Higher Order Cognition using Computers: Learning Abstract Concepts with Recursive Graph-based Self Organizing Maps

Peter J. Bentley[1, 2], Alexander Kurashov[1] and Soo Ling Lim[1, 2]

[1]Braintree Limited, London, United Kingdom
[2]Department of Computer Science, University College London, United Kingdom
p.bentley@cs.ucl.ac.uk

Abstract

Abstract concepts are rules about relationships such as identity or sameness. Instead of learning that specific objects belong to specific categories, the abstract concept of same/different applies to any objects that an organism might encounter, even if those objects have never been seen before. In this paper we investigate learning of abstract concepts by computer, in order to recognize same/different in novel data never seen before. To do so, we integrate recursive self-organizing maps with the data they are processing into a single graph to enable a brain-like self-adaptive learning system. We perform experiments on simple same/different datasets designed to resemble those used in animal experiments and then show an example of a practical application of same/different learning using the approach.

Introduction

Living organisms facing the challenges of survival must distinguish food from poison, friend from foe. In many simple organisms this may be a simple case of classification or categorical perception. For example, it is claimed that an earthworm can learn to avoid harmful stimuli by classifying the stimuli as harmful after some period of learning and avoiding them (Wilson et al., 2014), or a damselfish can recognize "faces" (Siebeck et al., 2015). But in more complex organisms, there is a need for a more complex form of learning: the recognition of abstract concepts.

Perhaps one of the most fundamental abstract concepts is same/different (S/D). Instead of learning that specific objects belong to specific categories, the abstract concept of S/D applies to any objects that the organisms might encounter, even if those objects have never been seen before. Four identical apples are the same, just as five identical cars are the same. A variety of species of mammals and birds are different, just as a variety of colors are different.

In theory, such a skill might have advantages – perhaps detecting differences in a group might indicate the presence of a predator, or differences within a nest might detect if eggs have been taken or replaced with those of another. However, experiments have shown that while organisms such as pigeons are capable of being taught the higher-level concepts of same/difference, this ability comes far more easily to animals with greater intelligence, such as chimpanzees, baboons, capuchin and rhesus monkeys (Katz et al., 2007).

Abstract concept learning is thus considered to form the basis of higher order cognition in humans (Katz et al., 2007). For many decades researchers have used the ability to judge same/different as a core theme in cognitive development, cognition, and comparative cognition (Goodman & Melinder, 2007; Mackintosh, 2000; Shettleworth, 2009; Thompson & Oden, 2000). The abstract concept of same/different is also thought to be necessary within mathematics, and learning language (Marcus et al., 1999; Piaget, 1970).

In computer science, the majority of research on learning focuses on classification or prediction, i.e., learning how to categorize or cluster specific types of data, or learning patterns and regularities within specific examples of data (Michie et al., 1994). Conventional machine learning typically does not study how abstract concepts can be learned.

In this paper we investigate learning of abstract concepts by computer, in order to recognize same/different in novel data never seen before. The study explores what type of information processing is required in order to perform this task, evaluating the algorithm with experiments similar to those performed with animals, and assesses whether same/different can truly be called higher order cognition. We also show an example of a more practical application of same/different learning using the approach.

The remainder of the paper is organized as follows. The next section describes existing work. The sections after that describe the method, experiments, and results. The final section provides our conclusions.

Background

Abstract Concept Learning

Abstract concepts are rules about relationships such as identity or sameness, and are considered to form the basis of much of our so-called higher order cognitive processing (Katz et al., 2007). Children develop cognition in stages and expand their abstract concept of sameness to include number, length, area, and volume (Piaget, 1970). In the laboratory, the abstract concept of sameness is studied in same/different (S/D) tasks where subjects view stimuli and then make one of two responses to indicate whether the stimuli are the same or different (Katz et al., 2007). The determination of abstract-concept learning is accurate performance with novel test stimuli, i.e., the subject learned an abstract rule that transcends the particular training stimuli. Such transfer performance

makes abstract-concept learning unique and different from other forms of concept learning (Katz et al., 2007).

It is important to differentiate abstract concepts from natural concepts. Natural concepts are categories of items which share specific features, such as cars, chairs, flowers, person, water, or trees. In contrast, abstract concepts do not involve learning specific stimulus features. Instead, they involve learning the relationships between items (Katz et al., 2007). Thus, abstract concepts involve relational learning as opposed to the item-specific learning of natural-concept learning. S/D experiments have been run on pigeons (Katz & Wright, 2006) and rhesus monkeys (Katz et al., 2002).

Self-Organizing Maps (SOMs)

This work uses graph-based SOMs in order to achieve abstract concept learning. The SOM is a biologically inspired brain-map model (Kohonen, 2013) which is often used for visualization of data to obtain a more abstract view (Kohonen, 1998). It is an automatic data-analysis method resembling the classical vector quantization, with the addition that more similar models will be associated with nodes that are closer in the grid, and less similar models will be situated gradually farther away in the grid (Kohonen, 2013).

