This write-up describes the key computational steps in the Fuzzy ARTMAP training and test cycles with the canonical Circle-in-the-Square (CIS) benchmark as the example dataset that the ARTMAP network learns to categorize. The CIS benchmark is a two-class, two-dimensional feature dataset with data points inside the unit circle assigned label blue, and the remaining data points (in the unit square) assigned label red.

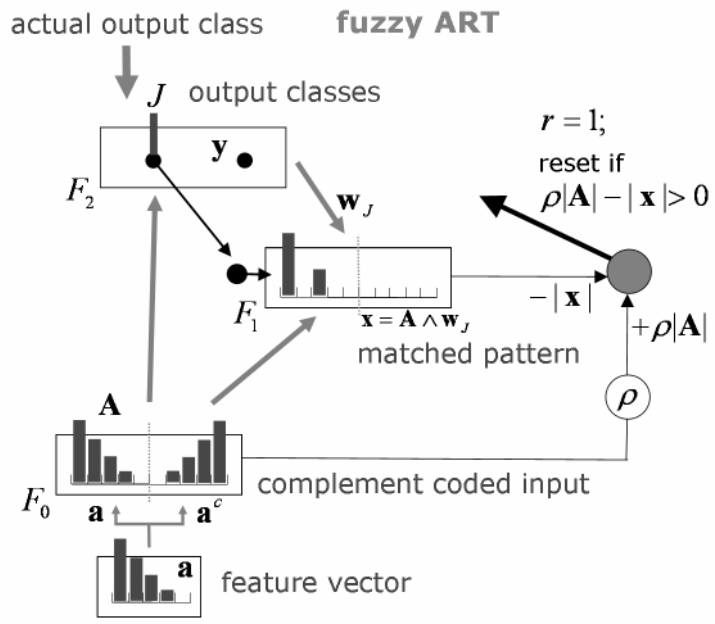

Figure 1: Fuzzy ARTMAP network.

Supervised Learning

Supervised learning is a learning paradigm where a function or mapping is learnt from training data. The *training dataset* provided to the supervised learning system consists of pairs of input *feature vectors* and desired *labels* or continuous-valued function outputs. The accuracy of the learned mapping is typically evaluated by measuring output

prediction accuracy on a *test dataset* containing pairs of feature vectors and labels not present in the training dataset.

Supervised Learning Example

An ocean trawler faced with the shortage of manual labor wants a system to automate the process of classifying whether the object on the conveyor belt is a fish or a shell, the two objects usually mixed together in the day's catch (Figure 2).

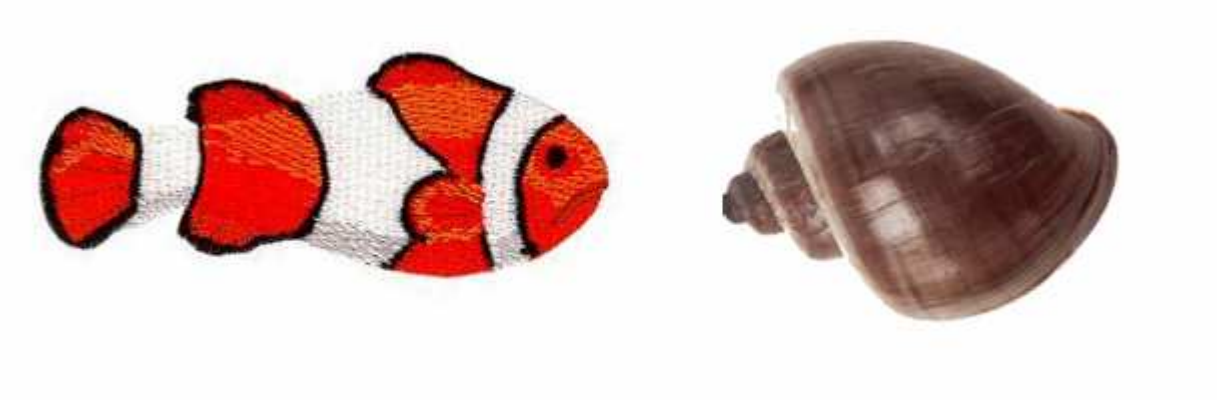

Figure 2: A supervised learning system on a trawler needs to learn how to correctly label the object on the conveyor belt as a fish or a shell.

However, the supervised learning system takes only the color, and the width and height of the fish or shell into account (Figure 3). That is, the input to the supervised learning system is a feature vector of length three [*color, height, width*] and the corresponding label (*Fish* or *Shell*). Classification problems requiring the determination of one of two possible labels, as in this case, are called two-class problems. Supervised learning problems with multiple output labels are also commonly encountered.

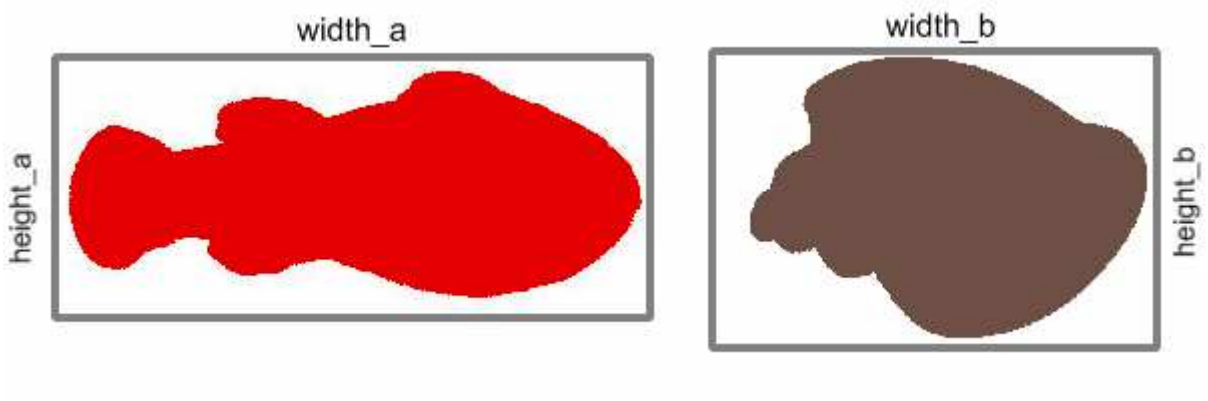

Figure 3: The supervised learning system only has access to the color, and the width and height of the object on the conveyor belt. That is, the supervised learning system learns the mapping of a three-dimensional vector [*color, height, width*] to one of two labels, *Fish* and *Shell*.

Fuzzy ARTMAP

Fuzzy ARTMAP is a neural network architecture that performs incremental supervised learning of recognition categories (labels) and multi-dimensional maps in response to input feature vectors. Fuzzy ARTMAP achieves a synthesis of fuzzy logic and Adaptive Resonance Theory (ART) neural networks. The mappings in Fuzzy ARTMAP are vectors in the same feature space as the inputs to the network. These vectors are analogous to long-term memories and are also referred to as *weight vectors (weights)* in machine learning terminology.

Complement Coding

ARTMAP networks employ a preprocessing step called complement coding, which allows for a representation of both, the presence and absence of features. The three individual features in [*color, height, width*] are scaled to lie in the [0,1] range and complement coding results in a six-dimensional input vector [*color, height, width* 1-*color,1- height,1- width*]. The complement coding module may also be downloaded separately from the Synapse repository.

Complement coding allows for the geometric representation of weights in Fuzzy ARTMAP as rectangles or **category boxes**. Hyper-dimensional rectangles of point width and height are called **point-boxes**. For the sake of clarity in presenting the internal geometry of Fuzzy ARTMAP training, a sequence of training on a **two-dimensional** example is presented here.

