# DECISION MAKING IN DYNAMIC ENVIRONMENTS: AN APPLICATION OF MACHINE LEARNING TO THE ANALYTICAL HIERARCHY PROCESS

Caelum Kamps
DRDC Ottawa Research Centre
Canada
caelum.kamps@drdc-rddc.gc.ca

Rahim Jassemi-Zargani
DRDC Ottawa Research Centre
Canada
rahim.jassemi@drdc-rddc.gc.ca

**ABSTRACT**

The purpose of this work is to propose a method of algorithmic decision making that builds on the Analytical Hierarchy Process by applying reinforcement learning. Decision making in dynamic environments requires adaptability as new information becomes available. The Analytical Hierarchy Process (AHP) provides a method for comparative decision making but is insufficient to handle information that becomes available over time. Using the opinions of one or many subject matter experts and the AHP, the relative importance of evidence can be quantified. However, the ability to explicitly measure the interdependencies is more challenging. The interdependency between the different evidence can be exploited to improve the model accuracy, particularly when information is missing or uncertain. To establish this ability within a decision-making tool, the AHP method can be optimized through a stochastic gradient descent algorithm. To illustrate the effectiveness of the proposed method, an experiment was conducted on air target threat classification in time series developing scenarios.

**Keywords:** threat assessments; AHP; machine learning; decision making

## 1. Introduction

During complex decision making for classification, the ability of a subject matter expert (SME) to reach a conclusion depends on the availability of information. In dynamic situations, where information is developing over time, the decision maker must be flexible as information becomes available or is lacking. Air target threat assessment is such a situation. The available systems dictate the rate and amount of target information that accumulates. Additionally, with the increasing amount of air traffic, it is not feasible for an individual or even a team to be able to evaluate and classify each target that is detected. The need for machine-aided classification is evident and increasing. It is desirable that the method of algorithmic classification be as follows:

1. Adaptive to updated information
2. Predictive of unknown information
3. Reflective of the classification of a SME

The Analytic Hierarchy Process (AHP) is a general theory of measurement. The method facilitates deductive and inductive algorithmic decision making by comparing the relative importance of evidence (Saaty, 1987). Threat assessment (TA) is the fusion of many sources of information for the purpose of classifying and predicting the intent, capability, and proximity of detected entities. (Johansson & Falkman, 2008) This fusion of information is used to infer an associated target threat value (TV) for each entity at any given point in time. This work is particularly concerned with the application of algorithmic target threat assessment to airborne targets during states of uncertainty and deficient information.

The objective of this paper is to present methods to improve the accuracy of the AHP network evaluations when not all the cues are known. The assumption is that under the complete information state (CIS), the state where all the cues are known, the optimal decision or reference decision is the complete AHP network.

The problem that is being addressed is what to do when not all the cues are known. How can the network be adapted to best predict the reference decision? Given a state of some information, what are the optimal weights and best techniques to improve the accuracy of prediction for the correct threat classification?

The proposed decision structure is a hierarchal decision tree. Through the AHP, the nodes are each assigned a weight corresponding to their importance. Machine learning techniques can then be used to optimize the node weights when not all the information is known. This structure can then be further improved by introducing special learning nodes called sigmoid nodes. This will be shown in the following sections covering the AHP, the scenario description, the proposed methodology, and finally, the experiments and results.

## 2. Analytical Hierarchy Process

The Analytical Hierarchy Process (AHP) is a well-known algorithm for decision making support (Saaty, 1987). It allows a SME to break the problem down into criteria groups to more easily compare evidences, called cues. The structure of the decision tree is shown in Figure 1.

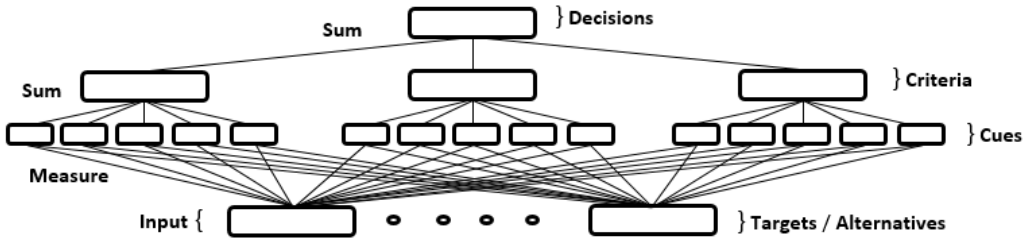



Figure 1 General AHP structure

A SME is given a list of cues to partition into criteria. Within each criteria, the cues are compared pairwise with one another. This information is stored in comparison matrices. Assuming the SME is relatively consistent in their comparisons, a principal eigenvector analysis on the comparison matrices can be used to obtain normalized weights for each of the cues. Once this is done, the same process is applied to the criteria to generate normalized criterion weightings. The structure then takes input from each of the targets or alternatives and makes a decision based on the previously defined weights. It is important to understand that since the weights are normalized, the sum of all the cue weights within a criterion is unity. Similarly, the sum of the weights of each of the criterion is unity (Saaty, 1987).

