Possibilistic Clustering Enabled Neuro Fuzzy Logic

Blake Ruprecht, Wenlong Wu, Muhammad Aminul Islam, Derek Anderson, James Keller, Grant Scott, Curt Davis, Fred Petry, Paul Elmore, Kristen Nock, and Elizabeth Gilmour

Abstract—Artificial neural networks are a dominant force in our modern era of data-driven artificial intelligence. The adaptive neuro fuzzy inference system (ANFIS) is a neural network based on fuzzy logic versus a more traditional premise like convolution. Advantages of ANFIS include the ability to encode and potentially understand machine learned neural information in the pursuit of explainable, interpretable, and ultimately trustworthy artificial intelligence. However, real-world data is almost always imperfect, e.g., incomplete or noisy, and ANFIS is not naturally robust. Specifically, ANFIS is susceptible to over inflated uncertainty, poor antecedent (fuzzy set) data alignment, degenerate optimization conditions, and hard to interpret logic, to name a few factors. Herein, we explore the use of possibilistic clustering to identify outliers, specifically typicality degrees, to increase robustness of ANFIS; or any fuzzy logic neuron/network. Experiments are presented that demonstrate the need and quality of the proposed solutions in the pursuit of robust interpretable machine learned neuro fuzzy logic solutions.

I. INTRODUCTION

The world is once again fixated on neural nets, due in large part to their recent performance leaps across numerous application domains; computer vision, natural language processing, etc. On the other hand, modern deep learning has a list of equally deep concerns, e.g., are we really just engineering machines that discover desirable correlations versus underlying causations [1]. Regardless, in all this excitement the field has more-or-less converged into a single mathematical foundation, convolution; which powers more complicated constructs like residual and recurrent networks. Furthermore, the vast majority of these deep nets have given rise to black box solutions— that have little emphasis on explainability or interpretability. Herein, we focus on a non-convolutional contribution from the field of fuzzy set theory, the adaptive neuro-fuzzy inference system (ANFIS) [2]. Specifically, we focus on a first order (linear) Takagi-Sugeno-Kang (TSK) type ANFIS.

A benefit of a fuzzy logic neuron/network (FLN), e.g., TSK ANFIS, is it holds the potential to help realize more explainable, interpretable, and ultimately trustworthy AI. There are a number of ways in which this can occur. For one, it is possible to insert human knowledge as rules into a FLN. Furthermore, a FLN can be derived from data then opened to study what variables, rules, and output combination strategies were learned. However, as shown by Keller and Yager in [3], fuzzy logic can be achieved using a multi layer perceptron (MLP), as both are well-known universal function approximators. A benefit of an approach like ANFIS, versus [3], is the information is explicit and centralized versus implicit and distributed. While numerous approaches exist to learn a fuzzy inference system, we focus on neural to support “plug-and-play” into existing deep learners and to maintain homogeneity for the sake of optimization (e.g., backpropagation and gradient descent).

One challenge with current FLNs, and ANFIS in particular, is they are not robust. Specifically, when noise is present it is likely that ANFIS will obtain variables with over inflated uncertainty: sets that are wider than they need to be. Also likely is poor placement: misalignment of the learned set relative to the underlying truth. Due to degenerate optimization conditions, if ANFIS is initialized with more uncertainty than what is required, then it is likely that uncertainty will not sufficiently decrease. Furthermore, it is possible to obtain rules that decrease the error function, but make little-to-no high level sense. Also, once a rule is dead (i.e., it never fires), it stays dead. Meaning, it does not get updated during training due to no data points contributing to the parameter update for that particular rule. Last, determining how many sets and rules are required for ANFIS has proven difficult, and not robust. As the reader can see, many challenges relate to ANFIS and FLN learning. The points mentioned above are explained and addressed in more detail later in the article.

