ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

Parallelized Unsupervised Feature Selection for Large-Scale Network Traffic Analysis

Bruno Ordozgoiti, Sandra Gómez Canaval and Alberto Mozo *

Universidad Politécnica de Madrid - Departamento de Sistemas Informáticos
{bruno.ordozgoiti, sgomez}@etsisi.upm.es, a.mozo@upm.es

Abstract. In certain domains, where model interpretability is highly valued, feature selection is often the only possible option for dimensionality reduction. However, two key problems arise. First, the size of data sets today makes it unfeasible to run centralized feature selection algorithms in reasonable amounts of time. Second, the impossibility of labeling data sets rules out supervised techniques. We propose an unsupervised feature selection algorithm based on a new formulation of the leverage scores. We derive an efficient parallelized approach over the Resilient Distributed Datasets abstraction, making it applicable to the enormous data sets often present in network traffic analysis.

1 Introduction

Dimensionality reduction (DR) is often a crucial step for the successful application of machine learning (ML). Some DR techniques, such as Principal Component Analysis, transform the data into a new subspace that can be hard to interpret for domain experts. To overcome this, feature selection (FS) can be employed. In fields such as bioinformatics, economics or network traffic analysis, model interpretability is a frequent requirement when using ML, making FS attractive as an approach for DR. However, two obstacles must be overcome. First, the huge size of data sets today can make FS algorithms too slow. Second, many FS techniques are supervised and need ground truth. Some of the existing proposals for unsupervised FS are based on the column subset selection problem (see section 2) following both randomized [1], and deterministic approaches [2]. The former lacks an efficient strategy for finding the minimizing subset, while the latter can be inaccurate in practice. Some recent proposals try to handle large volumes of data, like [3], designed for MapReduce —inefficient when repeatedly accessing data—, or [4], for MPI, which lacks fault tolerance, resilience and data distribution. In [5], a distributed algorithm is proposed, but it is just a straightforward implementation of an existing algorithm and does not exploit the nature of the problem for improved efficiency. In the context of ML for network traffic analysis, FS is often supervised, which is not feasible for big data, or done by expert criteria. An overview of the existing applications of ML to this domain can be found in [6]. We propose an FS algorithm that overcomes these issues. It is unsupervised, based on a new formulation of the leverage scores

*The research leading to these results has received funding from the European Union under the FP7 grant agreement n. 619633 (project ONTIC) and H2020 grant agreement n. 671625 (project CogNet)

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

independent of the target rank, and therefore applicable to unlabeled data. In addition, we derive an efficient parallelization over the recently surfaced Resilient Distributed Datasets paradigm (RDD) [7], the rapidly evolving new paradigm for Big Data processing whose in-memory computation capabilities enable it to outperform MapReduce. We implement it on the Apache Spark distributed computing platform, the official RDD implementation, making it able to handle huge data sets. To demonstrate the performance of our algorithm and draw domain-relevant conclusions, we test it on a 183GB, 1-month-long Internet traffic data set captured at the core network of an Internet Service Provider (ISP).

2 Algorithm Description

Notation: We denote sets by uppercase letters (S, Θ) and matrices by uppercase bold letters ($\mathbf{A}, \mathbf{\Sigma}$). Lowercase bold letters ($\mathbf{x}$) denote column vectors. $\mathbf{A}_{(i,:)}$ is the i-th row of $\mathbf{A}$ (understood as a column vector), and $\mathbf{A}_{(\Theta,:)}$ is the rows of $\mathbf{A}$ whose indices are the elements of Θ (we use analogous notation for columns, as well as vectors). By RDD(S) we denote a Resilient Distributed Dataset whose entries are the elements of S. $\mathcal{W}(c, \mathbf{x}_n, \boldsymbol{\pi})$ denotes a multivariate instance of Wallenius' noncentral hypergeometric distribution with parameters $c, \mathbf{x}_n, \boldsymbol{\pi}$.