Currently, the AHP assumes that all the inputs are known. It relies on the ability of a SME to rank each of the cues with respect to others within the category. One of the limitations of the AHP is that it does not consider the relative importance of cues with respect to multiple criteria. For example, if one cue is used when considering two criteria, then its relative importance might not be the same in both cases. The Analytical Network Process (ANP) offers the ability to do these multiple comparisons. In both methods, all of the cues must be known to make an accurate decision. This is not what is being addressed here. In the ANP and AHP community, dependency refers to the influence of the criteria under consideration on the importance of the cue (Mu, 2006). The type of dependency that is being addressed by this paper is different. Here we consider dependency in the calculation of the problem at hand. This means that cues might hold information about each other, either explicitly in their formulae for calculation or implicitly in correlation. The method proposed in this paper deals with accounting for the values of unknown cues that might have these dependency features with some of the known cues.

## 3. Cues

The calculation of some cues is dependent on others. For example, to calculate the estimated closest point of approach of a target to a defended asset, the orientation, range, and track history of the target must be known. The introduction of dependent cues creates potentially complex and unknown relationships between cues that are hypothetically accounted for during states of complete information but are not necessarily understood when not all the inputs are known. This is one of the challenges that will be addressed. Here, the difference between dependent and independent cues can be

established. Cues that require that other information be known before a measurement can be made are dependent. Conversely, cues that can be measured on their own are independent. Table 1 shows the dependency and criteria for each of the cues that will be considered for the target threat assessment.

Table 1
Target threat assessment cues

| **Cue** | **Criteria** | **Dependent/Independent** | **Dependencies** |
|---|---|---|---|
| Speed (Sp) | Capability | Independent | N/A |
| Altitude (Al) | Proximity | Independent | N/A |
| Range (R) | Proximity | Independent | N/A |
| Orientation (O) | Intent | Independent | N/A |
| Closest Point of Approach (CPA) | Capability | Dependent | Orientation & Range |
| Angle of Attack (AOA) | Intent | Dependent | AOA & Altitude |
| Time Before CPA (TBCPA) | Proximity | Dependent | CPA & Speed |

Using the cues from Table 1, a hierarchal decision tree is built and shown in Figure 2.

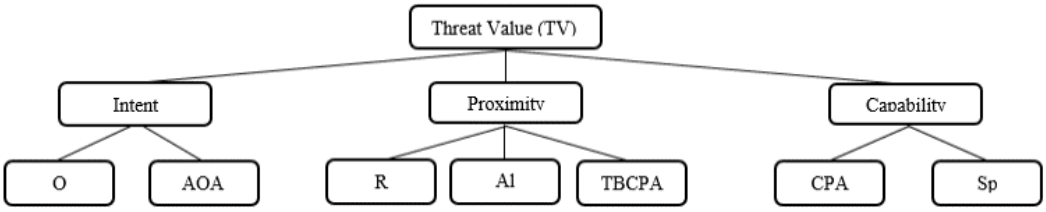



Figure 2 Hierarchal decision tree

To initialize the weights for each of the nodes in the decision tree using the AHP, four comparison matrices were generated and are shown in Table 2. The rankings provided by a study of cues and information order are used to create generalized pairwise comparisons for each cue (Liebhaber, Kobus & Feher, 2002).

Table 2
Comparison matrices for the hierarchy shown in Figure 2

| **Threat Value (TV)** | | | |
|---|---|---|---|
| | Proximity | Capability | Intent |
| Proximity | 1 | 2 | 2 |
| Capability | | 1 | 1 |
| Intent | | | 1 |

| **Intent** | | |
|---|---|---|
| | AOA | Orientation |
| AOA | 1 | 2 |
| Orientation | | 1 |

| **Proximity** | | | |
|---|---|---|---|
| | Alt | TBCPA | Range |
| Alt | 1 | 2 | 3 |
| TBCPA | | 1 | 2 |
| Range | | | 1 |

| **Capability** | | |
|---|---|---|
| | Speed | CPA |
| Speed | 1 | 2 |
| CPA | | 1 |

The weights generated through principal eigenvector analysis of the above comparison matrices are presented in Table 3.

