Online Feature Selection for Low-overhead Learning in Networked Systems

Xiaoxuan Wang[†], Forough Shahab Samani[†§], Andreas Johnsson[‡], Rolf Stadler[†§]
[†] KTH Royal Institute of Technology, Sweden – Email: {xiaoxuan, foro, stadler}@kth.se
[‡]Ericsson Research, Sweden – Email: andreas.a.johnsson@ericsson.com
[§]RISE Research Institutes of Sweden

***Abstract*—Data-driven functions for operation and management require measurements and readings from distributed data sources for model training and prediction. While the number of candidate data sources can be very large, research has shown that it is often possible to reduce the number of data sources significantly while still allowing for accurate prediction. Consequently, there is potential to lower communication and computing resources needed to continuously extract, collect, and process this data. We demonstrate the operation of a novel online algorithm called OSFS, which sequentially processes the collected data and reduces the number of data sources for training prediction models. OSFS builds on two main ideas, namely (1) ranking the available data sources using (unsupervised) feature selection algorithms and (2) identifying stable feature sets that include only the top features. The demonstration shows the search space exploration, the iterative selection of feature sets, and the evaluation of the stability of these sets. The demonstration uses measurements collected from a KTH testbed, and the predictions relate to end-to-end KPIs for network services.**

***Index Terms*—Feature Selection, Data-driven Engineering, Machine Learning, Network Management**

I. Background/Concepts

Data-driven network and systems engineering is based upon applying AI/ML methods to data collected from an infrastructure in order to build novel functionality and management capabilities. This is achieved through learning tasks that use this data for training. Examples are KPI prediction and forecasting through regression and anomaly detection through clustering techniques.

Data sources that feed the learning tasks include real-time measurements collected through monitoring. The number of available data sources can be very high, even in small systems. For example, on our testbed at KTH, which includes 10 compute servers, we can extract several thousand metrics from the operating system and orchestration layers. Since these metrics are dynamic, we monitor them periodically, e.g., once per second. It requires a significant overhead to extract and collect this data, as well as a significant computational overhead to train and update the machine-learning models that underlie the learning tasks. Considering the fact that the monitoring and computational overhead increases at least linearly with the number of measurements and the dimensionality of the input, i.e., the number of (one-dimensional) data sources, it becomes vital to reduce the number of data sources to the extent possible. Our earlier research has shown the feasibility, through offline analysis, to reduce the number of data sources significantly while still allowing for accurate prediction [1].

Based on this result, we developed a novel online source-selection algorithm called *Online Stable Feature Set algorithm (OSFS)*, which can reduce the number of sources needed for online training of models that are effective for learning tasks [2] [3]. In this paper, we describe a demonstration of OSFS, which processes measurement data from a KTH testbed.

Using the terminology of machine learning, we call a (one-dimensional, scalar) data source also *a feature*, and we refer to measurements taken from a set of data sources at a specific time as *a sample*.

OSFS builds on two main ideas, namely (1) ranking the available features using (unsupervised) feature selection algorithms and (2) identifying *stable feature sets* that include only the top k features. We call a feature set stable, if it remains sufficiently similar when additional samples are considered. During execution, OSFS explores the space of ranked features and available samples following a configurable search policy.

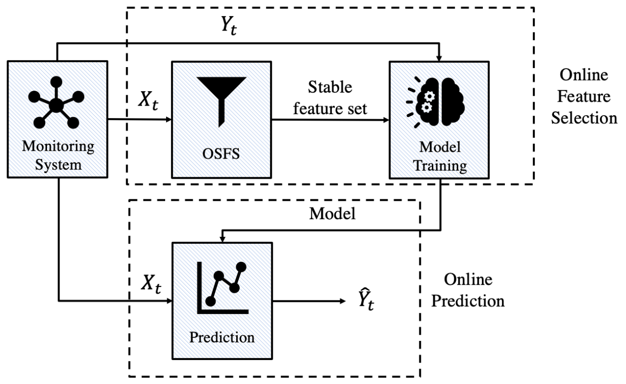

Fig. 1: Online learning: feature selection and prediction.

