Feature selection: a multi-objective stochastic optimization approach

Stefano Rovetta
DIBRIS,
University of Genova,
Genova, Italy
stefano.rovetta@unige.it

Grażyna Suchacka
Institute of Informatics,
University of Opole,
Opole, Poland
gsuchacka@uni.opole.pl

Alberto Cabri
DIBRIS,
University of Genova,
Genova, Italy
alberto.cabri@dibris.unige.it

Francesco Masulli
DIBRIS,
University of Genova,
Genova, Italy
francesco.masulli@unige.it

***Abstract*—The feature subset task can be cast as a multi-objective discrete optimization problem. In this work, we study the search algorithm component of a feature subset selection method. We propose an algorithm based on the threshold accepting method, extended to the multi-objective framework by an appropriate definition of the acceptance rule. The method is used in the task of identifying relevant subsets of features in a Web bot recognition problem, where automated software agents on the Web are identified by analyzing the stream of HTTP requests to a Web server.**

***Index Terms*—feature selection, subset selection, stochastic optimization, threshold accepting, multiobjective optimization**

I. Introduction

Although we think of the Internet as a means for people to communicate, actually a relevant fraction of traffic is generated by autonomous programs, termed *web robots*, or *bots* for short. Estimates vary, but about half of the unique visitors to websites ($42.2\% - 51.8\%$) may in fact be robots [1], [2].

Web bots are used to perform a variety of tedious tasks. Historically the first application is web crawling, but many types of bots exist: Social bots post automatically generated messages on services like Twitter or Reddit; conversational bots [3] reply to questions by human users with varying degrees of complexity in natural language; maintenance bots, as in Wikipedia, perform routine operations such as adding templates and replacing text.

Along with these "ethical" bots, however, there are other categories. For instance, botnets are ensembles of computers running bots that perform malicious activities from a number of different IP addresses, making it very hard to block them [4]. Apart from direct attacks, however, malicious bots are also increasingly used to gain undue advantage in online business. Ticketing bots can be use to buy large quantity of popular event tickets for illegal resale, and click bots are frequently used to simulate a higher number of clicks on web advertising [5].

The work presented here is part of a research activity aimed at filtering out bots that are trying to access a web site, in particular an e-shop. Online databases are available to identify known bots [6], for instance when the originating domain or user-agent strings are known [7]. However, in general no objective criterion exists to tell a bot from a legitimate human visitor. Previous work [8] has demonstrated that conventional machine-learning approaches can be very effective in this task when performed off-line on web logs, and sequential methods [9] achieve good results in online recognition. This paper describes one approach for *reverse-engineering* these and other succesful methods, by identifying subsets of features that are effective in the identification of bots. The solution space is discrete, and the evaluation of a classifier requires trading off different performance measures which conflict with each other. These aspects of the problem naturally suggest a feature subset selection technique based on stochastic multi-objective optimization.

II. The bot recognition problem

A. Web server data

Web servers process requests sent by Web clients using the HTTP application-level protocol [10]. Data from HTTP requests are logged in a standardized text file complying with a predefined format, set up in the server configuration. This research aims at recognising bots by directly looking at HTTP requests, so that the result is independent on the website implementation and largely also from the server software.

The interaction of Web clients with the website may be represented as sessions. A *Web session* is the sequence of HTTP requests coming from a client during a single visit. This concept is instantiated by the web server software, but is not supported by HTTP which is a stateless protocol. Therefore, as is common practice, session are heuristically defined: HTTP requests are grouped into the same session if they have the same IP address, the same user agent string and the time between consecutive requests does not exceed a threshold, here set at the common value of 30 minutes [11], [12].

