 using Matlab's command `imwrite`. The images were randomly divided into two halves for training and testing, respectively.

We steganalyzed three different JPEG domain steganographic algorithms: MB1 [16], YASS [17], and nsF5[1] [7],covering a wide range of payloads (YASS settings). These three algorithms represent three different

[1]The stego images were obtained using an nsF5 simulator available at `http://dde.binghamton.edu/download/nsf5simulator/`

embedding paradigms: MB1 is a model-preserving technique, YASS is a robust embedding that masks its impact by subsequent JPEG compression, and nsF5 minimizes the embedding impact. We used three different feature sets: CC-PEV [11] to detect MB1, CDF [14] to detect YASS (settings 3, 8, 10, 11, and 12 as reported in [14]) and the 7,850-dimensional $\mathcal{CF}^*$ set [13] to detect nsF5. These choices were made to cover a wide spectrum of feature types and dimensions.

We trained both types of ensemble classifiers (OOB and CV) for every payload (or YASS setting) separately, repeated all the experiments over 10 different splits of the CAMERA database into a training and testing set, and report the obtained median (MED) testing errors and median absolute deviation (MAD) values in Table 1. We conclude that both implementations of the ensemble classifier yield similar results and thus either of them can be used.

Steganalysis of MB1 using CC-PEV features

payload	MED		MAD	
(bpac)	OOB	CV	OOB	CV
0.01	0.3849	0.3874	0.00260	0.00195
0.02	0.2810	0.2827	0.00340	0.00185
0.03	0.1966	0.1970	0.00175	0.00240
0.04	0.1271	0.1277	0.00220	0.00180
0.05	0.0815	0.0825	0.00105	0.00220

Steganalysis of YASS using CDF features

payload	MED		MAD	
(bpac)$^+$	OOB	CV	OOB	CV
0.187 (3)	0.0230	0.0229	0.00080	0.00160
0.138 (8)	0.0753	0.0743	0.00125	0.00075
0.159 (10)	0.0514	0.0500	0.00240	0.00255
0.114 (11)	0.0718	0.0718	0.00150	0.00085
0.077 (12)	0.1281	0.1288	0.00080	0.00185

$^+$The number in parentheses denotes YASS setting

Steganalysis of nsF5 using $\mathcal{CF}^*$ features

payload	MED		MAD	
(bpac)	OOB	CV	OOB	CV
0.05	0.3393	0.3393	0.00155	0.00245
0.10	0.1727	0.1734	0.00200	0.00155
0.15	0.0726	0.0714	0.00140	0.00115
0.20	0.0264	0.0285	0.00085	0.00050

Table 1: Steganalysis of MB1, YASS, and nsF5 using three different feature sets. Two different implementations of the ensemble classifier are compared – the OOB-ensemble and the CV-ensemble. We report the median (MED) and median absolute deviation (MAD) over 10 different splits of the CAMERA database into a training and a testing set.

4 AdaBoost

AdaBoost [6] is an ensemble-based algorithm that trains individual base learners sequentially and every base learner focuses on those samples that were more difficult to classify by previous base learners. The final decision is formed as a weighted combination of individual predictions; the weights correspond to the individual accuracy of every base learner. It is a deterministic meta-algorithm, and any classifier capable of handling weighted training samples can be used as a base learner.

4.1 Overview

To formalize the concepts, given the training set $\mathcal{X}^{\text{trn}} = \{\mathbf{x}_m, \bar{\mathbf{x}}_m\}_{m=1}^{N^{\text{trn}}}$, let $w_m^{(l)}, \bar{w}_m^{(l)}$, $m = 1, \ldots, N^{\text{trn}}$ be the corresponding weights of the training cover samples ($\mathbf{x}_m$) and stego samples ($\bar{\mathbf{x}}_m$) after the lth base learner B_l is trained. Alternatively, we may use the vector notation $\mathbf{w}^{(l)}, \bar{\mathbf{w}}^{(l)} \in \mathbb{R}^{N^{\text{trn}}}$. We keep treating cover and stego samples separately as it will be useful later in this section. The weights are initialized as

$$w_m^{(0)} = \bar{w}_m^{(0)} = \frac{1}{2N^{\text{trn}}},\ \forall m = 1, \ldots, N^{\text{trn}}. \tag{2}$$

