How hyper-dimensional space can help automation systems to be smarter?

Evgeny Osipov
Luleå University of Technology, Campus Porsön, SE-97187 Luleå, Sweden
Tel: +46 920 49 15 78, E-mail: Evgeny.Osipov@ltu.se

Denis Kleyko
Luleå University of Technology, Campus Porsön, SE-97187 Luleå, Sweden
E-mail: Denis.Kleyko@ltu.se

Nikolaos Papakonstantinou
VTT Technical Research Centre of Finland, FI-02044 Espoo, Finland
E-mail: Nikolaos.Papakonstantinou@vtt.fi

KEY WORDS: Distributed representation, cognitive computing, fault detection.

ABSTRACT

This article introduces the usage of a biologically inspired information representation, called *distributed data representation in hyper-dimensional representation space* and a computing technique called *cognitive computing in the context of industrial automation.* The distributed data representation and the mathematical framework of cognitive computing have previously been used for semantic analysis of natural languages and symbolic reasoning for enabling online learning and reasoning. Specifically, two illustrative use cases discussed here are a.) Representation of time series (e.g. for processing of inputs from various sensors) for in-sensor classification applications and b.) An extension to IEC 61499 based multi-agent automation system for an intelligent, biologically-inspired fault detection in generic complex systems-of-systems.

1 INTRODUCTION

Distributing the intelligence across networked hardware devices is one of the main current trends in industrial automation development towards enabling flexible and intelligent industries. Specifically, condition monitoring and intelligent maintenance of industrial systems is becoming an increasingly important function of automation systems. By intelligence in modern industrial processes often understood an ability of the automation system to adapt to the changing environment conditions through self-configuration and reconfiguration as well as its ability to deduce simple logical relationships given the input and output data. In Section 3 an elaborated definition of intelligent automation is presented.

Talking about intelligent functions, the main shift of the paradigm observed over the past ten years concerns the transition from traditional artificial intelligence (AI) methods for data analysis and visualization, which require substantial manual work and knowledge engineering, to purely online autonomous learning of streaming data. The core idea with the online learning is to create an online model of temporal patterns in data streamed by multiple heterogeneous sources (sensors, actuators, controllers, etc.). In this way the model will evolve with the evolution of the underlying process. Such modeling method would address one of the main challenges of the current control systems – the outdating of underlying plant models – and in turn enable more efficient fault management.

This article introduces the usage of a biologically inspired information representation, called *distributed data representation in hyper-dimensional representation space* and a computing technique called *cognitive computing in the context of industrial automation.* The distributed data representation and the mathematical framework of cognitive computing have previously been used for semantic analysis of natural languages and symbolic reasoning for enabling online learning and reasoning. Specifically, two illustrative use cases discussed here are a.) Representation of time series (e.g. for processing of inputs from various sensors) for in-sensor classification applications and b.) An extension to IEC 61499 based multi-agent automation system /1/ for an intelligent, biologically inspired fault detection in generic complex systems-of-systems.

The article is organized as follows. The fundamentals of the cognitive computing framework and data representation in hyper-dimensional space are overviewed in Section 2. Section 3 presents an application of the mathematical framework for representing time series and illustrative operations. Section 4 presents the design of the biologically inspired fault detection methodology and its application in the context of the IEC 61499 based automation system. Section 5 concludes the article.

2 HYPER-DIMENSIONAL DISTRIBUTED REPRESENTATION AND THE FRAMEWORK OF COGNITIVE COMPUTING

The fundamental difference between distributed and traditional (also called *localist*) representations of data is as follows. In traditional computing architectures each bit and its position within a structure of bits are significant. For example a field in a database has a predefined offset amongst other fields and a symbolic value has unique representation in ASCII codes. In the distributed representation all entities are represented by binary random vectors of very high dimension also called Binary Spatter codes /2, 3/ Further in the article for brevity reasons HD-vector term is used when referring to hyper-dimension vectors. Hyper-dimensionality means here several thousand of binary positions for representing a single entity. In /3/ it is proposed to use vectors of 10000 binary elements.