Table 3
Weights of each criteria and cues

| **Criteria** | **Weight** | | **Sub-Criteria (Cues)** | **Weight** |
|---|---|---|---|---|
| Intent | 0.25 | | Orientation | 0.33 |
| | | | AOA | 0.66 |
| | | | | |
| Proximity | 0.5 | | Range | 0.54 |
| | | | Altitude | 0.30 |
| | | | TBCPA | 0.16 |
| | | | | |
| Capability | 0.25 | | Speed | 0.66 |
| | | | CPA | 0.33 |

## 4. Information domain partitioning

In real situations, information accumulates as a time series. Not all information is known right away because it takes time to run tests and acquire data. To represent each of the combinations of known information, the information domain is partitioned into an information state space (ISS). This is a list of every possible scenario that can occur with the information available. Each of the scenarios is referred to as an information state (IS). The number of information states is directly dependent on the number of independent cues. Each IS can be encoded as a binary string. If there are four independent cues, denoted 1, 2, 3, and 4, then the state in which cue 1 and 4 are known is 1001 or information state 9. Clearly, any combination of cues can be represented as a 4-bit string. Thus, there are $2^4$ information states. The ISS can now be explicitly defined. In Table 4, the dependent cues receive a designation of 1 (known) if their dependencies are satisfied.

Each IS can be referenced either by the string that is the first four binary digits in its row of the ISS table or by its state identification (ID).

Table 4
ISS table

| | Independent | | | | Dependent | | |
|---|---|---|---|---|---|---|---|
| State ID | Speed | Altitude | Orientation | Range | AOA | CPA | TBCPA |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 14 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## 5. Methodology

### 5.1 Initialization and base cases

To prove the validity of the method proposed in later sections, a base case must be generated in order to define some reasonable minimum accuracy of classification. A decision structure for each IS used to evaluate relative improvement is created using the AHP method. This ensures that the base case still reflects the relative weights of each of the known cues and is the best estimate available for the optimal classification given the information provided by the SME. Information state 11 or 1011 can be considered as an example to illustrate how a base case decision structure for that state can be generated. In this example, the CIS decision structure is recreated, however, all the cues that are not available and each criterion that becomes empty as a result of this process are removed. For IS 11, the decision structure is shown in Figure 3.

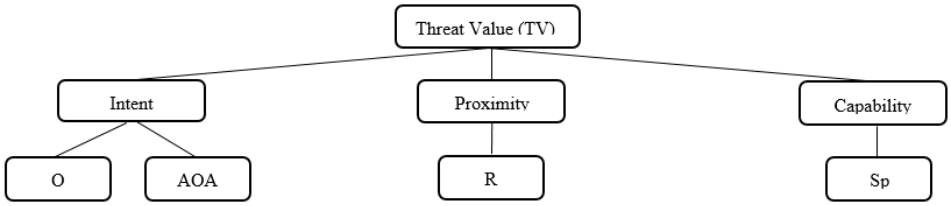



Figure 3 Decision structure

To produce weights for this decision structure, the AHP method must be applied again, and this time each comparison matrix only contains elements that appear in the decision structure. Since all the comparisons are pairwise, they still reflect the relative weights of each cue and criteria within their respective category. Thus, the AHP method is still a valid procedure for calculating the decision of this structure. This process is then applied to every other IS to produce a different base case for each. This base case initialization process is outlined in the following information flow chart shown in Figure 4.

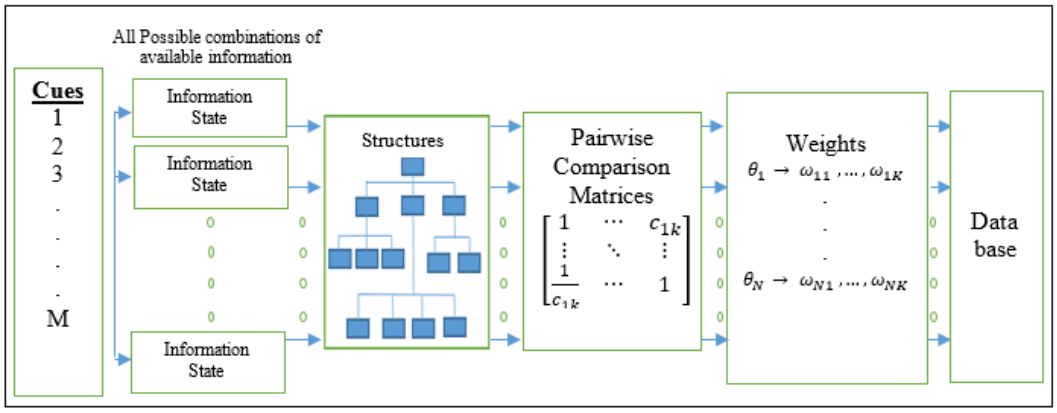



Figure 4 Initialization process

### 5.2 Mathematical representation of decision trees

When considering the state of complete information, the decision structure needs to be explicitly defined to be able to apply machine learning for optimization purposes. Taking the inputs from a specific target, a method to easily describe the propagation of the inputs to the outputs is described below.