The lth base learner B_l, $l = 1, 2, \ldots$, is trained on $\mathcal{X}^{\text{trn}}$ with weights $\mathbf{w}^{(l-1)}, \bar{\mathbf{w}}^{(l-1)}$. Its error on the training set $\mathcal{X}_l$ may be expressed as

$$\epsilon_l == \frac{1}{2N^{\text{trn}}} \sum_{m=1}^{N^{\text{trn}}} \left(B_l(\mathbf{x}_m) + 1 - B_l(\bar{\mathbf{x}}_m) \right), \tag{3}$$

where $B_l(\mathbf{x})$ is the cover (0) or stego (1) prediction of the base learner on the sample $\mathbf{x}$. The weight of the $l$the *base learner* B_l is defined as

$$\alpha_l = \frac{1}{2} \ln \left(\frac{1 - \epsilon_l}{\epsilon_l} \right). \tag{4}$$

Once the lth base learner is trained, the training sample weights are updated as

$$w_m^{(l)} = \frac{1}{Z_l} \cdot e^{-\alpha_l (-1)^{B_l(\mathbf{x}_m)}},\ m = 1, \ldots, N^{\text{trn}}, \tag{5}$$

$$\bar{w}_m^{(l)} = \frac{1}{Z_l} \cdot e^{+\alpha_l (-1)^{B_l(\bar{\mathbf{x}}_m)}},\ m = 1, \ldots, N^{\text{trn}}, \tag{6}$$

where Z_l is the normalization factor ensuring that $\sum_{m=1}^{N^{\text{trn}}} w_m^{(l)} + \bar{w}_m^{(l)} = 1$.

After L base learners are trained, the final ensemble prediction on a given (testing) sample $\mathbf{y}$ is formed as

$$B(\mathbf{y}) = \begin{cases} 0 & \text{if } \sum_{l=1}^{L} \alpha_l B_l(\mathbf{y}) < 0.5 \\ 1 & \text{if } \sum_{l=1}^{L} \alpha_l B_l(\mathbf{y}) > 0.5 \\ \text{random} & \text{otherwise} \end{cases} \quad (7)$$

4.2 Application to steganalysis

AdaBoost, as proposed in [6] and briefly described in the previous section, can be applied to the ensemble framework proposed in [13] in several ways.

The first option is to boost individual FLDs using AdaBoost, and use these boosted FLDs, each of them trained in a different feature space and on a different bootstrap sample of the training set, as base learners for the final ensemble framework. This idea was used for example in [21], where the authors applied it for image classification. We would need to replace the standard Fisher Linear Discriminant with its weighted counterpart, because the standard FLD does not accept weighted training samples at the input. This can be done for example as described in [15], using generalized definitions of a mean and a covariance matrix that incorporate sample weights. However, we would lose the ability to accurately estimate the testing error of the ensemble because the continuously monitored predictions of individual training samples would now be formed as different *weighted* combinations of different *number* of votes, and thus very few of these combined predictions would be formed in the same way as the final testing predictions would be. This is not an issue when we use the majority voting strategy of equally important predictions as there we have a guarantee that none of the predictions used for error estimation misses an *important* vote or is formed only by votes that are not important. Furthermore, even though the complexity of the generalized FLD is similar to the one of the standard FLD, the complexity of the whole system would multiplicatively grow by the number of iterations needed by AdaBoost for boosting every single FLD, which is not desirable. Therefore, we do not further pursue this direction.

The second option is to use the whole ensemble as a single base learner for AdaBoost, which is the other way of assembling AdaBoost and bagging. However, the complexity of this approach would grow in the same manner as with the previous idea, and even here we would lose the convenience of estimating the testing error simply as OOB estimates, and some sort of additional cross-validation would need to be used. Thus we do not pursue this direction neither.

The last option is to incorporate the ideas of AdaBoost into the framework *directly*, i.e., to keep adjusting the training sample weights as the training progresses, and form the final testing predictions according to the rule (7). However, unlike in the original AdaBoost, there are two non-trivial problems that need to be resolved as every base learner is trained:

1. in a different feature space (in the random subspace of the original space $\mathcal{F}$).
2. on different training samples (bootstrap samples from the original training set).

The first point, i.e. a different feature space every time a base learner is trained, can be seen as a property of a base learner itself – each learner is able to utilize only a portion of the feature space. This may be a problem in situations where a small number of features is responsible for majority of the classification accuracy, because the "importance" of a given training sample $\mathbf{x}_m$, described by a single weight w_m, may differ from feature space to feature space. Fortunately, this does not happen in modern steganalysis where the power of a feature space is typically spread across all the features (unless the steganography is fatally flawed).