2.1 Similarity metric

A similarity between two binary representations is characterized by Hamming distance, which measures the number of positions in two compared vectors in which they are different:

$$\Delta_H(A,B) = \frac{\|A \oplus B\|_1}{d} = \frac{\sum_{i=1}^{d} a_i \oplus b_i}{d}$$

where a_i, b_i are bits in *ith* position in vectors **A** and **B** of dimension d and $\oplus$denotes the bit-wise XOR operation.

2.2 Randomness

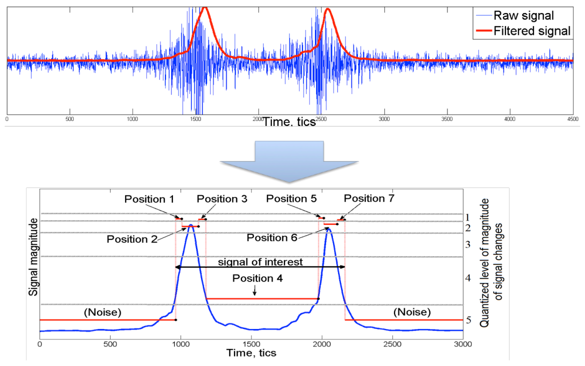
Randomness means that the values on each position of an HD-vector are independent of each other, and "0" and "1" components are equally probable $p_1=p_0=0.5$. The statistical properties of these HD-vectors are described by the binomial distribution: $Pr(k,d,p) = \binom{d}{k} p^k (1-p)^{d-k}$. On very high dimensions d, the distances from any arbitrary chosen HD-vector to more than 99.99 % of all other vectors in the representation space are concentrated around 0.5 Hamming distance.

2.3 Generation of HD-vectors

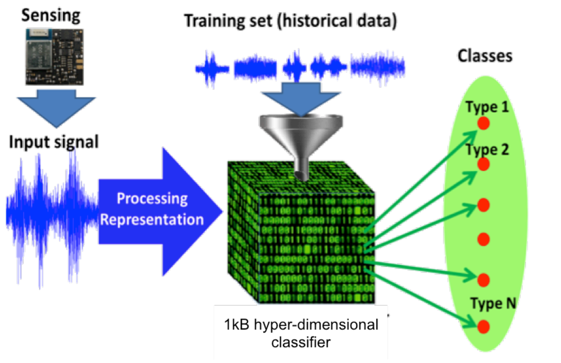
Random binary vectors with the above properties could be generated based on Zadoff-Chu sequences /4/ Using this principle a sequence of K pseudo-orthogonal vectors to a given initial random HD-vector A is obtained by cyclically shifting **A** on i positions, where $1< i \leq d$. Further in the article this operation is denoted as Sh(**A**, i).

2.3 Bundling of vectors

Joining several entities into a structure is done by a bundling operation. It is implemented by a (thresholded) sum of the HD-vectors representing entities. A bit-wise *thresholded* sum of n vectors results in 0 when n/2 or more arguments are 0, and 1 otherwise. Further term "majority sum" is used and denoted as [**A** + **B** + **C**]. The relevant properties of the majority sum are: The result is a random vector, i.e. the number of '1' components is approximately equal to the number of '0' components; The result is similar to all vectors included in the sum; Number of vectors involved into majority sum must be odd; The more HD-vectors are involved into majority sum operations the closer is Hamming distance between the resultant vector and any HD-vector component to 0.5. If several copies of any vector are included into majority sum the resultant vector is closer to the dominating vector than to other components. The algebra of hyper-dimensional representation includes also other operations e.g., binding and permutation. Since in the scope of this article they are not used the description of their properties is omitted.



a. The magnitude of a generic signal exhibiting certain pattern in time is quantized into a finite number of quantization levels after filtering out the noise.



Sensing
Training set (historical data)
Classes
Input signal
Processing Representation
Type 1
Type 2
Type N
1kB hyper-dimensional classifier

b. A 1 kB scalar is constructed using equation (1) for time efficient classification.

Figure 1. Application of hyper-dimensional representation for classification of temporal patterns.

