A Hierarchical Characterization of Knowledge for Cognition

Monte Hancock[1,3], Jared Stiers[2,3], Tyler Higgins[2,3], Fiona Swarr[2,3], Michael Shrider[2,3], and Suraj Sood[2,3(✉)]

[1] Digital, Los Angeles, USA
[2] Sirius19, Melbourne, USA
surajsoodx@gmail.com
[3] University of West Georgia, Carrollton, GA, USA

Abstract. A modestly formal systematic representation for "knowledge" is presented in the context of structured cognition. This representation places knowledge artifacts ("data", "facts", "rules", etc. to be defined later) into a hierarchy. This hierarchy aligns naturally with "stages" or "levels" often observed in intelligent biological and mechanical systems engaged in structured cognition.

Each level within the knowledge hierarchy consists of knowledge artifacts that arise as specific relations on the level prior to it. This establishes a recursive relational algebra by which knowledge at all levels can be specified in terms of percepts ("data") at the bottom. This is quintessential essentialism as a ground for cognition [1].

res est forma eius
" The thing is its form."

For the purposes of this work, an object of thought is regarded as equal to the assemblage of attributes it manifests. From the standpoint of cognition, nothing is sacrificed here, since percepts arising from this assemblage constitute the entirety of material available for structured cognition [2].

This formalizes a structured context for the analysis of cognition and the knowledge artifacts it uses. The U.S. Intelligence Community and the Department of Defense use similar but looser formalisms to support data fusion processes [3]. These are briefly described.

A brief case study is presented applying this knowledge representation to the analysis of an important pattern processing problem: the classification of distant military vehicles from their Doppler RADAR phase history.

Keywords: Knowledge model · Knowledge hierarchy · Cognitive model

1 Goals and Assumptions

Proposed here is an intuitive formulation, validation, and extension of a mathematical system within which high-fidelity models of a wide range of decision-support problems can be developed, assessed, and optimized. This optimization can be carried out on segments of a problem both individually, and in aggregate. By subjecting models

D. D. Schmorrow and C. M. Fidopiastis (Eds.): HCII 2019, LNAI 11580, pp. 58–73, 2019.
https://doi.org/10.1007/978-3-030-22419-6_5

derived within this formalism to mathematical analysis (e.g., visualization, modeling) rational solutions to real problems can be inferred in a principled, systematic way.

The essence of such a program will be a description of the fundamentals of the formalization to be created. This must include descriptions of representation of domain entities, methods for automated data preparation/processing, and a mechanism for automated reasoning. All of these must flow from the formal theory in a natural way.

2 Elements of the Approach

The customary approach to the analytic process is to employ analytic means to represent and prepare data, and discrete algebraic means for inferencing. This leads to a number of well-known difficulties:

The frame problem, reasoning that is inherently monotonic, super-polynomial inferencing time, and discrete inferencing that handles uncertainty as a "tacked on" afterthought.

We suggest that the representation be algebraic; the processing be geometric/topological; and, the inferencing be analytic/computational. This explicit choice of modalities allows principled formalization of the analytic process, and provides efficient satisficing solutions to these problems.

Representational foundation: an algebraic structure is proposed as the framework within which disparate domain entities are represented. This representation is applied so that

1. It makes extensive use of existing formalisms; we are not "starting from scratch".
2. Flexible abstraction is facilitated; the number and type of entity attributes made explicitly visible in the representation is adjustable.
3. The scope of entity definitions is labile; the representation allows an entity to be a signal sample, or a city.
4. It does not inherently limit the types of reasoning that can be applied to domain entities.
5. The representation is natural for machine implementation.
6. "Information" is an emergent property that arises through interpretation of entity relationships, rather than as context-free "magical contents" of individual, isolated entities. This means that relational structure is the basis of all domain knowledge, and geometric reasoning is the principal method by which it is derived.

Geometric toolkit: An interoperable collection of topological methods implemented in software is proposed for information extraction and refinement. These support

1. Parametric methods and unsupervised learning for the characterization of latent patterns.
2. Feature extraction, enhancement, evaluation, and winnowing.
3. User-centric, interactive high-dimensional visualization.

Inferencing mechanism: An analytic method is proposed as the means by which inferencing is performed. This allows

1. Linear-time non-monotonic reasoning, rather than the usual NP-Hard process required by graph-theoretic methods such as belief networks.
2. Declarative inferencing, that is, reasoning by direct computation. This makes knowledge "numeric", thereby facilitating the implementation of machine learning, as well as providing tractable approaches to the Frame Problem. Numeric knowledge is quantifiable, portable, and learnable by analytic methods (e.g., regression).
3. Natural representation, computation, and tracking of uncertainty, data pedigree, and conclusion confidence.

