**Anomaly Detection and Assessment Using Growing Neural Gas**

By Matt Paisner

**Abstract**

Metacognitive architectures provide one solution to the problem of brittleness for agents operating in complex, changing environments. The Metacognitive Loop (MCL), in which a system notes an anomaly, assesses the problem and guides a solution, is one form of such an architecture. This paper extends prior work on implementing the note phase of MCL in symbolic planning domains, using a growing neural gas algorithm to construct a network which represents various normal and anomalous states. Testing shows that this technique allows for improved detection of anomalies in the note phase as well as categorization of anomalies by severity and type.

**1 Introduction and Related Work**

Artificial intelligence agents guided by machine learning algorithms are often very successful at well-defined, consistent tasks, even when those tasks are very complex. However, for such agents to be effective in many real-world settings, they must be able to deal with situations that are more fluid. A pathfinding program might learn to navigate mazes better than a person ever could, but if it tries to apply that knowledge, unmodified, to the problem of climbing a wall, it will fail badly.

Of course, what the agent ought to do is to temporarily forget its maze navigation mastery and either start learning from scratch or apply some alternative algorithm that is more suited to its new task. Anderson and Perlis (2005) describe a metacognitive approach to this type of problem, in which an agent continuously monitors both the world and its own internal processes in order to *note* problems, *assess* their type and severity, and *guide* an appropriate response strategy. They call this process the metacognitive loop (MCL). Several iterations of MCL have been developed, including domain-specific versions that guide agents to improved performance in Q-learning (Anderson et al. 2006) and natural language dialog (Josyula et al., 2007) systems. A domain-independent version of MCL based on a Bayesian net was also developed (Schmill et al., 2011) and used to guide a simulated mars rover, as well as in other tasks.

Recently, work has begun on a version of MCL that works in symbolic domains, such as deterministic classical planning domains like blocks world (in which agents arrange blocks in stacks) and logistics (a package delivery scenario). Because they are defined by statements in a predicate logic, such domains are very amenable to logic- and explanation-based approaches to explanation. However, noting anomalies in the first place required the adaptation of techniques in statistical anomaly detection that were designed for real-valued, time series data. The problem of detecting anomalies in the inputs to intelligent systems has been studied extensively (e.g. Albrecht et al., 2000; Basseville and Nikiforov 1993; Crook and Hayes 2001; Fawcett and Provost 1999; Janssens, Postma, and Hellemons 2011; Pannell and Ashman 2010). There is also a large body of work on anomaly detection in data streams specifically (see Chandola, Banerjee, and Kumar 2009).

One such approach, the A-distance (Kifer et al., 2004), uses a statistical comparison between a baseline window of “normal” data and a sliding window of recent observations to detect shifts in the underlying distribution. This approach has been used in past domain adaptation work (Blitzer et al., 2006; Blitzer et al., 2007; Dredze et al., 2010). Cox et al. (2012) tested an implementation of the note phase of MCL in symbolic domains through the use of A-distance. This approach was successful as an anomaly detection procedure, but left some questions unanswered. First, A-distance requires the choice of a threshold value, epsilon, to determine how different a state must be from the baseline to be called an anomaly. One approach (used by Cox et al.) is to try several different values. However, in a real-time anomaly detection situation this may not always be practical. Additionally, while the adaptation of A-distance to these types of domains is creative, the analysis of results to this point has been fairly simplistic, making no attempt to move from noting an anomaly to attempting to assess it in terms of severity or type.

This paper aims to begin the work of filling that space. The next sections will describe the use of using the results from A-distance as inputs to a growing neural gas (GNG) network (Fritzke, 1995). This results in a network in which certain nodes correspond to anomalous data, providing two benefits. First, because the network grows or shrinks to fill whatever input space it is presented with, when fed a set of baseline data it can self-calibrate to recognize anomalies of various types without the necessity for a user-selected epsilon value. Second, the nodes of the network do not simply present a Boolean “anomalous” or “normal” result, but can be seen as archetypes, with each node representing an anomaly of a certain type and intensity. This information can then be used to help determine the most appropriate choice of strategies.