3 Hyper-dimensional classifier of temporal patterns

This section describes an application of the cognitive computing framework for classification of temporal patterns in data from low-level sensors. The problem of pattern classification appears in many areas ranging from semantic analysis of natural languages to intelligent transportation systems. During past years a vast number of methods based on statistical analysis, the usage of artificial neural networks and others has been established /5/. Many of these methods require manual training and a heavy processing over large data sets. The proposed usage of the hyper-dimensional representation for classification of temporal patterns (i.e. continuous signals) is illustrated in Figure 1.

Consider a generic signal exhibiting a certain pattern in time illustrated in Figure 1a. The task of raw sensory signal conversion into *distributive*-represented pattern is formulated as to find and to quantify distinct changes in signal level and representing them using hyper-dimensional vectors. For example the signal in Figure 1a features changes at 9 positions. The first and the last positions correspond to the level of the ambient noise experienced by the sensor in the absence of external stimuli. The task is to extract a set of significant level changes. There are several standard methods for detecting the magnitude of the signal's change /5/. Due to space limitation their description is omitted in this article, however, for the sake of further presentation suppose that as the output such an algorithm produces a series of values of magnitudes of significant signal's changes. The values of magnitudes are then quantized into a finite number of discrete quantization levels as illustrated in Figure 1a. The number of the quantization levels is application domain specific and can be selected automatically without manual pre-engineering. For a collection of signals describing similar underlying generating stochastic process the obtained from the algorithm series of the change-magnitude values form similar patterns. Thus the signal exemplified in Figure 1b features pattern: 1-2-1-4-1-2-1.

At the next stage each obtained pattern is encoded using distributed hyper-dimensional representation. For this purpose it is proposed to XOR the encoded levels into a single distributed representation as $1_{HD}\oplus2_{HD}\oplus1_{HD}\oplus4_{HD}\oplus1_{HD}\oplus2_{HD}\oplus1_{HD}$, where a number with index "HD" is the hyper-dimensional code of the particular quantization level. However, it is easy to spot that under such encoding XOR'ing HD representations of the same level would cancel each other. In order to avoid this, it is proposed to derive the code of for the particular level from its initial vector and its place in the pattern as $Sh(L_i, p_j)$. In this way the code for level L_i on the position p_j is obtained by cyclically shifting the initial vector for L_i on p_j positions. For further reasoning note that this representation of the entire pattern obviously results in a binary vector of the same dimension as the binary vectors for each component. In the case when vectors of dimension 8000 bits are used for encoding (which is a reasonable dimension exhibiting the properties stated in Section 2) each pattern will be represented by a scalar of approximately 1kB in size.

The classifier is then constructed in form (1) following similar line of reasoning as in /6/. In (1) **Σ** is the operation of MAJORITY summation, P_j is a pattern of a signal represented as described above, T_i is a hyper-dimensional representation of the vector representing the class to which the signal from the learning set belongs to, K is the number of classes and N is the size of the training set per class.

$$CLASSIFIER = \left[\sum_{j=1..N} P_j \otimes T_i | \{i = 1..K\}. \right] \quad (1)$$

Note that due to bit-wise operations the size of the representation of the entire classifier is obviously the same as the dimension of the binary vectors for each pattern, i.e. in the case of encoding using 8000 bits binary vectors, the size of the classifier remain constant and equals approximately 1 kB.

When the classifier is constructed by (1) it is ready for its main operation. The main operation is formulated as testing an input (previously unseen) pattern (P) on inclusion within the hyper-dimensional classifier. The result of the classification is the class to which the input pattern belongs:

$$Type = P \otimes CLASSIFIER. \tag{2}$$

The result of the above operation is the noisy version of the HD-vector encoding the type, which the pattern belongs to. The remaining operation is therefore performing a Hamming distance test with the clean versions of the type-representing HD-vectors:

$$\begin{cases} \Delta_H(Type, T_1) = \|Type \otimes T_1\|_1; \\ \Delta_H(Type, T_2) = \|Type \otimes T_2\|_1; \\ \Delta_H(Type, T_N) = \|Type \otimes T_N\|_1. \end{cases} \tag{3}$$