The Column Subset Selection Problem (CSSP) is an interesting framework for unsupervised feature selection methods, and is defined as follows:

Definition 1 *Column Subset Selection Problem. Given a matrix* $\mathbf{A} \in \mathbb{R}^{m \times n}$ *and a positive integer k, let* $\mathcal{A}_k$ *denote the set of matrices comprised of k columns of* $\mathbf{A}$. *Find* $\mathbf{C}$ *such that*

$$\mathbf{C} = \underset{\mathbf{X} \in \mathcal{A}_\mathbf{k}}{\operatorname{argmin}} \|\mathbf{A} - \mathbf{X}\mathbf{X}^{+}\mathbf{A}\|_F$$

where $\mathbf{X}^{+}$ *denotes the Moore-Penrose pseudoinverse of* $\mathbf{X}$.

The CSSP is believed to be NP-Hard. Inspired by [1], our proposal (Algorithm 1) approximately solves the CSSP in two phases: a random and a deterministic one. The **random phase** performs a biased column sampling process based on the statistical leverage scores, as suggested in [1] and [8]. However, we propose a new formulation of these scores that does not require a target rank (equation (1)), making it possible to choose the value of k *a posteriori*. For an arbitrary choice[1] of $\gamma \in \mathbb{N}^*$, we draw γ samples from $\mathcal{W}(k \log k, \mathbf{1}_n, \boldsymbol{\pi})$ (Algorithm 1, line 4), where $\mathbf{1}_n$ is a vector of n ones. This way, we obtain γ judiciously sampled random subsets of $k \log k$ non-repeating column indices with biased probability vector $\boldsymbol{\pi} = (p_1, p_2, \ldots, p_n)$. Column i is thus sampled with probability p_i, with

$$p_i \propto \| \ (\mathbf{\Sigma}\mathbf{V}^T)_{(:\rho,i)} \|_2^2 \tag{1}$$

where $\mathbf{V}$ is a matrix whose columns are the right singular vectors of $\mathbf{A}$, $\mathbf{\Sigma}$ is the diagonal matrix of the decreasing singular values of $\mathbf{A}$ and ρ is an estimation of the numerical rank. Intuitively, high values of p_i correspond to columns that are well aligned with a singular vector and that, because of the orthogonality of $\mathbf{V}$, are almost orthogonal to the space spanned by the rest of the columns. By

[1] We set $\gamma = \min(4^{\sqrt{0.3n-k}}, 40)$ to increase performance if k approaches the numerical rank.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Algorithm 1

procedure ChooseColumns($\mathbf{A}, k, \gamma$)
  rows $\leftarrow$ RDD($\{\mathbf{A}_{(i,:)} | 1 \leq i \leq m\}$)
  for $i = 1, \ldots, \gamma$ **do**
    $\tilde{\Omega} \sim \mathcal{W}(k \log k, \mathbf{1}_n, \boldsymbol{\pi})$
    $\Omega_i \leftarrow \mathrm{RRQR}_k\big((\boldsymbol{\Sigma}\mathbf{V}^T)_{(:,\tilde{\Omega})}\big)$
  end for
  $\mathbf{Q}^T\mathbf{A} \leftarrow$ rows.**map**{ $\mathbf{x}$ =>
    output $\mathbf{x}_{(\Theta)}\mathbf{x}^T$
  }.**reduce**{$(\mathbf{X}, \mathbf{Y})$ => $\mathbf{X} + \mathbf{Y}$}
  $P \leftarrow \emptyset$
  for $i = 1, \ldots, \gamma$ **do**
    $\mathbf{C}_i^T\mathbf{A} \leftarrow (\mathbf{Q}^T\mathbf{A})_{(\Omega'_i,:)}$;
    $\mathbf{C}_i^T\mathbf{C}_i \leftarrow (\mathbf{Q}^T\mathbf{A})_{(\Omega'_i,\Omega_i)}$
    $P \leftarrow P \cup \{(\mathbf{C}_i^T\mathbf{C}_i)^{-1}\mathbf{C}_i^T\mathbf{A}\}$
  end for
  broadcast(P, T, Ω)
  $\boldsymbol{\delta} \leftarrow$ rows.**map**{ $\mathbf{x}$ =>
  for $i = 1, \ldots, \gamma$ **do**
    $\mathbf{d}_i \leftarrow \mathbf{x} - \mathbf{x}^T_{(\Omega_i)}\mathbf{C}_i^+\mathbf{A}$
      ▷ Note that $\mathbf{C}_i^+\mathbf{A} \in P$
  end for
  output $(\|\mathbf{d}_1\|_2^2, \ldots, \|\mathbf{d}_\gamma\|_2^2)$
  }.**reduce**{$(\mathbf{x}, \mathbf{y})$ => $\mathbf{x} + \mathbf{y}$}
  output Ω_i with $i = \underset{i}{\mathrm{argmin}}\ \delta_i$
end procedure