The next section will explain the outputs generated by applying A-distance to a predicate-logic state representation, and how those outputs are used as inputs to a GNG network. Following that, we will detail the experiments performed, and present data on both the success of the GNG/A-distance combination in anomaly detection and in anomaly assessment. Finally, we conclude and discuss future research, including the ways in which we anticipate using the results of this work in a complete version of MCL.

**2 Domain [Change heading]**

Experiments were conducted in the logistics world (Veloso, 1992), a symbolic planning domain in which airplanes and trucks are tasked to deliver packages. Logistics is a deterministic world in which states are defined by sets of statements in predicate logic. As an example, consider the following visualization of a relatively simple state, consisting of two trucks, two planes and two packages each located in one of two cities. Each city has both a post office and an airport.



Figure

A simplified[[1]](#footnote-1) predicate representation of this state would be:

**At-Truck(TruckC, Airport\_A)**

**At-Truck(TruckB, Airport\_B)**

**At-Plane(PlaneA, Airport\_A)**

**At-Plane(PlaneB, Airport\_B)**

**At-Package(ObjectC, PostOffice\_A)**

**Inside-Truck(ObjectA, TruckA)**

**Inside-Truck(ObjectB, TruckB)**

From such a state, and given a list of goals, a classical planner can then generate a plan to achieve them. Cox et al. (2012) introduce a method for applying A-distance to series of plans of this sort in order to detect when changes have occurred in the planning process (for example, when the ability to unload airplanes has been removed). The full details of that method will not be described here, but it is important to the present paper to describe the output of the procedure.

First, we note that A-distance makes use of two “windows” of data, each of size n, to detect anomalies. The first window consists of the first n observations, which are assumed to come from a baseline distribution. The second window slides along the data stream, and always contains the n most recent data points. The value of A-distance at time t is a function of the difference between the distributions of data in the two windows, [0, n) and (t-n, t]. The output of running A-distance on a data stream, then, is another data stream, with the value at step t indicating how anomalous were the n data points up to and including t in the initial stream[[2]](#footnote-2).

Cox et al. use multiple data streams to represent a series of plans. Each of these streams is associated with a predicate, and A-distance is run separately on each stream. The output from A-distance, then, is a series of data streams, with the stream corresponding to the “at-truck” predicate, for example, reflecting the degree to which that predicate’s usage is anomalous at each time step. An anomaly is reported at time t when the value of *any* of the output streams at t is greater than parameter epsilon. Figure 2 shows an example of such output streams for three predicates.

**[0.13 0.24 0.30 0.36 0.36] (At-Package)**

**[0.16 0.20 0.20 0.20 0.18] (Inside-Plane)**

**[0.13 0.21 0.26 0.32 0.34] (Inside-Truck)**

The underlined values are those that would be considered anomalous at an epsilon value of 0.3. The last three time steps in the sequence shown would then be considered part of an anomaly. In this example, it seems likely that the anomaly has to do with packages being transferred to and from trucks in unusual numbers, since the At-Package predicate (which reflects packages being located at airports or post offices) and the Inside-Truck predicate are the ones that generate the anomaly.

Such sets of predicate-associated output vectors will be the inputs to the algorithm described below.

**2.1 Growing Neural Gas**

A basic neural gas network consists of only two layers, input and output. The algorithm uses one-shot single-winner competitive learning, in which the winning output node is picked based on Euclidean distance from the input vector. The learning rule for the winner is:

*dw = n(a - w)*

Where a is the input vector, n is the learning rate and w is the node’s weight vector. Neighbors of the winning node also learn, but with a smaller value of n. The output of the network also reflects a single-winner dynamic, with only that node turned on.

Growing neural gas (GNG) adds the dynamic of allowing output nodes to be created and eliminated to better fit the structure of the input data. In Fritzke’s (1995) classic model, an error value is stored for each node, based on the total squared distance from that node to all inputs for which it was the winner. New nodes are then added near the nodes with the highest error rates, at fixed intervals. Each new node has an edge connecting to the high error node near to which it was placed.