The value of the Hamming distance test (3) less than 0.5 indicates the correct type of the input pattern. The presented in this section classification scheme using hyper-dimensional distributed data representation was tested a real-life scenario of the classification of the car passages collected from a vibration sensor installed on the road surface in the university's testbed of Intelligent Transportation System. The details of the performance evaluation are reported in /7/. In summary, for the dimensions of the hyper-dimensional vectors discussed (1kB) the classifier can store up to 100 unique samples while exhibiting 100% classification accuracy. This result is of a practical interest in the scenarios with low variability in the patterns. In this case the classifier and the classification operations (self-learning, classification) could be deployed directly on a low-power sensing device. As part of the evaluation the classifier was implemented on a low-end sensor platform featuring 16-bits microcontroller. In this implementation one classification operation took approximately 130 milliseconds.

4 BIOLOGICALLY INSPIRED FAULT DETECTION

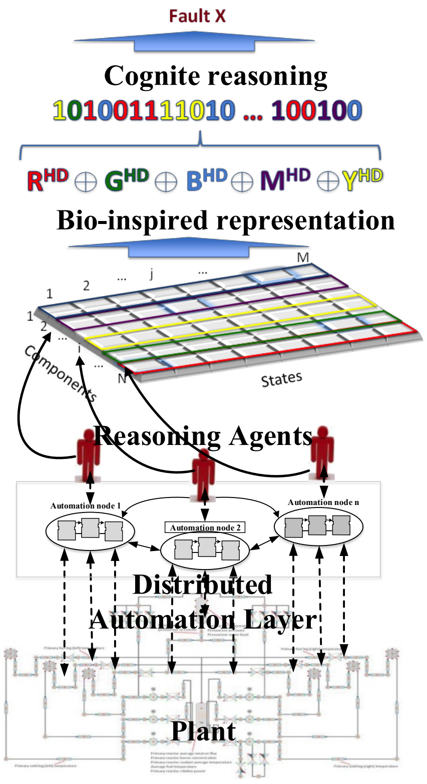


Figure 2. A distributed automation system enhanced with the autonomous intelligence capabilities based on the usage of multi-agent technology and distributed hyper-dimensional binary representation.

In the proposed architecture it is assumed that a complex system is equipped with a component based automation system built, for example, using the IEC61499 standard /8/. In this way the whole system is viewed as a collection of interacting components (or functional blocks), each of which could be characterized by its state encountering at the specific moment in time. Figure 2 demonstrates a high-level system model and the proposed solution.

The hyper-dimensional representation is applied to encode the components states. Referring to Figure 2, each component is assigned a system-wide unique initialization hyper-dimensional vector, denoted as IV. Without loss of generality suppose that the state of each system's component changes over time in discrete steps. Suppose also that the value range of the component's state is finite, bounded. Thus the state of a component at time t is encoded by cyclically shifting the initialization vector for this component by the number of steps corresponding to the index of the current state's value i in the value range for this particular state, i.e. $STATE_{j,i}^{HD} = \mathrm{Sh}(IV_j, i)$.

The aggregated system state is constructed using the historical data by encoding states of all components leading to the particular fault into a single *distributed* representation: $SYS_{FAULT1} = \left[\sum_{j=1}^{N} STATE_j^{HD}\right]$, where N in this case is the number of system's components and $STATE_j^{HD}$ is the state of component j encoded using hyper-dimensional distributed representation. The state of each component is encoded by cyclically shifting the initialization vector for this component. In the case of several historical cases leading to the particular fault i, the corresponding aggregated system states for this fault are bundled together as: $SYS_{FAULT^i} = [SYS^1_{FAULT^i} + SYS^2_{FAULT^i} + \cdots + SYS^M_{FAULT^i}]$, where M is the number of encountered cases leading to fault i.