- Calculate the threat value of target x1

$$T(x) = \sum_{i \in N} \theta_i \cdot \left( \sum_{j \in K} \omega_{ij} \mu_{ij}(x) \right)$$

Where:

- $\theta = [\theta_1, \ldots, \theta_N]$ is the vector of criteria weights
  - $N$ is the number of criteria
- $\omega = \begin{bmatrix} \omega_{11} & \cdots & \omega_{1K} \\ \vdots & \ddots & \vdots \\ \omega_{N1} & \cdots & \omega_{NK} \end{bmatrix}$ is a matrix of cue weights
  - The rows of $\omega$ represent the cues within each criteria.
  - $K$ is the size of the criteria with the most cues
  - Smaller criteria have trailing zero entries
- $\mu(x) = \begin{bmatrix} \mu_{11}(x) & \cdots & \mu_{1K}(x) \\ \vdots & \ddots & \vdots \\ \mu_{N1}(x) & \cdots & \mu_{NK}(x) \end{bmatrix}$ is the measurement of each of the cues from target x and has the same shape as omega with the same zero entries

It is important to note that because of the properties of the AHP method, the sum of $\theta$ or any row in $\omega$ is always unity. Additionally, it is desirable for the threat value to be normalized, which also normalizes the measurements $\mu_{ij}(x)$. With these properties, the threat value *T*, is always in the interval [0, 1].

For every other information state, different theta and omega values are generated using the AHP method. To create scenarios in which only some of the values of $\mu(x)$ are known, each information state is associated with a variable denoted as an input shape parameter. These are binary matrices with entries of either 0 or 1. They have the exact shape of μ, but if the cue exists in that information state then the entry in the position of the cue is 1, otherwise it is 0. This allows an easy partitioning of μ to create scenarios into a lesser IS for training purposes.

### 5.3 Weight optimization

When considering an incomplete IS, any IS that is not the CIS, the weights are calculated using the AHP method and are therefore intelligent in the sense that they take into consideration the relative importance of one another. In this situation, the problem is the uncertainty of the information that is lacking. Using weight redistribution through mini batch stochastic gradient descent (Ruder, 2017), it is possible to improve the accuracy of a correct threat classification of any information state that considers a minimum of two cues. To accomplish this, a cost function needs to be implemented to measure the error between the predicted threat value of an information state and the reference threat value.

- Quadratic Cost:

$$C_1 = \frac{1}{2B} \sum_{x \in B} ( Y(x) - T(x) )^2$$

- Entropic cost function:

$$C = -\frac{1}{B} \sum_{x \in B} [Y(x) \ln(T(x)) + ( 1 - Y(x) ) \ln( 1 - T(x) )]$$

Where:

- *C* is the cost to be minimized
- *B* is the batch of training targets
- *X* is a target
- *Y*(*x*) is the reference threat for target *x*
- *T*(*x*) is the threat calculated in the information state being trained

Both cost functions have their advantages; however, the entropic cost has a much steeper derivative and provides improved optimization when learning slowdown occurs. The method of stochastic gradient descent takes the gradient of the cost with respect to the weights $\theta$ and $\omega$ for that information state and shifts them in the opposite direction of the gradient with respect to some learning rate, $\alpha$.

$$\theta_i = -\alpha_\theta \frac{\partial C}{\partial \theta_i}$$

$$\omega_{ij} = -\alpha_\omega \frac{\partial C}{\partial \omega_{ij}}$$

Where:

- *C* is entropic cost
- $\alpha_\theta$ is the learning rate for the criteria weights
- $\alpha_\omega$ is the learning rate for the cue weights
- $\frac{\partial C}{\partial \theta_i} = -\frac{1}{|N|} \sum_{x \in |\boldsymbol{N}|} \left( \frac{Y(x) - T(x)}{T(x)(1 - T(x))} \right) \frac{\partial T(x)}{\partial \theta_i}$
- $\frac{\partial C}{\partial \omega_{ij}} = -\frac{1}{|N|} \sum_{x \in |\boldsymbol{N}|} \big( Y(x) - T(x) \big) \frac{\partial T(x)}{\partial \omega_{ij}}$
- $\frac{\partial T}{\partial \theta_i} = \sum_{j \in K} \omega_{ij} \mu_{ij}\,(x)$
- $\frac{\partial T}{\partial \omega_{ij}} = \theta_i \mu_{ij}(x)$

This process is repeated until some end condition is met to produce optimized weights for that information state. These weights are then saved and are the new weights used to calculate threat for that information state. The reason the cost function is averaged over a batch of training examples is to reduce the noise of any single scenario which helps avoid overfitting. It smooths the learning process and under some conditions results in better weight optimization. The weights of the CIS do not change. They are the weights which are used to calculate the reference threat value. The weights that do change are those of each information state.

### 5.4 Adapted decision method

One of the largest limitations of the above process is the inability to improve the classification accuracy unless many cues exist in the information state that can be reweighted. Next, a new structure of the decision tree is proposed that allows for much greater improvement in accuracy after optimization.