This dynamic is intended to induce higher sensitivity in areas where there is a greater input density. However, it is less effective at accounting for small anomalous data clusters that are disconnected from the larger mass of normal data, especially when those clusters are present only during a short interval and then cease. One reason why this pattern is less easily captured is that anomalous data will occur much less frequently, so that it is difficult for a node to move all the way out to the location of the actual cluster without being pulled back towards the center by the more frequent input patterns. Also, even if a node does reach the correct location, if the outlying input patterns stop being expressed if will drift away due to the pull of its neighbors, and the network will “forget” about the anomaly rather quickly.

The solution to this issue employed in the present experiments is two-fold. First, the method of adding nodes has been changed. For the purposes of anomaly detection, it is actually counterproductive to add more nodes in a small area of high concentration, as the old algorithm tends to do. Therefore, the version of GNG used herein adds nodes based on distance – when an input pattern is not near any currently existing node, a new node is created at the point specified by the input. This allows brief or unusually anomalous events to be detected. Also, the neighbor learning rate has been reduced to near zero, preventing the pull of outlying nodes back to the center. This prevents GNG from effectively performing its function as a topological map, but makes it more useful as a flexible clustering method, our present purpose.

**2.2 A-Distance as input to GNG**

As described above, A-distance produces a series of vectors as output, with one vector reflecting the abnormality of the distribution of a single predicate at each time step. The input data to GNG is the series of vectors created by, at each time step, concatenating the values from all input streams. So, for a system with k predicate streams, observed at T time steps, the input to GNG would be T vectors of length k, with each vector corresponding to a single time step t and reflecting the A-distance values for all predicates at t.

Over time, the GNG network develops a set of nodes which correspond to archetypes of normal and anomalous world states. Since the input data reflected a measurement of abnormality, nodes with larger values can be seen as more anomalous. The degree to which each archetype is anomalous, then, can be crudely determined by the distance from the node to the origin, and the type of any new input state can be determined by referring to the network’s activated output node.

To provide a Boolean anomaly detection result, this algorithm is run on a set of baseline data. The requirement for such a guaranteed set is not a significant issue, since A-distance already makes this assumption for the calibration of the baseline window. Once a network is generated using this data, the minimum “anomalous” distance d from the origin a node is defined as the maximum distance of any node in the baseline network, subject to a minimum number of updates that were performed on it. This last restriction is designed to eliminate outliers, especially those that occur near the end of the data set. Once d is calculated, the algorithm is run normally on the test data, and if the distance from the activated node for a given input to the origin is greater than d, that input is considered anomalous.

**Experimental Setup**

All experimental data were drawn from randomly generated world states and goals in the logistics domain described above. To create anomalous data, planning was done with one operator removed, and only world state/goal combinations for which a plan could be successfully generated were used.

There were three data sets used for testing. The first, the “airplane anomaly” set, consisted of 500 normal plans and 100 plans with the “unload-airplane” operator removed. The second, designated “truck anomaly”, also had 500 normal plans and 100 with the “unload-truck” operator removed. The third, “two anomaly”, had 500 normal plans and two separate anomalous sections of 100 plans, the first of which had the “unload-airplane” operator removed, and the second “unload-truck.”

Trials were conducted using variable concentrations of anomalous plans in the anomalous sections, simulating anomalies of different intensities. For example, at intensity 0.2, only 20% of the 100 plans in each anomalous section were actually anomalous, signifying a very faint anomaly. For comparison, we ran trials on each data set using A-distance alone, using several epsilon values, as well as the combined process. 50 trials were run for each pair of (data set, intensity level), where intensities ranged from from 0 to 1 by units of 0.1.

**Results**

For each method, including the A-distance/GNG combination as well as A-distance alone using several epsilon values, reported anomalies at each step were recorded, and expected results calculated. The expected outcome at a time step was ‘anomalous’ if at least P% of the values within the sliding window came from anomalous plans, i.e. those that were generated with an operator removed. Values for precision, recall, and F1 were then calculated for each method and at several values of P from 0 to 50[[3]](#footnote-3). Figure 3 shows the average F1 values by test setup for each of the three data sets.