B. Definition of the problem

The input for our approach is one or more log files, containing traffic data for an e-commerce website over some period of time. We assume to have access to at least the information contained in the *NCSA Combined* log format. The following fields are used:

- `host`, corresponding to the IP address of the Web client;
- `date:time`, the time stamp of the request;
- `request` – information on the *HTTP method* (e.g., GET for downloading a resource, HEAD for downloading just

978-1-7281-6654-4/20/$31.00©2020 European Union

the resource header) and the URI (Uniform Resource Identifier) specifying the requested resource;
- `statuscode`, the numeric code informing about the result of request processing at the server (e.g., codes 2xx for the success, 4xx for client errors);
- `bytes` – volume of data transferred to the client;
- `referrer`, the URL (Uniform Resource Locator) from which the client was directed to the current page;
- `user_agent`, the string specifying the client Web browser and platform.

Based on the fields available, various session features may be further determined, e.g., total number of requests, session duration, mean time per page and many others.

The problem of off-line session-based bot recognition can be formalized as follows: given the features characterising a Web session, label the session as performed either by a bot or by a human. The information about all requests in a session is entirely available at the time of decision making.

In this research, we are looking for a small set of features that provide a good recognition performance. This is a classic *feature subset selection* problem.

III. Feature selection

The feature selection task [13] can have different goals and can search for different types of solution. Common goals include dimensionality reduction and analysis of cause-effect relations. The output can either be a feature subset, or a ranking of all features by relevance. In general, however, the following elements have to be designed:

- a quality criterion or metric;
- a search strategy;
- an evaluation strategy.

In particular, here we focus on the feature subset selection problem.

A. Quality criteria

A quality criterion is the measurement of the relevance of feature subsets. It is possible to distinguish between univariate criteria, that assign a relevance value to individual features, and multivariate criteria, that take into account feature subsets. Univariate criteria are usually computationally advantageous. However, due to the possibly complex interactions existing between features, they are suboptimal.

B. Search strategies

Search strategies can be either bottom-up, starting from a minimal set and iteratively adding new features (*forward selection*), or top-down, starting from the complete set and iteratively eliminating the less relevant features (*backward elimination*). Mixed stategies also exist, adding or removing possibly different numbers of features at each step [14]

C. Evaluation strategies

Evaluation strategies can be categorised as *wrapper*, *filter* and *embedded*. Wrapper strategies "wrap" around a given machine learning method, for instance a classifier, whose performance directly acts as the quality criterion. The search loop includes multiple re-training of the method, resulting in a computationally expensive process. Filter strategies "filter" features as a pre-processing step, so that they need a separate quality criterion and optimize the desired performance only indirectly. Embedded strategies are "embedded" within some machine learning methods, for instance classification trees, and are therefore only available for these methods.

D. Feature selection methods

A feature subset selection method, therefore, is an optimization algorithm that implements the desired search strategy for optimising the quality criterion, which constitutes its objective function.

We choose to implement a wrapper approach, but using a "proxy" classifier that relieves the computational burden of the optimization. The search strategy is a combined forward selection and backward elimination method, which provides global search ability.

IV. A multi-objective optimization approach

In the following, to fix the ideas, we will imply a minimization problem where the optimal solution has a low value of the objective(s). As usual, for objectives that need to be maximised it is sufficient to change their sign.

A. Multi-objective optimization

Multi-objective optimization (MOO) deals with problems involving more than one objective function to be optimized simultaneously. In general this applies to *conflicting* objectives, so there is no single solution that optimizes all of them; otherwise the problem would collapse into a mono-objective one.

As a consequence, a MOO algorithm outputs a set of *Pareto-optimal* solutions. A solution is Pareto optimal if none of its objective function values can be improved without worsening any of the other objective values. More formally, a configuration x is said to dominate another configuration y if (a) all objectives are no worse in x than in y, and (b) at least one objective is strictly better in x than in y, or, in formulas:

$$\begin{cases} \forall i \in \{1, \ldots, c\} & : \quad f_i(x) \leq f_i(y) \\ \exists j \in \{1, \ldots, c\} & : \quad f_j(x) < f_j(y) \end{cases} \tag{1}$$

A solution is said to be Pareto-optimal if no other configuration dominates it.