SOMs have been used extensively for a variety of clustering, classification and visualization applications. For example. Merelo et al. (1994) used SOMs to develop a protein classification algorithm. Aly et al. (2008) and Lawrence et al. (1997) used SOMs for face recognition. Kamimoto (2005) used SOMs to evaluate the vibration of motor-operated electric tools. Teranishi (2009) used supervised SOM to estimate the bending rigidity of real bills using only the acoustic energy pattern. (Fatigued bills affect the daily operation of automated teller machines). Okada et al. (2009) used multiple SOMs for control of a visuo-motor system that consists of a redundant manipulator and multiple cameras in an unstructured environment. Tateyama et al. (2004) proposed a pre-teaching method for reinforcement learning using a SOM in order to increase the learning rate using a small amount of teaching data generated by a human expert. Amor and Rettinger (2005) developed a genetic algorithm that uses SOMs to enhance the search strategy and confront genetic drift. They found that representing the search history by the SOM provides visual insights into the state and course of evolution. De Buitleir et al. (2012) created an artificial life population where the agents have sufficient intelligence to discover patterns in data and to make survival decisions based on those patterns using diploid reproduction, Hebbian learning, and SOMs. Saunders and Gero (2001) used SOMs in their artificial life society of agents, enabling the agents to determine the novelty of new artifact without sampling the entire design space.

In this work we propose a recursive SOM, where the results from lower level SOMs are fed into higher level SOMs in order to enable abstract concept learning. Similar recursive SOMs have been proposed in the past, for example SOM^2 (Furukawa, 2009) and ASSOM (Kohonen, 1996), which use lower level SOMs to find the building blocks used by higher level SOMs, when visualizing similar data. In this work we propose a novel approach whereby datatype-independent interpretations or summaries of many distinct and different lower level SOM results are fed as input into parent SOMs.

In our work we also make extensive use of graph databases. Graph databases use graph structures for semantic queries with nodes, edges and properties to represent and store data (Angles & Gutierrez, 2008). SOMs have been used with semantic networks in the past with good results (Allinson et al., 2001) but in this work we also introduce another novel design feature: the SOMs are integrated into the network they are processing. By representing both data and the SOM in the same graph-based network the system becomes a dynamic "brain-like" network that integrates data and learning in the same structure, automatically reconfiguring itself when it needs to learn new concepts, and providing complete provenance for all nodes.

Method

Abstract concepts are not absolute concepts. In real life, the abstract concept of same/different means "degree of difference". Objects with few or no apparent differences can be called "same". Objects with many distinguishing features can be called "different". Organisms may have a different concept of same/different depending on their experiences. For example, if we see many different species, then we learn to detect the differences between species – a flock of birds is different from a herd of buffalo. If we see many animals from one species, then we learn to detect the difference between individual animals – each buffalo is different from its companions.

Therefore, to achieve the learning of same/different, it is not enough to learn the concept from one set of data. One must learn from multiple sets of data that some things within each set are the same and others within each set are different, and the exact meaning of the concept same/different will depend on the data sets presented. To achieve this, a recursive learning approach is needed:

1. For each data set: the data is clustered into groups containing similar items.
2. The clustered data sets are then clustered according to overall datatype-independent features of each set (in this case the number of clusters).
3. (Optional) Step 2 is repeated for even higher-level concept learning.

After performing second-order clustering, the abstract concepts of same and difference can be constructed. Those clusters containing data sets with fewer clusters, have items more similar to each other. Clusters containing datasets with more clusters, have items more different to each other.

By feeding the results of the learning algorithm back into itself in this way, the higher-order, abstract concepts can be learned (and indeed ever-higher level, more abstract concepts can be learned by recursing further). Uniquely to this work, the input data, the SOM grids, and the resulting clusters derived from the SOMs, are all stored within the same graph structure. In order to achieve this while making every data item traceable from high-level concept to input vector, this work uses a graph database to store all data, learned models, and concepts. Unlike traditional relational databases, graph databases enable *provenance*. The provenance of a data item is the lineage of that data item, describing what it is and how it came to be. The provenance about a data item includes details about processes and input data used to create the item. This is

of great benefit when constructing new networks of concepts and abstract concepts, as it becomes possible to trace exactly which data resulted in which abstract concepts. Graph databases also enable fast traversals of large-scale network-based data - not something that traditional relational databases support well.

These approaches are also chosen because of their similarity to neural systems and their power and scalability to large datasets and large numbers of features per data item (Kohonen, 2013). The system is implemented in Java and the Neo4j graph database[1] is used, also chosen for its ability to scale and support high-speed queries.

In more detail, the method works as follows.

Build Dataset Graph

Each dataset is encoded as a graph, see Figure 1. A new Vector node is created for each data item in the set, with data attributes encoded as a list of numeric properties in each Vector. The Dataset node links to each Vector node (with a HAS_MEMBER link). The Dataset node also links to a Datatype node, containing a type variable with unique value for each dataset (with HAS_TYPE link).

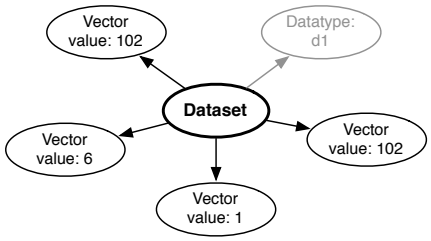


Figure 1. Each dataset is stored in a graph database (HAS_MEMBER links are in black, HAS_TYPE in grey).

