Abstract

Classifying novel terrain or objects from sparse, complex data may require the resolution of conflicting information from sensors working at different times, locations, and scales, and from sources with different goals and situations. Information fusion methods can help resolve inconsistencies, as when evidence variously suggests that an object’s class is car, truck, or airplane. The methods described here address a complementary problem, supposing that information from sensors and experts is reliable though inconsistent, as when evidence suggests that an object’s class is car, vehicle, and man-made. Underlying relationships among classes are assumed to be unknown to the automated system or the human user. The ARTMAP information fusion system uses distributed code representations that exploit the neural network’s capacity for one-to-many learning in order to produce self-organizing expert systems that discover hierarchical knowledge structures. The fusion system infers multi-level relationships among groups of output classes, without any supervised labeling of these relationships. The procedure is illustrated with two image examples, but is not limited to the image domain.

Keywords: ARTMAP; Adaptive resonance theory (ART); Information fusion; Pattern recognition; Data mining; Remote sensing; Distributed coding; Association rules; Multi-sensor fusion

1. Introduction: deriving consistent knowledge from inconsistent information

Image fusion has been defined as “the acquisition, processing and synergistic combination of information provided by various sensors or by the same sensor in many measuring contexts.” (Simone, Farina, Morabito, Serpico, & Bruzzone, 2002, p. 3) When multiple sources provide inconsistent data, such methods are called upon to select the accurate information components. As quoted by the International Society of Information Fusion (http://www.inforfusion.org/terminology.htm): “Evaluating the reliability of different information sources is crucial when the received data reveal some inconsistencies and we have to choose among various options.” For example, independent sources might label an identified vehicle car or truck or airplane. A fusion method could address this problem by weighing the confidence and reliability of each source, merging complementary information, or gathering more data. In any case, at most one of these answers is correct.

The methods developed here address a complementary and previously unexamined aspect of the information fusion problem, seeking to derive consistent knowledge from sources that are inconsistent—yet accurate. This is a problem that the human brain solves well. A young child who hears the family pet variously called Spot, puppy, dog, dalmatian, mammal, and animal is not only not alarmed by these conflicting labels but readily uses them to infer functional relationships. An analogous problem for information fusion methods seeks to classify the terrain and objects in an unfamiliar territory based on intelligence supplied by several reliable sources. Each source labels a portion of the region based on sensor data and observations collected at specific times and based on individual goals and interests. Across sources, a given pixel might be correctly but inconsistently labeled car, vehicle, and man-made.

Corresponding author. Tel.: +1 617 353 9483; fax: +1 617 353 7755.
E-mail addresses: gail@cns.bu.edu (G.A. Carpenter), sig@cns.bu.edu (S. Martens), ogiogas@cns.bu.edu (O.J. Ogas).

0893-6080/© 2005 Elsevier Ltd. All rights reserved.
DOI: 10.1016/j.neunet.2004.12.003
A human mapping analyst would, in this case, be able to apply a lifetime of experience to resolve the paradox by placing objects in a knowledge hierarchy, and a rule-based expert system could be constructed to codify this knowledge. Alternatively, an analyst could be faced with complex or unfamiliar labels, or the structure of object relationships may vary from one region to the next.

The current study shows how an ARTMAP neural network can act as a self-organizing expert system to derive hierarchical knowledge structures from inconsistent training data. This ability is implicit in the network’s learning strategy, which creates one-to-many, as well as many-to-one, maps of the input space. During training, the system can learn that disparate pixels map to the output class car; but, if similar or identical pixels are later labeled vehicle or man-made, the system can associate multiple classes with a given input. During testing, distributed code activations predict multiple output class labels. A rule production algorithm uses the pattern of distributed predictions to derive a knowledge hierarchy for the output classes. The resulting diagram of the relationships among classes can then guide the construction of consistent layered maps.