The second point is more challenging – the lth base learner is trained on $\mathcal{X}_l = \{\mathbf{x}_m, \bar{\mathbf{x}}_m\}_{m \in \mathcal{N}_l^{\mathrm{b}}}$, where $\mathcal{N}_l^{\mathrm{b}}$ is the lth bootstrap sample of the set of indices $\{1, \ldots, N^{\mathrm{trn}}\}$, i.e., roughly 37% of the training cover-stego pairs are omitted in its training. This poses the following questions. Should we update the weights, using formulas (5) and (6), of all the training samples or only those that belong to $\mathcal{X}_l$? When calculating error ϵ_l in order to obtain the base-learner weight α_l, should we use all the training samples or only those that belong to $\mathcal{X}_l$? The situation gets complicated due to the fact that every time a *different* set of 37% samples is omitted. We could either use all the training samples for updating, knowing that some weights may be adjusted incorrectly as some points would be classified differently if they were part of the training, or the weights would be updated unevenly, only to the samples from $\mathcal{X}_l$ at the lth iteration. This may influence the convergence properties of the ensemble and, more importantly, it would have a negative impact on the accuracy of out-of-bag estimates, a crucial element of the system needed for determination of parameters d_{sub} and L.

4.3 The proposed system

An appealing (and simple) way of resolving the problem is to use the cross-validation variant of the ensemble described in Section 3. Since the folds in k-fold cross-validation are formed at the very beginning and remain the same for the rest of the training process, every k-th base-learner is trained *exactly* on the same training

samples, i.e., on all folds but the kth one. Therefore, viewing the entire process as training k parallel sub-machines, we could apply AdaBoost to each of them individually by keeping track of k different sets of weights. This way, each of the boosted sub-ensembles produces $1/k$ of equally important predictions for error estimating purposes, the CV error estimate corresponds to the way testing predictions will be made, and the algorithms for automatic determination of parameters d_{sub} and L can be used as before with the caveat that we use the CV error estimates instead of the OOB estimates.

The implementation of the system is straightforward: we need k sub-classifiers trained in parallel, each of them implemented as weighted FLDs boosted via AdaBoost as described in Section 4.1. After every step of the training process, the predictions of the folds left out are updated using the weighted rules (7), and combined from all k folds together to form the updated value of the CV error estimate. The training stops once this error estimate converges, for which we may use the same stopping criterion as in the original ensemble in [13]. We also apply the same search algorithm for finding the optimal value of d_{sub}.

Before we proceed to the experimental evaluation of the described system, we need to make the following comment. As already mentioned in Section 3.2, binary classifiers used for steganalysis should be trained on pairs of cover-stego features. Keeping this in mind, not only do we need to create the folds for cross-validation in a way to preserve these pairs of features, but we also have to modify the weights updating rules of the classical AdaBoost. In the original formulas (5) and (6), the weight of every training sample is increased if it is misclassified by the current base learner, and it is decreased if the sample is classified correctly. We modify the update procedure as follows. If *both* features from every cover-stego pair are classified correctly, their weight is decreased. If at least one of them is misclassified, the weight of *both* of them is increased. This is a logical modification of the AdaBoost for steganalysis that guarantees that the focus of every subsequent base learner will be on more and more difficult training samples, while at the same time preserving the intrinsic cover-stego pairing.

4.4 Experimental comparison

We implemented the system described in the previous section and subjected it to a steganalysis test, comparing its performance to the standard OOB-ensemble described in Section 2.1. Experiments were conducted in a similar manner as in Section 3.2. We used the same image database and steganalyzed the same three algorithms – MB1, YASS, and nsF5 using the feature sets CC-PEV, CDF and $\mathcal{CF}^*$, respectively. We trained the ensemble classifier for every payload (or YASS setting) separately. The comparison is shown in Table 2 in terms of the median error over 10 different splits of the CAMERA database into a training and testing set. We conclude that boosting the ensemble through AdaBoost does not bring any performance gain. It even seems to deliver slightly worse results than the original ensemble implementation, which is more apparent for larger payloads when the classes are more distinguishable.

Steganalysis of MB1 using CC-PEV features

payload	MED		MAD	
(bpac)	OOB	CV+Ada	OOB	CV+Ada
0.01	0.3849	0.3871	0.00260	0.00140
0.02	0.2810	0.2833	0.00340	0.00220
0.03	0.1966	0.1977	0.00175	0.00235
0.04	0.1271	0.1301	0.00220	0.00200
0.05	0.0815	0.0870	0.00105	0.00215

Steganalysis of YASS using CDF features

payload	MED		MAD	
(bpac)$^+$	OOB	CV+Ada	OOB	CV+Ada
0.187 (3)	0.0230	0.0250	0.00080	0.00170
0.138 (8)	0.0753	0.0762	0.00125	0.00245
0.159 (10)	0.0514	0.0515	0.00240	0.00215
0.114 (11)	0.0718	0.0756	0.00150	0.00170
0.077 (12)	0.1281	0.1359	0.00080	0.00160

$^+$The number in parentheses denotes YASS setting

Steganalysis of nsF5 using $\mathcal{CF}^*$ features

payload	MED		MAD	
(bpac)	OOB	CV+Ada	OOB	CV+Ada
0.05	0.3393	0.3402	0.00155	0.00215
0.10	0.1727	0.1815	0.00200	0.00330
0.15	0.0726	0.0853	0.00140	0.00220
0.20	0.0264	0.0377	0.00085	0.00105

Table 2: Steganalysis of MB1, YASS and nsF5 using three different feature sets. Two different implementations of the ensemble classifier are compared – the OOB-ensemble and the CV-ensemble boosted with AdaBoost (column CV+Ada). We report median (MED) and median absolute deviation (MAD) over 10 different splits of the CAMERA database into a training and a testing set.