Figure 1 shows the role of OSFS for online learning and prediction. The monitoring system continually collects samples and provides them as input to OSFS, which identifies a stable feature set, i.e. a subset of all features, to be used for model training. Whenever such a set becomes available, a (new) model is trained and then used for online prediction.

Formally, OSFS reads a stream of samples $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_3$,... and returns the number of features k, the number of samples

t needed to determine k, and the stable feature set $\boldsymbol{F}_{k,t}$. To keep costs and overhead low, we want k and t to be small while still allowing for effective learning and prediction.

Algorithm 1: Online Stable Feature Set (OSFS)

```
Input: Sample sequence $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_3$,...;
       Feature ranking algorithm $A$;
       Metric and condition for stable feature set $C$;
       Search space $S$ for $(k, t)$;
       Search policy $P$ to traverse $S$.
Output: Stable feature subset $\boldsymbol{F}_{k,t}$, $k$, $t$.
1 Initialize $(k, t)$;
2 while True do
3     Compute $\boldsymbol{F}_{k,t}$ using $A$ on $\boldsymbol{X}_1$,..., $\boldsymbol{X}_t$;
4     if $\mathbf{F}_{k,t}$ is stable or $S$ is exhausted then
5         return $\boldsymbol{F}_{k,t}$, $k$, $t$;
6     else
7         choose next $(k, t)$ following $P$;
```

Algorithm 1 presents OSFS in a generic form, which allows for a range of instantiations and configurations. First, we select a feature ranking algorithm. Second, we provide a condition for a feature set $\boldsymbol{F}_{k,t}$ to be stable. Third, we choose the search space S, i.e. the space of possible values for (k, t), and a search policy to traverse S.

Once the execution of the instantiated OSFS algorithm starts, initial values for k and t are selected (line 1). It then enters a loop where, during each iteration, one point in the grid is evaluated (line 2-7). First, the feature set $\boldsymbol{F}_{k,t}$ is computed using the ranking algorithm A and available samples $\boldsymbol{X}_1$,..., $\boldsymbol{X}_t$. Then, the stability condition is evaluated for $\boldsymbol{F}_{k,t}$. If $\boldsymbol{F}_{k,t}$ is stable or the grid has been completely searched, the algorithm terminates and returns $\boldsymbol{F}_{k,t}$, k, t. Otherwise, the loop continues with the next (k, t) pair.

In the course of developing OSFS, we have studied and evaluated a range of feature selection and ranking algorithms, as well as feature set stability concepts [2]. For this demo, we instantiate OSFS with a ranking algorithm called Adapted Relevance Redundancy Feature Selection (ARR), which uses unsupervised feature selection and is based on the Relevance Redundancy Feature Selection (RRFS) method [4]. It uses two criteria for assessing the rank of a feature: (a) relevance, which relates to the distance of a feature vector to the mean of all feature vector vectors, and (b) redundancy, which relates to the cosine similarity between a feature vector and the vectors of all other features. High relevance and low similarity result in a high rank. Regarding the condition for feature set stability, we instantiate OSFS with a metric that is based on the set-similarity of two consecutive feature sets [2]. The search policy employed by OSFS traverses a rectangular grid of (k, t) pairs by proceeding row by row, from lower to higher k values. Within each row, the traversal evolves from smaller to larger t values (see Figure 3).

II. Testbed and Traces

Figure 2 outlines our testbed at KTH. It includes a server cluster, an emulated OpenFlow network, and a set of clients. The server cluster is deployed on a rack with ten high-performance machines interconnected by a Gigabit Ethernet. Nine machines are Dell PowerEdge R715 2U servers, each with 64 GB RAM, two 12-core AMD Opteron processors, a 500 GB hard disk, and four 1 Gb network interfaces. The tenth machine is a Dell PowerEdge R630 2U with 256 GB RAM, two 12-core Intel Xeon E5-2680 processors, two 1.2 TB hard disks, and twelve 1 Gb network interfaces. All machines run Ubuntu Server 14.04 64 bits, and their clocks are synchronized through NTP [5].

