Efficient Parallel Feature Selection for Steganography Problems

Alberto Guillén[1], Antti Sorjamaa[3], Yoan Miche[3], Amaury Lendasse[3], and Ignacio Rojas[2]

[1] Department of Informatics - University of Jaen, Spain
[2] Department of Computer Architecture and Technology University of Granada, Spain
[3] Department of Information and Computer Science Helsinki University of Technology, Finland

Abstract. The steganography problem consists of the identification of images hiding a secret message, which cannot be seen by visual inspection. This problem is nowadays becoming more and more important since the World Wide Web contains a large amount of images, which may be carrying a secret message. Therefore, the task is to design a classifier, which is able to separate the genuine images from the non-genuine ones. However, the main obstacle is that there is a large number of variables extracted from each image and the high dimensionality makes the feature selection mandatory in order to design an accurate classifier. This paper presents a new efficient parallel feature selection algorithm based on the Forward-Backward Selection algorithm. The results will show how the parallel implementation allows to obtain better subsets of features that allow the classifiers to be more accurate.

1 Introduction

Steganography has been used and known for a very long time and it aims at hiding some content (usually called *message*) into an apparently innocuous document – mostly digital files nowadays.

Image steganography is currently one of the most investigated field of steganography, since there are many images available online and hence, potentially embedding a secret message.

Steganalysis is the "opposite" process: the main goal is to detect, with the highest possible accuracy, the presence of a secretly embedded content in another document. This can be seen as a typical classification problem, since an optimal separation between images embedding a content (stego images) and genuine ones (cover images) is to be searched.

There exists different ways to perform steganalysis, but the most classical one remains to extract features from each considered image (considered suspicious). Fridrich *et al.* feature set for image steganalysis is widely used [1] since it enables achieving a high detection rate of stego images. Even though Fridrich's set of features is very large (274 features), there are other steganalysis feature sets with even greater number of features.

This paper presents an application of a new parallel feature selection scheme for the steganalysis problem. The methodology is a parallelized version of the Forward-Backward Selection method, which has been recently successfully applied in Time Series prediction problems [2].

The rest of the paper is organized as follows: Section 2 presents the original Forward-Backward algorithm. Then, Section 3 introduces the new improvements incorporated to the algorithm. Afterwards, Section 4 describes the dataset used and the experiments performed. Finally, in Section 5, conclusions are drawn.

2 Forward-Backward Selection

Forward-Backward (FB) Selection is an algorithm that results from the joining of two methodologies: Forward and Backward selections [2]. Both the Forward Selection and the Backward Elimination (or Pruning) methods suffer from an incomplete search. FB offers the flexibility to reconsider input variables previously discarded and *vice versa*, to discard input variables previously selected. It can start from any initial input set, including empty, full or randomly initialized input set.

Let us suppose a set of inputs $\mathbf{X}^i$, $i = 1, 2, \cdots, d$ and an output $\mathbf{Y}$, the procedure of the Forward-Backward Selection is summarized in Fig. 1. In the procedure and in this paper, the k-Nearest Neighbors (kNN) criteria is used as an example criteria for evaluating the input set, but the criteria can be almost any criteria or a suitable approximator or classification algorithm.

1. (Initialization)
   Let S be the selected input set, which can contain any input variables, and F be the unselected input set containing the inputs, which are not in set S. Compute k-NN(S, Y).
2. (Forward-backward selection)
   Find:
   $$\mathbf{X}^s = \arg\max_{\mathbf{X}^i, \mathbf{X}^j} \{k\text{-NN}(\{S, \mathbf{X}^j\}, \mathbf{Y})\} \cup \{k\text{-NN}(S \backslash \mathbf{X}^i, \mathbf{Y})\}, \mathbf{X}^i \in S, \mathbf{X}^j \in F.$$
   If the previously computed k-NN is larger than the new k-NN, stop; otherwise, update set S and save the new k-NN, repeat step 2 until S is equal to any former selected S.
3. (Result)
   The selection result is in set S.

Fig. 1. Forward-Backward Selection Strategy

It is noted that the selection result depends on the initialization of the input set. In this paper, several options are considered and the options are discussed more deeply in Section 3.3.

The number of input sets to be evaluated varies and is dependent on the initialization of the input set, the stopping criteria and the nature of the problem. Still, it is not guaranteed that in all cases this selection method finds the global optimal input set.