Point-box creation

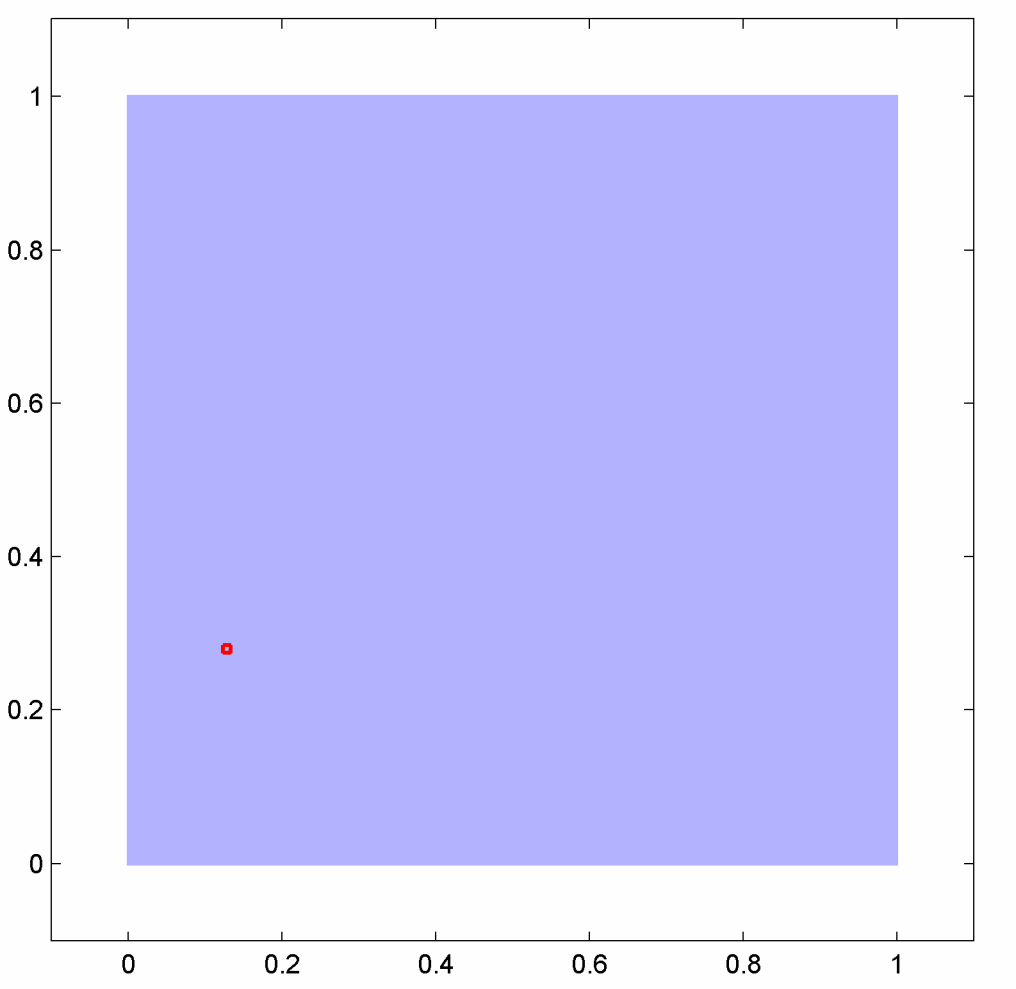

Figure 4: The input feature vector and the corresponding geometric representation of the newly created category node are depicted in the two-dimensional input feature space. For clarity, point boxes are shown with a small, non-zero area and slightly offset from the location of the corresponding input feature vector.

The two-dimensional (M = 2) training feature vector **a** presented to the network is complement coded to produce a 4-dimensional (2M = 4) input vector **A.** In the case of the first feature vector in the CIS dataset, **a** = [.12 .27] is complement coded as **A** = [.12 .27 .88 .73]. All nodes at F_2 are *uncommitted* at the start of training and thus the first input vector creates a **point-box** corresponding to the position of the vector **a** in the input space. This is shown in Figure 2.

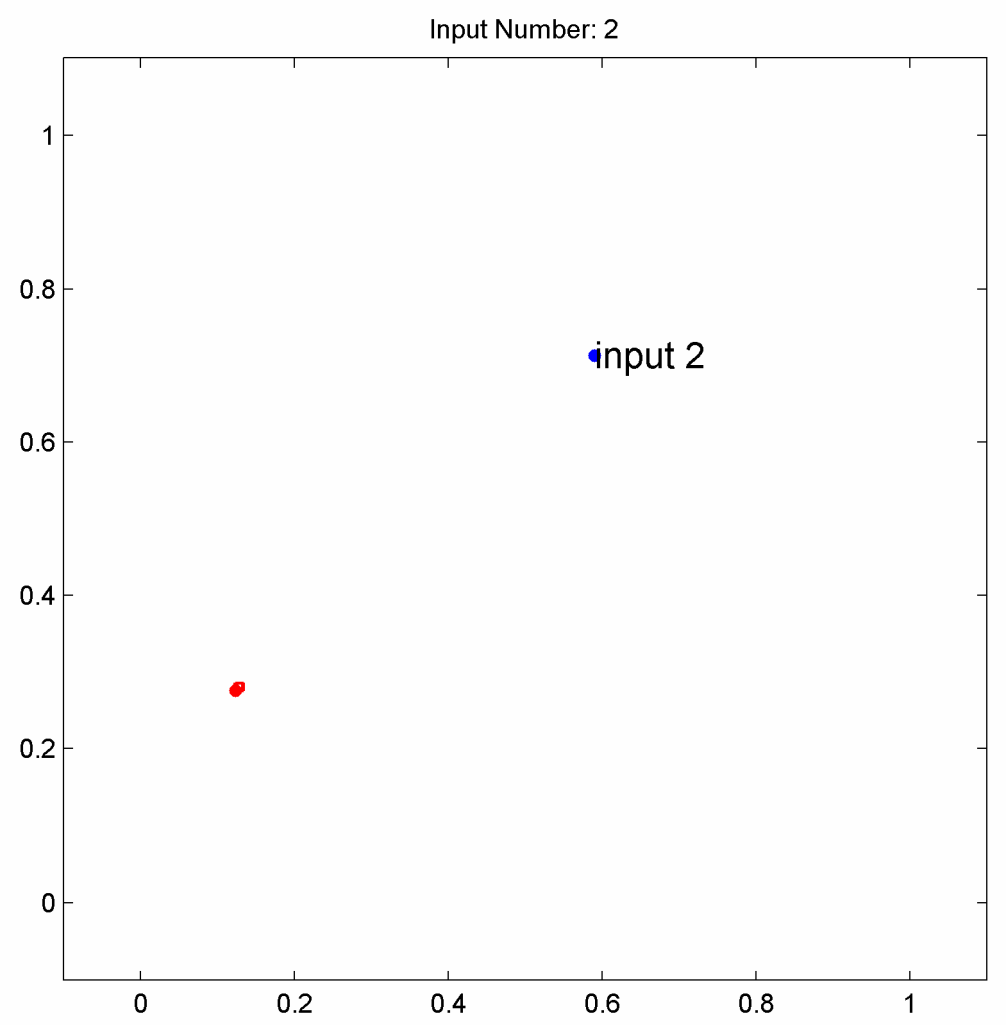

Figure 5: The second input feature vector **a** = [.59 .71] lies inside the circle and is labeled as class Blue.

On presentation of the second training vector, the single committed category node is evaluated as a possible candidate for learning. This node is rejected as it belongs to the wrong class and a second F_2 node mapped to the blue class is committed following the presentation of the second training vector (Figures 5 and 6). F_2 nodes are committed when all other committed category nodes belong to the wrong fail or belong to right class but fail the vigilance test.

