Feedback-based Learning for Self-Managed Network Elements

Panagis Magdalinos, Apostolos Kousaridas, Panagiotis Spapis, Giorgos Katsikas, Nancy Alonistioti

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece

{panagis, akousar, pspapis, katsikas, nancy}@di.uoa.gr

Abstract— **Autonomic network management systems will operate in a volatile network environment; thus they should be able to continuously adapt their decision making mechanism through learning from the behavior of the communication system. In this paper, a novel learning scheme is proposed based on the network-wide collected performance experience, targeting the enhancement of network elements' decision making engine. The algorithm employs a fuzzy logic inference engine in order to enable self-managed network elements faults or optimization opportunities identification, which is enhanced by applying data mining techniques on the accumulated observations.**

Keywords-component; Network Self-Management, Future Internet, Fuzzy Logic, Data Mining

I. INTRODUCTION

The majority of autonomic systems are based on Monitor-Decide-Execute Cycle (MDE) [1], [2]. Each autonomic element consists of the autonomic manager (AM) that instantiates the MDE cycle and the respective managed resource. The AM monitors resources' state and builds a knowledge model that is used in conjunction with the monitoring data in order to analyze the current status of the resource and decide the best action. Cognition development is an important aspect of future Internet self-managed systems which complements and advances the automation of actions. An in-network cognitive cycle will allow a communication system to improve its inference and reasoning capabilities, by exploiting feedback from previous events that are stored locally [3].

Future Internet self-managed networks shall incorporate the necessary algorithms that will facilitate the network administrator and thus render each network element capable to learn from previous adaptations (configuration actions) by assessing their effectiveness. There are many discussions on the applications of machine learning schemes and on the introduction of the learning capabilities, from an architectural point of view [4], [5]. However, the majority of the machine learning algorithms for communication systems evolution are targeting a specific network environment or a specific application e.g., traffic management [6], radio management [7].

The scope of this paper is to propose a holistic feedback-based learning scheme for an autonomic network management system, which has been designed following the principles of a hierarchical architecture of cognitive managers placed per network element as appearing in the Self-NET project [8] architectural framework.

Each learning scheme affects and is affected by the decision making engine of the cognitive managers for situation perception and/or configuration action selections. In this paper we focus on enhancing the quality of a fuzzy logic-based decision making engine. The latter is accomplished through a two step procedure. Initially we develop the knowledge base of a network node's experience by exploiting the merits of fuzzy logic in conjunction with the benefits of the *k Nearest Neighbor* (*k*-NN, [9]). Then, network wide accumulated information is used as input for the enhancement of the decision making process through a novel learning framework based on high dimensional Euclidean geometry and *k-Means* data mining algorithm [9].

II. FEEDBACK-BASED LEARNING ALGORITHM

The identified algorithm consists of three distinct steps, namely *monitoring*, *classification* and *fuzzy logic enhancement*. Throughout the presentation we assume that every network device is managed by a Network Element Cognitive Manager (NECM) and periodically monitors its operational environment. Each observation comprises a *d*-dimensional vector which can be classified as *True*, indicating that a particular problem has appeared, *False* –no problem- or *Neutral*, implying that although there is currently no problem there is a chance that a problematic situation may appear in the future. Additionally, given a problem, the device's NECM triggers a remedy action which is guaranteed to solve the problem; in other words it will enable the device to transit from a *True* state to either a *False* or *Neutral* state.

Each extracted tuple is evaluated by set of pre-installed fuzzy logic rules and attributed a label X_i which stores the result of the procedure. If X_i denotes a problematic situation then a predefined solution is applied. Given the fact that we assumed that the applied solution will always solve the

problem we can compare the i+1th tuple (i.e. $t_{i+1}^{\rightarrow}$) with the ith (i.e. $t_i^{\rightarrow}$) thus assessing the validity of the procedure. The comparison is done through the Euclidean distance metric; in case their distance is less than a predefined bound ε (i.e. $\|t^{\rightarrow}_{i+1}-t^{\rightarrow}_{i}\|<\varepsilon$) then we performed an incorrect classification. Thus we attribute the correct label Y_i that corresponds to the actual conditions (ground truth) (i.e. **if** X_i=True **and** X_{i+1}=True **and** $\|t^{\rightarrow}_{i+1}-t^{\rightarrow}_{i}\|<\varepsilon$ **then** Y_i = Neutral). In any other case, we cannot decide about the label and leave it as it is. As soon as a significant amount of vectors (i.e. *N*) is aggregated we halt the procedure and proceed with the application of the *k*-NN classifier.