Before we delve into ANFIS and its extensions, it is important to note that ANFIS is not the only neuro-fuzzy architecture. In [3], Keller and Yager proposed a multi layer perceptron (MLP) to learn fuzzy logic. In [4], Keller and Tahani discussed the implementation of a conjunctive and disjunctive fuzzy logic rules with neural networks. In [5], Pal and Mitra explored an array of topics on neuro-fuzzy related to pattern recognition. In [6], Rajurkar and Verma put forth a deep fuzzy network with Takagi Sugeno fuzzy inference

TABLE I: Acronyms, Notation, and Parameters

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFIS</td>
<td>Adaptive Neuro Fuzzy Inference System</td>
</tr>
<tr>
<td>SPOM</td>
<td>Sequential Possibilistic One-Means</td>
</tr>
<tr>
<td>N</td>
<td>Number of input samples</td>
</tr>
<tr>
<td>K</td>
<td>Number of input features</td>
</tr>
<tr>
<td>R</td>
<td>Number of rules in an ANFIS rule-base</td>
</tr>
<tr>
<td>N × K</td>
<td>Dimensionality of input dataset</td>
</tr>
<tr>
<td>x_k</td>
<td>Input of feature length K</td>
</tr>
<tr>
<td>w_r</td>
<td>Antecedent vector of the rule base, of length R</td>
</tr>
<tr>
<td>z_n</td>
<td>Consequent vector of the rule base, of length R</td>
</tr>
<tr>
<td>A_r^k(·)</td>
<td>Membership function of rule r</td>
</tr>
<tr>
<td>p_k</td>
<td>Component weight of rule r</td>
</tr>
<tr>
<td>p_r^bias</td>
<td>Component bias of rule r</td>
</tr>
<tr>
<td>y_n</td>
<td>Output of ANFIS for x_k</td>
</tr>
<tr>
<td>c</td>
<td>Number of clusters</td>
</tr>
<tr>
<td>u_{ij}</td>
<td>Typicality of ith sample to cluster j</td>
</tr>
<tr>
<td>v_j</td>
<td>jth cluster prototype</td>
</tr>
</tbody>
</table>

Blake Ruprecht is with the Electrical Engineering and Computer Science (EECS) department, University of Missouri (MU), MO. Wenlong Wu, Muhammad Aminul Islam, Derek Anderson, James Keller, Grant Scott, and Curt Davis are with EECS, MU. Fred Petry, Kristen Nock and Elizabeth Gilmour are with the U.S. Naval Research Laboratory, Paul Elmore is with Johns Hopkins Applied Physics Laboratory.
Fig. 1: High-level overview of the proposed work. Traditionally, ANFIS is applied to the raw full data. Here, possibilistic clustering is used to acquire data typicality degrees, which are fed to ANFIS during learning. Blue dots denote noise points (outliers) and black dots belong to a cluster. Red and green, respectively, triangles are learned membership functions, where red indicates rule one and green indicates rule two. Solid lines are ANFIS learned solutions and dotted is extended ANFIS.

ANFIS performs three steps on \( x_n \) to determine \( y_n \). The first two steps, the Antecedent Firing and the Consequent Component Building, can be run in parallel, and they produce an antecedent vector, \( w_n \), and consequent weighting vector, \( z_n \), respectively. The third step, Aggregation, takes \( w_n \) and \( z_n \), performs a weighted sum, and it produces the final output, \( y_n \).

The lengths of \( w_n \) and \( z_n \) are \( R \). The rules in ANFIS follow the familiar \textit{IF-THEN} format, each of which has an antecedent and consequent clause.

**Antecedent Firing:** consists of calculating the firing strength of all the antecedent clauses of each rule, \( w_{rn} \), which uses a t-norm (herein, we use the product operator), i.e.,

\[
\begin{align*}
  w_n &= \prod_{k=1}^{K} A_k^r(x_n(k)).
\end{align*}
\]

The membership functions \( A_k^r(\cdot) \) are unique to each rule and feature, and can be learned. This step can be thought of as how well the input vector matches each individual rule.

**Consequent Component Building:** consists of calculating the components of the consequent clause of each rule, \( z_k^r \), which is determined based on the summation of each input feature and weight \( p_k^r \), plus a bias term, \( p_{bias}^r \), i.e.,

\[
\begin{align*}
  z_k^r &= \left( \sum_{k=1}^{K} x_n(k)p_k^r \right) + p_{bias}^r.
\end{align*}
\]

The input feature weight \( p_k^r \) is a unique scalar to each rule and feature, which can be learned. This step can be thought of as the output each rule would make for the input vector that perfectly fires the antecedent clauses.