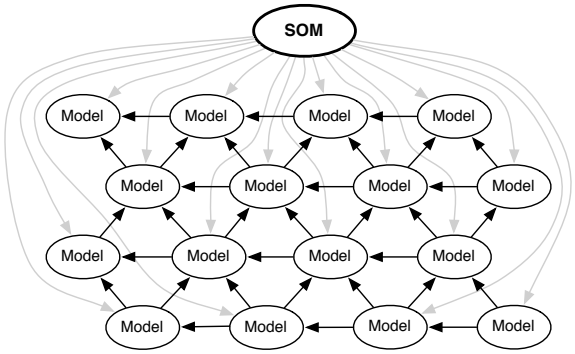


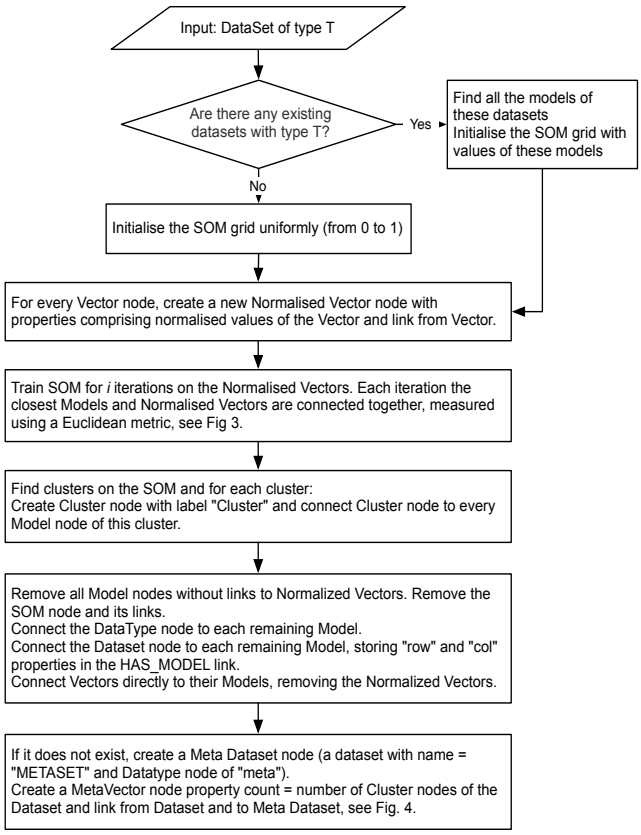
Figure 2. 4x4 SOM graph. HAS_MODEL links are shown in grey. NEIGHBOUR links are shown in black.

Build SOM graph

An SOM graph is created, see Figure 2. The SOM is encoded as an SOM node connected to a grid of $n \times n$ set of Model nodes (using a HAS_MODEL link), that are connected to their neighbors (using a NEIGHBOUR link). Each Model node contains a list of properties corresponding to the attributes of the data. Each HAS_MODEL link has integer properties "row" and "col", which represent the position of the Model within the grid.

First Order Learning

To classify the Dataset using the graph-based SOM, Algorithm 1 is used. Where no existing Models exist for a dataset of this type, each SOM is initialized by setting half of the attributes in each Model to *r/(N-1)* and the other half to *c/(N-1)*, where *(r, c)* denote the rectangular coordinate of the Model, and *N* is the size of the grid. This common method of initialization was chosen after preliminary experiments on standard datasets (e.g., Iris, Car, Wine, and Zoo from the UCI Machine Learning repository[2]) showed it was more effective than other alternatives. It also has the advantage of being deterministic and fast. Where a dataset of this type has been seen before, there will exist Models linked to the Datatype node; these are used to initialize the SOM to make use of past learning and improve speed. (Preliminary experiments showed that the reuse of previously learned Models can reduce the number of iterations needed for the SOM to converge.)



Algorithm 1. Learning and abstracting concepts from a dataset using the graph-based SOM.

[1] http://neo4j.com/

[2] http://archive.ics.uci.edu/ml/

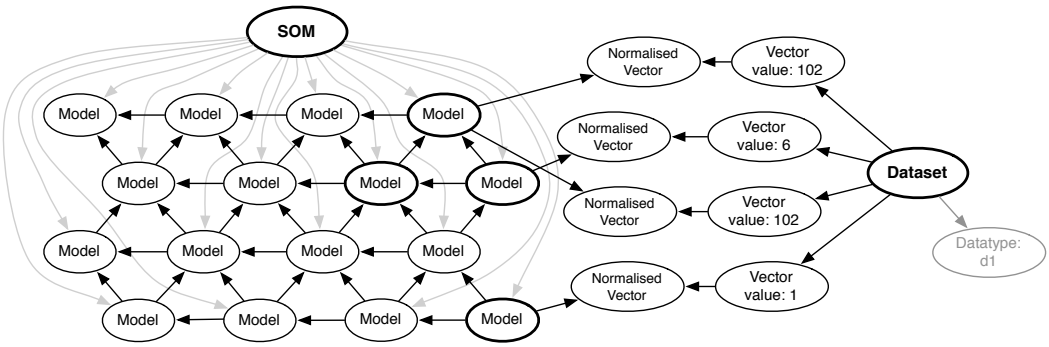


Figure 3. A graph-based SOM is trained on normalized vectors from the Dataset.