The *k*-NN classifier enables the identification of all missing Y_i labels. The set of labeled instances (S) is used as the training set, while all unlabeled records (T) are used as testing set. Recall from the last step that although we can accurately predict the labels of all observations appearing in S, we only have tuples from *Neutral* and *True*. In order to overcome this, we add a small number of tuples which are in advance labeled as *False*. It should be pointed out that this step appears only the first time that the algorithm is executed. In subsequent executions, the training set is populated with previously labeled records.

Periodically the stored tuples are validated in order to provide an indirect assessment measure with respect to the quality of the fuzzy logic rules. The evaluation is done according to formula $A=\sum^{N}_{i=1}|X_i-Y_i|/N$. *A* essentially quantifies the percentage of cases that we made an erroneous decision (i.e. the ground truth label Y_i is different than the fuzzy logic label X_i) and is compared with a predefined tolerance bound A_p. If the number of mistakes is not tolerable ($A>A_p$) then the network element sends all data to the domain controller (Network Domain Cognitive Manager - NDCM). NDCM receives data from all network elements for which condition $A>A_p$ holds true. All measurements lay in a *d*-dimensional space and by exploiting the ground truth labels (Y_i) we can categorize them in three distinct classes $C_i \in \{True, Neutral, False\}$ which correspond to three high dimensional manifolds (D_T, D_N, D_F). For ease of presentation, in the context of this work, we will assume that data points form hyperspheres, however the study can be extended to address a multitude of high dimensional manifolds.

Each sphere is centered at $CE_i = \Sigma_{j=1}^{|Ci|}S_j/|C_i|$, $S_j \in C_i$ and has radius $R_i=\max_{j=1}^{|Ci|}\|CE_i-S_j\|$. We assume that the cluster of tuples labeled as *True* is centered at $CE_T=(x_1, x_2, \ldots, x_d)$ and has radius R_T, while tuples corresponding to *Neutral* are centered at $CE_N=(y_1, y_2, \ldots, y_d)$ with radius R_N. Similarly, *False* points are centered at $CE_F=(z_1, z_2, \ldots, z_d)$ with radius R_F. Without loss of generality we consider only points CE_T and CE_N. These two points define a line ε which is described by the set of equations:

$$p_i = x_i + u\,(y_i - x_i),\ i = 1 \ldots d \quad (2)$$

This line intersects with spheres D_T and D_N in four points which can be retrieved by substituting p_i values into the following hypersphere equations:

$$D_T \rightarrow \sum_{i=1}^{d}(p_i-x_i)^2=R_T^2 \quad (3)$$

$$D_N \rightarrow \sum_{i=1}^{d}(p_i-y_i)^2=R_N^2 \quad (4)$$

Consequently, a simple way of identifying the bounds for the fuzzy logic rules would be to extract the intersection points which i) belong to different hyperspheres and ii) exhibit minimum distance from each other. Simply stated, the two intersection points are provided by:

$$\{B_1, B_2\} = \min\{\|P_{1T}-P_{1N}\|, \|P_{1T}-P_{2N}\|, \|P_{2T}-P_{1N}\|, \|P_{2T}-P_{2N}\|\} \quad (5)$$

By applying a similar procedure we can also extract points B_3 and B_4 from spheres D_N, D_F. Notice at this point that each B_i corresponds to a tuple $\{x_1, x_2, \ldots, x_d\}$; thus each B_i can be directly set as a new bound for the fuzzy logic rules. More precisely: i) Point B_1 would correspond to the bound for the *True-Neutral* situation, ii) Point B_2 would corresponds to the upper bound for the *Neutral-True* situation (obviously $B_1=B_2$) iii) Point B_3 would correspond to the lower bound for the *Neutral-False* situation iv) Point B_4 corresponds to the bound for the *False-Neutral* situation (obviously $B_3=B_4$). A graphical representation of this procedure appears in Figure 1. where we demonstrate the application of the procedure in R^3.