**Aggregation Step:** consists of calculating the weighted aggregation of the consequent clauses, \( z_n \), using the antecedent clauses, \( w_n \), as weights, to produce the output scalar \( y_n \), i.e.,

\[
\begin{align*}
  y_n &= \frac{\sum_{r=1}^{R} z_n^r w_n^r}{\sum_{r=1}^{R} w_n^r}.
\end{align*}
\]
B. ANFIS Optimization

For sake of article completeness, Table II are the partial derivatives for ANFIS, based on a first order TSK model. The reader can refer to [2], [7], and [13] for full mathematical explanation and derivations.

C. Open Source Codes

For reproducible research, in [7] we provided free PyTorch codes for shallow and deep ANFIS; https://github.com/BlakeRuprecht/Fuzzy-Fusion. For the current article, we have placed our possibilistic clustering extended ANFIS at https://github.com/BlakeRuprecht/ANFIS-SP1M.

D. Limitation of Traditional ANFIS

ANFIS [2] is a powerful tool, but not without flaw. While a number of shortcomings have been identified, addressed to varying degrees, we highlight a few relevant limitations.

1) Noise and Over Inflated Uncertainty: Noise pulls the estimated membership functions away from their true state of nature. For example, consider a trapezoidal membership function. Noise in the data will increase the core region (membership value one) and the support (membership value greater than zero) will widen. From a semantic standpoint, this means noise has the impact of “inflating” or creating greater uncertainty than what actually exists.

2) Initialization: A number of works have been proposed to estimate the number and parameters of rules from data. ANFIS, by default, requires the number of rules, and number of antecedents per rule, to be specified before optimization can begin. ANFIS is not a structure learning algorithm, it is a parameter estimation algorithm. Furthermore, what values should the membership functions (and TSK coefficients) be set to initially, i.e., how should ANFIS be initialized? This selection can have an impact on the result in the hands of an algorithm like stochastic gradient descent.

3) Non-Decreasing Uncertainty: An underappreciated aspect of ANFIS is its inability to reduce uncertainty during learning. Consider two inputs and two rules. Let each rule correspond, antecedent-wise, to an underlying Gaussian distribution and let these clusters have no overlap and sufficient separation. Suppose that the standard deviations are initialized to three times the true standard deviation. This is a simple example. The point is, if no data points exist in the space between the two underlying clusters, then there is no part of ANFIS learning that naturally tries to reduce uncertainty. There are no errors driving the sets to be as specific as possible (see Proposition 2). From a control standpoint this might
be desirable if the goal is to partition the underlying space. However, if the goal is to listen to the data and to have the membership functions fit the underlying samples, then this is problematic.

4) Once Dead, Always Dead: Consider an ANFIS with one or more rules initialized such that their membership functions do not fire, i.e., return a rule firing strength greater than approximately zero for any sample in the training data. Here, we refer to such rules as dead. In ANFIS, once a rule is dead, there is little-to-no force (sufficiently scaled gradient) that drives that rule towards a location in space to remedy the problem (see Proposition 1). This is a massive limitation with respect to learning quality ANFIS solutions. It places a large burden on the initial position of the antecedent fuzzy sets and/or it requires us to initialize with wide uncertainties which conflicts with the above non-decreasing uncertainty challenge.

In summary, this subsection exists to raise awareness of ANFIS limitations that need addressing. The combination of possibilistic clustering and modified ANFIS learning leads to more robust neuro fuzzy logic learning.

III. SEQUENTIAL POSSIBILISTIC ONE MEANS (SP1M)

The sequential possibilistic one means (SP1M) algorithm [14] is based on the possibilistic c-means (PCM) algorithm [15]. The PCM abandons the membership sum-to-one constraint in the fuzzy c-means (FCM) [16] algorithm and it has been shown to be robust against outliers. Each cluster in PCM is independent of the others and the user and/or algorithm may have to address coincident clusters. The SP1M was created to combat the coincident clusters problem of PCM by generating one cluster at a time until all the “dense” regions are found.

Pseudocode for the latest version of SP1M is shown in Algorithm 1, where X is the input samples, c is the estimated input for cluster number, ε is the threshold, m is the fuzzifier, K is defined to be the number of points whose maximum typicality is smaller than 0.5. Note that the (*) detail of dynamic η computation in Algorithm 1 is discussed in [14].