3 What Is Knowledge? Meta-Meta-Knowledge Models

We begin by defining a representation schema that is adequate to handle "knowledge" at every level of abstraction. The natural way to do this is to define knowledge according to a recursive method, so that, at the lowest level of abstraction, "knowledge" is nothing but data: fundamental percepts that are experienced and measured in the world. At higher levels of abstraction, "knowledge" posits relationships among entities that are at a lower level.

In this schema, "knowledge" exists at different formal levels. Advancement to the next level is recursive (from "knowledge" to "meta-knowledge" to "meta-meta…").

The simple relational algebra is best understood by considering an intuitive example:

Each level in the hierarchy will have a finite list of relation operators.

Level 0:
Relations: IS-A, ISNT-A (associates a percept with an attribute)

An element of knowledge at level = 0 would be a pair consisting of the null symbol and an attribute arising from some sensation or other measurement, such as a designation of time, place, or condition.

Example 1: 64 degrees.
Example 2: Tampa, Florida
Example 3: 2:00 p.m. today

At level 0, these attributes are not predicated of anything… they can be regarded as abstract properties.

Level 1:
Relations: all lower level relations, and some relational comparators and logical connectives (e.g., >, logical AND)

An element of knowledge at this level is a relation on data, such as the "ISA" relation (this datum ISA that datum), or any *wff* ("well-formed formula") in some predicate calculus.

Example 1: The temperature in Tampa, Florida today at 2:00 p.m. is 64 degrees.
Example 2: The temperature in Orlando, Florida today at 2:00 p.m. is 84 degrees.
Example 3: Power consumption in Florida increases with increasing temperature.
Level 2
Relations: all lower level relations, and Logical IMPLICATION.

An element of knowledge at this level is a relation on level 1 elements (such as a proof, or "argument"). For example

P1: The temperature in Tampa, Florida today at 2:00 p.m. is 64 degrees.
P2: The temperature in Orlando, Florida today at 2:00 p.m. is 84 degrees.
P3: Power consumption in Florida increases with increasing temperature.

Therefore,

C: The power consumption in Orlando, Florida is higher than the power consumption in Tampa, Florida at 2:00 p.m. today.
Level 3
Relations: all lower level relations, possibly others

This consists of what we normally call meta-mathematics, that is, the theory of formal systems.

Level k: an element of knowledge at this level is a relation on elements in levels 0 – $(k-1)$

These relations are formal associations among elements. If they are reflexive, symmetric and transitive, they are called equivalence relations. If they also preserve an arithmetic, they are called congruences. These can be represented as directed graphs and/or association matrices.

At levels above 0, it will probably be necessary to define operators that carry out transformations on elements (e.g., closures, proofs, resolution).

An expert system, for example, is just a collection of level 2 objects (rules) organized and executed by a level 3 object (an inference engine). In this way, we explicitly define what it means for a system to "know what it is doing": it is composed of level k objects, and has a level $k+1$ component to support learning.

General machine learning techniques would work naturally within this schema. There are learning paradigms that are appropriate to different levels. Manual learning can occur easily at levels 0 and 1, but automated techniques are probably required at higher levels. However, in a pure sense, each level is a formal relation, so all can be analyzed and represented using graph-theoretic methods.

As will be seen, black-box regression methods are just sophisticated ways of building what we will call level 1 ontologies.

4 The Representation Scheme as an Analogy

There is no such thing as formalism apart from representation. Formal reasoning in any domain requires an appropriate underlying symbology supported by inferencing machinery. Establishing a scheme capable of representing domain entities and their interactions is logically prior to everything else. This is where we begin [4].

The United States Intelligence Community (IC) has shown the way by its prescient attempts to create frameworks for such a scheme.

Two of these have gained wide acceptance in the IC. One, the NSA Reference Model, views the analytic process from a data-centric perspective. The other, the JDL Fusion Model, views the analytic process from a functional perspective. Placed side-by-side, they are seen to be similar attempts to represent the analytic activity in terms of a discrete hierarchy.

These hierarchies depict the systematic elevation of processing artifacts from mere phenomena (pure syntax) to cognitive artifacts (pure semantics). In this way, the analytic process winnows and refines sense experience, concentrating its latent value to produce actionable, decision-ready input for the human user [5].