Two types of service are running on the testbed. *The VoD service* uses VLC media player software [6], which provides single-representation streaming with varying frame rate. It is deployed on six PowerEdge R715 machines —one HTTP load balancer, three web server and transcoding machines, and two network file storage machines. The load balancer runs HAProxy version 1.4.24 [7]. Each web server and transcoding machine runs Apache version 2.4.7 [8] and ffmpeg version 0.8.16 [9]. The network file storage machines run GlusterFS version 3.5.2 [10]. The VoD client is deployed in another PowerEdge R715 machine and runs VLC version 2.1.6 over HTTP.

The KV store service uses the Voldemort software [11]. It executes on the same machines as the VoD service. Six of them act as KV store nodes in a peer-to-peer fashion, running Voldemort version 1.10.22. The OpenFlow network includes 14 switches, which interconnect the server cluster with clients and load generators. The load generators emulate client populations.

A more detailed description of the testbed setup is available in [12].

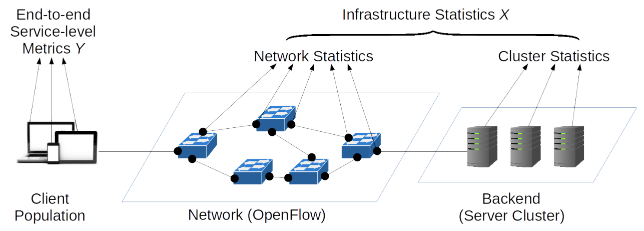

Fig. 2: The KTH testbed with data sources (statistics).

In this demo, measurements from the KV service under a flash-crowd load pattern are processed by OSFS. The trace contains 1723 features and 19444 samples [13].

III. Demonstration

We demonstrate the evolution of key metrics and data structures during the execution of OSFS when continually reading measurements from the monitoring system. Instead of reading from the testbed, the demonstrator reads from a trace file.

Figure 3 pictures the main graphics from the demonstration screen. The graphic on the upper left side shows the