The SOM training procedure is taken from (Kohonen, 2013). On every iteration, each Model value is updated:

$$m'_i = \frac{\sum_j n_j h_{ij} x_j}{\sum_j n_j h_{ij}}, \text{where } h_{ij} = \frac{1}{gd(i,j)+1} \quad (1)$$

and where: x_j denotes the mean of the vectors that are closest (according to the Euclidean metric) to the Model j; n_j denotes the number of those vectors; $gd(i, j)$ denotes the grid distance between Models i and j; h_{ij} is a neighbourhood function (Kohonen, 2013).

The grid distance gd between Models is the distance between the cubic coordinates (ax, ay, az) and (bx, by, bz) of each Model M and is defined in Eqn. 2.

$$gd(M_{ax,ay,az}, M_{bx,by,bz}) = \frac{|ax-bx|+|ay-by|+|az-bz|}{2} \quad (2)$$

The cubic coordinate of a Model (x, y, z) is a transformation of its rectangular coordinate (r, c) defined in Eqn. 3.

$$(x, y, z) = (c - \frac{(r-(r \text{ XOR } 1)}{2}, -x-z, r) \quad (3)$$

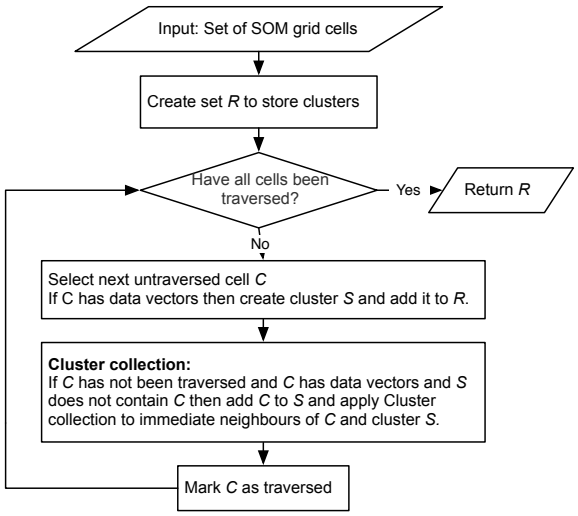
This method to compute the grid distance is chosen to optimize performance of a heavily used neighborhood function calculation.

After every Model is updated, the mean energy of the iteration is calculated (Eqn. 4).

$$E^* = \frac{\sum_i d(m_i, m'_i)}{N^2} \quad (4)$$

where m_i denotes the Model value before update; m_i' denotes the Model value after update; $d(x, y)$ denotes the Euclidean distance between vectors x and y; N denotes the size of the grid, and N^2 thus denotes the number of Models.

Each SOM is trained for a maximum of 50 iterations or until the mean energy E^* is less than 10^{-8}. The choice of neighborhood function and other values were also chosen after performing preliminary experiments on standard UCI datasets. Figure 3 illustrates the graph during the SOM learning phase. After learning is complete, the clusters are identified (see Algorithm 2) and Cluster nodes are linked to Models. Any unused Models and all normalized vectors are then removed from the SOM graph. This helps reduce the space taken for learning, while retaining useful Models in case the same type of data is presented again in the future. A MetaVector is then created, with a value that equals the number of clusters found. Figure 4 illustrates the graph after learning is complete and the SOM has been removed.



Algorithm 2. Finding clusters within the graph-based SOM.

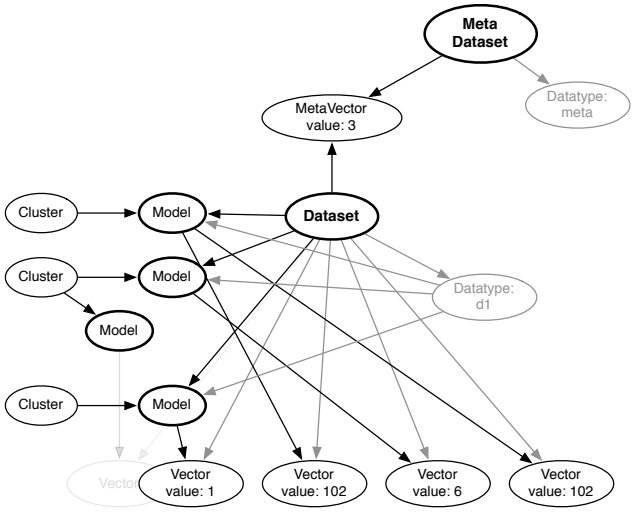


Figure 4. Similar Models are grouped into Clusters, the SOM and its unused Models are discarded, the normalized vectors are removed, and a MetaVector is created, summarizing the number of clusters in the Dataset, and added to the Meta Dataset. (A vector connected to the third Model is shown in pale grey for clarity.)