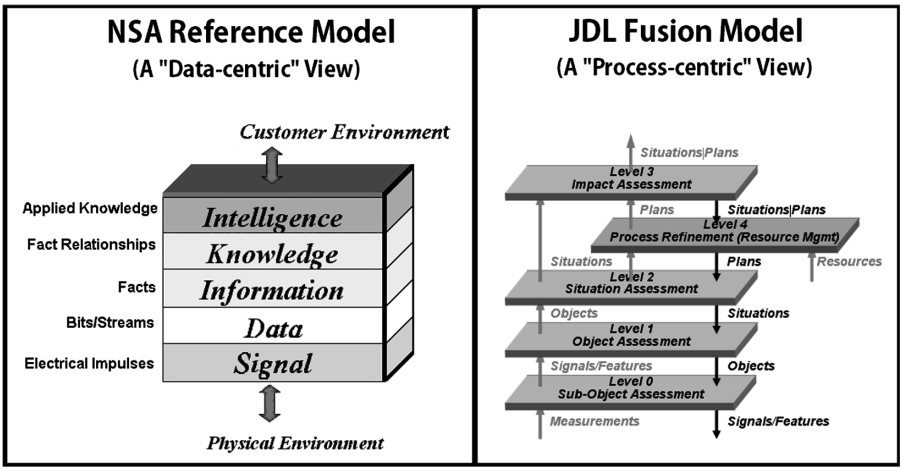

Fig. 1. Analysis viz. "data" and "process" according to the IC

These frameworks are notional depictions of what analysis does, but they are not formalizations. There is, however, one very important ingredient they share that makes them indispensable to any credible effort to formalize the analytic process: they were created by humans to describe what analysis conducted by humans is. A formalization congruent to these schemes will be comprehensible to humans.

5 Relation Towers

In Fig. 2 is a notional pictorial representation of a knowledge hierarchy.

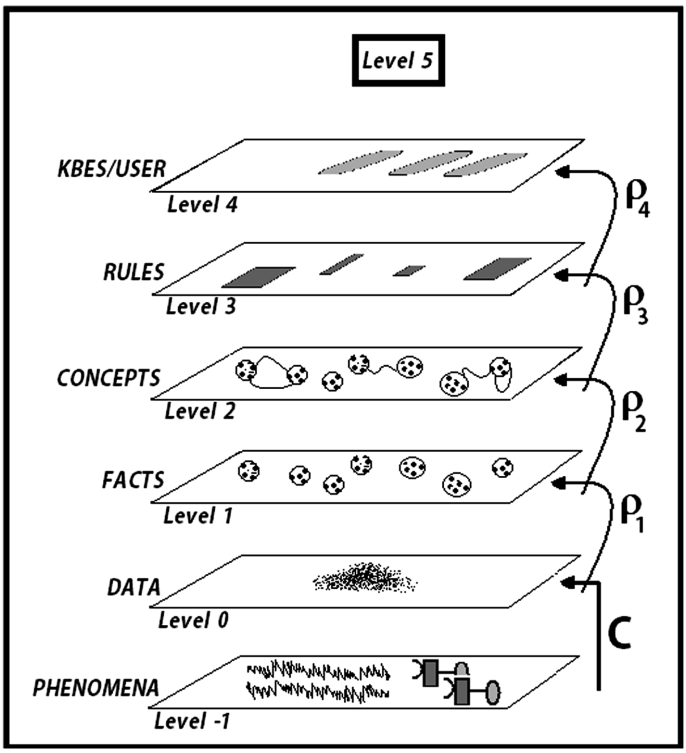

Fig. 2. Relation tower provides sufficient representation scheme.

We now describe the formalism for the representation scheme; it is depicted in Fig. 2.

Figure 2 shows a "tower" consisting of multiple layers. At layer -1 is the phenomenology of the real world. This is where waveforms, text, noumenal actors, and the phenomenology of reality reside.

At the next level above level 0, is *data*. Data consist of isolated measurements with no imposed context. They are depicted here as a collection of disconnected dots (a "sandbox"!). The next layer above that, level I, is the layer of *facts*. The next layer above that, layer 2, is the *concepts* layer. The next higher layer, layer 3, is the *rules* layer. Layer 4 is the reasoning layer.

The mathematical model proposed is based upon an established mathematical formalism.

The "lifting transform" that elevates entities through the successive layers of the tower is now described. It must not only accomplish the identifications with data, facts,

etc., it must do this in the same way for each layer transition. That is, the transform used to elevate entities from the data level to the fact the level should be mathematically equivalent to the transform used to elevate facts to concepts, then from concepts to rules, and finally from rules reasoning (KBES).

This allows the definition of the lifting transform to be varied a single layer at a time, perhaps with slight customization at certain levels. By recursive application, entities are elevated from the lowest level of phenomenology (individual data items) to layer 4 (system having multiple reasoners), within which the reasoning user resides.

The lifting transform that moves phenomena from level -1 to level 0, C, is conventional-technology data conformation. Conformation recasts disparate data and places them into a common context for manipulation, correlation, comparison, and analysis.

Conformation is not just "reformatting"; it must also accommodate variations in precision, timescale, space-scale, variation in units and representational schemes, and so on. This is a complex problem. It is assumed that conformation has been performed at layer -1, and data exist at layer 0 ready for the application of the formalism.