The reader would have already noticed that these bounds offer poor discrimination quality, in the sense that a lot of *True* and *Neutral* situations as well as *Neutral* and *False* cases may have been placed together. The latter, is due to the fact that the hyperspheres provide only an abstraction of the actual manifold formed by the data points. In order to overcome this we employ a hierarchical divisive clustering (HDC [8]) approach based on *k*-means [8].

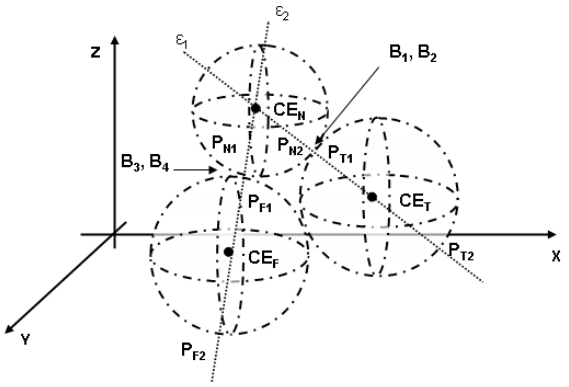

Figure 1. Graphical representation of data records according to ground truth labels

At first, k-Means is applied to the two spheres which correspond to *False* and *True* and direct the algorithm to split it into two clusters, *False/True* and *Neutral*. The result will be two adjacent spheres maintaining elements belonging to both classes. The division continues on the resulting *True* and *False* clusters until we start experiencing loss in the *Recall* (Recall = Retrieved Relevant Records / Total Relevant Records) or high *Precision* (Precision = Retrieved Relevant Records / Total Retrieved Records) in conjunction with high *Recall* (high F-measure value, where F-measure = 2*Recall*Precision / Recall + Precision). The geometric interpretation of our approach is depicted in Figure 2. The algorithm simply augments the sphere corresponding to *Neutral* cases and shrinks the other two by extracting falsely classified points. When the procedure is halted then we have three overlapping hyperspheres. The

intersecting points of the line defined by the spheres' centers with the spheres correspond to the desired solution.

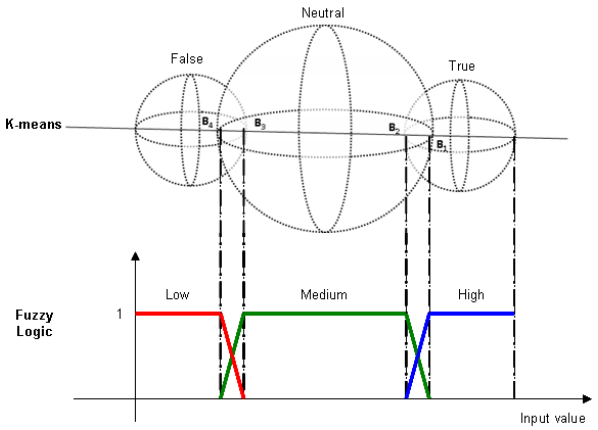

Figure 2. Geometric interpretation of the approach

We can quantify the amelioration induced by the algorithm by employing a simple procedure. Due to the fact that every network element holds a copy of the fuzzy logic rules, NDCM adapts its local rules and re-evaluates the whole set of observations. In the end, we quantify the amelioration induced by our algorithm by comparing the derived fuzzy logic labels with the existing ground truth labels. If the derived value is lower than the predefined bound A_p, we define a new stricter bound (e.g. A_p=0.8A_p), otherwise we consider than no further amelioration can take place. The new rules together with the new evaluation bound of the fuzzy logic membership functions are transmitted back to the underlying NECMs, where the local fuzzy logic inference engine is updated.

III. Performance Analysis

The performance analysis was carried out for the problem of load identification. In this use case, access points monitor their operational environment and attempt to identify potential (high) load situations. If such a problem appears then they collaborate in order to select the most appropriate configuration action, which in this case is the optimal reallocation of the associated terminals in the corresponding network area.

