Negative feature selection algorithm for anomaly detection in real time

Krzysztof Hryniów[1], Andrzej Dzieliński[2]
[1] Warsaw University of Technology
Institute of Control and Industrial Electronics
Koszykowa St. 75, 00-662, Warsaw, Poland
hryniowk@isep.pw.edu.pl
[2] Warsaw University of Technology
Institute of Control and Industrial Electronics
Koszykowa St. 75, 00-662, Warsaw, Poland
Andrzej.Dzielinski@ee.pw.edu.pl

Abstract. Anomaly detection methods are of common use in many fields, including databases and large computer systems. This article presents new algorithm based on negative feature selection, which can be used to find anomalies in real time. Proposed algorithm, called Negative Feature Selection algorithm (NegFS) can be also used as first step for preprocessing data analyzed by neural networks, rule-based systems or other anomaly detection tools, to speed up the process for large and very large datasets of different types.

Keywords. Anomaly detection, feature selection, frequent pattern mining, neural networks, rule-based systems

1 Introduction

Anomaly detection is a method of detecting patterns in given datasets that are not part of normal behavior of data in given set. It is a sort of whitelisting - allowing only data that can be identified as normal to be run. Anomaly for purposes of this article is defined by Hawkins definition of outliers in [1]:

An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.

Detected patterns are classified as either normal or anomalous, with classification based mainly on rules or patterns, with use of some form of artificial intelligence - be it neural network or rule-based system. Anomalous patterns detected in real-life can be found in such critical situations as frauds, intrusions or data corruption, so real-time detection is often needed.

Anomaly detection techniques now constitute an essential part of security in many fields - including banking, database security, network traffic, national defense and telecommunications [2] [9].

There are three main categories of anomaly detection - supervised, semi-supervised and unsupervised; analogous to neural network learning methods [2] [9]. Unsupervised anomaly detection in many cases is most useful one, as it is able to detect anomalous patterns in unlabeled datasets (with the assumption that majority of data in set is not anomalous). At the moment, there are many anomaly detection algorithms, but most of them is either supervised or semi-supervised, and cannot be used with unlabeled datasets [2] [8] [9]. Also, most of anomaly detection techniques presented in the last years can be used strictly for outlier detection in graphically represented data (like satellite images) or anomaly detection of network traffic [9]. Others, tend to be more generic, like algorithms based on Moving Averages (ARMA, ARIMA, SARIMA), but work best for sequence data which was previously labeled [3] [8] [10]. For record data and sequence data, algorithms based on statistical methods or filtering (especially using Kalman filter) are often used - there are very precise, but most of them are too slow for real-time processing of large, unlabeled data [4] [7] [8]. Methods based on SVM (Support Vector Machines), like SCFAR-AD, are also used for anomaly detection but as shown in [5] on benchmark datasets, although their detection rate is impressive, their working time is quite slow and are unfit for real-time anomaly detection for very large datasets.

This article introduces unsupervised anomaly detection technique based on negative feature selection called Negative Feature Selection algorithm (NegFS). It is designed to work best for large and very large sets of unlabeled record and sequential data (like telecommunication data or records in databases) for real-time anomaly detection, but is generic enough that can be used for other types of data as preprocessing tool. What is important, algorithm is designed in such a way, that it can be also used successfully for preprocessing of unlabeled datasets for use by other anomaly detection techniques, making them work faster.

2 NegFS algorithm overview

NegFS algorithm operates on given dataset, record after record. Each record is treated as a string of symbols, which are assigned to one of 3 predefined groups - sequences, links and separators. Separators divide record into segments and sequences in the same segment are separated by links. In the data matrices records are assigned as rows and each segment is put in separate column. All other symbols are sequences, and are assigned to one of three possible sequence types:

- n-character long string of letters (marked as Zn),
- n-character long string of digits (marked as Ln),
- n-character long string of alphanumerical symbols (marked as Dn).