Section 2 outlines how distributed coding in the default ARTMAP network supports multi-class prediction. Section 3 describes two remote sensing testbed examples, with sensor data from Monterey, CA, and from the Boston area. Section 4 specifies the algorithm that derives hierarchical knowledge structures from distributed class label predictions, and Section 5 demonstrates system performance of the ARTMAP information fusion system on the Monterey and Boston testbed examples. Section 6 points to the use of the new methods in other application domains. A software implementation of both default ARTMAP and the complete ARTMAP information fusion system is available from http://cns.bu.edu/techlab.

2. Multi-class predictions by ARTMAP neural networks

Adaptive Resonance Theory (ART) neural networks model real-time prediction, search, learning, and recognition. ART networks function both as models of human cognitive information processing (e.g. Carpenter, 1997; Carpenter & Grossberg, 1993; Grossberg, 1980, 1999, 2003; Page, 2000) and as neural systems for technology transfer (e.g. Aggarwal, Xuan, Johns, Li, & Bennett, 1999; Gopal, Woodcock, & Strahler, 1999; Griffith & Todd, 1999; http://cns.bu.edu/techlab). Sites of early and ongoing transfer of ART-based technologies include industrial venues such as the Boeing Corporation and government venues such as MIT Lincoln Laboratory. A review of industrial uses of neural networks (Lisboa, 2001) states: “[The] Boeing...Neural Information Retrieval System (Caudell, Smith, Escobedo, & Anderson, 1994) is probably still the largest-scale manufacturing application of neural networks. It uses [ART] to cluster binary templates of aeroplane parts in a complex hierarchical network that covers over 100,000 items, grouped into thousands of self-organised clusters. Claimed savings in manufacturing costs are in millions of dollars per annum.” At Lincoln Lab, a team led by Waxman developed an image mining system which incorporates a number of models of vision and recognition introduced in the Boston University Department of Cognitive and Neural Systems (BU/CNS) (Streilein et al., 2000; Waxman et al., 2001, 2002). Over the years a dozen CNS graduates have contributed to this effort, which is now located at Alphatech, Inc.

Design principles derived from scientific analyses and design constraints imposed by targeted applications have jointly guided the development of many variants of the basic networks, including fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds, & Rosen, 1992), simplified fuzzy ARTMAP (Kasuba, 1993), ART-EMAP (Carpenter & Ross, 1995), ARTMAP-IC (Carpenter & Markuzon, 1998), Gaussian ARTMAP (Williamson, 1998), and distributed ARTMAP (Carpenter, 1997; Carpenter, Milenova, & Noeske, 1998). Across many variations of these models, a neural computation central to both the scientific and the technological analyses is the ART matching rule (Carpenter & Grossberg, 1987), which represents the interaction between bottom-up sensory inputs and on-center/off-surround top-down learned expectations. This interaction creates a focus of attention which, in turn, determines the nature of stored memories.

While the earliest unsupervised ART (Carpenter & Grossberg, 1987) and supervised ARTMAP networks (Carpenter, Grossberg, & Reynolds, 1991) feature winner-take-all code representations, many of the networks developed since the mid-1990s incorporate distributed code representations. Comparative analyses of these systems have led to the specification of a default ARTMAP network, which features simplicity of design and robust performance in many application domains (Carpenter, 2003). Selection of one particular a priori algorithm is intended to facilitate technology transfer. This network, which here serves as the recognition engine of the information fusion system, uses winner-take-all coding during training and distributed coding during testing. Distributed test outputs have helped improve various methods for categorical decision-making. One such method, in a map production application, compares a baseline mapping procedure, which selects the class with the largest total output, with a procedure that enforces a priori output class probabilities and another one that selects class-specific output thresholds via validation (Parsons & Carpenter, 2003).

Distributed coding supports each method, but the ultimate prediction is one output class per test input. This procedure also specifies a canonical training/testing method which partitions the area in question into four vertical or horizontal strips. A given simulation takes training pixels from two of these strips; uses the validation strip to choose
parameters, if necessary; and tests on the fourth strip. Methods are thus compared with training and test sets that are not only disjoint but drawn from geographically separate locations. This separation tests for generalization to new regions, where output class distributions could typically be far from those of the training and validation sets.