Moving from a level to the next higher level is a matter of imposing a relation on entities in the lower level to obtain entities at the next higher level. That is, in moving from data to facts, a relation is imposed on data.

For example, the isolated datum "72°" has no information content without context. Similarly, the isolated datum "2 PM", has no information-content out of context. When tied together and associated with a third datum, "Paris", the aggregate becomes a fact bearing latent information. This fact exists at level 1 as a relation on data items residing at level 0.

Figure 1 depicts level 1 facts as aggregations of data objects (drawn to suggest that they are "clusters of data"). In general, the number of entities will decrease at higher levels in the tower, since each upward motion results from aggregation of lower level entities [5].

Formally, a binary relation on a set S is a subset of $S \times S$ ($S \times S$ is the Cartesian product of S with itself).

Each ordered pair depicts an association from the first item to the second item. This can be depicted graphically as showing the two items as vertices in a directed graph: small icons connected by an arrow starting at the first element and terminating at the second.

Moving from level 1 to level 2 in the relation tower is done by aggregating facts, that is, by imposing a binary relation on layer 1. These digraphs are referred to as concepts. Layer 2 is the concept layer, where collections of facts about entities in the domain of discourse reside.

Moving next from level 2 level 3 is accomplished in the same way: by imposing a binary relation on the concept layer. Here there is an opportunity to inject some logical formalism by assigning types to the edges in the digraphs. If the associations are logical connectives (conjunction, disjunction, and implication), the lifting transform can generate well-formed formulas in propositional logic. A digraph of ANDed and OR'ed antecedent concepts followed by an implication is a *rule*.

Finally, at level 4, collections of rules are aggregated to obtain rule clusters. An appropriately selected collection of rule clusters constitutes the raw material for a

reasoner. Such systems are capable of inferencing within a domain, and when supported by the following

1. An appropriate non-monotonic formalism
2. A means of handling the frame problem
3. An adjudication logic

are able to perform automated cognition in a principled way.

By this process of imposing relations on lower-level objects to elevate entities to a higher level, the relation tower organizes entities of varying logical complexity within the domain of discourse in a principled way. Further, the same lifting transform is used at each level. After four applications of this lifting process, we have moved from the layer of uninterpreted phenomenology at level 0 to automated cognition, adaptive systems, and trainable software at level 4.

The relation tower does not describe how inferencing is conducted; rather, it provides a representation scheme within which inferencing will occur. There are many ways in which inferencing can occur in a structure of this sort. Because any mapping can be represented as a relation, the lifting transform is theoretically sufficient to support any kind of aggregation or connection one might want to impose on lower-level objects to obtain higher-level constructs.

5.1 Unification

The relation tower supports several important implementation considerations

1. Separate relation towers can be created for different problem spaces, areas-of-interest, or other decompositions of the analytic problem. This allows the creation of customized methods for specific problems; parallelization; and distributed computing.
2. More importantly, perhaps, is the fact that multiple towers can be unified at any desired level as a means to implement multi-level fusion.

Figure 3 is a notional diagram depicting the use of a relational tower to unify (fuse) two other towers at an intermediate level of processing

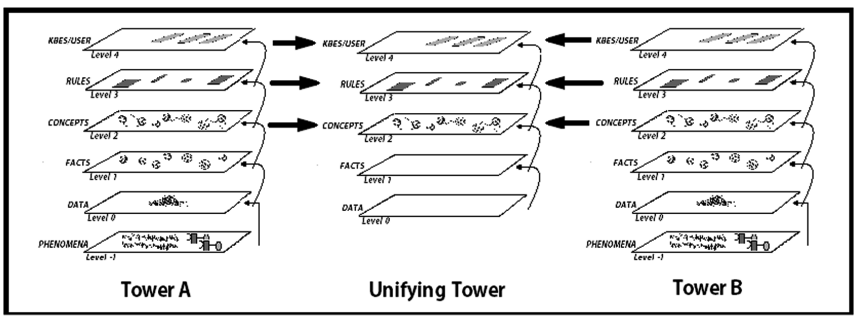

Fig. 3. Fusion can be implemented without developing additional algorithmics.

It is hypothesized that animal brains engage in parallel pattern association, matching percepts with sets of extracted image features previously annealed into plastic brain structures. Images are perceived to be members of the category which developed the associations giving the best match.

These plastic structures are emulated here by a collection of associative memories (AM's), each corresponding to a particular image category. Incoming percepts are matched against each AM, giving a matrix of responses. These responses are passed to a reasoning component (described below) responsible for adjudicating them to produce a classification to category for the image.