The set of all Pareto-optimal solutions is called the Pareto set, front, or frontier; it is typically not feasible to return all the nondominated solutions, so the output is usually just a finite sampling of the Pareto set. It is important to note that the Pareto front contains solutions that are all of equivalent quality according to the objectives, so external criteria should be employed to select one final solution.

B. Objectives

In the feature selection problem as applied to two-class classification, the goal is inherently multi-objective. In particular, the problem involves finding a good balance between classification performance and feature set size. In turn, classification performance is a multi-faceted problem because both false positives and false negatives should be minimised.

The natural objectives for feature subset selections will therefore be, as a minimum:

1) Feature set size
2) Precision
3) Recall

where precision and recall could in principle be replaced by other indexes carrying the same information, for instance sensitivity and specificity.

C. Feature subset selection as a discrete optimization problem

The feature subset selection problem consists in deciding which features, from an available pool, are to be kept and which ones are to be discarded. A natural representation of this problem is as a binary string with as many bits as features. For one instance of the problem, the available feature pool is of size 50. This is by no means a large input size, and yet the possible subsets are $2^{50}-1$ (we afford the luxury of discarding the solution with zero features).

In view of this complexity, stochastic optimization seems a viable solution, provided that the problem has some locality structure that can be exploited. However, it is reasonable to think that solutions including a set of n features will not have a performance that differs much from that of the same set with just one feature added or removed. Indeed, previous studies using the concept of Markov blanket [13] reassure us about the existence of such a structure that can guide a stochastic local search algorithm.

V. The Multi-Objective Threshold Accepting Method

A. Multi-Objective Simulated Annealing

Simulated Annealing (SA) [15] is a well-known metaheuristic optimization method that explores stochastically a discrete optimization space. A candidate solution $\overline{x}$ is accepted over the current solution x with probability

$$P = \min\left\{1,\ e^{(\,f(x)-f(\overline{x})\,)/T}\right\} \tag{2}$$

When the candidate solution improves the objective, the exponential is larger than 1 and the solution is always accepted. When it does not improve, it can still be accepted, but with a probability depending on its quality: the higher the distance from the current objective value, the lower the acceptance probability. This probability also depends on a "computational temperature" parameter T which is gradually decreased ("annealed") during the iterations of the algorithm.

Multi-objective SA [16] extends this idea to multiobjective optimization. Several multiobjective generalizations have been proposed, because there are many ways to adapt criterion (2) to the multiple criteria case.

Require: Initial candidate solution: $x \leftarrow x_0$
  Threshold schedule: $\{\tau_0, \ldots, \tau_f\}$
  Objective functions: $f_1(x), \ldots, f_c(x)$
  Neighborhood size: s
  Duration of each exploration step: n
  Initial Pareto set $\Pi = \{x\}$
  for $\tau : \{\tau_0, \ldots, \tau_f\}$ **do**
    for $i : 1 \ldots n$ **do**
      Select a random config. $\overline{x}$ among the s nearest neighbours of x
      Evaluate $f_1(\overline{x}), \ldots, f_c(\overline{x})$
      if $\overline{x}$ is quasi-Pareto-optimal in Π **then**
        Add $\overline{x}$ to Π
        Remove from Π all configurations that are strictly dominated within Π
      end if
    end for
  end for
  Remove from Π all configurations that are not Pareto-optimal within Π
  return Pareto set Π

Fig. 1. Pseudocode of the multi-objective threshold accepting procedure.

Since objectives can be incompatible in magnitude, it is necessary to introduce weights $w_1, \ldots, w_c$ to scale them onto the same range.