Algorithm 1 SP1M Pseudocode

1: INPUT: X, c, ε
2: OUTPUT: U: final typicality partition
3: OUTPUT: V: final cluster prototypes
4: Initialize U, V as empty
5: while j++ < c and #(P1M) < K do
6:     repeat
7:         Pick v ∈ X with probabilities Eq. 6
8:         repeat
9:             Compute η_j dynamically (*)
10:            Compute typicality u_{ji} = \frac{1}{1 + \frac{\eta_j^2}{m-1} \sum k u_{ki}^m x_i}
11:            Compute cluster center: v_{j} = \frac{\sum_{i=1}^{N} u_{ji}^m x_i}{\sum_{i=1}^{N} u_{ji}^m}
12:         until |Δu_{ij} < ε
13:     until \min_{w \in V} \|v_j - w\| ≥ 2\eta
14: Append v_j to U
15: Append u_j to V
16: end while

In SP1M, the cluster centers are not initialized purely randomly. They are initialized from probabilities based on the typicalities of the previously found clusters. The initial cluster centers are picked from dataset X with probabilities

\[ p(x_i) = \begin{cases} \frac{1}{n} & \text{if } j = 1 \\ 0 & \text{if } \max_{k=1,...,j} u_{ki} > 0.5 \\ N - \sum_{s=1}^{N} \max_{k=1,...,j} u_{ks} & \text{otherwise} \end{cases} \]

The returned matrix U from the SP1M is the typicality matrix that measures how “typical” a particular point is to each cluster, that is, how close the point is to each cluster prototype. Outliers have a large distance to all the existing clusters so that they naturally have low typicality to all clusters. SP1M open source MATLAB code: https://github.com/waylono/sp1m-de.

IV. POSSIBILISTIC CLUSTERING INFORMED ANFIS

A. Initialization

Herein, we do not rely on random ANFIS parameter initialization. ANFIS is a supervised learning algorithm. First, the number of underlying clusters must be determined from the data (we will create one rule for each underlying cluster). Next, user-specified membership functions are fit to each cluster. In the case of a Gaussian, we calculate the respective mean and standard deviation. In the case of a trapezoidal membership function, we set the core to three standard deviations and the support to five standard deviations. The point is, the user has flexibility over what function to use and how to best fit the antecedent clause membership functions to the data. Next, once the membership functions are estimated for each cluster, we use the least means squared (LMS), like Jang [2], to estimate the ANFIS consequent weights \( p(e)^k \). Note, in traditional ANFIS the user selects the number of rules and antecedents per rule. In our possibilistic clustering-based ANFIS, SP1M informs us about the number of underlying clusters, which is used to pick the number of rules.

B. Method 1: Pre-Processing or Data Filtering

Our first proposal is to use the possibilistic clustering algorithm results to pre-process the training data. Training data typicalities are generated using the SP1M algorithm. These typicalities are then analyzed, and any typicality below a user defined threshold, Γ, are removed from the training data. Specifically, we take the max typicality of a data point to all clusters, \( t_i = \max_{j=1,...,c} u_{ji}, \) i.e., the strongest degree that it belongs to any cluster. This process has numerous benefits. First, there are fewer data points, which lets an algorithm train faster. Second, there are fewer outliers, which improves initialization and helps us combat challenges like non-decreasing uncertainty. However, this procedure is crisp in the fact that it partitions data into use/not use, versus utilizing the degree to which a data point is an outlier. Furthermore, results will likely vary based on parameter Γ.
C. Method 2: Gradient Scaling

Our second approach is to use SP1M typicalities to modify the ANFIS algorithm itself, not the data presented to it. Specifically, we modify the learning algorithm. Herein, gradient descent is used to optimize ANFIS. The criteria function used is the sum of squared error (SSE); which leads to the derivatives in Table II. We first summarize our data point typicalities according to their max typicality, \( t_i = \max_{j=1,...,e} u_{ji} \in [0, 1] \). Next, we scale the partials, i.e.,

\[
\left[ \frac{\partial y_n}{\partial z^*_n} \right]_{\text{scaled}} = t_i \frac{\partial y_n}{\partial z^*_n}.
\]

Thus, if a sample has a high degree of belonging to a cluster then ANFIS operates "as is". However, outliers have a low typicality, which dampens their impact on learning. Note, as this is uniform scaling, i.e., all gradients are scaled the same way, it does not change the direction of the gradient, just its magnitude. In summary, Method 2 is different from Method 1 since it utilizes the typicality information to scale gradients during learning, rather than partition data during pre-processing.