#### 5.4.1 Sigma node

The sigma node is a function that is frequently used in machine learning. Specifically, its properties are used for training neural networks. The sigmoid node is a special case of the logistic function. It is a simple, non-linear function that provides a balance between linear and non-linear behavior (Menon, 1996). Sigma functions are of the following form.

- Sigma function

$$S(\rho, \beta, x) = \frac{1}{1 + e^{-\rho(\mu - \beta)}}$$

Where:

- $\rho$ is a shape parameter
- $\mu$ is the input
- $\beta$ is a shift parameter

For these purposes, it is useful to consider the standard values for the shape and shift parameter as:

- $\rho$=4.5
- $\beta$= 0.5

The sigma function with these parameters is denoted the standard sigma function (SSF). This is because of the similarity of this function to the identity function on the interval [0,1], as seen in the Figure 5. This is desirable because it means that the use of SSF nodes within the decision structure will result in a largely unchanged threat value from the original method given the same cue measurements.

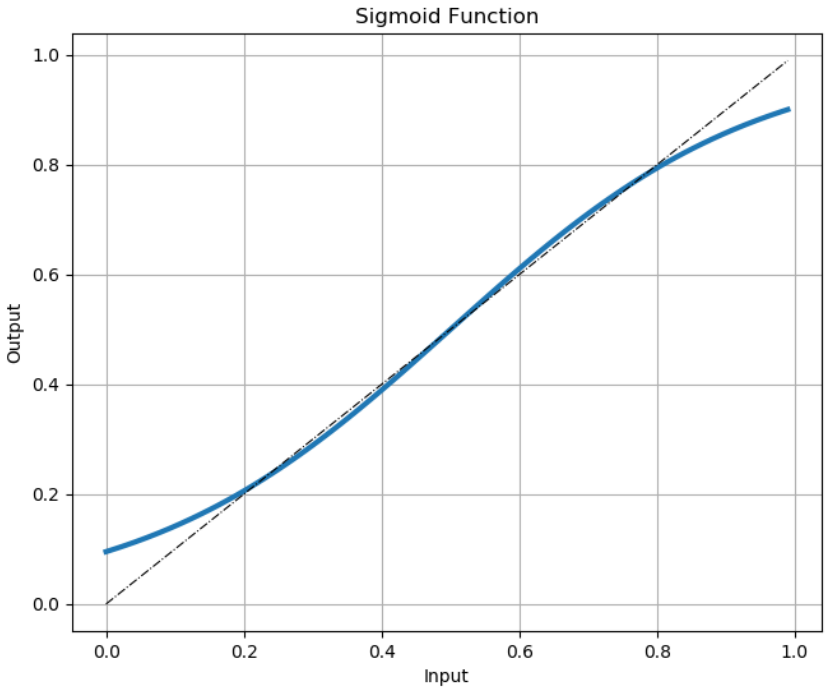



Figure 5 Sigma function

**5.4.2 Updated decision structure**

To improve the ability of a learning function to optimize the threat value prediction in any IS, an update to the decision structure needs to be made. When considering the CIS, the decision shape takes the same form as in the previous case but with an added step. Before input to any of the weight nodes, the input passes through an SSF node. This structure is called a sigmoid decision tree as seen in Figure 6.

When sigmoid operations are being considered (i.e., in the adapted decision method), the reference threat value becomes the output of the decision tree as seen in Figure 6. To create the sigmoid decision trees for each IS, the process described in section 5.1 is applied. The difference is the addition of SSF nodes before input to any weight node. These decision structures become the new base cases for each of the ISs used as comparisons for how much the learning algorithm that is described in the following section can improve the prediction.