multiplying by the singular values we favor the leading singular vectors. It is not necessary to compute all vectors, since the entries of $\mathbf{V}^T$ that are multiplied by a small singular value vanish and do not contribute noticeably to the final score. In practice, the top-ρ values and vectors of a numerically rank-ρ matrix will suffice. For each column index sample $\tilde{\Omega}$ we then run an RRQR factorization [9] on $(\boldsymbol{\Sigma}\mathbf{V}^T)_{(:,\tilde{\Omega})}$, keeping only the first k elements from the resulting permutation. The **deterministic phase** computes the approximation error on the Frobenius norm for all candidates and chooses the one that minimizes it. We define the residual error δ_i corresponding to candidate column choice i, with $1 \leq i \leq \gamma$, as

$$\delta_i = \|\mathbf{A} - \mathbf{C}_i\mathbf{C}_i^+\mathbf{A}\|_F \tag{2}$$

The computation of all these errors can be expensive, since it involves the computation of γ pseudoinverses, as well as γ large matrix products and subtractions. To significantly decrease the computation time needed, we derive a parallelized algorithm that yields the δ_i's efficiently and in a scalable manner. Since for any real $m \times k$ matrix $\mathbf{C}$, with $k \leq n$,

$$\mathbf{C}^+\mathbf{A} = \underset{\mathrm{rank}\mathbf{X}=k}{\arg\min} \|\mathbf{A} - \mathbf{C}\mathbf{X}\|_F \tag{3}$$

it can be easily seen that the computation of the different columns of $\mathbf{C}^+\mathbf{A}$ is equivalent to solving n instances of linear regression (it suffices to see that $(\mathbf{C}^+\mathbf{A})_{(:,i)} = \arg\min_{\mathbf{x}} \|\mathbf{A}_{(:,i)} - \mathbf{C}\mathbf{x}\|_2$ for $i \in \{1, \ldots, n\}$). Therefore, the problem (3) for a particular column subset $\mathbf{C}_i$ has the following analytical solution [10]:

$$\mathbf{C}_i^+\mathbf{A} = (\mathbf{C}_i^T\mathbf{C}_i)^{-1}\mathbf{C}_i^T\mathbf{A} \tag{4}$$

Since $\mathbf{C}_i^T\mathbf{C}_i \in \mathbb{R}^{k \times k}$ for all i, the computation of this inverse in the context of feature selection, where normally $k \ll n$, can be done fast. Singular matrices can be discarded, since they correspond to candidates with rank less than k. In any case, the distribution of equation (1) significantly decreases the chance of obtaining a singular matrix at this step. In practice, the different candidate column subsets tend to share many columns in common. Therefore, it is desirable to avoid computing all instances of $\mathbf{C}_i^T\mathbf{C}_i$ and $\mathbf{C}_i^T\mathbf{A}$ separately (each of which

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

is a $k \times m$ by $m \times k$ and a $k \times m$ by $m \times n$ matrix product respectively), since it would involve redundant computations. To this end, we propose the following method. Let Ω_i denote the set of column indices obtained after running the RRQR factorization on the i-th random column sample of $\mathbf{\Sigma V}^T$, and we define $\Theta = \bigcup_{i=1}^{\gamma} \Omega_i$ and $\mathbf{Q} = \mathbf{A}_{(:,\Theta)}$. We can then obtain any instance of the desired matrices as $\mathbf{C}_i^T\mathbf{A} = (\mathbf{Q}^T\mathbf{A})_{(\Omega_i',:)}$ and $\mathbf{C}_i^T\mathbf{C}_i = (\mathbf{Q}^T\mathbf{A})_{(\Omega_i',\Omega_i)}$ (lines 12, 13), where Ω_i' denotes the set of indices of the columns of matrix $\mathbf{Q}$ that correspond to the columns whose indices in matrix $\mathbf{A}$ are those in Ω_i, i.e. $\mathbf{Q}_{(:,\Omega_i')} = \mathbf{A}_{(:,\Omega_i)}$. This way we can compute (4) for all i through one $\tau \times m$ by $m \times n$ matrix product (i.e. $\mathbf{Q}^T\mathbf{A}$), for some $\tau \leq n$. This derivation allows for a very efficient parallelization on the RDD paradigm. The product $\mathbf{Q}^T\mathbf{A}$ is computed via the map-reduce operation shown in lines 7–9. We then build and broadcast the set $P = \bigcup_{i=1}^{\gamma} \mathbf{C}_i^+\mathbf{A}$ (lines 14, 16), incurring just $O(\gamma kn)$ network usage[2], and compute all the residuals by the map-reduce operation detailed in lines 17–22.