The information fusion techniques developed in the current study modify the baseline mapping procedure by allowing the system to predict more than one output class during testing. A given test pixel either predicts the $N$ classes receiving the largest net system outputs or predicts all classes whose net output exceeds a designated threshold $\Gamma$. A preliminary version of the ARTMAP information fusion system (Carpenter, Martens, & Ogas, 2004) chose a global selection parameter $N$ or $\Gamma$ based on analysis of the validation strip. This method succeeds when most validation and test items share a common number of correct output classes. The new procedure introduced here allows each test exemplar to choose its own number $N$ of output class predictions. This per-pixel filtering method thus does not rely on the strong assumption that the correct number of output classes per item is approximately uniform across the test set.

3. Monterey and Boston testbed examples

An image of the Monterey Naval Postgraduate School (Fig. 1a) has previously served (Parsons & Carpenter, 2003) as the basis of a benchmark testbed developed for classifier comparisons within the context of the Lincoln Lab spatial data mining system (Section 2). Ground truth construction for this supervised learning example specified eight target output classes (red car, other car, roof, road, foot path, grass, tree, other), with pixel subsets located by observation of the Monterey image. In order to maintain a valid comparison of candidate recognition networks, this testbed retained the same feature vectors and some of the target classes (esp. red car) that had previously been used in Lincoln Lab demonstrations of the Monterey image (Ross et al., 2000).

The present study extends the Monterey testbed by designating multiple labels for each ground truth pixel. Namely, red car and other car pixels are also labeled vehicle; road and foot path pixels are also labeled pavement; grass and tree pixels are also labeled vegetation; vehicle, roof, and pavement pixels are also labeled manmade; and vegetation pixels are also labeled natural.

Fig. 1. Testbed images for ARTMAP information fusion methods. (a) Monterey image. Dimensions: 987×1510 pixels (0.5 m resolution) $\cong$500×750 m. (b) Boston image, in false color representation of preprocessed inputs: The city of Revere is at the center, surrounded by (clockwise from lower right) portions of Winthrop, East Boston, Chelsea, Everett, Malden, Melrose, Saugus, and Lynn. Logan Airport runways and Boston Harbor are at the lower center, with Revere Beach and the Atlantic Ocean at the right. The Saugus and Pines Rivers meet in the upper right, and the Chelsea River is in the lower left of the image. Dimensions: 360×600 pixels (15 m resolution) $\cong$5.4×9 km. Each testbed image is divided into four vertical strips: two for training, one for validation (if needed), and one for testing. This protocol produces geographically distinct training and testing areas, to assess regional generalization. Typically, class label distributions vary substantially across strips.
Two or three output class labels were thus assigned to each pixel in the ground truth set. During training, however, the ARTMAP network is never given any information about relationships among the target classes.

A second testbed demonstrates the robustness of the ARTMAP information fusion procedure (Fig. 1b). This example was derived from a Landsat 7 Thematic Mapper (TM) image acquired on the morning of January 1, 2001 by the Earth Resources Observation System (EROS) Data Center, US Geological Survey, Sioux Falls, SD (http://edc.usgs.gov). The 5.4 × 9 km area includes portions of northeast Boston and suburbs. Whereas the resolution of the Monterey image is approximately 0.5 m² in each spectral band, the resolution of the Boston image is 30 m² in six TM bands, 60 m² in two thermal bands, and 15 m² in one Panchromatic band. Urban ground truth labels are therefore necessarily coarser in the Boston example.

The Boston region encompasses mixed urban, suburban, industrial, water, and park spaces. Ground truth pixels are labeled ocean, ice, river, beach, park, road, residential, industrial, water, open space, built-up, natural, man-made. As in the Monterey example, ARTMAP is given no information about relationships among the Boston target classes during training. Note that class relationships may vary from one image to another. In Monterey, for example, natural is equivalent to vegetation. In Boston, the class natural includes water (which in turn includes ocean, ice, and river) and open space (which in turn includes beach and park).