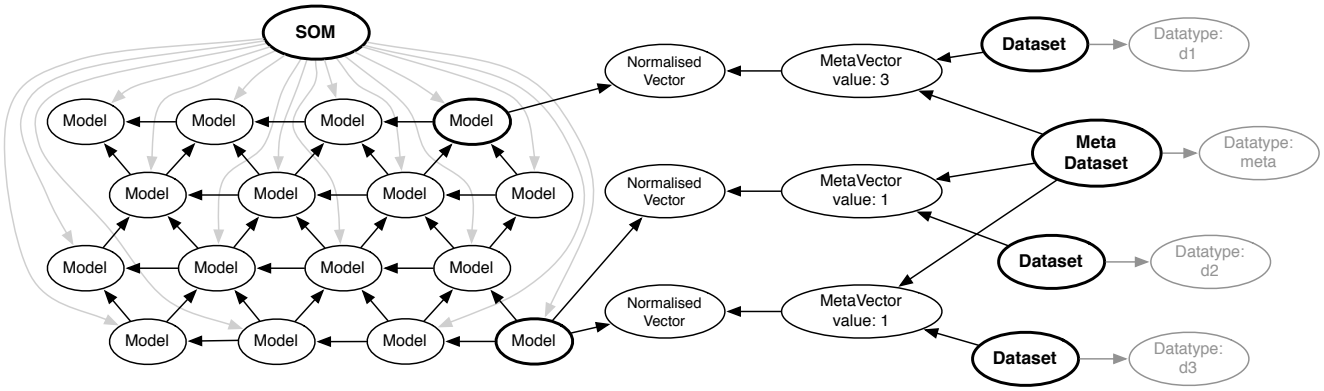


Figure 5. A new SOM is created and trained on the normalized MetaVectors from multiple different Datasets, which may comprise different types of data. (Graphs linked to each Dataset are not shown for clarity.)

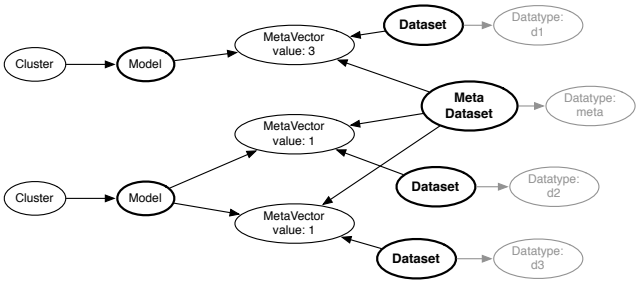


Figure 6. Similar Models in the SOM are linked to Clusters, the SOM and its unused Models are discarded, and the Normalized MetaVectors are removed as before. (New links between Models and Meta Datasets and Datatypes are not shown for clarity.)

Abstract Concept Learning

Each time First Order Learning is performed on a new Dataset, a new MetaVector node (linked to its corresponding Dataset) is created and linked to the Meta Dataset. Abstract Concept Learning is then performed by creating a new SOM to train on the set of MetaVectors. To perform Abstract Concept Learning, Algorithm 1 is used again on this Meta Dataset, where T = "meta", and Vectors are MetaVectors. Figure 5 illustrates the learning phase as the SOM is trained on the Normalised MetaVectors. Figure 6 illustrates the graph after learning is complete and the SOM has been removed. The resulting emergent clusters represent abstract concepts created – in Figure 6, one cluster represents the concept "same" as it points to Datasets containing only single clusters; the other concept represents "different" as it points to a Dataset containing multiple clusters (the MetaVector value is 3, representing 3 clusters). The final step of Algorithm 1 is optional (its execution would produce an even higher level of abstraction). If desired, such recursion could continue until everything is grouped into a single cluster.

Experiments

Experiment 1

The first experiment investigates whether the system can learn the higher order concept of "sameness" when presented with simple data similar to the simple symbols shown to animals during S/D experiments, see Table 1 (Katz et al., 2007). Table 2 shows the six simple sets of data, each set containing no value that appears in an earlier set. These sets were presented to the system in turn, allowing learning to complete before the next set was presented. SOMs were 4x4 in size with settings as described earlier.

Results 1

The results were fascinating. As each new dataset was presented, the first level SOM clustered the data, the clusters were identified, and the MetaVectors summarizing the clusters were then clustered to produce abstract concepts. Initially the first two datasets were clustered together – the system had correctly identified that both A and B contain items that are the Same. After dataset C was presented, the system created a new cluster containing C, representing the abstract concept Different. Datasets D and E were also placed into the Different cluster, see Figure 7 (Left). However, after the final dataset F was presented, the system had gained enough insight to reorganize the datasets, finally clustering A and B together (Same), C and D (Slightly Different), and E and F together (Very Different), see Figure 7 (Right). This seems a valid view of the data given that datasets E and F contain 10 and 9 unique values whereas C and D only comprise 4 values, of which only 1 or 2 are different.

- - - - - - - -	Same Display
+ U + + + + + +	Different Display

Table 1. Typical animal experiment data (Katz et al., 2007).

Data set	Values	S/D
A	33, 33, 33, 33	S
B	7.1, 7.1, 7.1, 7.1	S
C	102, 6, 102, 102	D
D	31, 1.5, -3, 31	D
E	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	D
F	1.0001, 1.0002, 1.0003, 1.0004, 1.0005, 1.0006, 1.0007, 1.0008, 1.0009	D

Table 2. Input data sets for Experiment 1.