2.1 *k*-Nearest Neighbors

The k-Nearest Neighbors (k-NN) approximation method is a very simple but powerful method. It has been used in many different applications, particularly for classification tasks [3]. The key idea behind the k-NN is that samples with similar inputs have similar output values. Nearest neighbors are selected, according to Euclidean distance, and their corresponding output values are used to obtain the approximation of the desired output.

In this paper, since the problem is a binary classification task, the estimated output value of a sample is the class of the majority of the nearest neighbors k. The k has to be determined beforehand.

3 Parallel Forward-Backward Selection

The following sections describe the modifications that have been done to the original Forward-Backward in order to improve the performance in terms of computational cost and classification accuracy. The parallel implementations were done in MATLAB using the MPI standard and the interface used in [4].

3.1 Improvement 1: Efficient Computation of the Distance Matrix

As described in the previous Section, the k-NN has to be computed each time a subset of variables has to be evaluated. This requires the computation of the distances between an input vector and the others to determine the k closest ones. This results in a heavy computational load and in order to save some time, a distance matrix will be computed in advance. These calculations become crucial when dealing with the feature selection using scaling, that is, weighting the importance of a variable instead of merely selecting it. The distance matrix will have an element for each couple of values, storing the distances separately between each input and each sample.

Nonetheless, because the computation of this matrix is quite time consuming, an another improvement has been done: parallel calculation of the distance matrix. The matrix is divided between a set of processes and each one of them will perform locally a part of the calculations. Once they are done, they will communicate with the other processes in a collective way so that all processes have the complete matrix. Each process computes its starting and ending columns as $(rank) * ceil(d/size_w) + 1$ and $(rank + 1) * ceil(d/size_w)$, except for the last process that ends in d, where d is the number of all possible input variables, $size_w$ is the number of all processes and $rank$ is the process identifier from 0 to $size_w - 1$.

3.2 Improvement 2: Parallel FB

The search for the best solution is also distributed to several computers so that more solutions can be evaluated in less time. The parallel implementation is quite straightforward and consists of the division of the newly generated subset of variables in the first iteration of the FB. This division is shown in Fig. 2. Once each process has a part of the subset, they proceed as the original FB. The algorithm stops when all the processes have converged to a solution. Then, the best solution is found among the final solutions of the individual processes.

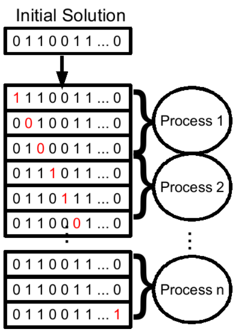

Fig. 2. Parallel scheme for the Forward-Backward Selection

The advantage of this approach is that each process has a different starting solution. Thus, it is possible to find different local minima and have a more thorough exploration of the solution space than in the original FB.

3.3 Improvement 3: FB Initialization

One of the main issues, when using the FB, is how to choose the initial solution from where the search will begin. Due to the locality of the FB, the risk of falling into a local minimum is quite high. Therefore, the starting point should be close enough to a good local minimum in order to get proper results. In this paper, two approaches are used: Mutual Information and Slice Division. A comparison of their performances will be shown in the Experiments section.

Mutual Information Based Initialization. The Mutual Information (MI) is used as a heuristic to find an adequate starting point for the FB selection. Let $\mathbf{X}^l = \{\boldsymbol{x}_m^l\}$ with $l \in 1, ..., d$ (i.e. $\mathbf{X}^l$ is the l-th input variable) and $\mathbf{Y} = \{y_m\}$ with $\{m = 1...M\}$. The Mutual Information between $\mathbf{X}^l$ and $\mathbf{Y}$ can be defined as the amount of information that $\mathbf{X}^l$ provides about $\mathbf{Y}$, and can be expressed as:

$$I(\mathbf{X}^l, \mathbf{Y}) = \sum_{y \in \mathbf{Y}} \sum_{x \in \mathbf{X}} \mu_{\mathbf{X}^l,\mathbf{Y}}(x, y) \log \frac{\mu_{\mathbf{X}^l,\mathbf{Y}}(x, y)}{\mu_{\mathbf{X}^l}(x)\mu_{\mathbf{Y}}(y)}. \tag{1}$$

$\mu_{\mathbf{X}^l,\mathbf{Y}}$ is the joint probability distribution function of $\mathbf{X}$ and $\mathbf{Y}$, and $\mu^l_{\mathbf{X}}(x)$ and $\mu_{\mathbf{Y}}(y)$ are the marginal probability distribution functions of $\mathbf{X}$ and $\mathbf{Y}$ respectively.