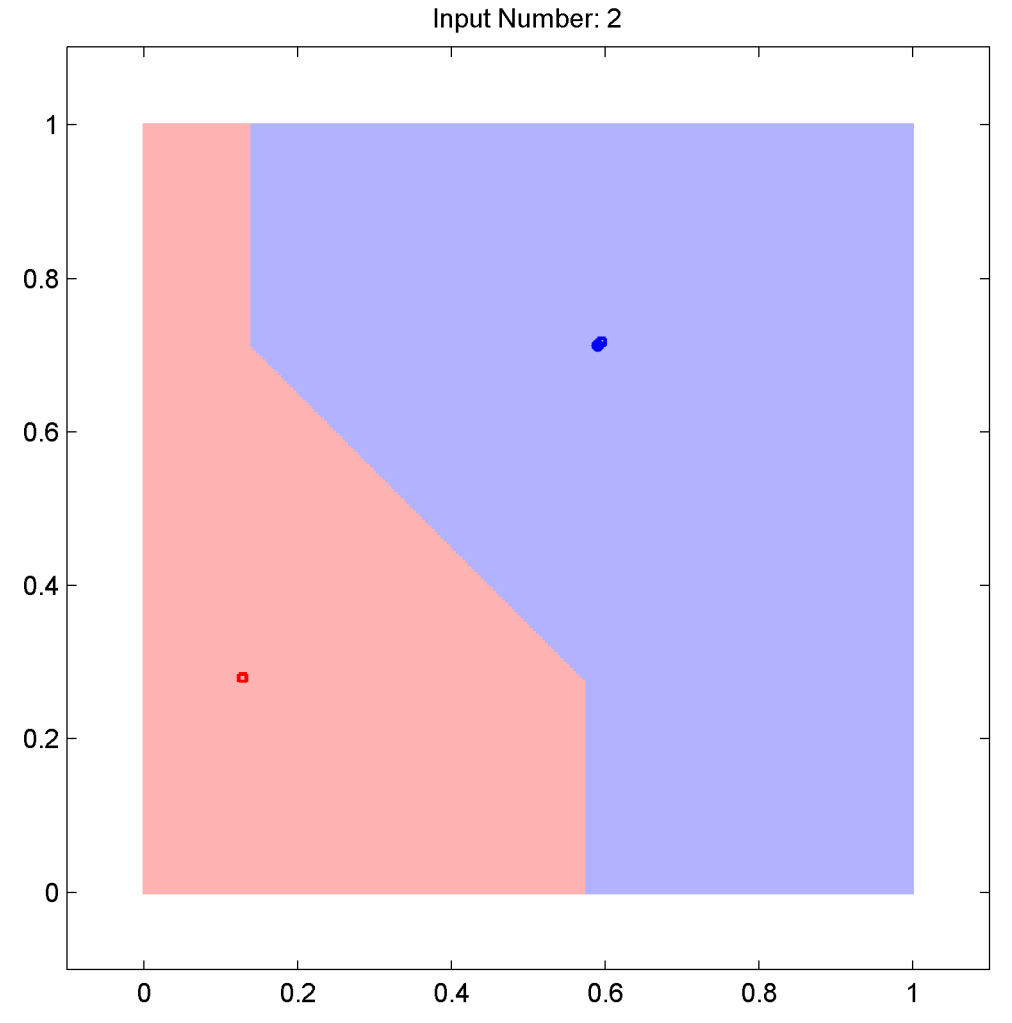

Figure 6: A second category node mapping to class Blue (circle) is committed. The two committed nodes would results in a labeling of the inputs space as shown by the two adjoining colored segments.

Update of F_2 weights

The third training vector **a** = [.41 .65] also lies inside the circle and is thus labeled as Blue (Figure 8). Committed coding nodes are evaluated in the order of the signal amplitudes from the input field. $T_J = |\mathbf{A} \wedge \mathbf{w}_J| + (1-\alpha)(M - |\mathbf{w}_J|)$. Accordingly, the second category node, committed during the previous input presentation, is chosen and the corresponding weight is modified to $\mathbf{w}_2^{new} = \beta(\mathbf{A} \wedge \mathbf{w}_2^{old}) + (1-\beta)\mathbf{w}_2^{old}$.

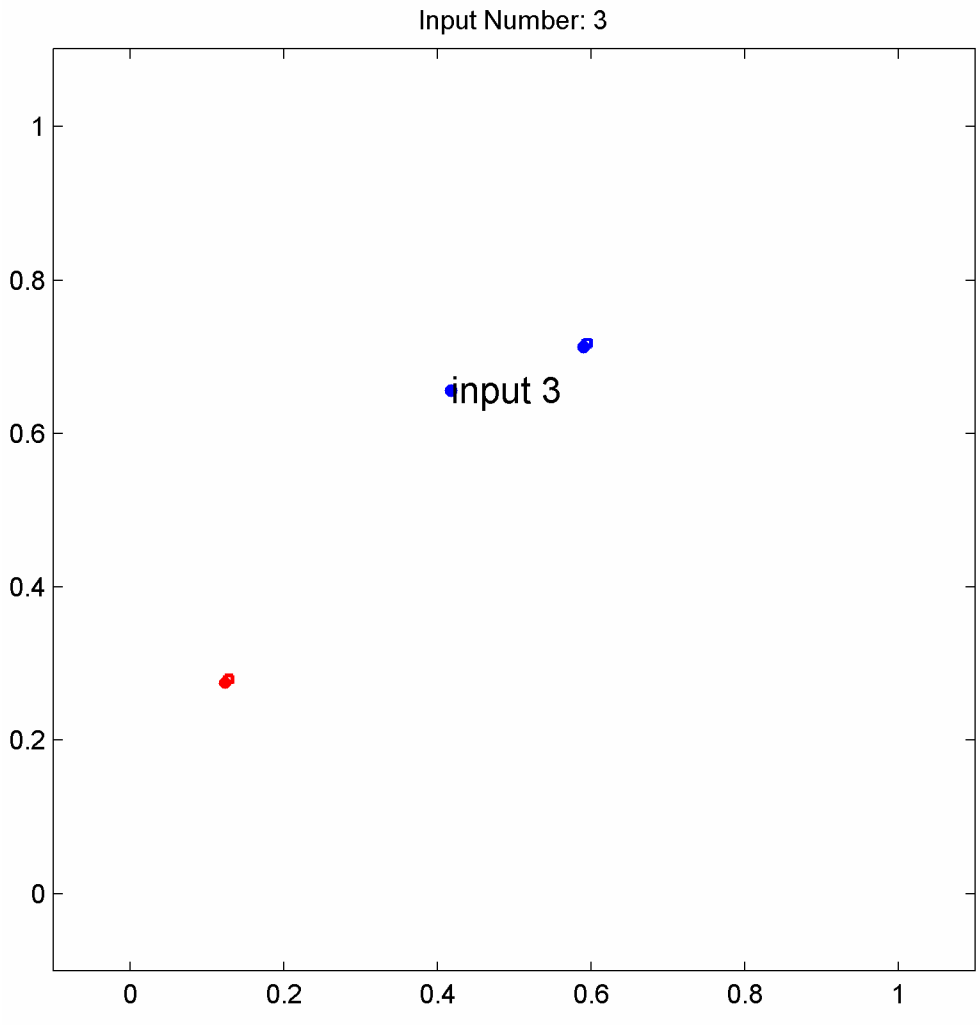

Figure 7: The third input **a** = [.41 .65] picks the second committed category node which maps to class Blue.

Fast updates, as shown in these simulation results, are obtained with a β value of 1. The result of a fast update is the modification of an existing category box such that the updated rectangle subsumes the input vector (Figure 8).

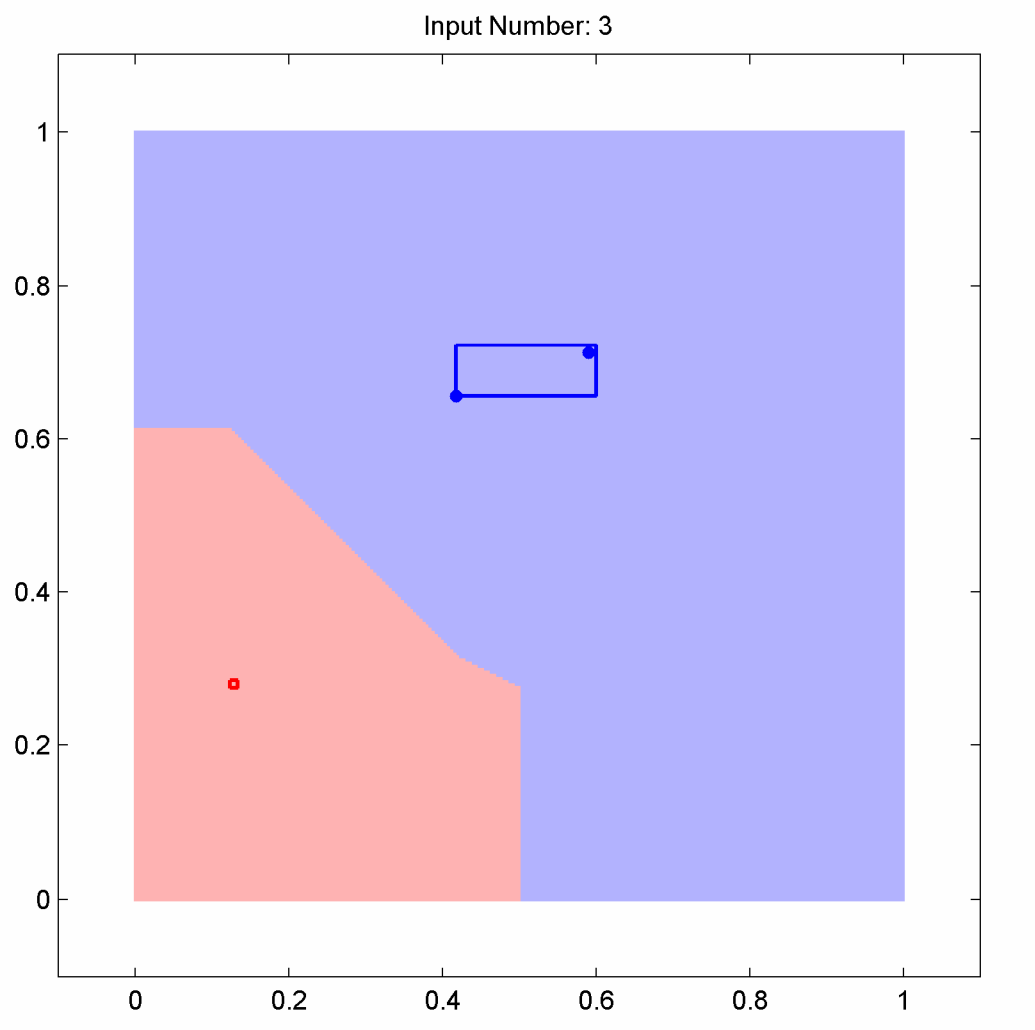

Figure 8: The partitioning of the input space observed in Figure 6 is modified as a result of the weight update following the presentation of input 3.