Sequence can be combined with other sequences in the same segment, creating larger sequence. For example Z4 and Z2 sequences with link '@' between them can be merged into Z4@Z2 sequence or D7 sequence. Also some characteristics of record and each of its segments are noted - length, number and type of links. They will be used in later step of algorithm to create more general sequences in the way described above allowing a broader range of sequences to fit into a dominating type.

In the next step NegFS creates matrices M and C. In matrix M there are stored sequence types - records as rows and segments as columns. Additionally it separates records with full number of columns from those with lesser number - which can be potential anomalies.

At that stage of algorithm those are only noted as potentially anomalous, information to be used in last part of algorithm or by neural network or rule-based system using NegFS as preprocessing tool. In matrix C are stored, in the same order, common characteristics of each segment in record - length, number of links and type of links. Basing on common types of sequences found in each column, the algorithm creates a vector of dominating types and gives to each type of sequences certain compatibility score. Additionally, basing on matrices M and C, algorithm is able to create new general types of sequences, based on common characteristics that allow it to fit a broader number of types, which are proper.

After that NegFS creates a matrix S of dominating sequences. For each column it is created according to one of given rules:

- percentage of occurrences of given sequence S in given column is above 1% (for large and very large datasets; this percentage threshold should not be used for small datasets),
- number of occurrences of given sequence S in given column is above numerical threshold T_d,
- given sequence S belongs to intersection of most common sequence types in given column.

All records with sequences not belonging to dominating types are given some measure of incompatibility I, the larger the more different the sequence is from any of dominating sequence types. Potentially anomalous records (i.e. records without full column number) are given increase in incompatibility score I depending on used criteria. In the last step records with incompatibility score above given anomaly threshold T_i are noted as anomalies.

Brief summary of the above algorithm is as follows:

<u>Algorithm</u>

Step 1 - creation of matrices M and C

Step 2 - creation of matrix of dominating elements S

Step 3 - calculating incompatibility score I

Step 4 - determining if records are proper or anomalous

<u>Example 1</u>

Lets show how the NegFS works with example consisting of simple 3-record dataset presented below:

John Doe	01018400001	j.doe@test.info	doe@test.com.uk	500000000	600 – 100 – 200	No data
Mike Black	04017216302	black@pw.edu.pl	black@test.com.uk	461328114	500 - 111 - 111	No data
Corrupted	01020300004	youtube.com	www.pw.edu.pl	300400500	100 - 200 - 300	

is divided into seven segments divided by separator (in this case a tabulator). Matrix M for those records looks like:

Z4 Z3	L11	Z1.Z3@Z4.Z4	Z3@Z4.Z3.Z2	L9	L3 - L3 - L3	Z2 Z4
Z4 Z5	L11	Z5@Z2.Z3.Z2	Z5@Z4.Z3.Z2	L9	L3 - L3 - L3	Z2 Z4
Z9	L11	Z7.Z3	Z3.Z2.Z3.Z2	L9	L3 - L3 - L3	

And matrix C looks like:

(D8 L1)	(D11 L0)	(D15 L3.@.)	(D15 L3@..)	(D9 L0)	(D11 L2 - -)	(D7 L1)
(D10 L1)	(D11 L0)	(D15 L3@..)	(D17 L3@..)	(D9 L0)	(D11 L2 - -)	(D7 L1)
(D9 L0)	(D11 L0)	(D11 L1)	(D13 L3...)	(D9 L0)	(D11 L2 - -)	

In matrix C in brackets there are characteristics of each segment noted - length (D), number of links (L) and types of links. It is also noted that record 3 has less segments then other records.

For this simple example dominating matrix is created by taking sequence with simple majority, treating records 1 and 2 as having much greater weight then record 3 (we can imagine that they are representing typical records for given larger dataset and record 3 is anomalous sample that appears only once) and the matrix S should look like the one below:

(D10 L1) (D11 L0) (D15 L2@.) (D17 L3@..) (D9 L0) (D11 L2 - -) (D7 L1)
(D8 L1) (D15 L3@..)

As we can see, both e-mail addresses in e-mail column should be considered as normal – as there is a common part that is dominant in that column, NegFS will classify both addresses as (D15 L2@.) - string consisting of 15 symbols with 2 separators in order - '@' and '.'. It is worth noting that additional separators in that case are treated as normal symbols.