Figure 3

On each data set, the GNG network performed slightly worse than the best epsilon value. However, to select the best epsilon value required testing using a priori knowledge of which data came from anomalous sources, while GNG was initialized only using a set of baseline data. These results, then, suggest that while the addition of processing with GNG may not be useful given a domain and anomaly set whose characteristics are well known, it does come very close to optimizing the performance of A-distance without requiring any prior information on the type of anomalies that will occur. In the context of MCL, which is intended to be employed in a wide spectrum of domains that may change unpredictably, this is a useful feature.

Besides improving the flexibility of anomaly detection, the second purpose of this method is to provide useful information as the MCL cycle moves into the assessment phase. GNG’s contribution to this goal is to provide groupings of anomalies by type and severity so that a newly noted anomaly can be quickly categorized, allowing a more robust assessment procedure to narrow down its list of options. Figure 4 graphs assignments of world states to nodes during the anomalous segments of the two anomaly data set. The x-axis counts steps from the sliding window’s entry into the anomalous section. The y-axis reflects which output node was selected at that time step, listed from least anomalous (1) to most anomalous (11).

Figure 4

There are two significant features of this graph. First, other than at the very edges of each anomaly (when most of the data in the sliding window is normal), the two anomaly types map to two non-intersecting sets of nodes: the truck anomaly maps to nodes 4, 6, 8 and 9, while the airplane anomaly maps to 3, 5, 7, 10 and 11. So, the GNG network is able to differentiate between anomalies of different types. Second, in both cases there is a clear progression, as a larger percentage of the window slides into the anomalous region, from nodes that are slightly anomalous (3, 4, 5), to nodes that are moderately anomalous (6, 7), to those that are fully anomalous (8, 9, 10, 11) as the window’s back edge enters the anomaly. This pattern is reversed as the window slides out of the anomalous section, showing that the network has generated anomaly archetypes which reflect intensity as well as type.

**Discussion**

Both A-distance and Growing Neural Gas are established techniques. The use of GNG to perform online clustering of A-distance data, however, opens up new possibilities for an agent that is attempting to adapt to a changing world. The first of these is the ability to dispense with user-selected epsilon values. Because a GNG network grows to fill its input space, it can organically generate a threshold that provides results comparable with the best possible epsilon choice.

Secondly, the network generated by GNG provides a unique perspective on anomaly analysis. A simpler approach might have been to run GNG, or another online clustering algorithm, on raw state data. By using A-distance data instead, we have a network that, instead of generating clusters of similar states, creates clusters of similar types of *anomalies*. Further, because the input data have a consistent meaning – a larger coordinate value corresponds to a larger disparity from normal for the corresponding predicate – we have actually gained more information than simply a set of clusters. If a selected node is just above the anomaly threshold, we can surmise that it might simply be an outlier or a weak anomaly. If the selected node has a very high value for the “inside-airplane” coordinate and a very low one for “inside truck,” we know that something strange is happening with the airplane fleet, but we probably do not need to spend much time worrying about our trucks.

Because the project of MCL is to fix problems by first deducing what has gone wrong, the added semantics of a net constructed A-distance predicate data is important. When presented with a weak anomaly, MCL might want to wait for confirmation, or pursue a conservative strategy. If the anomaly is caused by changes to the plane-related predicates, it should keep trucks running as they have been while employing reasoning or learning techniques to improve the agent’s airplane control strategies in light of the changes. The ability to use data-driven techniques like GNG and A-distance to generate semantic knowledge about the type of failure that is being experienced gives MCL a significant head start in more conventional approaches to reasoning, as well as a clearer target for future learning.

**Future Work**

As it stands, the present paper describes testing that has only been performed on randomly generated data. Further, while this process has demonstrated the capability to discover anomalies and categorize them into types that provide some semantic information, we have not taken the next step of using that information to direct a strategy change. The next stages of work on MCL will go in two directions. First, we want to use these results directly to inform an agent that is trying to make choices. In the logistics domain, a simple example might feature an agent that has two choices of planning algorithms: one that emphasizes ground transport and one air transport. When an anomaly occurs, it could decide whether to switch algorithms on the basis of anomaly type and severity.