Presentation of input 4 results in another update of $\mathbf{w}_2$ as show in Figure 9.

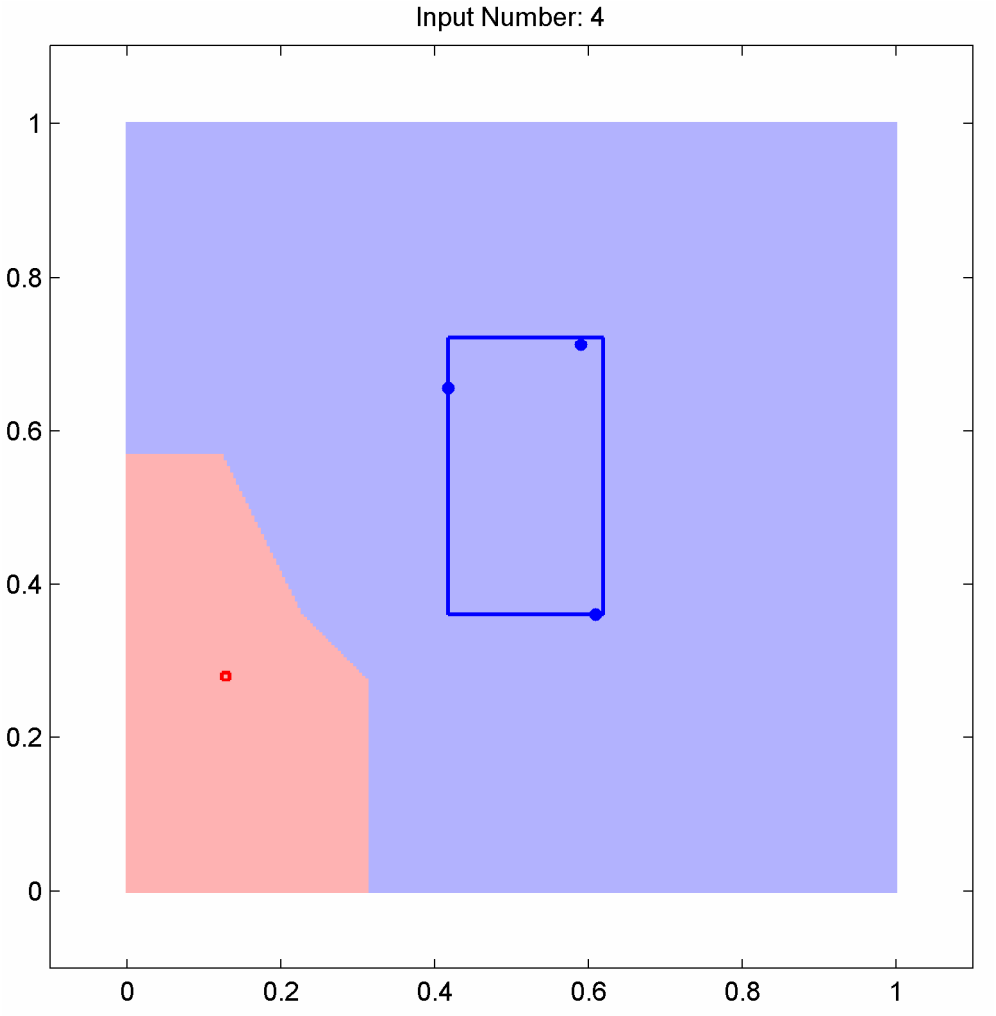

Figure 9: The partitioning of the input space observed in Figure 8 is again modified as a result of the weight update following the presentation of input 4.

The fifth input **a** = [.97 .20] lies outside the circle and is labeled Red (Figure 10). Following presentation, the second category node is chosen first according to the choice function, but is rejected as it codes the opposite class blue (Figure 11). Match tracking

results in an increase of vigilance to $\rho = \frac{|\mathbf{A} \wedge \mathbf{w}_J|}{M}$. In this particular instance, the first committed category node passes the vigilance test and the weight is updated (Figure 12).

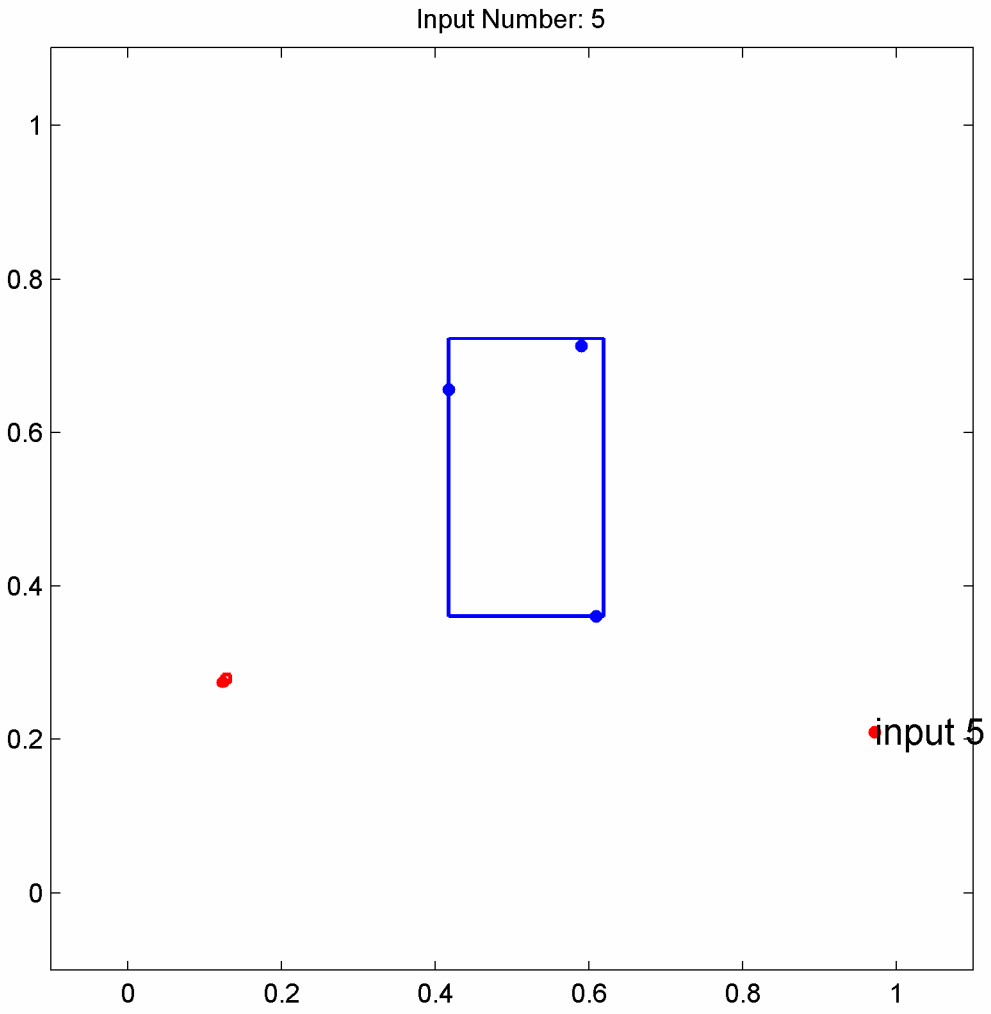

Figure 10: The fifth input **a** = [.97 .20], which lies outside the circle, picks the second committed category node which maps to class Blue.