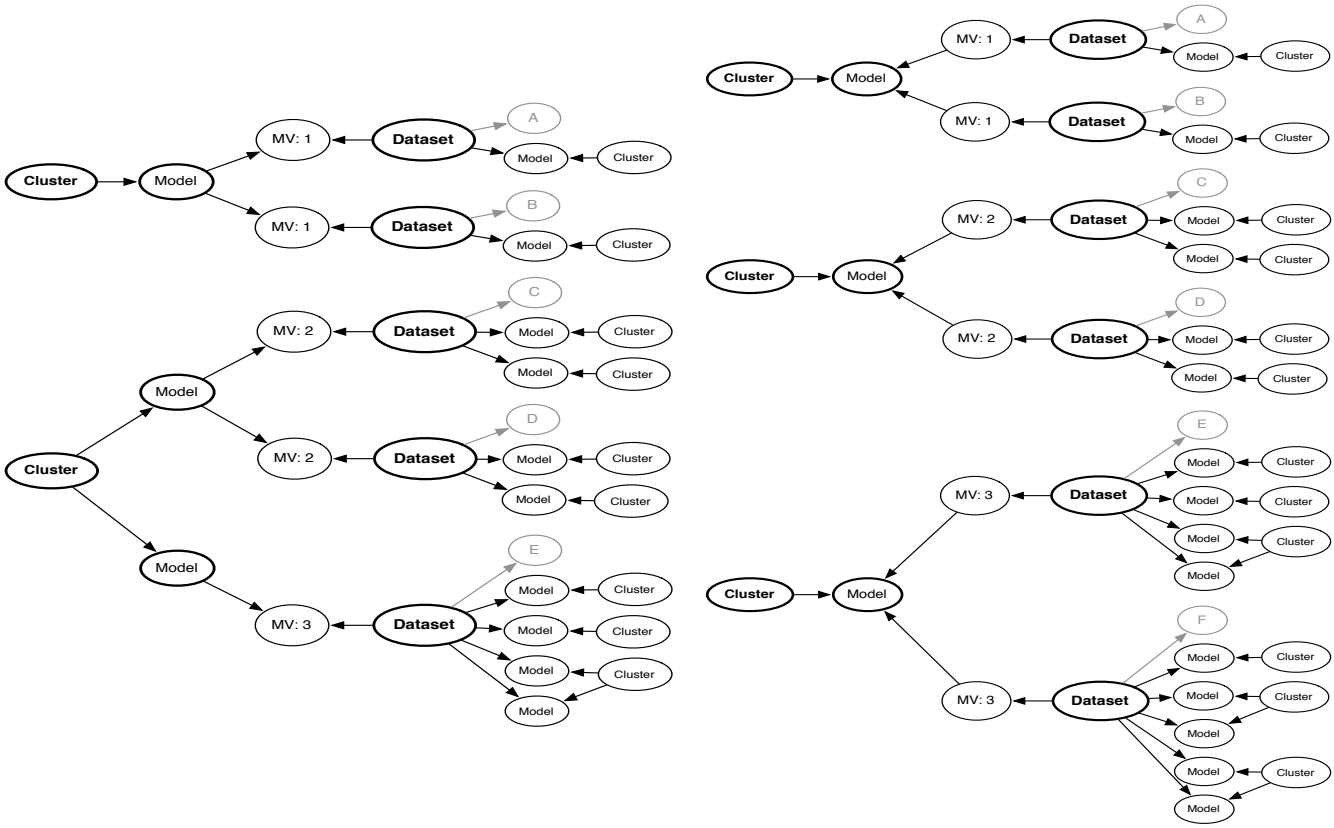


Figure 7. Left: the graph after presenting the first 5 datasets A to E. MetaVectors are shown as MV. The second-order SOM has discovered that there are two types of dataset: A,B (corresponding to Same) and C,D,E (corresponding to Different). Right: the graph after presenting the final dataset F. With more experience, the second order SOM now better distinguishes between the different types of dataset, by identifying three kinds: A,B (corresponding to Same), C,D (corresponding to Slightly Different) and E,F (corresponding to Very Different).

Experiment 2: Visual data

In many animal S/D experiments, instead of showing simple symbols, actual images of objects such as cars or flowers are shown (Katz & Wright, 2006). In the second experiment we take inspiration from these experiments and allow the system to "watch" an animation in which an event occurs after a specific period of time. Here we wish to determine if the system can understand when the frames of the animation are mostly the Same, discover the point in time when they become Different, and understand that new, unseen frames after the event may also be considered the Same once again. This is a more practical application of the ability to detect S/D and could, for example, be used to detect some anomalous event occurring in a video feed from a security camera. It is a more challenging task as the data presented to each SOM is considerably larger compared to Experiment 1. To enable the input to be carefully controlled, in Experiment 2 we use an 8x8 10-frame color animation of the video game character Mario, see Figure 9. Each frame was converted to a vector of 8x8x3 = 192 RGB values. An 8x8 SOM grid was used, and all other settings were the same as in Experiment 1. Three different scenarios were presented to the system.

Scenario 1: Jump after 10 frames: 100 frames of animation are presented, 5 at a time (e.g., 2-3-4-2-6, 7-2-3-4-2). The first 10 frames are Mario walking. The next 5 frames depict Mario jumping out of the picture. The remaining 85 are black.

Scenario 2: Jump after 50 frames: 100 frames of animation are presented, 5 at a time. After 50 frames of walking, the next 5 depict Mario jumping, and the remaining 45 are black.

Scenario 3: Jump after 90 frames: 100 frames of animation are presented, 5 at a time. After 90 frames of walking, the next 5 depict Mario jumping, and the final 5 are black.