Fault identification in the proposed architecture

The fault identification procedure has its foundation in the statistical properties of the hyper-dimensional random binary vectors used for encoding states of system's components and the similarity property of the majority sum is used for bundling of several vectors together. The joint system state SYS_{FAULT^i} is an associative memory of all combinations of system-wide states, which characterize the particular fault. The fault identification is, therefore, performed by testing the inclusion of the current pattern of system states with holographic representations for different faults SYS_{FAULT^i}. This is done by computing the Hamming distance $\Delta_H(SYS_{Current}, SYS_{FAULT^i})$, between the current system state $SYS_{Current}$ and the state of all possible faults SYS_{FAULT^i}. The closer this metric to 0.5 for the particular fault the less likely is that the current state is an indication of this fault.

Performance benchmarking with related approaches

The case study used for demonstrating the proposed cognitive fault detection system is a generic nuclear power plant model provided by Fortum Power and Heat, a power utility with nuclear power plant operation license in Finland. The Apros 6 process simulator is used to run the model. Apros 6 is a dynamic process simulator owned by the VTT Technical Research Centre of Finland and Fortum. Two main process loops are used for power generation using nuclear energy, the primary and secondary circuit. The primary circuit contains the reactor vessel and the nuclear fission within the fuel generates thermal energy, which heats the water in the vessel. The coolant pumps in the primary circuit circulate water through the steam generators and the reactor vessel and thus thermal energy is transferred from the primary to the secondary circuit. The pressurizer, also part of the primary circuit, is a vessel is partially filled with water and it is designed for pressure regulation using heaters and water sprays. The secondary circuit is connected to the primary through the steam generators. Heat from the primary circuit converts water flowing in the secondary side of the steam generators into steam. Turbines use the high-pressure steam flow and drive electric generators. Condensers are used to convert the low-pressure steam after the turbines back to water.

Data sources

The functional failure identification and propagation framework was used to analyze 116 automation components as the sources of hardware fault /9/. Most of these components were pump and valve actuators. For each automation component type three specific failure modes were chosen (e.g. a valve actuator can be set to the "failed open", "failed closed" or "no electric supply" fault mode which will result in the opening, closing or stop controlling the valve). A component – failure mode pair defines a fault in this paper (e.g., "Valve valveID" – "Fails - Open"). From the 348 total possible faults (116 components x 3 failure modes per component), 92 faults actually affected the steady state operation of the power plant model and thus can be detected by a data driven fault detection system.

In the case study the model was driven to 11 power production levels in order to get more simulation data for the set of faults. The 92 faults, which are detectable in the steady state of the plant, are simulated for every one of the 11 power levels (for a total number 1012 simulations). The results of these simulations were used to build data sets for training and testing the fault detection systems. Each simulation (fault - power level) contains 180 seconds of simulation time. Data set entries are generated by 37 monitored simulation signals. Three statistical values are generated per signal leading to a 111 inputs per entry (37signals x 3statistical values per signal). The data sets had a total of 1012 entries (same as the number of simulations), each with 111 inputs (generated by the 37 logged signals) and at the end every entry contains the classification attribute.

Reference technologies and selected results of benchmarking

The performance of the proposed approach was compared to the performance of a multilayer perceptron Artificial Neural Network (ANN), a decision tree, and a K-nearest-neighbors (KNN) classifier. The WEKA tool and MATLAB were used to benchmark the accuracy of the fault detection of the proposed architecture. Note that the decision tree and ANN produce a single prediction for a given input. The KNN classifier and the proposed approach produce a ranked list of possible results. During the benchmarking process an accuracy of the fault identification was assessed. The accuracy was measured as the ratio of the correctly identified faults over all presented cases from the test set.

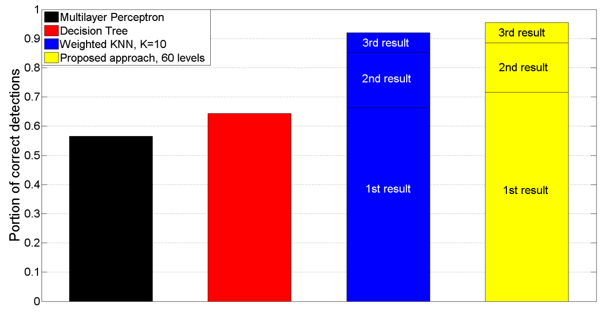


Figure 3. The results of benchmarking: ANN, decision tree, KNN and the proposed approach.