In the last step NegFS checks incompatibility score for records with sequences not belonging to dominating sequence types, in this case record number 3. I(1) = 0, I(2) = 0, I(3) = 2, 6.

Record number 3 is clearly separated as anomalous. Calculation and meaning of incompatibility score will be shown in detail in the next section.

2.1 Selection of parameters

Threshold T_d determining dominating sequence types should be set according to data used. For small databases (below 1000 records where 1% rule can be inconvenient as few anomalies of the same type can fit into it) it should be set to around 3-5% of records. In larger databases it can be used as *precision criterion*, changed to try to find some of the more frequent anomalies (in this case 1% criterion is omitted) or to penalize sequences below both 1% and Td thresholds, giving them bigger incompatibility score. In the situation where 1% criterion is omitted it should be noted that casual use of big T_g thresholds can lead to marking normal sequences as anomalies.

Threshold T_i is used to differentiate normal records from anomalies based on their incompatibility score I. It is found that for different types of data different thresholds are optimal and it is hard to find one in the first run of the algorithm. As in most cases data types in given dataset are changing only slightly with time passing thus it is suggested to run last step of NegFS on given dataset many times with different T_i until we find one we deem optimal for given dataset. As last step of algorithm is simple comparisons of incompatibility score I of each record with threshold T_i its computational complexity is only O(r), where r is number of records, not the size of dataset.

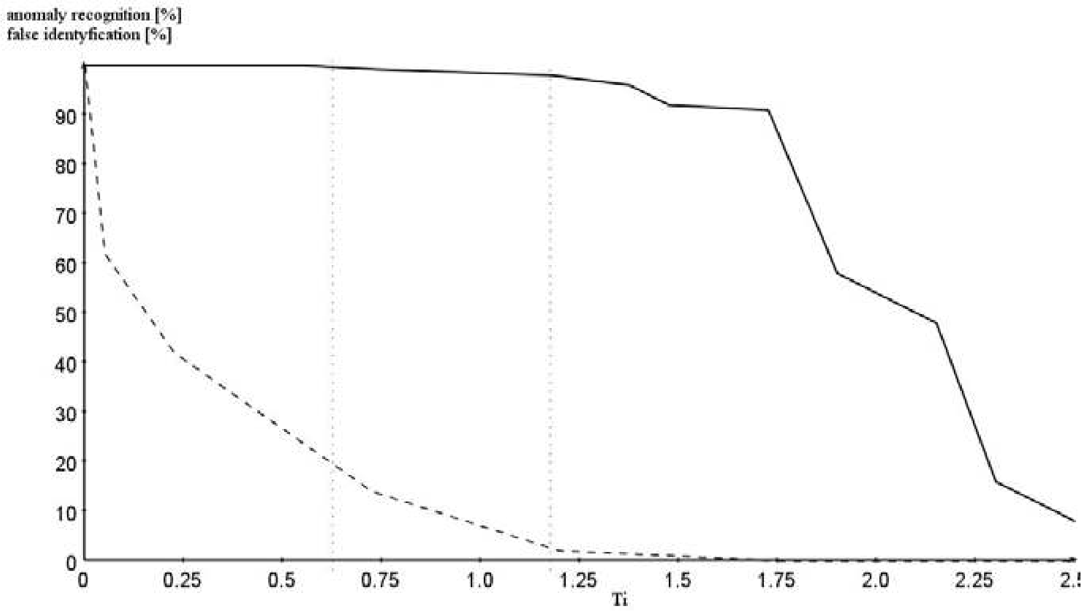

Figure 2.1. Dependence between threshold Ti, anomaly identification and false recognition