The reason for the suboptimality of the boosted ensemble may consist in an inappropriate combination of random subspaces and weighted voting. The original OOB-ensemble is a random forest, and each of its base

learners has a random parameter that is drawn independently and from the same distribution for all base learners (random subspace generation). Therefore, it make sense to treat each of the base learners *equally* in the final voting. However, once we incorporate AdaBoost, the first few base learners produce the most important votes for the final decision because they are trained on the training samples with similar weights. On the other hand, later base learners are trained on an increasingly more difficult training set, producing classifiers with lower accuracy and thus lower weight of their vote. The cover/stego class distinguishing ability of random subspaces formed later in the training process has therefore lower influence on the final decision than random subspaces formed at the beginning of the training, in spite of their equal forming strategy.

5 Conclusion

In this technical report, we proposed two alternative designs to the ensemble classifier proposed in [13]. First, we replaced the out-of-bag error estimation with cross-validation and showed that both approaches yield similar results. The second alternative was an incorporation of AdaBoost into the cross-validation based ensemble. This involved two major modifications of the original design: 1) continuous adjustment of the training sample weights so that every subsequent base learner focuses more on those training samples that were previously misclassified, 2) replacement of the final majority voting with a weighted sum of individual votes. These votes correspond to the standalone performance of individual base learners.

According to the steganalysis experiments performed on three different JPEG-domain steganographic algorithms and three different feature sets, the boosted version of the ensemble classifier performed slightly worse than the original implementation. We note that there are other ways of incorporating the idea of AdaBoost into the ensemble implementation. One can form the ensemble classifier from boosted FLDs, or vice versa, take the trained ensemble as a base learner for AdaBoost. We did not implement any of these two variants as they would markedly increase the complexity of the overall steganalysis framework.

6 Acknowledgments

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

References

[1] H. Ahn, H. Moon, M. Fazzari, N. Lim, J. Chen, and R. Kodell. Classification by ensembles from random partitions of high-dimensional data. *Comput. Stat. Data Anal.*, 51:6166–6179, August 2007.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996.

[3] L. Breiman. Random forests. *Machine Learning*, 45:5–32, October 2001.

[4] R. Bryll, R. Gutierrez-Osuna, and F. Quek. Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302+, 2003.

[5] R. O. Duda, P. E. Hart, and D. H. Stork. *Pattern Classification*. Wiley Interscience, New York, 2nd edition, 2000.

[6] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

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

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

[9] T. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, 1998.

[10] J. Kodovský. On dangers of cross-validation in steganalysis. Technical report„ Binghamton University, August 2011.

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

[12] J. Kodovský and J. Fridrich. Steganalysis in high dimensions: Fusing classifiers built on random subspaces. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XIII*, volume 7880, pages OL 1–13, San Francisco, CA, January 23–26, 2011.

[13] J. Kodovský, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*. under review.

[14] J. Kodovský, T. Pevný, and J. Fridrich. Modern steganalysis can detect YASS. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XII*, volume 7541, pages 02–01–02–11, San Jose, CA, January 17–21, 2010.

[15] I. Laptev. Improving object detection with boosted histograms. *Image Vision Comput.*, 27:535–544, April 2009.

[16] P. Sallee. Model-based steganography. In T. Kalker, I. J. Cox, and Y. M. Ro, editors, *Digital Watermarking, 2nd International Workshop*, volume 2939 of Lecture Notes in Computer Science, pages 154–167, Seoul, Korea, October 20–22, 2003. Springer-Verlag, New York.

[17] A. Sarkar, K. Solanki, and B. S. Manjunath. Further study on YASS: Steganography based on randomized embedding to resist blind steganalysis. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 16–31, San Jose, CA, January 27–31, 2008.

[18] R. E. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

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

[20] G. Serpen and S. Pathical. Classification in high-dimensional feature spaces: Random subsample ensemble. In *Machine Learning and Applications, 2009. ICMLA '09. International Conference on*, pages 740 –745, dec. 2009.

[21] Z. Yu and H.-S. Wong. Image classification based on the bagging-adaboost ensemble. In *ICME*, pages 1481–1484, 2008.