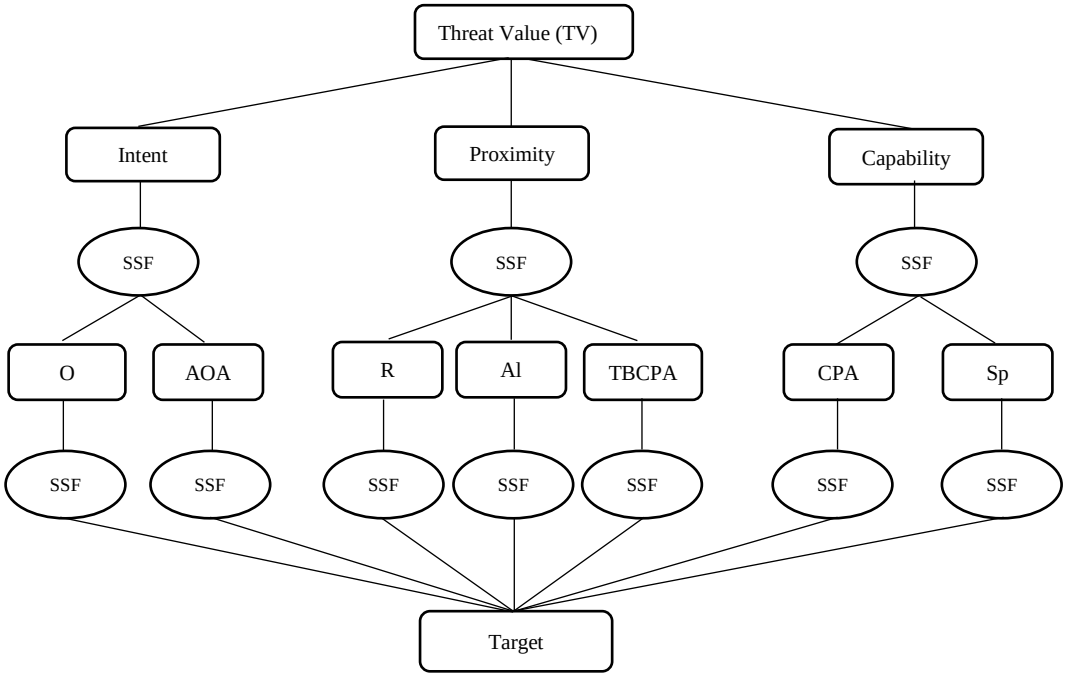
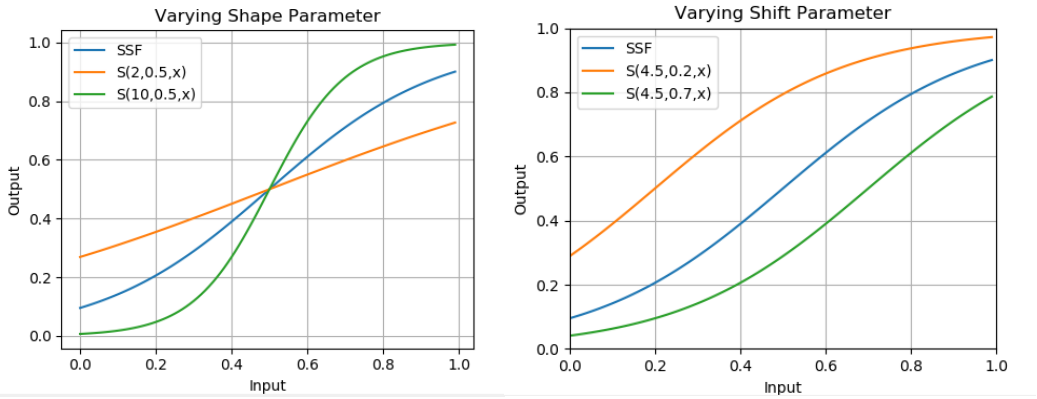



Figure 6 Sigmoid decision tree

### 5.5 Sigma node justification

The sigmoid decision structure offers a method to quantify sensitivity to dependency within the cues. Input from cues with more influence can be adjusted to reflect their impact on future calculations. Using the same learning process described in section 5.3, the value of the sigma parameters can be optimized to produce further improved threat value predictions. Using the shape and shift parameters, a broad spectrum of I/O relations can be achieved through sigma optimization. To illustrate the flexibility of the sigma node to adapt to countless circumstances, Figure 7 offers some examples of different I/O relationships.




(a) (b)

Figure 7 (a) Varying shape parameters, (b) Varying shift parameter

An added improvement of including sigma nodes in the threat calculation lies in the ability of the sigma function to easily adjust the threat value output even when few cues are available. This means the structure can adapt to reflect a more accurate prediction even when only one cue is known.

### 5.6 Sigma calculations and optimization

Given the new decision structure in Figure 6, the threat value calculation must be established.

- Expanded Sigma threat calculation

$$T(x) = \sum_{i \in N} \left[ \theta_i * S\left( \varphi_i, \delta_i \sum_{j \in K} \left( \omega_{ij} S\left( \rho_{ij}, \beta_{ij}, \mu_{ij}(x) \right) \right) \right) \right]$$

- Substitute

$$\gamma_i(x) = \sum_{j \in K} \left( \omega_{ij} S\left( \rho_{ij}, \beta_{ij}, \mu_{ij}(x) \right) \right)$$

- Simplified version

$$T(x) = \sum_{i \in N} \theta_i * S(\varphi_i, \delta_i, \gamma_i(x))$$

Where:

- $\theta$, $\omega$, and $\mu$ are as defined in 2.2
- $S$ is a sigma function

- $\beta = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1K} \\ \vdots & \ddots & \vdots \\ \beta_{N1} & \cdots & \beta_{NK} \end{bmatrix}$ and $\rho = \begin{bmatrix} \rho_{11} & \cdots & \rho_{1K} \\ \vdots & \ddots & \vdots \\ \rho_{N1} & \cdots & \rho_{NK} \end{bmatrix}$ are the matrices of cue shift and shape parameters, respectively.