In figure 2.1 above there is shown an example of one of artificial datasets (chosen because of clearly visible borders of proper T_i threshold, due to mixed incompatibility of proper and anomalous records), that choosing proper T_i threshold can seriously affect results gained. Solid line shows recognition (in %) of anomalous records and dotted line, false recognition percentage. It can be seen, that for that dataset we will get best results for T_i threshold between 0.62 (where we get 99,5% recognition of anomalous records and 21% false recognition value) and 1.18 (with 98% recognition of anomalous records and 1,5% of false alarms). Parameters outside of that spectrum are clearly inferior for given dataset - giving increase in false recognition value without sufficient gain in recognition of anomalous records or decrease in recognition percentage with only a slight decrease of false alarms generated. By modifying T_i we can set what is more important to us - finding all anomalies with slightly higher rate of false alarms or allowing some anomalies (the least visible ones) to remain undetected in this step (for example in preprocessing) while gaining minimal number of false alarms, which speeds up human response to most visible (and probably dangerous) anomalies.

Incompatibility score I can be calculated in many ways and it is hard to find way of calculating an optimal incompatibility score for any dataset with unknown properties. Proposed ways of calculating incompatibility score are simple and should give good, but not in any way optimal, results for most used datasets. It is possible to run NegFS with different formulae for calculating I and its computational complexity is only O(rs) for each run (where s is number of segments in record). For first run of NegFS on given large and frequently checked dataset it can be useful to run it with different formulae for I and different T_i thresholds, task with additional $O(r^2s)$ computational complexity.

Incompatibility score for record i can be calculated using the following simple formula:

$$I_i = \sum_j^s \frac{R_j(i)}{s}$$

where:

$$R_j(i) = \min[R_{1j}, R_{2j}, \ldots, R_{zj}, \ldots R_{xj}]$$

$$R_{zj} = \Delta t_{zj} + \Delta l_{zj} * 0{,}3 + \Delta s_{zj} * 0{,}1$$

and

$$z = 1,2,\ldots,x$$

$$\Delta t_{zj} = \begin{cases} 1 \text{ if type is different in } S_{ij} \text{ and } M_{zj} \\ 0 \text{ if type is the same in } S_{ij} \text{ and } M_{zj} \end{cases},$$

Δl_{zj} is number of different links in S_{ij} and M_{zj},

Δs_{zj} is number of different symbols in S_{ij} and M_{zj},

s is number of segments in given record,
x is maximum number of dominating sequences

3 Algorithm evaluation

NegFS algorithm was evaluated both theoretically (with computational complexity) and experimentally - by running it on artificial and real-life datasets, including benchmark dataset used for testing other new algorithms [5] [6].

3.1 Theoretical evaluation

Algorithm's computational complexity for each step can be calculated as follows:

- for dividing dataset into records and segments; creation of matrices M and C NegFS needs one run on full dataset and this step's computational complexity is O(n),

- creation of matrix S requires operations on all records and segments - step's computational complexity is O(rs), where r is number of records in dataset and s is number of segments in record. In the best case (big records with few separators) step's complexity is marginal (as n >> r * s); in the worst case (records with single symbol in each segment, NegFS operation for such a case is shown later in the article) r * s = n,
- incompatibility score I is determined - as mentioned earlier it is O(rs) task,
- anomalies are detected - as mentioned earlier it is O(r) task.

As shown above NegFS' computational complexity is O(n) making it fast algorithm capable of real-time operations on large datasets.

3.2 Experimental evaluation

For experimental evaluation algorithm was compiled using Dev C++ 4 and run on two computers:

- Intel Core 2 4300 @ 1.80Ghz, 4 GB RAM, Windows XP Professional with SP2, multiple software installed,
- laptop Gateway NV-7915U, Intel Core i3-330M 2.13GHz, 4 GB RAM, fresh installation of Windows 7.

Time shown is average from 10 runs of each algorithm on second computer, biggest difference in running times on the same computer was 7%. Running times on first computer set were about 23-29% slower.

The proposed method has been successfully verified and validated with benchmark dataset (Credit Approval dataset) from the UCI Machine Learning data repository [6].

Below are shown algorithm working times and accuracy (detection rate) for different datasets, both artificial and real-life.