In the Monterey example, the Lincoln Lab preprocessor transformed the spectral bands of the original image into a 20-dimensional input vector for the recognition system. Inputs for the Boston example were similarly generated by a more recent version of this system, called the Neural Fusion Module, which was developed by Waxman and colleagues working in the CNS Technology Laboratory during 2001–2002 (Fay, Ivey, Bomberger, & Waxman, 2003; Waxman et al., 2002). For the Boston image, this Module, implemented on an ERDAS Imagine (http://gis.leica-geosystems.com) platform, produced 41-dimensional input vectors representing local contrast, color, and texture attributes at each pixel.

4. Deriving a knowledge hierarchy from a trained network: predictions, rules, and graphs

The ARTMAP fusion system provides a canonical procedure for assigning to each input an arbitrary number of output classes in a supervised learning setting. Information implicit in the distributed predictions of a trained ARTMAP network (Section 4.1) can be used to generate a hierarchy of output class relationships. To accomplish this, each test pixel first produces a set of output class predictions (Section 4.2). The resulting list of test predictions determines a list of rules \( x \rightarrow y \) which define relationships between pairs of output classes, with each rule carrying a confidence value (Section 4.3). The rules are then used to assign classes to levels, with rule antecedents \( x \) at lower levels and consequents \( y \) at higher levels (Section 4.4). Classes connected by arrows that codify the list of rules and confidence values form a graphical representation of the knowledge hierarchy.

4.1. ARTMAP fusion system training protocol

Although learning in the ARTMAP information fusion system is carried out by a previously defined neural network (Section 2), a number of additional design elements need to be specified to complete the training protocol. This section describes the cross-validation, training set selection, post-processing, and voting procedures employed in the simulation examples reported below.

According to a standardized cross-validation procedure (Section 2), each image is divided into four vertical strips (Fig. 1). In the Boston and Monterey examples, training pixels are drawn from two of the strips and test pixels from another strip. A single system would be trained, for instance, on pixels from Strips 1 and 4 and tested on pixels from Strip 2. Note the challenge presented by the different distributions of classes, such as water, across vertical strips in the Boston image. The strip reserved for validation is not used in the Boston and Monterey examples because the current version of the rule production algorithm has no free parameters that need selection. The simulations reported here are the result of cross-validation across the 12 possible train/test strip combinations for each image. The training/testing protocol also allows for dramatic disparities between class percentages across the whole image. In Monterey, for example, the important class vehicle is sparsely represented compared to tree.

Ground truth labels typically reflect inequities in class distribution across image regions. For example, Strip 1 (left) of the Boston image contains 75 pixels labeled road and no pixels labeled ocean, while Strip 4 (right) contains 19,919 pixels labeled ocean and only 4 pixels labeled road in the ground truth set. In order for the learning system to encode imbalanced exemplars, the training protocol imposes a cap (here set equal to 250) on the maximum number of labels from each class. Early in training, a chosen pixel is associated with each one of its output class labels, presented sequentially in random order. Once a class reaches the cap, however, no more pixels are associated with the label of this class. In the Boston image, for example, road pixels also carry the ground truth label of the highly represented class man-made. Once 250 pixels of any sort have been labeled man-made during training, all subsequent road pixels can be labeled only road.

When an ARTMAP training input activates a coding node \( j \) for the first time, this node is said to become committed, and the output weight \( W_k \) from node \( j \) to the associated output class \( k \) is set equal to 1 for the duration of training (Fig. 2). This procedure partitions the coding nodes according to the output class to which they were first linked. A post-processing training step, which was tested with the distributed ARTMAP network (Carpenter et al., 1998), presents the input-output
pairs once more, this time distributing the activations \( y \) at the coding field and hence also distributing the output predictions. The output weights \( W_{jk} \) are then retrained to minimize the total least-squared error between predicted and actual outputs. This procedure is akin to the second stage of training in a radial basis function network (Moody & Darken, 1989). Final weights \( W_{jk} \) are here computed in batch mode using the Matlab pseudo-inverse function pinv (Moore, 1920; Penrose, 1955).