- $\delta = [\delta_1, \dots, \delta_N]$ and $\varphi = [\varphi_1, \dots, \varphi_N]$ are the vectors of criteria shift and shape parameters, respectively.

The relationship between the sigma threat structure and the original structure is highly correlated. The following plot (Figure 8) outlines the correlation between threat value calculations of one versus the other.

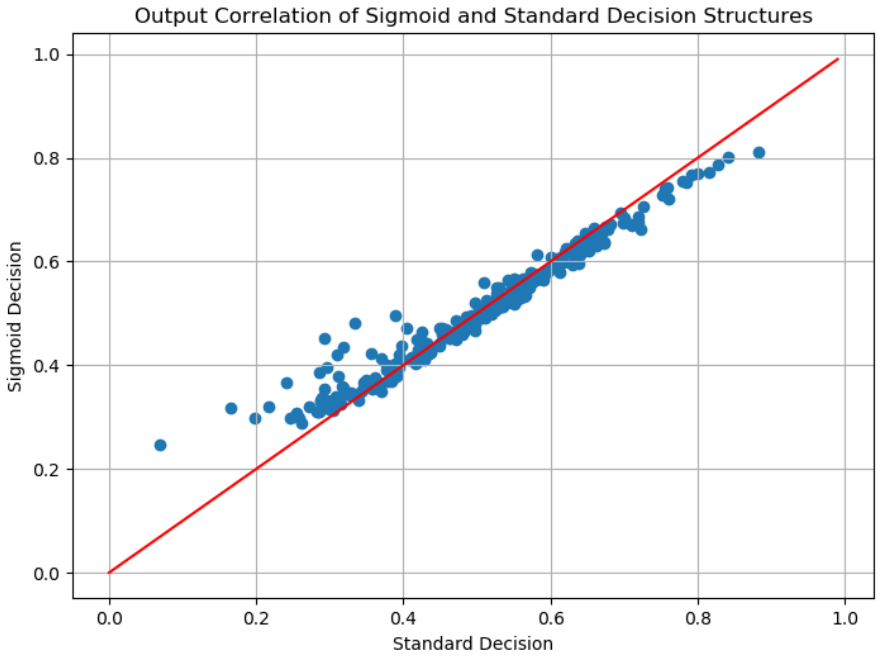



Figure 8 Correlation of sigmoid and standard decision structures

The correlation shows that in the intermediate threat value section the output is functionally the same. At the extremes, there is a slight deviation from the perfect correlation. The Pearson correlation coefficient for the two outputs is 0.971. This is a strong indication that the sigma threat calculation very strongly supports the results from the original decision structure method. Therefore, the argument that the validity of the SSF supplemented decision tree is representative of the original AHP method is very strong.

### 5.7 Sigmoid (SGD) optimization learning

Through SGD, the ability of any IS to predict the correct classification can be greatly improved. The entropic cost function is used for its ability to improve the speed at which the algorithm converges to optimization. Consider the following progression of

calculations that will lead to the gradient of the cost function with respect to the sigma parameters.

- Partials of sigma function with respect to generic placeholder variables, x, y, and z.

$$\frac{\partial S(x,y,z)}{\partial x} = [S(x,y,z)]^2 (z-y) e^{-x(z-y)}$$

$$\frac{\partial S(x,y,z)}{\partial z} = [S(x,y,z)]^2 x e^{-x(z-y)}$$

$$\frac{\partial S(x,y,z)}{\partial y} = -\frac{\partial S(x,y,z)}{\partial z}$$

1. Partial of sigma threat with respect to a criteria shape parameter

$$\frac{\partial T}{\partial \varphi_i} = \theta_i \frac{\partial S(\varphi_i, \delta_i, \gamma_i)}{\partial \alpha_i}$$

Where:

- $\gamma$_i is defined above in section 2.6
- $\alpha$_i is the criteria shape parameter
- $\varphi$_i is the criteria shift parameter

2. Partial of sigma threat with respect to cue shape parameter

$$\frac{\partial T}{\partial \rho_{ij}} = \left(\frac{\partial T}{\partial \gamma_i}\right)\left(\frac{\partial \gamma_i}{\partial \rho_{ij}}\right)$$

Where:

$$\frac{\partial T}{\partial \gamma_i} = \theta_i \frac{\partial S(\varphi_i, \delta_i, \gamma_i)}{\partial \gamma_i} \quad \text{and} \quad \frac{\partial \gamma_i}{\partial \rho_{ij}} = \omega_{ij} \frac{\partial S(\rho_{ij}, \beta_i, \mu_{ij})}{\partial \rho_{ij}}$$