We used 50.000 tuples; each tuple was described by three variables indicating the status of an access point at time t_x, namely *Packet Error Rate (PER)* ranging in [0…1], *Channel Utilization (CU)* ranging in [0…1] and *Number of Associated Terminals (AT)* in [0…25]. 10% of the dataset has been sampled from the deployment of the testbed described in [11] and has been manually labeled (Y_i labeling) while the rest has been artificially generated according to the distribution derived from the initial set. The resulting dataset consists of 6667 tuples marked as *Load* (*True* according to the algorithm description of the previous section), 9956 marked as *No Load* (*False*) while the remaining 33377 correspond to the *Medium Load* case (*Neutral*).

In the first experiment we validate the ability of k-NN to support the derivation of the ground truth label for the first step of the algorithm. In order to accomplish that we experimented with various values for the number of nearest neighbors using the implementation provided by Weka [12]. Results have been assessed through a 10-fold cross validation procedure while all experiments verified our initial intuition regarding the applicability of k-NN in the context of our problem exhibiting a classification rate larger than 98%. Precisely, for kNN=1 we reached a correct classification rate of 98.7% while for kNN=5 a rate of 98.6% and finally for kNN=10, 98.5%.

At a second step, attempting to simulate a real life situation, we evaluated all observations with a pre-configured fuzzy logic decision making controller (FL). The rules used for the decision making procedure follow the format of equation (6) while membership functions have been configured according to the bounds appearing in TABLE I. [1].

IF ***PER IS low*** AND ***CU IS low*** AND ***AT IS High*** THEN Interference IS low, ***Load IS low*** (6)

TABLE I. Bounds used in the membership functions of the fuzzy logic rules

	PER	CU	AT
No Load	FL:[0…0.01]	FL: [0…0.2]	FL: [0…10]
Medium	FL: [10^{-4}…0.05]	FL: [0.1…0.9]	FL: [9…16]
Load	FL: [7*10^{-3}…1]	FL: [0.8…1]	FL: [15…25]

An interesting outcome after observing the original dataset is that the results will be heavily influenced by the values of the *AT* parameter while on the other hand *PER* will provide little information in the clustering process. The latter is due to the fact that these variables are in different scale. Indeed, $AT \in N$ takes values from the range [0…25] while $PER \in R$ and specifically in [0…1] with the majority of its values concentrated in [0…0.015]. In order to overcome this issue we choose to normalize the values of PER and AT through formulas (7) and (8) respectively.

$$PER_i = PER_i / \max_{j=1}^{N}(PER_j),\ i = 1...N \quad (7)$$

$$AT_i = AT_i / \max_{j=1}^{N}(AT_j)\ i = 1...N \quad (8)$$

Figure 3. presents the results after executing the algorithm on the dataset. The algorithm uses the decision obtained from the fuzzy logic controller and divides the input tuples into three classes according to the identified state of the network element (i.e. *Load, Medium Load, No Load*). Then the tuples identified as *Load* are used as input to the algorithm that iteratively divides the aforementioned tuples into two clusters, *Load* and *Medium Load,* until the halting condition is satisfied; the same procedure is being held for the tuples identified as *No Load*.

When the halting condition is validated we acquire a good approximation of the actual sets while any potential overlapping corresponds to the intersection of the membership functions. The bounds obtained from these experiments appear

[1] The full dataset, accompanied by the source code and a description of the testbed is available at http://kandalf.di.uoa.gr/IM2011/

in TABLE II. Using the derived points as the new bounds for the membership functions we apply the validation algorithm and obtain a classification accuracy of 76.73% which induces an amelioration of 16.8% with respect to the original results of 65.64%.

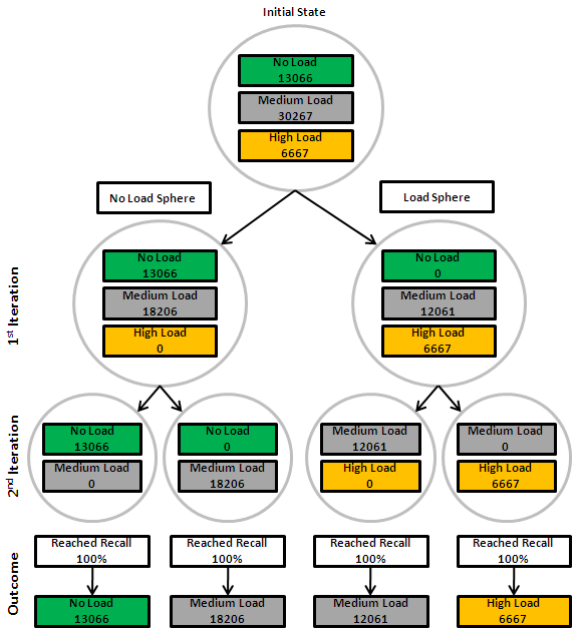

Figure 3. Dendograms derived after applying our algorithm on the original dataset.