Finally, many applications benefit from voting across several ARTMAP systems produced by a given training set. This feature derives from the fact that fast learning produces different networks, and hence different error patterns, for different input orderings. Simple voting procedures typically produce improved accuracy compared to that of any single network, with five voters normally serving as a good default choice. Here, the Monterey and Boston examples show results using only one voter per train/test strip combination, both because of the size of the simulations and the accuracy of single-voter networks. Increasing the number of voters remains an option for systems that need to improve test-set predictive accuracy.

4.2. Predictions

A critical aspect of the default ARTMAP network is the distributed nature of its internal code representation, which produces continuous-valued predictions across output classes during testing. In response to a test input, distributed activations in the default ARTMAP coding field send a net signal \( \sigma_k \) to each output class \( k \) (Fig. 2). A winner-take-all method predicts the single output class \( k = K \) receiving the largest signal \( \sigma_k \). Alternatively, a single test input can predict multiple output classes. The per-pixel filtering method employed here allows the output activation pattern produced by each test pixel to determine the number of predicted classes. Namely, if the net signals \( \sigma_k \) projecting to the output classes \( k \) are arranged from largest to smallest, the system predicts all the classes up to the point of maximum decrease in the signal size from one class to the next. This strategy is motivated by the behavior of a hypothetical system that accurately represents all the output classes. In such a system, if a pixel should predict three classes (e.g. road, pavement, man-made), then the output signals \( \sigma_k \) to each of these classes would typically be large compared to those of the remaining classes. The maximum decrease in size would then occur between the third and fourth largest signal, and the per-pixel filtering method would predict three classes.

4.3. Rules

Once each test pixel has produced a set of output class predictions \( \{x, y, \ldots\} \) from its distributed signals \( \sigma_k \), according to the per-pixel selection method, the list of multi-valued test set predictions is then used to deduce a list of output class implications of the form \( x \Rightarrow y \), each carrying a confidence value \( C\% \). This rule creation method is related to the Apriori algorithm in the association rule literature (Agrawal, Imieliński, & Śwami, 1993; Agrawal & Srikant, 1994).

The steps listed below produce the list of rules that label class relationships. The algorithm introduces an equivalence parameter \( e\% \) and a minimum confidence parameter \( c\% \). Rules with low confidence (\( C < c \)) are ignored, with one exception: if all rules that include a given class have confidence below \( c \), then the list retains the rule derived from the pair predicted by the largest number of pixels. Although this ‘no extinction’ clause may produce low-confidence rules, these may occasionally correspond to cases that are rare but important. The user can easily take these exceptions under advisement, since the summary graph displays each confidence value. Two classes \( x \) and \( y \) are treated as equivalent \( (x \equiv y) \) if both rules \( x \Rightarrow y \) and \( y \Rightarrow x \) hold with confidence greater than \( e \). In this case, the class
predicted by fewer pixels is ignored in subsequent computations, but equivalent classes are displayed as a single node on the final rule summary graph.

Reasonable default values set the equivalence parameter \( e \) in the range 90–95% and the minimum confidence parameter \( c \) in the range 50–70%. In all simulations reported here, parameter values were set a priori to \( e = 90\% \) and \( c = 50\% \). Alternatively, \( e \) and \( c \) may be chosen by validation.

The rule creation method below includes illustrative computations from the test set pixels of the Boston example. Pixel numbers, which are averages across the 12 train/test strip combinations, indicate the wide range of predicted output class fractions. The complete Boston rule graph will be seen in Section 5.2.

**Step 1:** List the number of test set pixels predicting each output class \( x \). Order this list from the classes with the fewest predictions to the classes with the most.