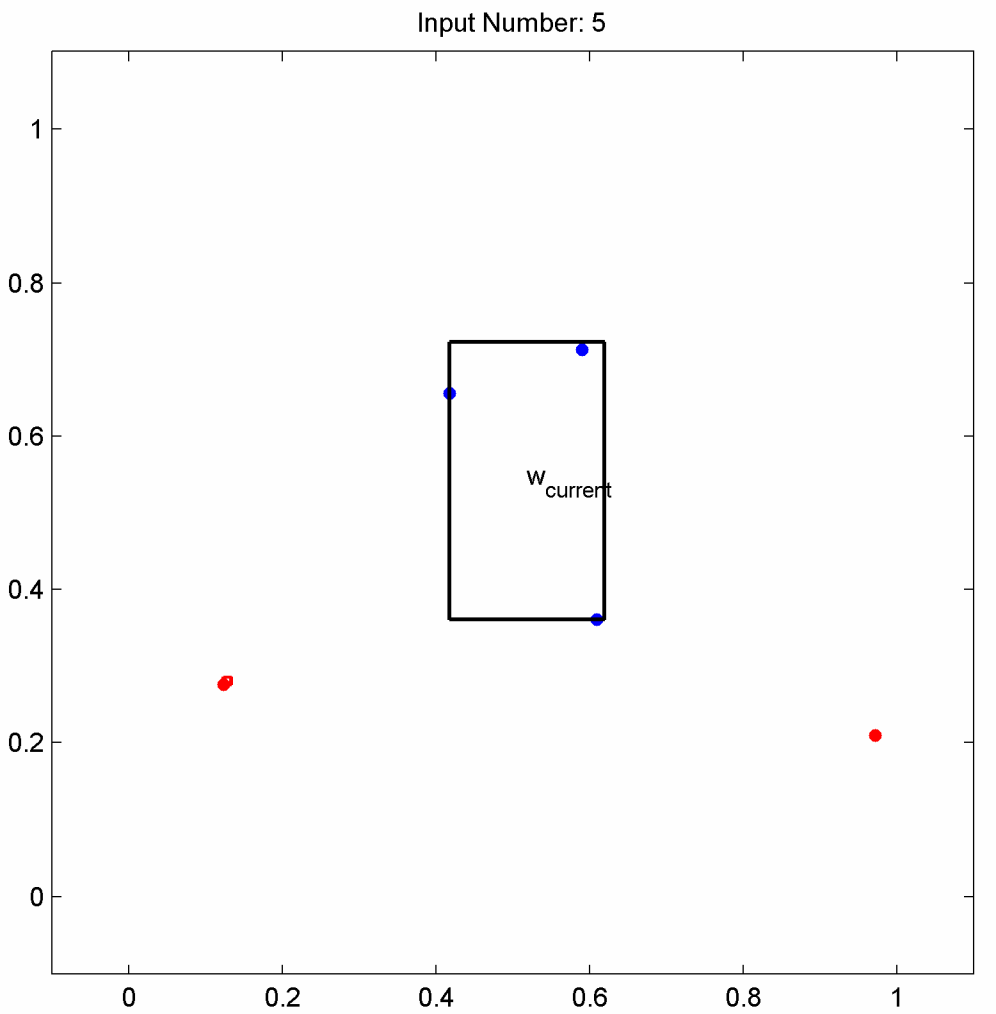

Figure 10: The second committed category node is chosen first and rejected as it belongs to the wrong class.

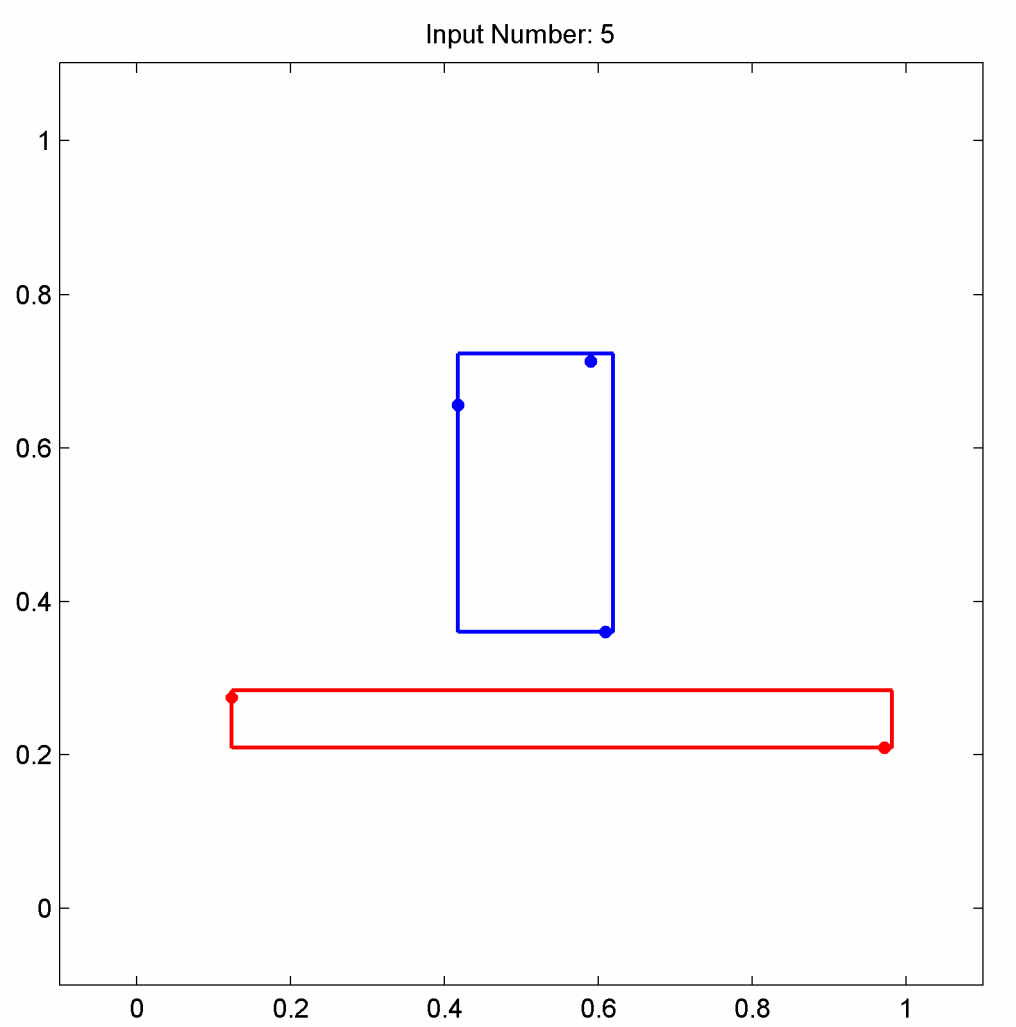

Figure 11: The first committed category node, mapping to the right class, passes the vigilance test and learning occurs.

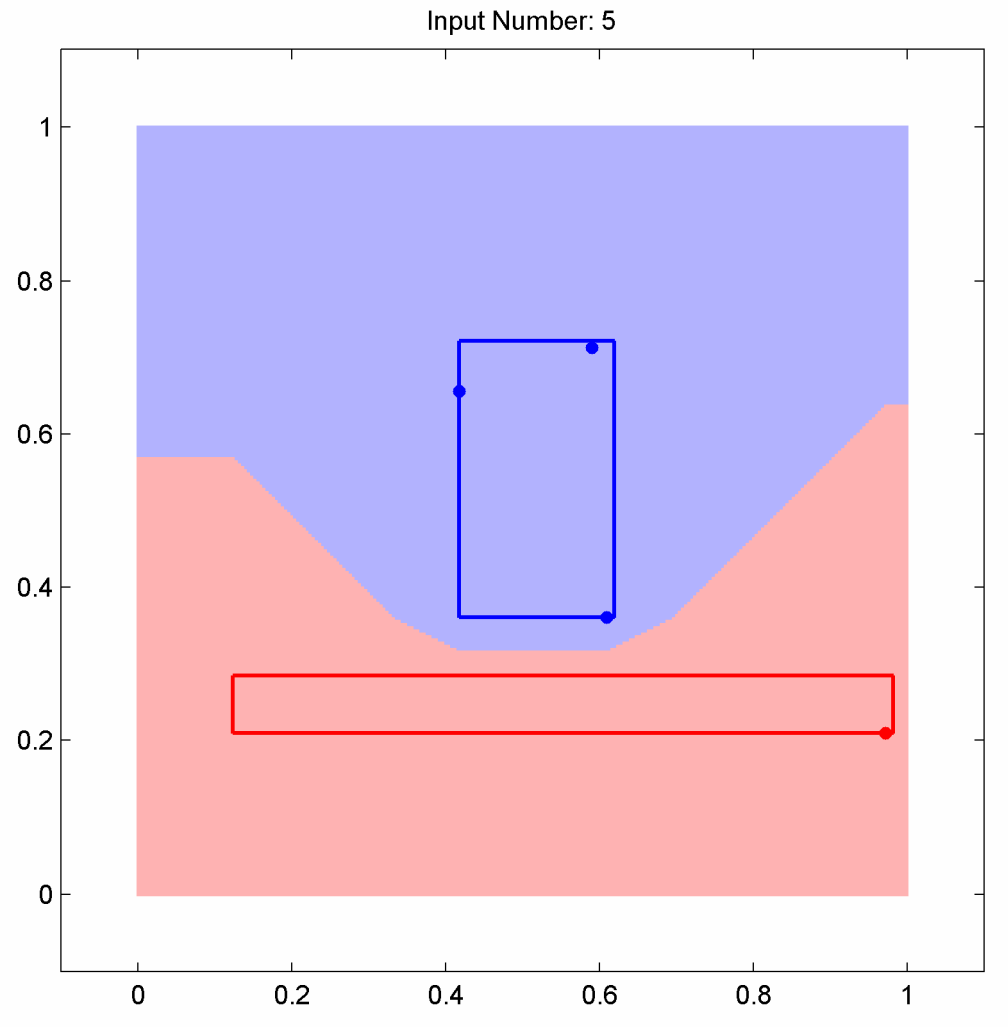

Figure 12: The partitioning of the input space as a result of weight update occurring at the first category node.