V. Preliminary Experiments and Results

In this section, we demonstrate a set of controlled experiments to show the before and after effects of clustering informed versus traditional ANFIS. Synthetic data is used because we can control the conditions and range and we know the answer. We use two inputs/dimensions because the results can be visualized. Furthermore, we focus on the quality of the results. ANFIS, as is, can be used to achieve minimum error relative to a user specified criteria function. By quality, rather than accuracy or an index like F1 score, we specifically mean results. ANFIS, as is, can be used to achieve minimum error relative to a user specified criteria function. By quality, rather than accuracy or an index like F1 score, we specifically mean relative to a user specified criteria function. By quality, rather than accuracy or an index like F1 score, we specifically mean.

A. Experiment 1: Low Amount of Noise

In Experiment 1, we focus on a dataset that should have five rules, which is intended to model a few close rules whose noise can influence each other. Furthermore, five rules were selected because it can be visualized; i.e., there are not too many points and overlaid resulting clustering information to inhibit the readers viewing and understanding. Each cluster has 300 samples and 10% of the data is noise. Specifically, noise samples are generated beyond four standard deviations of the underlying generative Gaussian clouds. The five class (rule) centers are \([0.080, 0.501], [0.074, 0.680], [0.496, 0.077], [0.918, 0.679], [1.151, 0.903]\), with corresponding \(x\) and \(y\) dimension standard deviations of \([0.033, 0.0256], [0.0353, 0.0193], [0.0397, 0.0148], [0.0384, 0.0091], [0.0251, 0.0272]\). It is worth noting that we selected close, but not overlapped clusters because in a real scenario if two clusters/rules overlap, it makes less sense. Meaning, two rules with similar IF but different THEN counterparts. A single data point could potentially belong to multiple different rules. Such a scenario is semantically confusing. The number of features may be inadequate to properly separate rules, or something else may be causing issues. Figure 3 shows Experiment 1.

Figure 3 shows the result of traditional ANFIS, clustering-based pre-processing, and gradient scaled ANFIS. Note, in traditional ANFIS one has to either select or engage in some external method to pick \(R\). Herein, we use SP1M to select \(R\), which provides ANFIS more benefit than what the core algorithm affords. Clearly, clustering-based pre-processing produces better rule structures. Namely, the core (membership value one) region is a tight fit to the underlying data and the support (membership value greater than zero) has a reasonable footprint; specifically, it does not include all of the noise. On the other hand, traditional and gradient scaled ANFIS have over inflated cores and support regions. Furthermore, both have a problem with rule placement for the two close clusters (green and bluegray). Overall, typicality-based pre-processing yields a good fit of the underlying data, making the resulting fuzzy sets more faithful and interpretable descriptions of the data.

B. Experiment 2: Moderate Amount of Noise

In Experiment 2, we keep the same dataset, allowing Experiment 2 to be compared to Experiment 1. The only thing that has changed is that 25% of each cluster’s data is now noise, compared to 10% in Experiment 1. The reason for Experiment 2 is to observe the impact and behavior of the proposed approaches in light of a greater amount of noise. Figure 4 shows the dataset.

Figure 4 also displays the result of the three methods. The take away is as follows. As expected, an increased level of noise impacts ANFIS, but the power of possibilistic clustering is able to address and overcome shortcomings, namely typicality-based pre-processing.

C. Additional Experiments

Figure 5 shows nine additional arbitrary ANFIS experiments with varying numbers of \(R, C\), and cluster overlap. The reason for these mini experiments is to give the reader a feel for performance in different contexts. As can be seen, typicality weighted ANFIS is more resilient and its largest shortcoming is if SP1M can estimate the true underlying generative cluster structure. While we have only demonstrated results for two dimensions and a few clusters \((2 \leq C \leq 10)\), the reader can download and experiment with our open source codes, https://github.com/BlakeRuprecht/Fuzzy-Fusion and https://github.com/BlakeRuprecht/ANFIS-SP1M.

VI. Insights and Summary

Experiments 1 and 2 highlight the benefit of using possibilistic typicality degrees in neuro fuzzy logic. This allows us to combat a number of underlying qualitative concerns about a learned neuro fuzzy logic solution. However, it was our expectation that Method 2, gradient scaling, would be the top performer. Because this is not the case, we dig deeper
Fig. 3: Experiment 1. Each class is color coded. The traditional and gradient scaled ANFIS trapezoidal membership functions are shown in grey. Solid is the core and dashed is support. Red solid and dashed lines are clustering-based pre-processed ANFIS. The data points are scaled based on typicality (note that the size has a lower bound to prevent points from disappearing).