Two examples involve computing the probability of acceptance according to the following rules:

$$P(\text{acceptance}) = \prod_{j=1}^{c} \min\left\{\ 1,\ e^{\,w_j(\,f(x)-f(\overline{x})/T}\ \right\} \tag{3}$$

which accepts with certainty only solutions that improve the current one; and

$$P(\text{acceptance}) = \min\left\{\ 1,\ \max_{j=1,\ldots,c} e^{\,w_j(\,f(x)-f(\overline{x})/T}\ \right\} \tag{4}$$

which also accepts solutions that are neither dominating nor dominated by the current one (*noncomparable* solutions). For a more extensive survey, we refer to [17].

B. Multi-Objective Threshold Accepting

The Threshold Accepting optimization procedure [18] is a variant of Simulated Annealing that replaces the probabilistic acceptance rule with a simpler threshold of acceptance. A candidate solution $\overline{x}$ is accepted over the current solution x if

$$f(\overline{x}) - f(x) < \tau \tag{5}$$

where τ is the current value of a threshold to be "annealed", similarly to the temperature parameter in SA. Therefore, all candidate solutions that do not worsen the current objective by more that τ are accepted. This simpler rule can be interpreted

as a mean field approximation to the stochastic one, and has indeed been proved to be optimal [19].

To the best of our knowledge, a multi-objective variant of this rule has not been proposed, or at least prominently studied, in the literature. To fill in the gap, we propose the algorithm whose pseudocode is illustrated in Figure 1, which we now describe.

Given a threshold τ, we introduce the concept of *quasi-dominated* configurations. We will say that configuration x quasi-dominates another configuration y if (a) all objectives in x are within a margin τ from the corresponding values in y, and (b) at least one objective is strictly better in x than in y, or:

$$\begin{cases} \forall i \in \{1, \ldots, c\} & : \quad f_i(x) \leq f_i(y) + \tau \\ \exists j \in \{1, \ldots, c\} & : \quad f_j(x) < f_j(y) \end{cases} \tag{6}$$

We will term a configuration that is not quasi-dominated by any other configuration a *quasi-Pareto-optimal* solution.

As previously noted, weights can be used. They are also a practical tool for mixed minimization-maximization problems: objectives that should be maximised will receive a negative weight. The actual condition for non quasi-dominated solutions therefore becomes:

$$\begin{cases} \forall i \in \{1, \ldots, c\} & : \quad f_i(x) \leq f_i(y) + w_i\tau \\ \exists j \in \{1, \ldots, c\} & : \quad f_j(x) < f_j(y) \end{cases} \tag{7}$$

Every solution that is accepted is added to the current Pareto set (let it be Π). Afterwards, all solutions in Π which after the addition are not Pareto-optimal any more are removed.

C. Multi-objective threshold accepting for feature selection

The implementation of the above algorithm in the feature selection case requires instantiating some details that are left unspecified in the description of the general case.

a) Encoding: Since the optimization space consists of sets of features, the encoding of a solution is a bit string with one bit per feature, in a predefined order. Bits are interpreted as set indicator functions, i.e., 1 means "included" and 0 means "excluded".

b) Neighborhood: The natural metric in the space of bit strings is the Hamming distance $h()$. A neighborhood of solution x of radius s is the set of all solutions y such that $h(x, y) \leq s$. Equivalently, y is obtained from x by "flipping" at most s bits.

c) Objective: In a filter approach, the objective evaluation is the most computationally demanding operation. Rather than using a full-featured machine learning method, we compute the performance indexes (precision, recall) using a basic nearest neighbour rule. Not only it provides a large gain in computation time when compared to neural network training or support vector machine learning, for instance, but it has also been found that simple rules similar to this one perform close to the best methods [20].