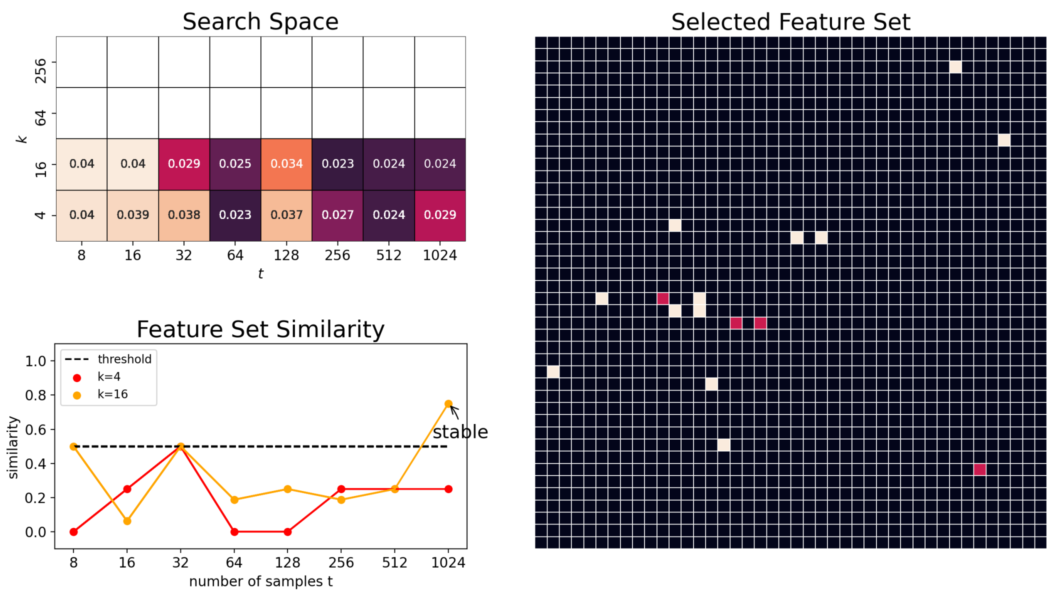

Fig. 3: Graphics from the demonstration screen at the time when OSFS terminates: the part of the search space that has been explored, the values of the similarity metrics of the feature sets that have been evaluated, and the stable feature set identified by the algorithm.

search space of the algorithm. It is a rectangular grid of (t, k) values which increase exponentially. As the algorithm processes, the grid is traversed from left to right and from bottom to top. At grid point $t = 1024, k = 16$, OSFS finds a stable feature set and terminates. For this reason, only half of the space has been explored. For illustration purposes, the graphic shows the prediction error of the model computed by OSFS with the current feature set $\boldsymbol{F}_{k,t}$. The error is given as Normalized Mean Absolute Error (NMAE) [2].

The graphic on the lower left side shows the values of the similarity metric for the feature sets $\boldsymbol{F}_{k,t}$. The red line gives the values for feature sets with 4 features, the yellow line for feature sets with 16 features. The algorithm identifies the feature set with $t = 1024, k = 16$ as stable.

The graphic on the right side shows the selected feature set for $t = 1024, k = 16$. Each of the 1723 features in the data trace is represented as a cell in the $42 * 42$ matrix. Features that are associated with a white or red cell belong to the selected feature set. The white cells indicate features which also belong to the preceding feature set that the algorithm identified. The current feature set contains 16 features. This means that OSFS has selected these 16 out of 1723 features for training the prediction model.

Acknowledgements

This research has been supported by the Swedish Governmental Agency for Innovation Systems, VINNOVA, through projects ITEA3 AutoDC and Celtic ANIARA, as well as by the Digital Futures project Democritus.

References

[1] F. S. Samani, H. Zhang, and R. Stadler, "Efficient learning on high-dimensional operational data," in *2019 15th International Conference on Network and Service Management (CNSM)*. IEEE, 2019, pp. 1–9.
[2] X. Wang, F. Shahab Samani, and R. Stadler, "Online feature selection for rapid, low-overhead learning in networked systems," *arXiv preprint*, 2020.
[3] X. Wang, "Osfs algorithm implementation for "online feature selection for rapid, low-overhead learning in networked systems" paper, cnsm 2020," 2021. [Online]. Available: https://github.com/Xiaoxuan-W/OSFS
[4] A. J. Ferreira and M. A. Figueiredo, "An unsupervised approach to feature discretization and selection," *Pattern Recognition*, vol. 45, no. 9, pp. 3048–3060, 2012.
[5] NTP, 2016. [Online]. Available: http://www.ntp.org/
[6] VLC, 2016. [Online]. Available: http://www.videolan.org/vlc/
[7] HAProxy, 2016. [Online]. Available: http://www.haproxy.org/
[8] Apache HTTP Server, 2016. [Online]. Available: http://httpd.apache.org/
[9] FFmpeg, 2016. [Online]. Available: https://www.ffmpeg.org/
[10] Gluster FS, 2016. [Online]. Available: http://www.gluster.org/
[11] Voldemort, 2016. [Online]. Available: http://www.project-voldemort.com/voldemort/
[12] R. Stadler, R. Pasquini, and V. Fodor, "Learning from network device statistics," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 672–698, 2017.
[13] F. Shahab, "Data traces for "efficient learning on high-dimensional operational data" paper, cnsm 2019," 2018. [Online]. Available: https://github.com/foroughsh/CNSM2019-traces