<table>
<thead>
<tr>
<th>Boston</th>
<th>Classes ( x )</th>
<th>#(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beach</td>
<td>182</td>
<td></td>
</tr>
<tr>
<td>Ice</td>
<td>1489</td>
<td></td>
</tr>
<tr>
<td>Industrial</td>
<td>2179</td>
<td></td>
</tr>
<tr>
<td>Open space</td>
<td>3742</td>
<td></td>
</tr>
<tr>
<td>Road</td>
<td>7046</td>
<td></td>
</tr>
<tr>
<td>Water</td>
<td>13,395</td>
<td></td>
</tr>
<tr>
<td>Built-up</td>
<td>18,456</td>
<td></td>
</tr>
<tr>
<td>Natural</td>
<td>19,625</td>
<td></td>
</tr>
<tr>
<td>Man-made</td>
<td>37,917...</td>
<td></td>
</tr>
</tbody>
</table>

**Step 2:** List the number of test set pixels \( #(x \text{ and } y) \) simultaneously predicting each pair of distinct output classes. Omit pairs with no such pixels. Order the list so that \( #(x) \leq #(y) \): classes \( x \) observe the order established in Step 1; and for each such class \( x \), classes \( y \) observe the same order.

<table>
<thead>
<tr>
<th>Boston</th>
<th>Class pairs ( x \text{ and } y )</th>
<th>#(x and y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beach and open space</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>Beach and natural</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>Beach and man-made</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Ice and water</td>
<td>1285</td>
<td></td>
</tr>
<tr>
<td>Ice and natural</td>
<td>1394</td>
<td></td>
</tr>
<tr>
<td>Ice and man-made</td>
<td>439</td>
<td></td>
</tr>
<tr>
<td>Industrial and road</td>
<td>249</td>
<td></td>
</tr>
<tr>
<td>Industrial and built-up</td>
<td>1962</td>
<td></td>
</tr>
<tr>
<td>Industrial and man-made</td>
<td>2102...</td>
<td></td>
</tr>
</tbody>
</table>

**Step 3:** Identify equivalent classes, where \( x \equiv y \) if \( #(x \text{ and } y)/#(y) \geq e\% \). Remove from the list all class pairs that include \( x \) (where \( #(x) \leq #(y) \), as in Step 2).

**Step 4:** Each pair remaining on the list produces a rule \( x \Rightarrow y \) with confidence \( C = #(x \text{ and } y)/#(x) \). If Step 3 determined that \( x \equiv y \), record the confidence \( C \geq e \) of each rule in the pair \( \{x \Rightarrow y, y \Rightarrow x\} \).

<table>
<thead>
<tr>
<th>Boston</th>
<th>Rules ( x \Rightarrow y )</th>
<th>Confidence ( x \Rightarrow y )</th>
<th>Converse ( y \Rightarrow x ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beach ( \Rightarrow ) open space</td>
<td>( C = \frac{#(\text{beach and open space})}{#(\text{beach})} = \frac{160}{182} = 88% )</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Beach ( \Rightarrow ) natural</td>
<td>( C = \frac{#(\text{beach and natural})}{#(\text{beach})} = \frac{158}{182} = 87% )</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>Beach ( \Rightarrow ) man-made</td>
<td>( C = \frac{#(\text{beach and man-made})}{#(\text{beach})} = \frac{32}{182} = 18% )</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>Ice ( \Rightarrow ) water</td>
<td>( C = \frac{#(\text{ice and water})}{#(\text{ice})} = \frac{1,285}{1,489} = 86% )</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Ice ( \Rightarrow ) natural</td>
<td>( C = \frac{#(\text{ice and natural})}{#(\text{ice})} = \frac{1,394}{1,489} = 94% )</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Ice ( \Rightarrow ) man-made</td>
<td>( C = \frac{#(\text{ice and man-made})}{#(\text{ice})} = \frac{439}{1,489} = 29% )</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Industrial ( \Rightarrow ) road</td>
<td>( C = \frac{#(\text{industrial and road})}{#(\text{industrial})} = \frac{249}{2,179} = 11% )</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Industrial ( \Rightarrow ) built-up</td>
<td>( C = \frac{#(\text{industrial and built-up})}{#(\text{industrial})} = \frac{1,962}{2,179} = 90% )</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Industrial ( \Rightarrow ) man-made</td>
<td>( C = \frac{#(\text{industrial and man-made})}{#(\text{industrial})} = \frac{2,102}{2,179} = 96% )</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
**Step 5:** Remove from the list all rules with confidence $C < c$. Exception *(no extinction)*: If all rules that include a given class have confidence below the minimum confidence $c$, then retain the rule or rules $x \Rightarrow y$ with maximal $\#(x \text{ and } y)$. 