Markov Random Fields (MRF's) are used to implement the AM's. MRF's (certain implementations are referred to as Boltzmann Machines) are regressions that estimate binary system state variables from binary observables (e.g., condition true/false, present/absent). MRF's represent domain knowledge as a weight matrix $W = \{W_{ij}\}$ which quantifies the joint-likelihoods that binary state variables s_i and s_j are simultaneously "true". We use this matrix to impose a "soft" relation on the image features space to generate image responses and cross-responses for the image categories.

If we insist on exact matches in reasoning as above, the resulting inferencing scheme will be brittle. MRF's provide a natural remedy by replacing the crisp relation $r \sim s$ with a real-valued symmetric association weight matrix $W = \{W_{rs}\}$. The association weight W_{rs} will have a positive value when r and s usually occur together (are usually consistent), and will have a negative value when s and r usually do not occur together (are usually inconsistent). More precisely, when both $p(s|r)$ and $p(r|s)$ are large, W_{rs} will be positive; when both $p(not\ s|r)$ and $p(not\ r|s)$ are large, W_{rs} will be negative. Under this new definition, concepts are arbitrary sets of symbols whose elements generally have pairwise positive association weights, but might also include some element pairs having negative association weights. In keeping with the notion of crisp concepts above, it is customary to require W to be symmetric.

Unlike a simple cooccurrence matrix, W captures the relative strength of the associations of state variables in the context of the values assumed by all other state variables: it retains "context".

6 Constructing Markov Random Fields from Percepts

As described above, a MRF is a graph consisting of a finite number of fully interconnected vertices (called "units"). The edge from unit s_i to s_j has an associated real weight W_{ij}, where it is required that $S_{ij} = 0$ for all i, and $W_{ij} = W_{ji}$. Specifically, there are M units $s_1, s_2, \ldots, s_M$, and a real, symmetric matrix of interconnection weights $W = \{W_{ij}\}$, $i, j = 1, \ldots, M$, having zeros on the major diagonal. The units are binary; a unit is said to be active or inactive as its value is 1 or 0, respectively.

Certain units can be designated as inputs, and others as outputs. Units that are neither input nor output units are called processing units.

Inferencing in MRF's proceeds according to the following update rule

1. Assign the (binary) values of the input units ("input clamping")
2. Randomly select a non-input unit s_i. Sum the connection weights of the active units connected to s_i, and make s_i active if the sum exceeds a preselected threshold value t_i, else make it inactive.
3. Repeat step 2 until the unit values stop changing
4. Read off the (binary) values of the output units

In this scheme, positive weights between units are excitatory, and negative weights inhibitory: an active unit tends to activate units with which is has a positive connection weight, and deactivate units with which it has a negative connection weight.

Formal mathematical analysis of the inferencing procedure above is facilitated by the introduction of an energy function (due to John Hopfield): Let V be a vector of binary inputs. The energy of the MRF at $V = (b_1, b_2, ..., b_L)$ is

$$E_W(V) = -(1/2) \sum_{i \neq j} s_i s_j w_{ij+} + \sum_{i=1}^{M} S_i t_i$$

With this definition of network energy, Hopfield's updating rule is equivalent to the following: randomly select an unclamped unit; if toggling its state will lower the network energy, do so. Repeat until no single state change can lower the energy, then read the states of the output units.

The values of the input nodes are fixed, but the changes in the states of the processing and output units under updating cause changes in the energy of the network. This suggests that a supervised reinforcement learning algorithm called "simulated annealing" (first applied to Hopfield nets by Hinton and Sejnowski) can be used to train the network to associate desired outputs with given inputs

1. Assign the (binary) values of the input units ("input clamping")
2. Assign the desired (binary) values of the output units ("output clamping")
3. Randomly select a processing unit s_j
4. Set the state of s_i to active with a certain one.
5. Repeat step 3 for many epochs, allowing the network to relax to a local energy minimum
6. Once an energy minimum has been reached, step around the network, incrementing the weights between pairs of units that are simultaneously active (encourage future simultaneity!), and decrementing the weights between pairs having just one active unit (discourage future simultaneity!)

To avoid saturation of the weights, an additional updating sequence having both the processing and output units unclamped is often included.

Here T is a parameter (the *temperature*) that is gradually reduced as training proceeds. At low temperatures, the weight matrix is easily modified; as the temperature is reduced, the weights "lock" into their final values.

Simulated annealing develops a weight matrix that gives local minima in energy when the input pattern is "consistent" with the output pattern: the lower the energy, the

stronger the consistency. The interconnect weights, then, codify constraints on outputs the machine is likely to produce when the inputs units are clamped and updating is applied. Notice that the machine can give different outputs for the same input owing to the random element in the update rule. It can be shown that the probability of "incorrect" outputs can be made arbitrarily small (Fig. 4).