Figure 13 depicts the state of committed category nodes at the presentation of the 70th training point. While the circular shape has not yet been attained for the blue class, the configuration of committed nodes now combine to partition the space into a polygonal inner segment categorized as Blue (Circle) and the outer region categorized as Red (Square).

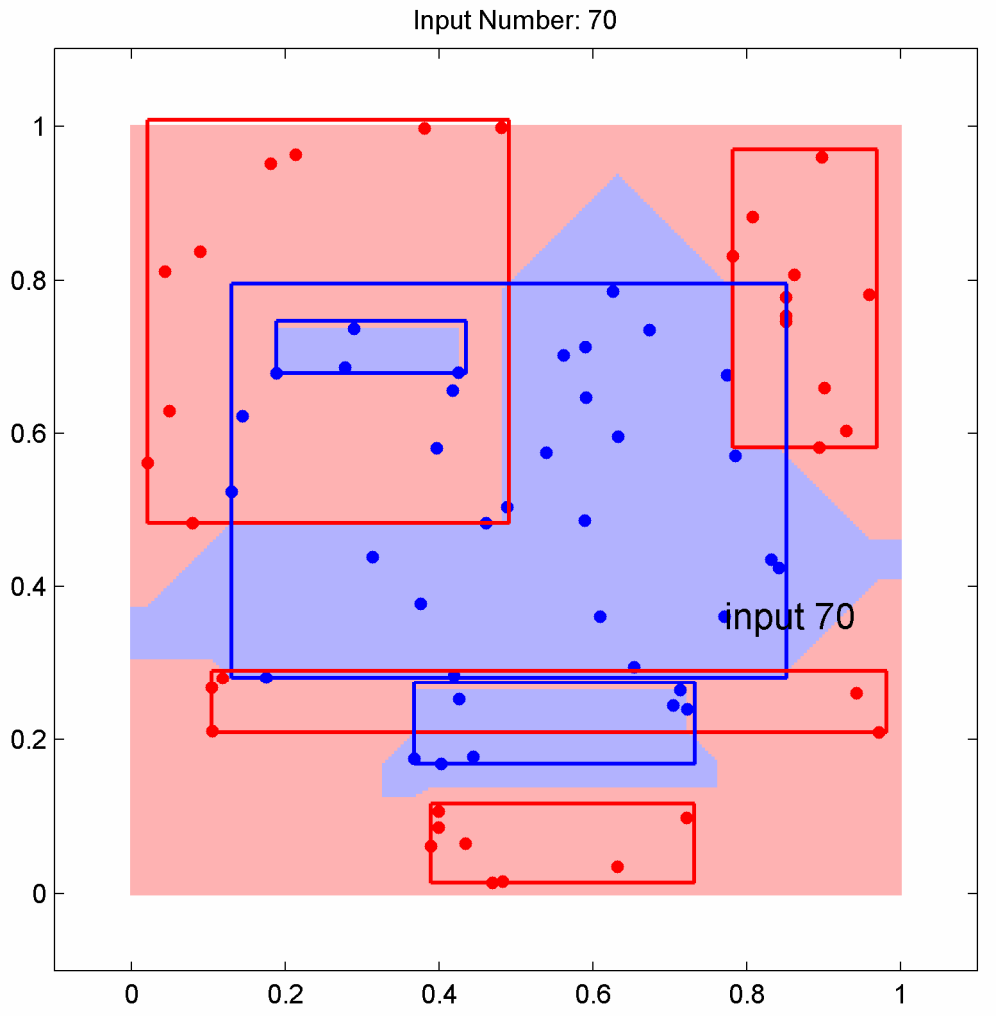

Figure 13: The partitioning of the input space at the start of the 70th training feature vector presentation.

The accuracy of this partitioning is made clearer in Figure 14, which depicts the partitioning of the feature space and also highlights regions (in darker shades or red and blue) where the ARTMAP categorized label is incorrect.

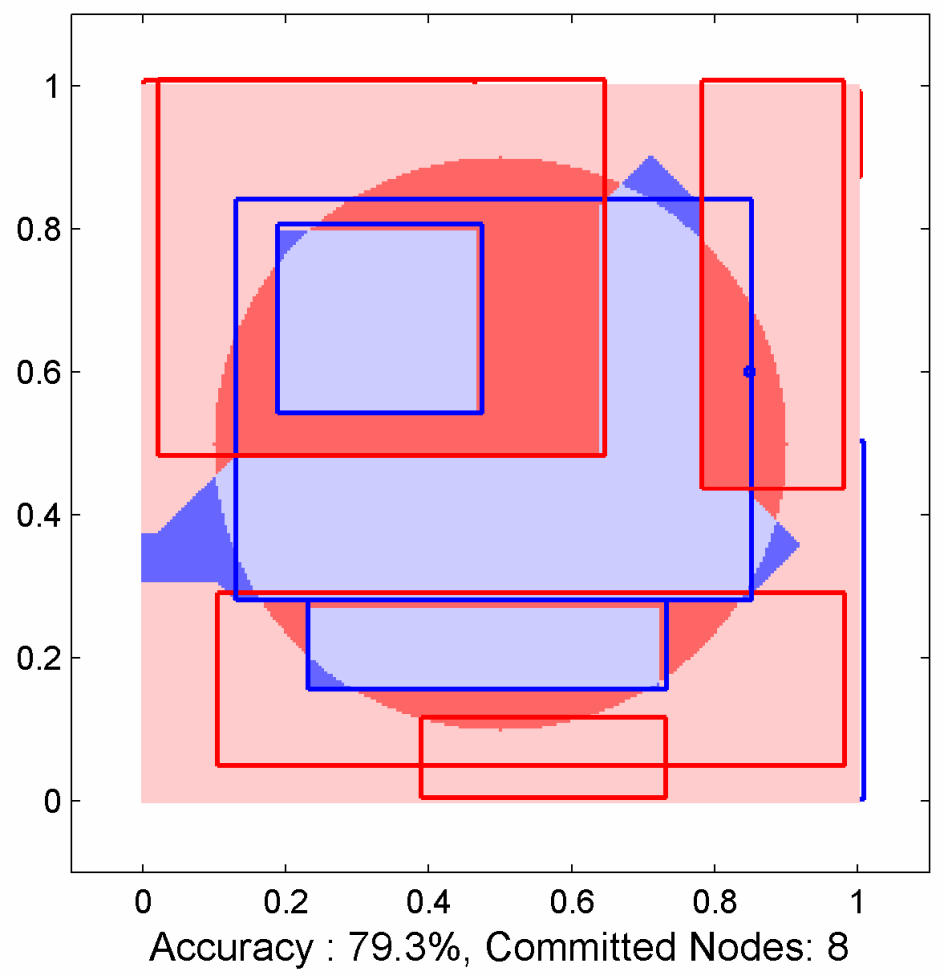

Figure 14: The partitioning of features space after training with 100 randomly chosen training feature vectors. Darker shades of red and blue indicate regions where the ARTMAP network classifies incorrectly. Overall accuracy is 79.3%, or, 79.3% of the feature space is categorized correctly by the Fuzzy ARTMAP network.

Fast or slow learning

All the figures and simulations described until this point were performed with β = 1 in the equation $\mathbf{w}_J^{new} = \beta\left(\mathbf{A} \wedge \mathbf{w}_J^{old}\right) + (1-\beta)\mathbf{w}_J^{old}$, thus learning maximally on each presentation. What effect does slow learning have? (β < 1)

Figure 15 depicts the feature space partitioning achieved with the exact same sequence of training vector presentation, but with a β value of 0.1. As category nodes are not allowed to expand to the fullest extent when learning for a particular training presentation, the category boxes stay small. As a result, more category nodes are committed initially, resulting in a greater final number of committed nodes (16, as opposed to 8 with β = 1). However, classification accuracy is also increased as a result of reduced overlap.