3 Experimental Results

To validate the performance of our algorithm, we employed a traffic data set captured at the core network of a medium-sized ISP during one month[3]. Each day of the month consists of a 4-10GB file containing from 12 to 27 million 5-tuple TCP flows[4], totalling 534 million data samples and 183 gigabytes, expressed in 95 features. The experiments were run on a cluster of 10 worker nodes with an Intel quad-core processor and 4 GB of RAM each, connected through an ethernet switch with a capacity of 100 Mbps on each link. Our algorithm was implemented using Scala 2.10.4 on Spark 1.4.1. The Hadoop 2.6 Distributed File System was used for data storage and access. We ran the algorithm on each day separately (obtaining 30 feature subsets) and evaluated the results. We set $k = 10$ to obtain a manageable number of features for domain expert analysis.

Ratio to SVD. To evaluate the quality of the chosen feature subsets, we measure the ratio of the residual obtained with our algorithm to that of the best rank-k approximation, $\frac{\|\mathbf{A}-\mathbf{C}\mathbf{C}^+\mathbf{A}\|_F}{\|\mathbf{A}-\mathbf{A}_k\|_F}$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the data, $\mathbf{C} \in \mathbb{R}^{m\times k}$ is a subset of its columns and $\mathbf{A}_k$ is the best rank-k approximation to $\mathbf{A}$. Figure 1 shows the described ratio on each day for three feature subsets: the one chosen by the algorithm on that day (*algorithm_ratio*), the 10 features that appear the most often among the daily chosen subset throughout the month (*best_app*), shown in table 1, and the top-10 features according to their accumulated leverage scores (*best_scores*). The value of *algorithm_ratio* remains consistently close to 1.05, revealing that the analyzed data contains feature subsets that can approximate the full data set almost as well as the top-k singular vectors, and that the proposed algorithm can find such a subset. The ratio for *best_app* is slightly higher, but moderate throughout the month, suggesting that there exists a single subset of features that can consistently provide good approximation errors.

[2]In network traffic analysis n is usually small, around a few hundred, and normally $k \ll n$. Finally, γ is a user-defined parameter that need not be more than a few dozen.

[3]The data span the period from April 7th to May 6th 2015.

[4]We aggregated the network packets into flows using Tstat 3.0. http://tstat.polito.it/