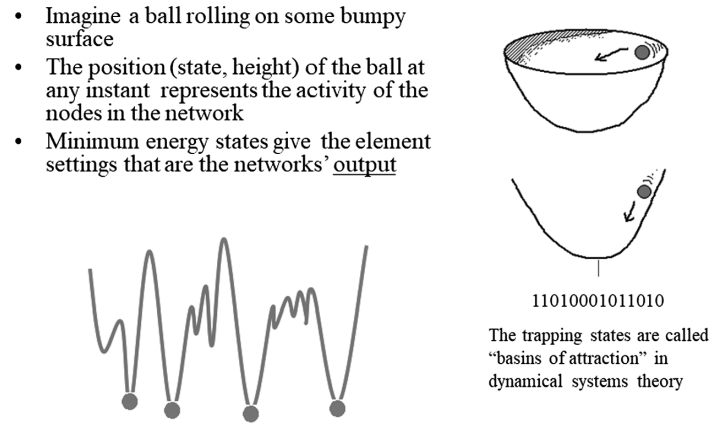

Fig. 4. The Hopfield Network adjust the strengths (both positive and negative) between related entities seeking a set of connection weights that willo enable the machine to match news patterns of activated elements with previsously learned patterns of activation. It does this by "falling into" a minimum energy configuration when learned patterns are present.

Parameters associated with training include the number of input, processing, and output units; the initial and final temperatures; how the temperature is to be reduced (annealing schedule); the number of random updates applied during each relaxation epoch; the number of relaxation epochs; the values of the thresholds; and the amount by which to increment/decrement the weights after relaxation is complete. As with most trainable systems, there are no universally applicable heuristics for these assignments.

7 Reasoning with Knowledge: An Executable Ontology

The term ontology here refers to information structures which enumerate the entities, concepts, and relationships that obtain in a domain. We introduce the notion of an executable ontology, within which these relations are made dynamic and actionable by being embedded in adaptive algorithms. A natural way to do this is to bind domain relationships in parameterized rulesets.

By adjusting the parameters based upon experience, the executable component of the ontology can be made adaptive and trainable. This overcomes the difficult problems associated with developing conventional knowledge-based systems, Bayesian belief networks, and the like.

Human experts typically do a fairly good job of assessing the impact, relevance, and quality of individual heuristics in isolation. However, when knowledge-based systems are placed into operation, it is the entire of mix of heuristics in context that determines the effectiveness of the system.

Even humans having deep domain expertise have difficulty understanding the complex interactions of a large number of rules. For this reason, the optimization of existing rules is best done by automation of information theoretic techniques. A gradient-based rule optimization strategy can be used to calibrate existing rulesets to perform reasoning tasks. This is described in the following.

7.1 Hopfield Networks

The Hopfield Network Architecture was proposed at Caltech in 1982 by physicist John Hopfield. A Hopfield Net is an artificial neural network that is able to store certain memories or patterns in a manner that is functionally similar to animal brains.

The nodes in the network are vast simplifications of real neurons - they can only exist in one of two possible states - *firing* or *not firing* (there are no input or output neurons).

Every node is connected to every other node with some strength (or weight) but not with itself; connections are symmetrical "association weights". At any instant of time a node will change its state (i.e. start or stop firing) depending on the inputs it receives from the other nodes.

There exists a rather simple way of setting up the connections between nodes in such a way that any desired set of patterns can be made a stable firing pattern – minimize its energy. Thus, any desired set of "memories" can be burned into the network at the beginning using an "annealing" process.

8 Case Study: Learn Patterns in the Phase Structure of Doppler RADAR Reflections from Military Vehicles

Doppler RADAR operates by computing the way in which the motion of a reflecting surface affects a stream of incident radio waves. If the earliest wave sent out indicates the same distance to the surface as subsequent waves, that surface must be stationary with respect to the RADAR. However, if later waves return earlier or later than earlier waves, the surface must be respectively, approaching the RADAR, or receding from it. The amount of this change gives a direct measure of the relative speed of surface and RADAR.

This is the principle by which Doppler Weather RADAR detects complex atmospheric movements: winds moving in different directions and at different speeds hasten or delay RADAR waves in a way that allows the detection of tornadic vortices, distance storms, wind-shear, and micro-bursts.