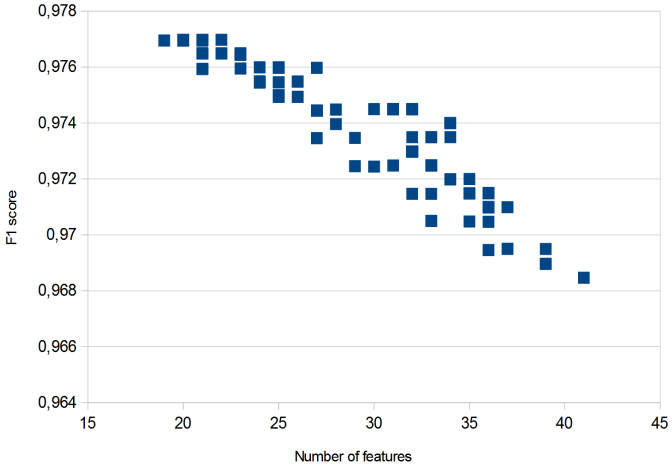

Fig. 2. The graph of the Pareto front: F1 score plotted against number of features.

VI. Experiments

A. Data and software implementation

The data used are a set of 13741 sessions from one specific Web server running an e-shop. The sessions descriptions are obtained by preprocessing web server log files to yield a set of 50 candidate features. These include descriptors of the type of request, type and volume of data served, and temporal aspects.

The algorithm was implemented as a Python 3.8 program running under the Linux operating system on a dual Xeon dedicated multiprocessor machine. The machine is also equipped with a GPU for computation, but this was not exploited in this research.

As usual with global random-search-based methods, a multi-start strategy was set up. Due to the independence of the individual trials, a straightforward multi-processing implementation with MPICH[1] was produced to take advantage of the 48 computational threads (24 cores) available on the machine.

The software was therefore run in 20 trials, each starting from a different random configuration with an expected fraction of 25% of all features (this is a bit string of length 50 and an expected number of 12.5 bits set), with linearly decaying thresholds varying from 0.05 to 0 in 50 steps; at each step, 50 neighbours were explored in a neighborhood of radius 3.

B. Results and discussion

Figure 2 shows the set of solution obtained from the algorithm. The problem is actually three-dimensional (precision, recall, number of features), but for illustration purposes precision and recall are summarized with their harmonic mean, the F1 score, so that the graph is not precisely a line. However, the correlation coefficient between the two quantities is -0.94. This means that the stochastic search procedure has identified a region in the solution space where the problem with objectives (F1 index, number of features) collapses into a locally mono-objective one, where the objectives are not conflicting.

[1] http://www.mpich.org/

TABLE I
FEATURE NAMES FOR THE SOLUTION WITH MAXIMUM F1 SCORE

intervalPag_avg	%otherFileType	%HEAD
intervalPag_stdev	imageToPageRatio	%2xxWithout200
maxSustClickRate6	embToPageRatio	%304
%page	avgReqPerURI	%3xx
%image	robots.txt	%5xx
%binaryDoc	totalVolume	%empReferrerPag
%ascii	volume_avg	

TABLE II
OBJECTIVE VALUES FOR THE SOLUTION DESCRIBED IN TABLE I

recall	0.982
precision	0.972
number of features	20

(However, the two objectives which we have summarized in the F1 index for the sake of visualization, namely precision and recall, are still contradictory.)

It is interesting to note that the correlation is actually negative, which means that smaller feature sets improve the performance of the classifier in terms of the F1 measure. However, since solutions with less than 20 features were seldom contained in the returned Pareto sets, we assume that this correlation is inverted when features are too few, so we can make the hypothesis that the set of the most relevant features is of size 20.

The size of the solutions ranges from 19 to 41 features, with the majority included in the range [20,39]. We can "zoom in" on one specific solution to inspect it. Table I shows the feature set of the solution corresponding to the best value of F1. This solution has the objective values indicated in Table II.

In this context we are not specifically interested in the meaning of the features (this "reverse engineering" task, as we termed it, is the subject of current research). Once a complete Pareto set of solutions has been obtained, the next step will be to group feature sets and analyse which ones are appearing most frequently in the best-performing sets.