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

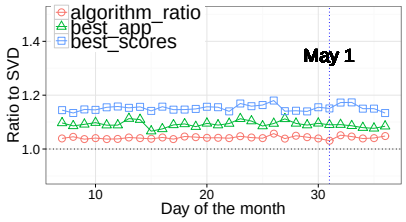

Fig. 1: Ratio to SVD

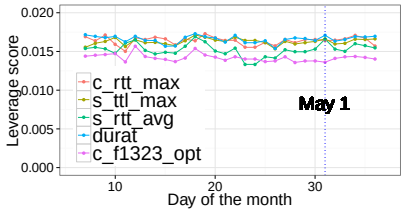

Fig. 2: LS of the top-5 features

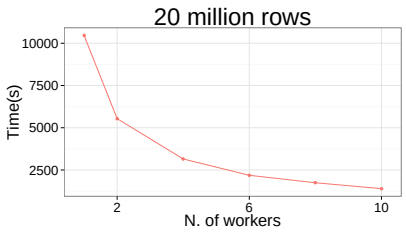

Fig. 3: Time with respect n. of workers

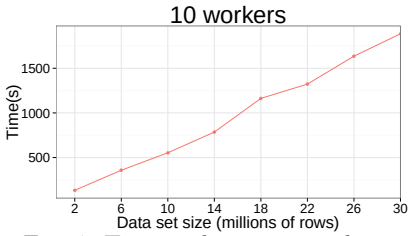

Fig. 4: Time with respect to data size

Applicability of our algorithm. To assess the usefulness of our approach in the domain of network traffic analysis, we examine the chosen features and their leverage scores. We collect the top-10 that appear the most often among the chosen subset throughout the month (table 1, where prefixes *c* and *s* indicate client-to-server and server-to-client respectively). The leverage scores of these features are shown in figure 2. The fact that these features have consistently high leverage scores[5] suggests that the sampling strategy proposed in equation (1) is a good heuristic for randomly sampling candidate features. It is interesting to point out the relationship between the obtained features and those most frequently present in the literature. Some of them (*durat*, *c_cwin_ini*) are consistent with choices often made when applying machine learning techniques to network traffic, usually relying on expert criteria or supervised methods [6]. Others (to the best of our knowledge) had not been previously identified as being especially relevant or informative (*s_ttl_max*, *c_ttl_max*, *c_f1323_opt*, *c_pkts_retx*, *s_ack_cnt_p* and *s_bytes_retx*). Finally, we did not detect *c_rtt_max* or *s_rtt_avg* in the literature, although related metrics such as inter-packet arrival times do appear. It would be interesting to study the linear dependencies between these sets of features and determine which tend to approximate data best.

Scalability. Figures 3 and 4 show running times with respect to the number of workers in the cluster (with 20 million rows) and the size of the data set (in millions of rows), exhibiting significant benefits to be gained from parallelization and roughly linear scalability with respect to the number of data samples.

4 Conclusions and future work

We presented an efficient parallelized unsupervised feature selection algorithm and applied it to the domain of network traffic analysis. We showed that a feature

[5]The highest observed leverage scores overall are below 0.18.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Top-10 appearing features		
Code	Description	Days present
c_rtt_max	Maximum RTT	19
s_ttl_max	Maximum Time To Live	18
s_rtt_avg	Average RTT	16
durat	Flow duration	15
c_f1323_opt	Window scale option sent (boolean)	14
c_pkts_retx	Number of retransmitted segments	12
c_cwin_ini	First in-flight size	11
s_ack_cnt_p	Segments with ACK=1 and no data	11
c_ttl_max	Maximum Time To Live	11
s_bytes_retx	Number of retransmitted bytes	10

Table 1: Top features, based on the times they were chosen throughout the month.

subset can retain almost as much information as the SVD, and that a fixed feature subset can consistently provide a good approximation. We identified network traffic features previously proposed in the literature, as well as others that had not been previously recognized as useful. We verified the efficiency and linear scalability of our algorithm, which make it applicable to huge data sets. In the future we plan to explore alternative sampling strategies, non-linear approximations and online versions of the algorithm. We also plan to study what the leverage scores and the chosen features can reveal about network behavior.

References

[1] C. Boutsidis, M. W Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proc. of the 20th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 968–977. Soc. for Industrial and Applied Mathematics, 2009.

[2] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of the 20th ACM SIGKDD*, pages 997–1006. ACM, 2014.

[3] A. K Farahat, A. Elgohary, A. Ghodsi, and M. S. Kamel. Distributed column subset selection on mapreduce. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 171–180. IEEE, 2013.

[4] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle. Massively parallel feature selection: an approach based on variance preservation. *Machine learning*, 92(1):195–220, 2013.

[5] B. Ordozgoiti, S. Gómez, and A. Mozo. Massively parallel unsupervised feature selection on spark. In *New Trends in Databases and Information Systems*. 186–196, Springer, 2015.

[6] T. TT Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys & Tutorials, IEEE*, 10(4):56–76, 2008.

[7] Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proc. of the 9th USENIX conf. on Networked Systems Design and Implementation*, pages 2–2. USENIX Assoc., 2012.

[8] P. Drineas, M. W Mahoney, and S Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[9] T. F Chan. Rank revealing qr factorizations. *Linear algebra and its applications*, 88, 1987.

[10] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.