To the best of the authors' knowledge benchmarking machine learning techniques operate only in the centralized mode. That is the data set presented for the fault detection should have the same structure as the training data set. In this article a zero knowledge about the underlying component interdependencies is assumed. Therefore all the compared approaches were trained by presenting the state values from all system components.

The results of comparison of the considered approaches are shown in Figure 3. The main result indicated by the figure is that the accuracy of the fault identification by the proposed approach outperforms the other approaches even when considering the top first result. Namely the best ANN's accuracy index was nearly 0.6; the accuracy of the decision tree was 0.65. The best accuracy was demonstrated by both KNN classifier and the proposed approach (0.68 and 0.71 correspondingly). When considering top three results both KNN and our architecture show the accuracy above 0.9 with the accuracy of the proposed approach toping at 0.95. The presented above result is a very positive for the architecture. Showing either matching or in most of the cases superior performance over the traditional approaches our approach has one more unique advantage: it is suitable for the distributed implementation.

5 SUMMARY

This article introduced selected applications of a mathematical framework for cognitive computing in the context of industrial automation systems. The cases for classification of temporal patterns and fault detection in complex systems were considered. The main message delivered by the article is that the usage of distributed data representation and specifically its binary variant (binary hyper-dimensional codes) enables rather complex analysis directly on or close-to the resource-constrained nodes (controllers and sensors). Streamlining and automating the root cause analysis is on the research agenda since many years now. In /10/ the authors overview the progress of the event correlation techniques and provide a set of recommendations for future development of event correlation techniques in the context of system management. These recommendations include: a.) The next generation event-correlation systems must be able to deal with uncertain knowledge; b.) Better learning techniques to improve the accuracy of case-based systems; c.) Faster algorithms based on binary vector mapping which would convert a problem of correlating spatiotemporal events from complex cross-matching of "IF-THEN" rules into binary vector mapping operations. The authors of this article conjectures that applications of the biologically inspired computing framework presented in herewith has an excellent potential for addressing these challenges.

6 REFERENCES

1. Mousavi, V. Vyatkin: 2eA-FB: a semantic agent approach to IEC 61499 function blocks in energy efficient building automation systems, Automation in Construction, Elsevier, 2014, in print.
2. Gallant S. I. and Okaywe T. W.: Representing objects, relations, and sequences, Neural Comput., vol. 25, pp. 2038-2078, 2013.
3. Kanerva P.: Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors, Cognitive Computation, vol. 1, pp. 139-159, Jun 2009.
4. Kleyko D. and Osipov E.: On Bidirectional Transitions between Localist and Distributed Representations: The Case of Common Substrings Search Using Vector Symbolic Architecture, Procedia Computer Science, vol. 41, pp. 104-113, 2014.
5. Stork D, Duda R., Hart P.: Pattern Classification, John Wiley and sons, LTD, 2nd edition, 2000.
6. Levy S, Bajracharya S., and Gayler R.: Learning behavior hierarchies via high-dimensional sensor projection, In Learning Rich Representations from Low-Level Sensors: Papers from the 2013 AAAI Workshop.
7. Kleyko D. and Osipov E.: Brain-like classifier of temporal patterns, in Computer and Information Sciences (ICCOINS), 2014 International Conference on, 2014, pp. 1-6.
8. Vyatkin V.: IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review, Industrial Informatics, IEEE Transactions on, vol. 7, pp. 768-781, 2011.
9. Papakonstantinou N., Proper S., O'Halloran B., and Tumer I. Y.: Simulation Based Machine Learning For Fault Detection In Complex Systems Using The Functional Failure Identification And Propagation Framework, In proceedings of ASME CIE, Buffalo NY, USA, 2014.
10. Martin-Flatin J-P, Jakobson G., and Lewis L.: Event Correlation in Integrated Management: Lessons Learned and Outlook. J. Netw. Syst. Manage. 15, 4 (2007).