Table 3.1. Working time and detection rate of NegFS algorithm for artificial datasets

Dataset size	Working time [s]	# of anoma-lies	False detection	Detection rate
70x255	1,14	1	4%	100%
4000x1000	17,06	70	1%	98%
50000x1000	212,93	200	0%	99%

Table 3.2. Comparison of working times of neural network and NegFS+neural network for artificial datasets

Dataset size	Working time for NN [s]	Working time for NegFS+NN [s]	Detection rate for NN	Detection rate for NegFS+NN
70x255	2,04	3,72	98,5%	100,0%
4000x1000	101,21	96,51	98,9%	99,3%

As shown in table 3.1 NegFS has good anomaly detection rate and low false detection with larger datasets, it is also fast enough to work in real-time. In table 3.2 it can be seen, that for larger datasets NegFS improves efficiency and speed of simple neural network.

Table 3.3. Working time and detection rate of NegFS algorithm for benchmark dataset, real-life data from UCI repository

Test type	Working time [s]	# of false detections	Detection rate
single run	0,78	12	100%
Ti configuration run	1,49	4	100%

Results shown in table 3.3 are for Credit Approval benchmark dataset from UCI repository, real-life dataset of 690 records in 16 columns. There are 37 anomalies – records with one or more missing values. Columns in dataset have different forms (continuous, nominal with small numbers of values, and nominal with larger numbers of values) and attributes and many columns are stored in worst for NegFS form - single symbol in column. Despite this algorithm works fast and it's detection rate for basic settings is satisfying (100% detection of existing anomalous records, false alarms below 1% for T_i configuration run). Using other incompatibility score formulae, that take into account specific form of presented data, instead of general ones, would give even better results, reducing false detection rate.

4 Conclusions

New algorithm for anomaly detection based on negative feature selection is proposed. In the article it has been evaluated for both artificial and real-life data.

The proposed NegFS algorithm gives new possibilities for anomaly detection in real-time systems either as unsupervised anomaly detection tool or as preprocessing tool used to speed up other commonly used anomaly detection techniques. It allows semi-supervised and supervised detection techniques to be used on unlabeled datasets of unknown structure. Furthermore experimental investigation shows that it allows to save the computing time when working with large and very large datasets. Still the optimal values of parameters T_d, T_i (for common dataset types) and I are to be researched, as there are also some possibilities open of increasing both speed and detection rate of algorithm.

References

1. D. Hawkins, (1980), *Identification of Outliers*, Chapman and Hall.
2. Sandeep Kumar, (1995), *Classification and detection of computer intrusions*, PhD. Thesis, Purdue University.
3. T. Lane, C.E. Brodley, (1997), *Sequence Matching and Learning in Anomaly Detection for Computer Security*, AAAI (American Institute of Aeronautics and Astronautics) Workshop: AI Approaches to Fraud Detection and Risk Management, pages 43-49.

4. C.C. Aggarwal, P. Yu, (2001), *Outlier Detection for High Dimensional Data*, Proceedings of the ACM SIGMOD Conference, pages 37-46.
5. R. Xu, T. Qian, G. Zhang, (2010), *A Novel Anomaly Detection Technique based on Limited Anomalous Data*, AIAA (American Institute of Aeronautics and Astronautics) Infotech@Aerospace 2010, Atlanta.
6. A. Frank, A. Asuncion, (2010UCI *Machine Learning Repository*. [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
7. A. Soule, K. Salamatian, N. Taft, (2005), *Combining Filtering and Statistical Methods for Anomaly Detection*, *IMC (Internet Measurement Conference) '05 Proceedings*, pages 331-344.
8. Gerhard Münz, (2009), *Traffic Anomaly Detection and Cause Identification Using Flow-Level Measurements*, PhD Thesis, Technische Universität München.
9. Varun Chandola, Arindam Banerjee, Vipin Kumar, (2009*), Anomaly Detection: A Survey*, ACM Computing Surveys, Vol. 41(3), Article 15.
10. D. Olszewski, *Fraud Detection in Telecommunications Using Kullback-Leibler Divergence and Latent Dirichlet Allocation*, *In A. Dobnikar, U. Lotrič, and B. Šter (Eds.): ICANNGA 2011, Part II, LNCS 6594, pp. 71-80*, Springer 2011.