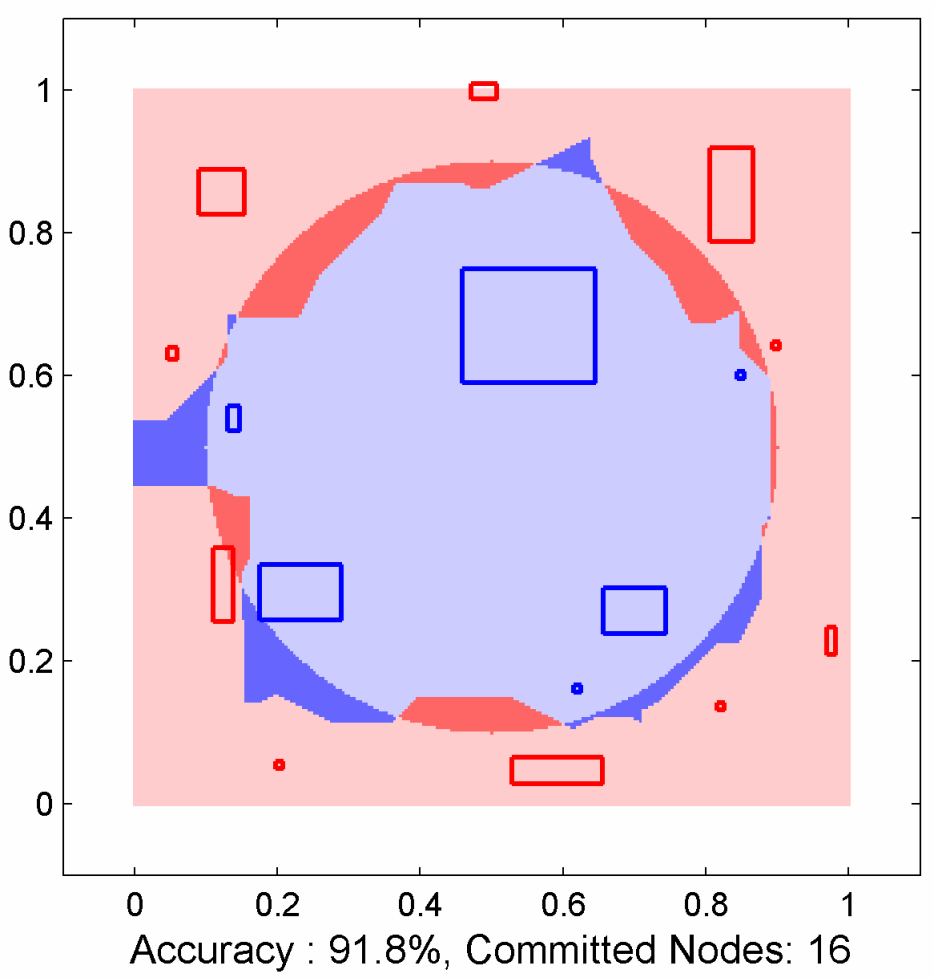

Figure 15: The partitioning of features space after training with the same sequence of 100 training feature vectors, but with β = 0.1.

The difference in learning accuracy and overlap is also seen with arbitrary partitioning of the feature space as in the Checkerboard benchmark. Figures 15, 16, and 17 depict the feature space partitioning and committed category boxes for β values of 1, 0.5, and 0.1 respectively.

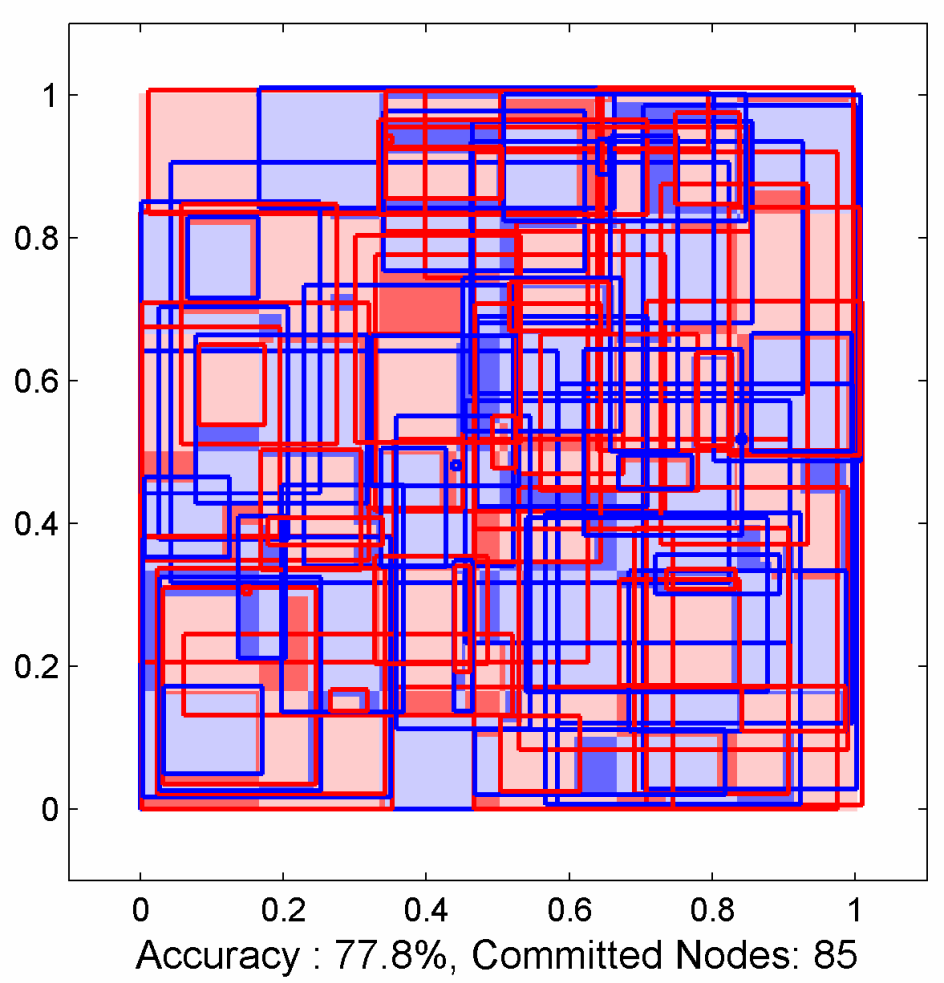

Figure 14: Checkerboard benchmark: The partitioning of features space after training with a random sequence of 2000 training feature vectors, β = 1.

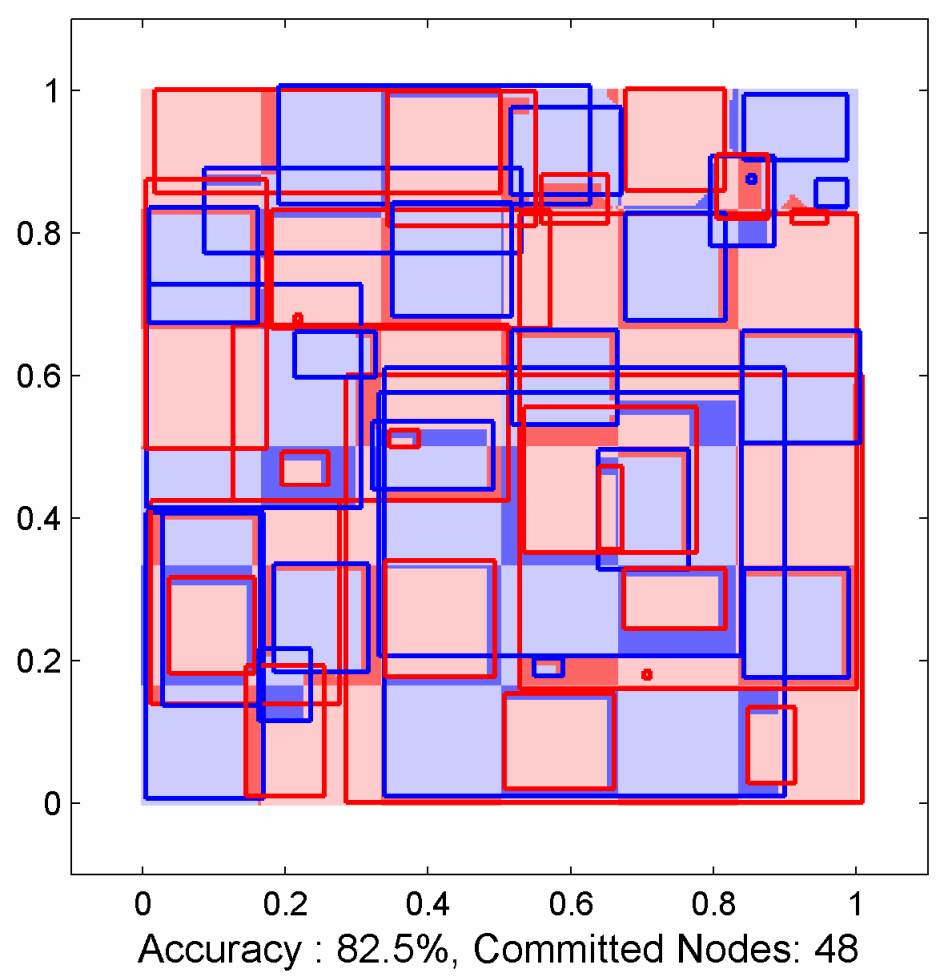

Figure 17: Checkerboard benchmark: The partitioning of features space after training with a the same random sequence of 2000 training feature vectors, but with β = 0.5. Higher classification accuracy is achieved with a lower number of committed category nodes.