Results 2

Figure 8 shows a summary of the clusters for each scenario. In the fist scenario the frames are considered Same (one meta cluster) until the Event on chunk 3 (frames 10-15) when Mario jumps. Here the first SOM finds two clusters, resulting in the higher level SOM forming a new meta cluster corresponding to the abstract concept Different. Following the Event, new identical black frames are presented, which are correctly classified as Same. This pattern occurs in the other two scenarios, except that during the Event the first level SOM finds three clusters. In all cases, the system learns that most of animation frames in each chunk can be considered the Same, but during each Event, one chunk of frames comprises Different frames.

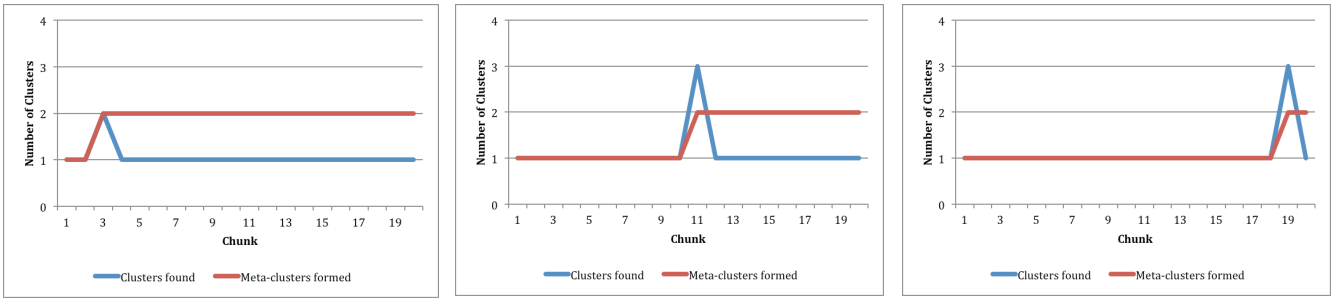


Figure 8. The number of clusters found by the first level SOM for each 5-frame chunk and meta clusters found so far by the second level SOM. The Event (Mario jumping out of shot) occurs during chunk 3 for Scenario 1, chunk 11 for Scenario 2 and chunk 19 for Scenario 3. On each Event the system learns and remembers that there are two classes of animation frames: Same and Different.

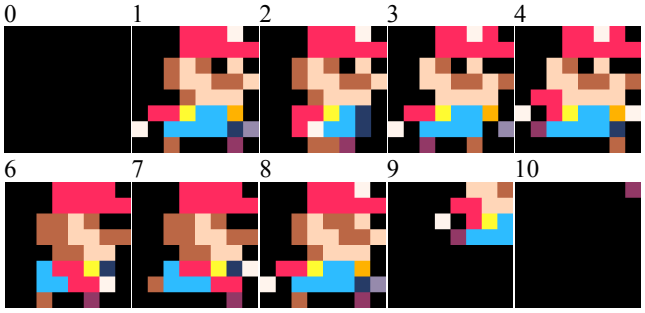


Figure 9. The 10 different frames of the Mario animation, comprising the walking sequence of frames: 2-3-4-2-6-7-2-3…, blinking sequence: 1-8-1-8-1…, and jumping sequence: 4-9-10-0-0-0…

Figure 10. (A) examples of a set of stored Models corresponding to the learned abstract concept Different. (B) and (C) show two examples of stored Models corresponding to the learned abstract concept Same, where (B) is a "hazily remembered Mario" and (C) comprises black frames.

A further test was performed by introducing the blinking Mario chunk (frames 1-8-1-8-1) at various times. This chunk of new frames could be regarded as Same (since each consecutive frames is different by a single pixel) or as Different (there are two clear groups of different frames: 1 and 8). Fascinatingly, the system seemed ambivalent towards this chunk of animation. It generally regarded the chunk as Same if it was presented after seeing many other Same chunks. It regarded the chunk as Different if it was presented after seeing the Event, which it considers Different.

Discussion

Abstract concept learning is considered to form the basis of higher order cognition in humans. This work has shown that in a very real sense, the notion of the abstract concept Same/Different is indeed higher order. For an SOM to discover this datatype-independent abstract concept, the output from one SOM must be fed into a higher-level SOM, which must learn about the general, datatype-independent features (in our case, the number of clusters) found by the first. This learning of datatype-independent features is an important requirement. Unlike previous work on recursive SOMs which build improved views of the same kind of data by using lower level SOM clusters as building blocks, here we show that it is necessary for the higher level SOM to learn about sets of overall interpretations or summaries of the datasets derived from the lower-level SOMs. By doing so, the system is able to accept radically and completely different datasets that share nothing in common, and still determine whether each dataset comprises items that should be considered largely Same or Different. With enough examples, the system can also start to differentiate further, and find Same, Slightly Different and Very Different. It should be stressed that these abstract concepts are automatically generated and they will change depending on what kinds of data are presented. Like the S/D abstract concept of living creatures, the system here only understands Same and Difference in terms of its experience of different data sets, not in terms of any absolute comparisons of values.

The use of a graph database for this approach also enables the system to explain its understanding. We are able to query the database and ask it for examples of Same or Different that it has experienced. Even when the original datasets have been removed, the stored SOM Models are able to provide a "hazy memory" of its notion of each abstract concept, see Figure 10. The connections from each Model via MetaVectors and Models to Dataset types enables the tracking of the provenance of each Model – it is possible to know which dataset resulted in each abstract concept or which "hazy memory".