Suppose now that the RADAR wave are incident upon the body of a military vehicle. Such vehicles usually have parts that themselves move as the vehicle moves: wheels, tracks, antennas, guns, hatches, etc. These moving objects add their own

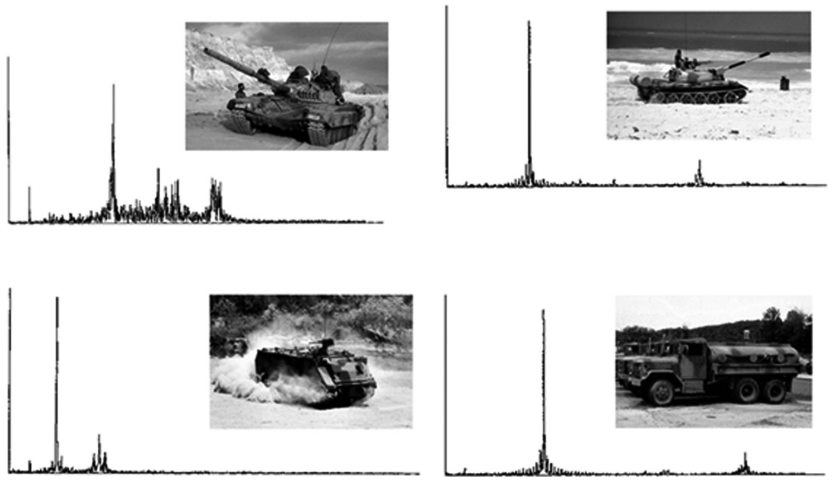

Fig. 5. The truck in the lower right-hand-corner is a low-value target… generally not of great concern, and usually not worth deploying expensive munitions to engage. The other three armored vehicles are high-value targets.

changes into a Doppler RADAR measurement, resulting in complex modulation of the radio waves incident on the vehicle.

If this modulation varies from vehicle to vehicle in a consistent way, it might be possible to infer the vehicle model from its Doppler RADAR modulation pattern. Distinguishing a convoy of trucks and a column of tanks from miles away offers clear tactical benefits.

When humans look at ragged waveforms, such as those in Fig. 5, they aren't drawn to the random-looking, disorganized stubble that constitutes most of the data; they naturally look for "clumps" that, taken together, tell a story. Humans can make sense of a few clumps of variable size and arrangement; they rest is inscrutable.

The difficulty with the vehicle classification problem, though, is that the "clumps" are not where the discriminating information lies.

The two vehicles on the left are both in the class "high-value targets"; yet their Doppler returns look nothing alike. The two vehicles on the right are in different classes; one high-value and the other low-value. Yet, their Doppler returns are virtually identical, "clumpologically" speaking.

The discriminating information, if it is present at all, is scattered across the random-looking stubble. It is distributed in a way, having cross-correlations of a sort that no human will ever be able to determine by visual inspection. What is needed is a method to determine which pieces of the Doppler RADAR trace being present/absent in combination (clumpy or not) that together indicate the vehicle model.

In other words, what is sought is the clique of energy peaks that is characteristic of each vehicle class.

This is exactly a relation. At level 0, there are bins for the various RADAR frequencies. Real numbers exist at level 0.

At Level 1, we will have a set of "facts" that link each RADAR bin with real number giving its relative energy when a vehicle of a given class is scanned.

At Level 2, we would like to have "rules" similar to:

{bin7 energy > 1%, bin12 energy > 1%, bin16 energy < 1%, bin25 energy > 1%} → HIGH-VALUE-TARGET

This takes us away from manual clumpology into the realm of trainable MRF. It is not necessary (in fact, not possible) to enter vehicle modulation information into a data base, because it isn't known. But an MRF can infer these patterns using simulated annealing.

A reference set of vehicles is assembled, and scanned with the RADAR, and their modulation patterns sampled over many trials. Using the relaxation training method described in section xxx, and MRF has these patterns annealed into it. When a test pattern is introduced at the machine's input elements, the output element corresponding to the vehicle model will become active.

Feature reduction begins by creating a bit string equal to the number of bins into which frequency has been quantized. Bit positions corresponding to energy levels above a threshold value are set to "1"; others are set to "0". This encodes the Doppler RADAR modulation as a vector of zeros and ones.

To enter this data into the Hopfield network, the input nodes corresponding to bit values of "1" are switched ON; nodes corresponding to bit values of "0" are switched OFF. The update cycle is then applied until the output bits stop changing, and the total energy is low. The output node active indicates the networks classification decision (Fig. 6).

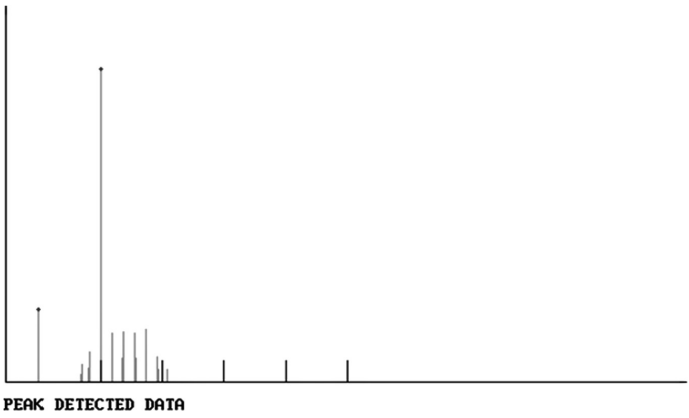

Fig. 6. If the energy of a bin exceeds a threshold value, the corresponding input bit is set to "1"; otherwise it is set to "0".