Therefore, in order to obtain the MI between $\mathbf{X}^l$ and $\mathbf{Y}$, only the estimate of the joint probability density function is needed. This value can be computed using several techniques based on histograms, kernels or the k-NN [5].

For each input variable, the MI between that variable and the output is computed and, once finished, it is possible to rank all the input variables according the values of MI. Then, the initial solution for the FB is defined as a number of *first* variables in the ranking. The problem now is to determine the actual number of variables, since the value obtained by the MI is not enough to perform this selection. Unfortunately, the only chance is to set this value manually, as it will be shown in the experiments section.

Slice Division. Being aware of the two significant drawbacks of the used MI heuristic (the determination of the k and the final number of variables to be chosen), another heuristic is considered, which requires the definition of only one parameter. This value can again be set manually as in the MI, or as a function of the available resources making the heuristic more flexible when executed in different computer architectures or systems.

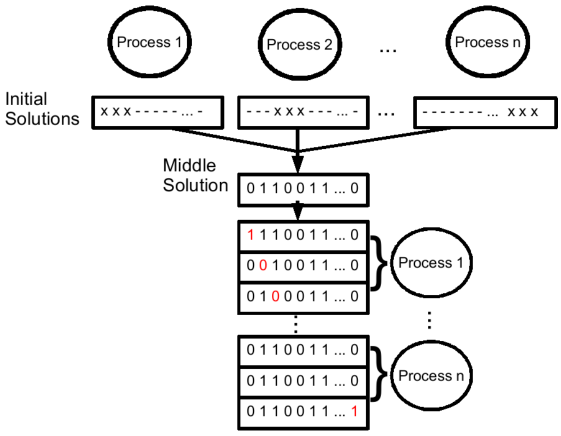

Fig. 3. Scheme of the algorithm using the Slice Division heuristic to initialize the starting point for the FB

The heuristic works as follows: it performs a preinitialization by dividing the original input vector into subvectors of a smaller dimension. Then, the local search is applied to each subvector although, the evaluation of a subvector implies the evaluation of the complete set of variables. Therefore, four starting alternatives arise: 1) all zeros, 2) subvectors ones, the rest zeros, 3) all ones, 4) subvectors zeros, the rest ones. Although the first one seems the most reasonable, since it is easier to find a local minimum with a reduced set of variables, the different alternatives were studied in the experiments. The issue remains, how large the subvectors should be? This value can be set manually or as a function of the number of available processors.

In the scheme shown in Fig. 3, the initial solutions are divided into slices of size 3 (depicted as x) and the remaining values are not changed (depicted as -) during the first FB. This first FB is done sequentially and separately in each process. Once all the processes have converged to a local optima, the processes perform a collective communication and share the results found by the other processes. Then, the initial starting point for the parallel FB (named *middlesolution* in Fig. 3) is computed by concatenating all local solutions.

4 Experiments

Two experiments are shown in order to illustrate the benefits of the newly added features: analysis of the computational cost and number of evaluations, and the accuracy of the classifications.

The dataset has been taken from a real-life steganalysis problem, where a set of features are extracted from an image. The dataset contains 10000 images from the BOWS2 challenge [6] database, hosted by Andreas Westfeld and it is publicly available in [7]. All images are 512×512 greyscale (color versions are also available). The steganographic algorithms have been used only on these specific dimensions of the images from the database, but any size would work just as well.

For half of the whole base of images, chosen randomly, steganography is applied. For this application, one well-known steganographic algorithm, called OutGuess [8], has been used. The OutGuess algorithm is known to be rather weak in steganalysis. An embedding rate of 15% of the maximum capacity of the OutGuess has been used.

Once half of the images are stego, the feature extraction is performed [1], leading to 274 features for each image. This feature extraction process is also carried out on the non-stego images. The final dataset is obtained by joining the two halves, giving a total of 10000 samples in 274-dimensional space.

The dataset is further divided into training and test sets, 1000 samples and roughly 9000 samples, respectively. Both sets are normalized using the mean and standard deviation of the training set.

4.1 Experiment 1: Parallel Performance Evaluation

The original FB algorithm and the improved versions were executed in identical machines in order to compare the computational times. The results are shown in Table 1.