The best balance of accuracy and number of committed coding nodes is achieved with a β value of 0.5. Overlap of category boxes can be visually estimated from the number of intersections of the category boxes. Overlap, as would be predicted, is highest in fast learning where category boxes expand maximally and lowest with β = 0.1, when a large number of coding nodes remain as point boxes after being committed (27 out of 182) or learn once (137 out of 182).

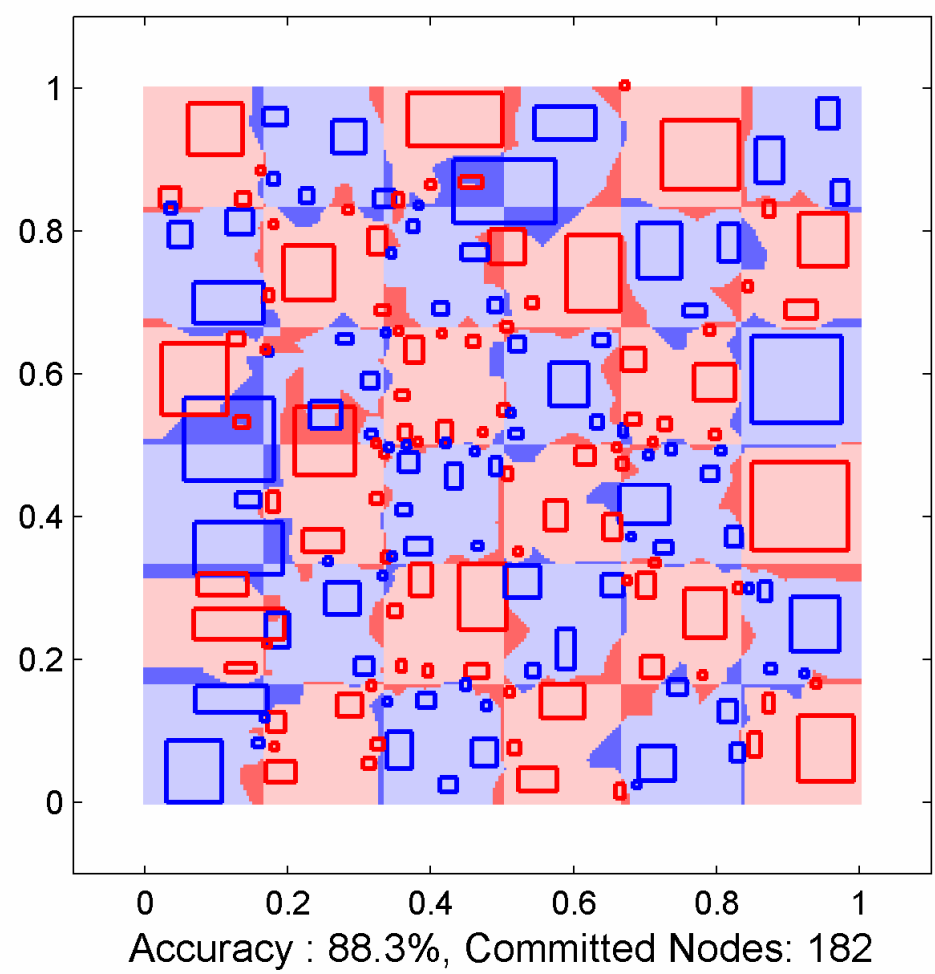

Figure 18: Checkerboard benchmark: The partitioning of features space after training with a the same random sequence of 2000 training feature vectors, but with β = 0.1. Higher classification accuracy is achieved, but with a higher number of committed category nodes.

Confusion matrix depiction for multi-class classification problems

Benchmarks like the Boston dataset have multiple classes. Consequently, learning dynamics cannot be represented geometrically as done with the two-class problems above. However, a confusion matrix depicting the predicted class label versus the actual (ground-truth) class label, as shown in Figure 19, can help in determining if the classifier is overtraining or over-predicting a specific class.

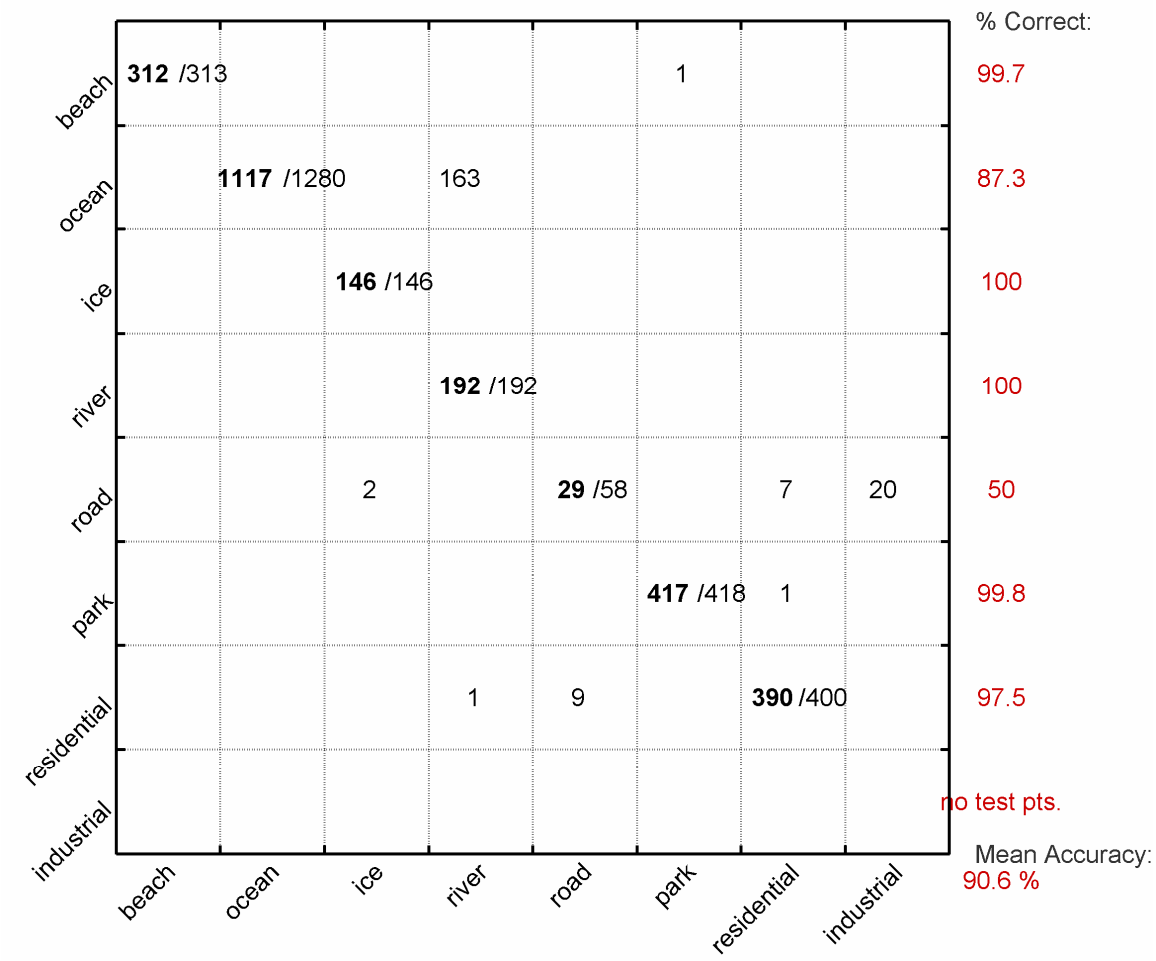

Figure 19: Boston benchmark: Confusion matrix depicting actual class labels (y-axis) versus predicted labels. Average accuracy is the average of per-class accuracy, which disallows overemphasis of any class whose exemplars are much more frequent in the test dataset.