VII. CONCLUSION

The feature subset task can be cast as a multi-objective discrete optimization problem. In this work, we have studied the search algorithm component of a feature subset selection method. We have proposed one algorithm based on the threshold accepting method, extended to the multi-objective framework by an appropriate definition of the acceptance rule.

This is a preparatory work that will be developed in the following two directions: (1) hyperparameter tuning of the algorithm itself, including threshold schedule and neighborhood size; (2) use of the algorithm for identifying relevant features that are shared by diverse Websites.

ACKNOWLEDGMENT

This work was partially supported by the National Science Centre (NCN) in Poland under Grant No. DEC-2017/01/X/ST6/01070.

S. Rovetta, A. Cabri and F. Masulli are affiliated with the National Institute of Higher Mathematics "F. Severi", National Group for Scientific Computing (GNCS), Italy.

REFERENCES

[1] GlobalDots, "2018 bad bot report," GlobalDots, Tech. Rep., 2018. [Online]. Available: www.globaldots.com

[2] I. Zeifman, "Bot traffic report 2016," Imperva Incapsula, Tech. Rep., 2017. [Online]. Available: https://www.incapsula.com/blog/bot-traffic-report-2016.html

[3] "Microsoft bot framework," visited on 2017-04-20. [Online]. Available: https://dev.botframework.com/

[4] N. Goodman, "A survey of advances in botnet technologies," *arXiv preprint arXiv:1702.01132*, 2017.

[5] "Invalid clicks," visited on 2017-03-21. [Online]. Available: https://support.google.com/adwords/answer/42995

[6] Udger, "Udger," Czech Republic, 2017. [Online]. Available: https://udger.com(accessdate:September4,2017)

[7] A. Staeding, "User-agents.org," 2017. [Online]. Available: http://www.user-agents.org(accessdate:September4,2017)

[8] S. Rovetta, A. Cabri, F. Masulli, and G. Suchacka, "Bot or not? a case study on bot recognition from web session logs," in *Quantifying and Processing Biomedical and Behavioral Signals*, A. Esposito, M. Faundez-Zanuy, F. Morabito, and E. Pasero, Eds. Springer, 2018, vol. 103, p. 197.

[9] A. Cabri, G. Suchacka, S. Rovetta, and F. Masulli, "Online web bot detection using a sequential classification approach," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2018, pp. 1536–1540.

[10] T. Berners-Lee, R. Fielding, and H. Frystyk, *Hypertext Transfer Protocol – HTTP/1.0*, IETF RFC 1945, May 1996.

[11] C. Bomhardt, W. Gaul, and L. Schmidt-Thieme, "Web robot detection - preprocessing Web logfiles for robot detection," in *New Developments in Classification and Data Analysis*. Berlin, Heidelberg: Springer, 2005, pp. 113–124.

[12] D. Doran and S. S. Gokhale, "Web robot detection techniques: overview and limitations," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 183–210, 2011.

[13] S. H. Huang, "Supervised feature selection: A tutorial." *Artif. Intell. Research*, vol. 4, no. 2, pp. 22–37, 2015.

[14] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119 – 1125, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167865594901279

[15] C. D. Kirkpatrick, S. and Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[16] P. Serafini, "Simulated annealing for multi objective optimization problems," in *Multiple criteria decision making*. Springer, 1994, pp. 283–292.

[17] K. Amine, "Multiobjective simulated annealing: Principles and algorithm variants," *Advances in Operations Research*, vol. 2019, 2019.

[18] G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, no. 1, pp. 161 – 175, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/002199919090201B

[19] A. Franz, K. H. Hoffmann, and P. Salamon, "Best possible strategy for finding ground states," *Phys. Rev. Lett.*, vol. 86, pp. 5219–5222, Jun 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.86.5219

[20] S. Rovetta, G. Suchachka, and F. Masulli, "Bot recognition in a web store: an approach based on unsupervised learning," *Journal of Network and Computer Applications*, 2020, in press. [Online]. Available: https://doi.org/10.1016/j.jnca.2020.102577