Based on the experiments we conclude that the algorithm performs significantly well and tends to provide rules, which converge to decisions closer to the ground truth, independently of the initial configuration of the network element's decision making engine. Indeed the algorithm induced an amelioration of almost 17% compared to the original configuration.

IV. CONCLUSIONS

In the context of this paper a novel solution, for enhancing the decision making capability of self-managed network elements is proposed. In a Future Internet environment, the network elements will be required to operate in diverse and volatile environments. Consequently additional operational effort is required by network operators and/or manufacturer, since static a priori deduction mechanism and policy deployment is not enough. The proposed feedback solution emerges as a viable answer to the aforementioned problem, enabling network elements to operate having acceptable success rates in their decision making mechanisms. Its success lays in the combination of fuzzy logic with traditional data mining techniques, a key factor for enabling the decision making mechanism to adjust its operation to a constantly changing network environment. The experimental analysis showcases a significant amelioration in the classification ability of NECM, which with adequate preprocessing reaches the scale of 17%. Future research will concentrate on incorporating more complex high dimensional manifolds in the original algorithm, introducing more complex clustering algorithms for HDC as well as transitioning to a fully autonomous, unsupervised system with no built in rules of other kind of knowledge.

TABLE II. THE BOUNDS EXTRACTED FROM K-MEANS AFTER CLUSTERING ON THE NORMALIZED DATASET

	PER	CU	AT
No-Load	[0 ... $3.2*10^{-3}$]	[0 ... 0.375]	[0... 5.23]
Medium	[$2.87*10^{-3}$ ... $1.17*10^{-2}$]	[0.357 ... 0.756]	[4.29 ... 16.18]
Load	[$1.16*10^{-2}$ ... 1]	[0.747... 1]	[15.89 … 25]

ACKNOWLEDGMENT

This work is supported by the European Commission Seventh Framework Programme ICT-2008-224344 through the Self-NET Project (https://www.ict-selfnet.eu).

REFERENCES

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing. Computer", Vol. 36, Issue: 1 pp. 41–52, 2003.

[2] Simon Dobson, et al., "A survey of autonomic communications", ACM Transactions on Autonomous and Adaptive Systems (TAAS), vol. 1 , Issue 2, pp. 223 – 259, 2006.

[3] J. Mitola, Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio, Ph.D. dissertation, KTH, 2000.

[4] Ryan W. Thomas et al , "Cognitive networks: adaptation and learning to achieve end-to-end performance objectives," Communications Magazine, IEEE , vol.44, no.12, pp.51-57, Dec. 2006

[5] Thomas G. Dietterich and Pat Langley, "Machine Learning for Cognitive Networks: Technology Assessment and Research Challenges" In Q. Mahmoud (Ed.), Cognitive Networks: Towards Self-Aware Networks. New York: John Wile, 2007.

[6] Soysal, M. and Schmidt, E. G. 2010. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. Perform. Eval. 67, 6 (Jun. 2010), 451-467

[7] Bagnasco, R.; Serrat, J. Multi-Agent Reinforcement Learning in Network Management. "3rd International Conference on Autonomous Infrastructure, Management and Security". Springer, 2009, p. 199-202.

[8] Self-NET Project, https://www.ict-selfnet.eu

[9] Jiawei Han and Micheline Kamber, "Data Mining: Concepts and Techniques", ISBN 1558604898, 2007

[10] A. Kousaridas, et al, "An experimental path towards Self-Management for Future Internet Environments", Book Chapter In: "Towards the Future Internet - Emerging Trends from European Research" [ISBN 978-1- 60750-539-6], 2010.

[11] P.Magdalinos et al "Coverage and Capacity Optimization of Self-managed Future Internet Wireless Networks", in ServiceWave, pp 201-202, 2010