<table>
<thead>
<tr>
<th>Boston</th>
<th>High confidence rules $x \Rightarrow y$</th>
<th>$C$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beach $\Rightarrow$ open space</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>Beach $\Rightarrow$ natural</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>Ice $\Rightarrow$ water</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>Ice $\Rightarrow$ natural</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>Industrial $\Rightarrow$ built-up</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>Industrial $\Rightarrow$ man-made</td>
<td>96</td>
<td></td>
</tr>
</tbody>
</table>

**Step 6:** The following optional information may be useful for purposes of analysis.

(a) List rules removed in Step 5 that have confidence in a marginal range, say $20\% \leq C < c$.

**Boston:** As computed in Step 4, the (incorrect) marginal rule ice $\Rightarrow$ man-made has confidence $C = 29\%$.

(b) List class pairs $x$ and $y$ (from Step 2) with equivalence values in a marginal range. For example, list the rule pairs $\{x \Rightarrow y, y \Rightarrow x\}$ for class pairs $x$ and $y$ for which $c \leq \frac{\#(x \text{ and } y)}{\#(y)} < e$.

4.4. **Graphs**

A directed graph summarizes the list of implication rules derived in Section 4.3. These rules suggest a natural hierarchy among output classes, with antecedents sitting below consequents. For each rule $x \Rightarrow y$, class $x$ is located at a lower level of the hierarchy than class $y$, according to the iterative algorithm below. Once each class is situated on its level, a listed rule $x \Rightarrow y$ produces an arrow from $x$ to $y$. Each rule’s confidence is indicated on the arrow, with lower-confidence rules (say $C < 90\%$) having dashed arrows. For arrows with no displayed confidence values, $C = 100\%$.

The following procedure assigns each output class to a level.

Top level: Items that appear only as consequents $y$.

Level 1: Classes that do not appear as consequents in any rule.

Remove from the list all rules $x \Rightarrow y$ where $x$ is in Level 1.

Next level: Classes that do not appear as consequents in any remaining rule.

Remove from the list all rules $x \Rightarrow y$ where $x$ is in this level.

Iterate: Repeat until all rules have been removed from the list.

Note that Level 1 includes classes that do not appear in any rule as well as those that appear only as antecedents.

5. **Graphical representations of knowledge hierarchies**

Graphs in Figs. 3 and 4 depict the implication rules, hierarchy levels, and confidence values derived for the Monterey and Boston examples.

5.1. **Monterey testbed**

Fig. 3 depicts the graph of the Monterey example. The ARTMAP fusion system here produces the complete set of correct rules, each with confidence values at least 67%. The next lower confidence (for the rule grass $\Rightarrow$ tree) is $C = 42\%$. This indicates that any value of the minimum confidence parameter $c$ between 42 and 67% would have given identical results. Note that the sparsest classes (red car, other car, path) produce rules that carry the lowest confidence values.

Note, too, that the class natural is correctly identified as equivalent to vegetation in the Monterey testbed, a result that would be the same for any value of the equivalence parameter $e$ between 83 and 95%. A value of $e$ below 83% would have equated road with pavement and tree with natural. That is, a higher-level class tends to merge with a lower-level class that represents a large relative majority of that class within this image.