Fig. 4: Experiment 2. Each class is color coded. The traditional and gradient scaled ANFIS trapezoidal membership functions are shown in grey. Solid is the core and dashed is support. Red solid and dashed lines are clustering-based pre-processed ANFIS. The data points are scaled based on typicality (note that the size has a lower bound to prevent points from disappearing).
into ANFIS. Consider the update equation for a membership function parameter,
\[
\frac{\partial y_n}{\partial \Theta^m_r} = \left( \sum_{j=1, j \neq r}^{R} w^j_n (z^j_n - z^r_n) \right) \times (8a)
\]
\[
\left( \prod_{j=1, j \neq k}^{K} A^j_r(x_n(j)) \right) \times (8b)
\]
\[
\frac{\partial A^j_r(\cdot)}{\partial \Theta^m_r}. \tag{8c}
\]

Consider the following two scenarios.

**Proposition 1.** Term 1 (Equation 8a) results in a zero magnitude gradient (Equation 8) for rule r when it is the only rule that fires, i.e., \(w^r_n > 0\), and \(\forall j \in \{1, ..., R\}, j \neq r, w^j_n = 0\).

**Proof.** The numerator in Equation 8a is all rules other than r. As each of these rules have \(w^r_n = 0\), the numerator is zero and the denominator is not. Thus, Equation 8 is zero.

Proposition 1 can be interpreted as diminishing the gradient when the rules are properly separated with no overlap. This means that the conditions for Proposition 1 are constantly being met each time a data point that belongs to rule r is being trained on. This prevents learning from occurring for the rule being met each time a data point that belongs to rule r (and other t-norms, e.g., the minimum) is zero. As a result, Equation 8 is zero.

**Proposition 2.** Term 2 (Equation 8b) results in a gradient (Equation 8) whose magnitude is 0 when a data sample (n) is not in a rule (r), i.e., \(w^r_n = 0\).

**Proof.** If data sample n is not covered by rule r, then \(A^j_r(x_n(j)) = 0\). As each of these terms are zero, their product (and other t-norms, e.g., the minimum) is zero. As a result, Equation 8 is zero.

Proposition 2 has a number of ramifications. For example, if a data point does not result in a rule firing strength greater than zero, then that rule is not updated. While by itself, this does not seem overly alarming, consider the case of a rule initialized to a region corresponding to no data points. The rule will never get updated. This is the definition of “once dead, always dead.”

These two propositions highlight two common cases. More scenarios that give rise to diminishing-to-dead gradients can be identified. The point is, it is clear why our typicality weighted and initialized procedure performs best. Until factors like these are remedied with ANFIS, a procedure like gradient scaling, while elegant in design, is rendered ineffective.

**VII. CONCLUSION AND FUTURE WORK**

In this article we explored the role of possibilistic clustering to generate data point typicality degrees to improve challenges in ANFIS, a neuro fuzzy logic learning tool. Specifically, we used SP1M, which helps us combat coincident clusters in PCM, and it allows one to support, if desired, realtime, online learning and/or Big Data. We explored two possible methods, pre-processing data by removing SP1M identified outliers, and gradient scaling during ANFIS learning utilizing SP1M typicalities. Pre-processing had the best results, namely due to a diminishing/dead gradient shortcoming in ANFIS.

In future work, we will take this preliminary investigation and explore real-world applications of neuro fuzzy logic. We will also find a way to remedy the diminishing/dead gradient problem in ANFIS, which should make gradient scaling the top performer. Other future work will include taking definitions of what constitutes a “good logic explanation”, and folding that into the learning algorithms to promote better explanations. Last, once our work related to understanding shallow neuro fuzzy logic networks is mature, we will open our processes up to deep inference nets.

**VIII. ACKNOWLEDGMENTS**

Petry, Kristen, and Elizabeth would like to thank the Naval Research Laboratory’s Base Program for sponsoring this research.

**REFERENCES**


Fig. 5: Additional experiments that show variety and ANFIS behavior in a range of contexts. The goal is a tight fit of boxes to data. C is number of underlying clusters, R is number SP1M found. (a)-(c) have 1500 data points. (a) and (b) have the same means. (d)-(f) have 1500 data points. (d) and (e) have the same means. (g)-(i) have 250 data points and the same means.

----