When a Hopfield Network having 32 input nodes, 6 input nodes, and 2 output nodes is trained and run on blind test data, accuracy of classification to a High-value/Low-value vehicle was 10-percentage points higher than the competing

regression classifier was realized. When a larger topology having 12 output nodes (the number of specific vehicle models) was trained on a large test set, accuracy improved again, and the machine was able to classify vehicle to specific model type about 60% of the time (Fig. 7).

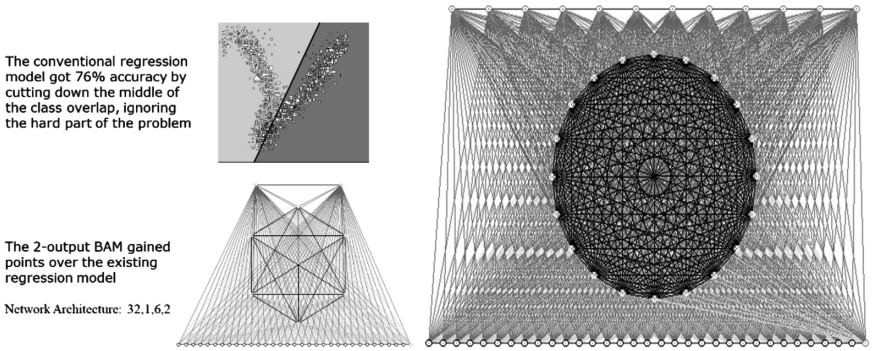

Fig. 7. Final Engine: lower-left is the high-low value MRF classifier architecture; the 32 input nodes are across the bottom, and the two output nodes are at the top. On the right is the larger 12-class MRF. In both of these architectures, every node is connected to every other node, and the input nodes are "clamped" to the input bit-string values.

Finally, since our particular MRF is a Boltzmann Machine, it is a bi-directional associative memory (BAM). Unlike most classifiers, trained BAMs can be run both forward and backward. In the usual operation mode, input nodes are clamped, and relaxation produces activation of the appropriate output node. But it a class output node is clamped, relaxation causes the relation defining the typical input pattern for that output class to appear at the input! For the Doppler RADAR vehicle classifier, the BAM is mathematically tricked into revealing the modulation patterns that correspond to the vehicle classes. These can then be used as vehicle templates in other classifiers. Our team has demonstrated that it is possible in this way to create a rule set that achieves the same performance of a trained BAM (Fig. 8).

```
NUMBER OF INPUT, PROCESSING, AND OUTPUT NEURONS: 32  6  2
NUMBER OF TRAINING PASSES: 200
LEARNING RATE: 2.999999932944775E-002
ANNEALING CONSTANT: 1.004999995231628
NEURAL ACTIVATION THRESHHOLD: .5
FINAL TEMPERATURE: .300000011920929

 CLASS     PROTOTYPICAL FEATURE VECTOR
-------   ------------------------------------------------------------------------

TRACKS     4 5 3 8 9 4 6 10 21 9 5 6 2 6 2 6 4 8 5 7 7 6 6 7 4 6 1 3 4 2 7 6

WHEELS     16 18 9 3 5 16 7 0 0 3 3 5 13 6 5 7 7 5 3 5 3 3 6 8 5 9 6 3 3 6 4 3

?
```

Fig. 8. Running the BAM is reverse produces the relative energy levels in each frequency bin that are the prototypical modulation patterns for both high-value and low-value vehicles.

References

1. Boyen, X., Koller, D.: Tractable inference for complex stochastic processes. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 33–42, Madison (1998)
2. Delmater, R., Hancock, M.: Data Mining Explained. Digital Press (2001)
3. Doucet, A., de Freitas, N., Gordon, N. (eds.): Particle Filters Sequential Monte Carlo Methods in Practice. Springer, New York (2001)
4. Friedman, N., Koller, D., Pfeffer, A.: Structured representation of complex stochastic systems. In: Proceedings of the 15th National Conference on Artificial Intelligence, pp. 157–164, Madison (1998)
5. Hancock, M.: Practical Data Mining. CRC Press, Boca Raton (2011)
6. Battaglia, F., Borensztajn, G.: Structured cognition and neural systems: from rats to language. Neurosci. Biobehav. Rev. **36**(7), 1626–1639 (2012)
7. Essentialism. https://www.britannica.com/topic/essentialism-philosophy. Accessed 15 Feb 2019