The second avenue of research will focus on combining these data-driven techniques with a knowledge-rich approach to analysis. A significant appeal of logically formulated domains like logistics is that they are by nature amenable to techniques of case-based reasoning. MCL should be able to utilize this type of process alongside, and with the aid of, data-driven techniques like the one described here. Once these two areas have been explored, we hope to have a version of MCL that can autonomously aid an agent in dealing with adversity across a wide range of domains and problem types.

**References**

S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, and P. Tavan, “Generalized radial basis function networks for classification and novelty detection: self-organization of optional Bayesian decision,” Neural Networks 13(10), pp. 1075–1093, 2000

M. L. Anderson, T. Oates, W. Chong, and D. Perlis, “The metacognitive loop I: enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance,” Journal of Experimental and Theoretical Artificial Intelligence, 18.3, pp. 387-411, 2006.

M. L. Anderson and D. Perlis . “Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness,” Journal of Logic and Computation, 15.1, 2005.

M. Basseville and I. V. Nikiforov, Detection of abrupt changes - theory and application. Englewood Cliffs, NJ, Prentice-Hall, 1993.

J. Blitzer, R. McDonald, and F. Pereira, “Domain adaptation with structural correspondence learning,” Proc. 2006 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2006.

J. Blitzer, M. Dredze, and F. Pereira, “Biographies, Bollywood, boom-boxes and blenders: domain adaptation for sentiment classification,” in Association for Computational Linguistics Vol. 7, pp. 440-447, 2007.

V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys 41(3), 2009

M.T. Cox, T. Oates, M. Paisner, D. Perlis. “Noting anomalies in streams of symbolic predicates using A-distance,” Advances in Cognitive Systems 2, pp. 167-184, 2012.

P. Crook and G. Hayes, “A robot implementation of a biologically inspired method for novelty detection,” in Proc. Towards Intelligent Mobile Robots Conference, Edinburgh, Scotland, Division of Informatics, University of Edinburgh, 2001.

D. Josyula, S. Fults, M. L. Anderson, S. Wilson, and D. Perlis, “Application of MCL in a dialog agent,” in Third Language and Technology Conference, 2007.

M. Dredze, T. Oates, and C. Piatko, “We're not in Kansas anymore: detecting domain changes in streams,” in Proc. EMNLP '10, pp. 585-595, 2010.

T. Fawcett and F. Provost, “Activity monitoring: noticing interesting changes in behavior,” in Proc. 5th International Conference on Knowledge Discovery and Data Mining (SIGKDD99), pp. 53-62, New York: Association for Computing Machinery, 1999.

B Fritzke, “A growing neural gas network learns topologies,” in Advances in Neural Information Processing Systems 7, MIT Press, Cambridge MA, 1995.

J. Janssens, E. Postma, J. Hellemons (Eds.), Proc. International Workshop on Maritime Anomaly Detection 2011, Tilburg, The Netherlands: Tilburg Center for Cognition and Communication, Tilburg University, 2011.

D. Kifer, S. Ben David, and J. Gehrke, “Detecting change in data streams,” In Proc. Thirtieth Very Large Databases Conference, pp. 80-191, 2004.

G. Pannell and H. Ashman, “Anomaly detection over user profiles for intrusion detection,” in Proc. 8th Australian Information Security Management Conference, Perth, Western Australia: Edith Cowan University, 2010.

M. D. Schmill, M. L. Anderson, S. Fults, D. Josyula, T. Oates, D. Perlis, H. Haidarian, S. Wilson, and D. Wright. “The metacognitive loop and reasoning about anomalies,” In M. T. Cox and A. Raja (Eds.), Metareasoning: thinking about thinking (pp. 183-198), Cambridge, MA: MIT Press, 2011.

M. M. Veloso, “Learning by analogical reasoning in general problem solving,” Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, PA, 1992.

1. Static predicates representing the locations of airports and post offices and their relations have been omitted. [↑](#footnote-ref-1)
2. For t < n, A-distance is defined to be 0 since the baseline window is not yet full. [↑](#footnote-ref-2)
3. P=50 means that an anomaly is defined as a step at which more than half of the data in the sliding window was generated by planning with an operator removed. [↑](#footnote-ref-3)