In Table 1, the Variable Selection time measured depends greatly on when the search converges to a local optimum. Regarding the time consumed for the calculation of the distances, the proposed parallel implementation turns out to be efficient and scalable. The scalability remains good also in terms of the number of solutions evaluated, which is growing linearly with the number of processes.

Table 1. Times (in minutes) spent on the distance matrix calculation and in the variable selection and the number of solutions of the sequential and the parallel implementations using the MI and the FBS methodologies

	Distance	Variable Selection	# Evaluated Solutions
Sequential	15	238	1650
Parallel 2p	11	297	3848
Parallel 4p	5	284	6596
Parallel 6p	4	273	10169
Parallel 8p	3	348	13742

4.2 Experiment 2: Classification Accuracy

The second experiment concentrates to the results obtained by the classifier after the feature selection. Table 2 shows the errors obtained for both training and test subsets for the algorithm using different heuristics to set the initial starting point.

Table 2. Results of the variable selection. For each variable selection scheme the following results are shown: a number of initial and finally selected variables, respective k-NN classification error and training and test errors of the OP-KNN classification methodology.

	Variables		k-NN	OP-KNN	
	Initial	Selected	Train	Train	Test
All Variables	274	274	0.190	0.172	0.210
FBS	30	33	0.118	0.120	0.146
	50	52	0.122	0.110	0.141
	70	73	0.131	0.134	0.145
	100	102	0.134	0.127	0.155
pFBS	30	35	0.113	0.115	0.141
	50	52	0.115	0.125	0.135
	70	73	0.126	0.130	0.145
	100	103	0.118	0.120	0.141
pFBSv2	All Ones	245	0.157	0.142	0.192
	All Zeros	56	0.114	0.110	0.138

The classification errors using the k-NN algorithm and the OP-KNN [9] show, how important it is to perform feature selection beforehand: all the models that were trained after the dimensionality reduction outperform the ones trained with all the variables. The results also show, how the two initialization heuristics have a good behavior, although it is remarkable that the last one, based on a local slice optimization, has the best performance. Furthermore, this initialization, starting from the empty subset of variables, was able to select 50 variables, avoiding many local minima.

5 Conclusions

The problem of identifying if an image has a hidden message is really interesting, because of its possible applications, specially in the World Wide Web. The work presented in this paper proposes a methodology for the dimensionality reduction in order to make the data samples easier to learn by the artificial models.

Regarding the feature selection problem, the Forward-Backward algorithm was used as a first choice, but it provided poor results requiring a significant computational cost. Therefore, three major improvements were presented in this paper: parallel implementation of the computation of the distance matrix, parallelization of the search methodology and two new heuristics to determine the starting point of the search. This last element is crucial due to the large amount of local minima in the solution space.

The results have shown how the new elements allow the algorithm to perform an adequate feature selection, allowing the models to provide accurate classifications.

Acknowledgement. This work has been partially supported by the projects TIN2007-60587, P07-TIC-02768 and P07-TIC-02906, TIC-3928. Part of the work of Antti Sorjamaa has been supported by a grant from Nokia Foundation, Finland.

References

1. Pevny, T., Fridrich, J.: Merging markov and dct features for multi-class jpeg steganalysis. In: IS&T/SPIE 19th Annual Symposium Electronic Imaging Science and Technology, January 29-February 1, 2007. LNCS, vol. 6505 (2007)
2. Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., Lendasse, A.: Methodology for long-term prediction of time series. Neurocomputing 70(16-18), 2861–2869 (2007)
3. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
4. Guillén, A., Pomares, H., González, J., Rojas, I., Herrera, L.J., Prieto, A.: Parallel multi-objective memetic rBFNNs design and feature selection for function approximation problems. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 341–350. Springer, Heidelberg (2007)
5. Kraskov, A., Stögbauer, H., Grassberger, P.: Estimating mutual information. Phys. Rev. 69, 066138 (2004)
6. The 2nd bows contest (break our watermarking system), watermarking virtual laboratory (wavila) of the european network of excellence ecrypt (2007)
7. Westfeld, A.: Reproducible signal processing (bows2 challenge image database)
8. Provos, N.: Defending against statistical steganalysis. In: 10th USENIX Security Symposium, April 13-17, 2001, pp. 323–335 (2001)
9. Miche, Y., Lendasse, A.: A faster model selection criterion for op-elm and op-knn: Hannan-quinn criterion. In: ESANN 2009: European Symposium on Artificial Neural Networks, Bruges, Belgium (April 2009)