---

**Fig. 3.** Graphs represent rules, confidence values, and knowledge hierarchies derived by the ARTMAP information fusion system. For the Monterey example shown here, the graph displays all the correct rules, equivalence relations, and class levels. The optional Rule Step 6a also points to three marginal rules with confidence in the range $20\% \leq C < c$: grass $\Rightarrow$ tree ($C = 42\%$), other car $\Rightarrow$ roof ($C = 34\%$), and vehicle $\Rightarrow$ road ($C = 27\%$); and to two marginal equivalence relations: road $\Rightarrow$ pavement ($C = 91\%$)/pavement $\Rightarrow$ road ($C = 83\%$) and tree $\Rightarrow$ natural ($C = 98\%$)/natural $\Rightarrow$ tree ($C = 83\%$).
Fig. 4. For the Boston example, the ARTMAP fusion system correctly produces all class rules and levels, and no equivalence relations. Rule Step 6a points to eight marginal rules with confidence 20% ≤ C < c, the two with confidence C ≥ 30% being: open space ⇒ man-made (C = 38%) and park ⇒ man-made (C = 36%); and to three marginal equivalence relations: park ⇒ open space (C = 86%); open space ⇒ park (C = 85%), residential ⇒ built-up (C = 82%); built-up ⇒ residential (C = 78%), and water ⇒ natural (C = 99%); natural ⇒ water (C = 68%).

5.2. Boston testbed

Fig. 4 shows that for the Boston testbed ARTMAP information fusion again places each class in its correct level and discovers all the correct rules. This example illustrates the robustness of the method across scales of pixel resolution.

Note that, while the classes natural and vegetation are equivalent in the Monterey example, in the Boston image, natural includes water, ice, beach, etc. in addition to vegetation, here called park. This difference illustrates that the correct class label hierarchy may be image-specific and cannot necessarily be defined by an analyst without ad hoc knowledge of the location. In the Boston example, if the equivalence parameter e were below 85%, then park would become equivalent to open space. As in the Monterey example, as e is reduced, lower level classes reasonably begin to merge with higher level classes in which they represent large relative majorities.

In the Boston graph, the rule with lowest confidence is residential ⇒ built-up (C = 82%). The marginal rule with next lowest confidence is open space ⇒ man-made (C = 38%). This again indicates the robustness of the minimum confidence parameter c, which is set equal to 50% throughout: any value of c between 38 and 82% would have produced an identical class hierarchy and rule set for the Boston image.

6. Conclusion: ARTMAP information fusion

The ARTMAP neural network processes one-to-many mappings from input vectors to output classes, as well as the more traditional many-to-one mappings, as the normal product of its supervised learning laws. During training, a given input may learn associations to more than one output class. Some of these associations could be erroneous: when different observers label an image dog, coyote, or wolf, at most one of these classes is correct. Inconsistent data may, however, be completely correct, as when observers variously label the image wolf, mammal, and carnivore. By resolving such paradoxes during everyday knowledge acquisition, humans naturally infer complex, hierarchical relationships among classes without explicit specification of the rules underlying these relationships. One-to-many learning allows the ARTMAP information fusion system to associate any number of output classes with each input. Although inter-class information is not given with the training inputs, the system readily derives knowledge of the rules, confidence estimates, and multi-class hierarchical relationships from patterns of distributed test predictions.

The testbed examples from the Monterey and Boston images demonstrate how ARTMAP information fusion resolves apparent contradictions in input pixel labels by assigning output classes to levels in a knowledge hierarchy. This methodology is not, however, limited to the image domain illustrated here, and could be applied, for example, to infer patterns of drug resistance or to improve marketing suggestions to individual consumers. One such pilot study has created a hypothetical set of relationships among protease inhibitors, based on resistance patterns from genome sequences of HIV patients.

Acknowledgements

This work was supported by research grants from the Air Force Office of Scientific Research (AFOSR F49620-01-1-0423), the National Geospatial-Intelligence Agency (NMA 201-01-1-2016), the National Science Foundation (NSF SBE-0354378), and the Office of Naval Research (ONR N00014-01-1-0624); by postdoctoral fellowships from the National Geospatial-Intelligence Agency and the National Science Foundation for Siegfried Martens (NMA 501-03-1-2030 and NSF DGE-0221680); and by a Department of Homeland Security graduate fellowship for Ogi Ogas. Sucharita Gopal, Junchang Ju, and Mutlu Ozdogan provided remote sensing data, and Suhas Chelian and Brad Rhodes helped create the ground truth sets for the Boston testbed.
References