Conclusion

In this work we have presented a novel recursive SOM, where a datatype-independent summary of the output from lower-level SOMs that have been applied to different datasets is fed into a higher-level SOM in order to learn the abstract concepts of Same/Difference. The implementation exploits graph-based computing, with data, SOM grid, learned clusters and all concepts represented in the same graph database. This provides the advantage of provenance for all nodes, enabling details about processes and input data used to create the item to be found highly efficiently. The graph representation reorganizes itself during and after learning, adding new concepts as they are discovered, removing nodes when they are no longer needed, and reusing stored nodes to improve efficiency of learning. The use of the graph database also enables this approach to scale.

The method was tested on simple same/different datasets designed to resemble those used in animal experiments and then a more practical application of same/different learning was investigated – finding anomalous frames within a short animation. In all cases the system demonstrated a clear ability to learn the datatype-independent abstract concepts of Same/Different correctly, and this "skill" was refined and improved as new data was presented.

There are many potential applications for this system, where learning of Same/Different is non-trivial. Examples include the identification of similar profiles in large databases, anomalous events in video streams, or the identification of other higher-level concepts such as homographs and synonyms. Future work will explore some of these ideas.

References

Allinson, N., Yin, H., Allinson, L., & Slack, J. (2001). *Advances in self-organising maps*. Springer, London.

Aly, S., Sagheer, A., Tsuruta, N., & Taniguchi, R.-i. (2008). Face recognition across illumination. *Artificial Life and Robotics, 12*(1-2): 33-37.

Amor, H. B., & Rettinger, A. (2005). Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1531-1538.

Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys, 40*(1): 1-39.

de Buitléir, A., Russell, M., & Daly, M. (2012). Wains: A pattern-seeking artificial life species. *Artificial Life, 18*(4): 399-423.

Furukawa, T. (2009). SOM of SOMs. *Neural Networks, 22*(4): 463-478.

Goodman, G. S., & Melinder, A. (2007). Child witness research and forensic interviews of young children: A review. *Legal and Criminological Psychology, 12*(1): 1-19.

Kamimoto, N., Yamada, Y., Kitamura, M., & Nishikawa, K. (2005). Evaluation of vibration in many positions by SOM. *Artificial Life and Robotics, 9*(1): 7-11.

Katz, J. S., & Wright, A. A. (2006). Same/different abstract-concept learning by pigeons. *Journal of Experimental Psychology: Animal Behavior Processes, 32*(1): 80-86.

Katz, J. S., Wright, A. A., & Bachevalier, J. (2002). Mechanisms of same-different abstract-concept learning by rhesus monkeys (Macaca mulatta). *Journal of Experimental Psychology: Animal Behavior Processes, 28*(4): 358-368.

Katz, J. S., Wright, A. A., & Bodily, K. D. (2007). Issues in the comparative cognition of abstract-concept learning. *Comparative Cognition & Behavior Reviews, 2,* 79-92.

Kohonen, T. (1996). Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biological Cybernetics, 75*(4): 281-291.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing, 21*(1): 1-6.

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks, 37,* 52-65.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks, 8*(1): 98-113.

Mackintosh, N. J. (2000). Abstraction and discrimination. In C. Heyes & L. Huber (Eds.), *The Evolution of Cognition* (pp. 123-141). Cambridge, MA, US: The MIT Press.

Marcus, G. F., Vijayan, S., Rao, S. B., & Vishton, P. M. (1999). Rule learning by seven-month-old infants. *Science, 283*(5398): 77-80.

Merelo, J. J., Andrade, M. A., Prieto, A., & Morán, F. (1994). Proteinotopic feature maps. *Neurocomputing, 6*(4): 443-454.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Upper Saddle River, NJ, USA.

Okada, N., Qiu, J., Nakamura, K., & Kondo, E. (2009). Multiple self-organizing maps for a visuo-motor system that uses multiple cameras with different fields of view. *Artificial Life and Robotics, 14*(2): 114-117.

Piaget, J. (1970). *Science of Education and the Psychology of the Child. Trans. D. Coltman*. Orion, New York.

Saunders, R., & Gero, J. S. (2001). A curious design agent. *CAADRIA (The Association for Computer-Aided Architectural Design Research in Asia)*, pages 345-350.

Shettleworth, S. J. (2009). *Cognition, Evolution, and Behavior*. Oxford University Press, USA.

Siebeck, U. E., Parker, A. N., Franz, M. O., & Wallis, G. M. (2015). Face discrimination in fish. *Behaviour*, pages 1.

Tateyama, T., Kawata, S., & Oguchi, T. (2004). A teaching method using a self-organizing map for reinforcement learning. *Artificial Life and Robotics, 7*(4): 193-197.

Teranishi, M., Omatu, S., & Kosaka, T. (2009). Continuous fatigue level estimation for the classification of fatigued bills based on an acoustic signal feature by a supervised SOM. *Artificial Life and Robotics, 13*(2): 547-550.

Thompson, R. K., & Oden, D. L. (2000). Categorical perception and conceptual judgments by nonhuman primates: The paleological monkey and the analogical ape. *Cognitive Science, 24*(3): 363-396.

Wilson, W. J., Ferrara, N. C., Blaker, A. L., & Giddings, C. E. (2014). Escape and avoidance learning in the earthworm Eisenia hortensis. *PeerJ 2:e250,* https://doi.org/10.7717/peerj.250.