3. Partial of sigma threat with respect to a criteria shift parameter

$$\frac{\partial T}{\partial \delta_i} = \theta_i \frac{\partial S(\varphi_i, \delta_i, \gamma_i)}{\partial \delta_i}$$

4. Partial of sigma threat with respect to a cue shift parameter

$$\frac{\partial T}{\partial \beta_{ij}} = \left(\frac{\partial T}{\partial \gamma_i}\right)\left(\frac{\partial \gamma_i}{\partial \beta_{ij}}\right)$$

Where:

$$\frac{\partial T}{\partial \gamma_i} = \theta_i \frac{\partial S(\alpha_i, \gamma_i, \varphi_i)}{\partial \gamma_i} \quad \text{and} \quad \frac{\partial \gamma_i}{\partial \beta_{ij}} = \omega_{ij} \frac{\partial S(\rho_{ij}, \mu_{ij}, \beta_{ij})}{\partial \beta_{ij}}$$

**5.8 Input generation**

All the independent cues are randomly generated based on their own probability density functions so that there are an infinite number of scenarios.

1. *Orientation:* The angle at which the target is oriented with respect to some defended asset.
2. *Range*: The horizontal distance of the target from the defended asset
3. *Speed:* The magnitude of the velocity of the target with respect to the ground
4. *Altitude:* The vertical distance of the target from the ground

Each cue is generated using a realistic interval for random value selection. Depending on the cue, the probability of the value is either based on a uniform or normal distribution. Speed and altitude are based on a normal distribution since most targets will exhibit standard speed and altitude values. Range and orientation are generated with a uniform distribution since there is no bias towards a particular position of the target.

Using this generation method for the independent cues, CPA, AOA and TBCPA can be calculated when the other required cues are available. Figure 9 visually represents the geometric cues.

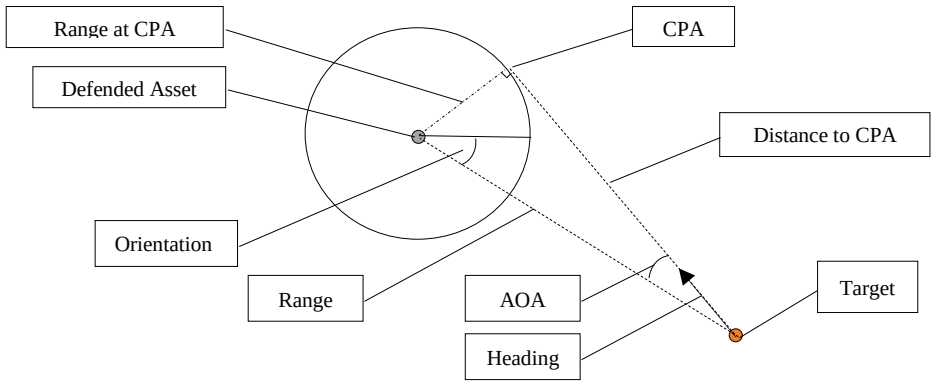



Figure 9 Graphical representation of cues

The following formulae are used to calculate the dependent cues.

$$1.\quad AOA = |Heading - Orientation|$$

$$2.\quad CPA = \left(\left|\left|x_{cpa}, y_{cpa}\right|\right|^2 + Altitude^2\right)^{\frac{1}{2}}$$

Where:

$$(x_{cpa}, y_{cpa}) = \min_{t>0} ||R * \cos(O) + t * \cos(H), R * \sin(O) + t * \sin(R)||$$

- $R$ = Range
- $O$ = Orientation
- $H$ = Heading

$$3.\quad TBCPA = \frac{CPA}{Speed}$$

Heading is not used in threat value structure, but is used in the intermediate calculation of cues. It is also generated randomly and is independent of every other cue.

Once the cues have been generated or calculated, they need to be normalized with respect to the severity of their threat value. Recall that all the inputs need to be normalized to a [0, 1] scale so that the decision structure can work properly. The general process is to assume some average case and relate the severity of the cue to how improbable the value. For this example, the severity normalizations are very simple. For example, when considering altitude, there is some PDF for the random generation of altitude. The severity of the altitude is directly related to the deviation from the average value. If a target is much higher or much lower than generally expected, then it is considered suspicious.

$$P(Altitude = x) = Normal.PDF_{altitude}(mean = a, STD = b, x)$$

$$Severeity\ of\ Altitude = CDF_{altitude}\big(mean = a, STD = b, (b + |b - x|)\big)$$

Where:

- Severity of altitude is the normalized threat value for altitude
- $Normal.PDF_{altitude}$ is a normal probability distribution function for the generation of altitude values
- $CDF_{altitude}$ is the cumulative density function of altitude
- $a$ is the mean of the *PDF*
- $b$ is the standard deviation of the *PDF*

In a similar way, each of the cues can be assigned a threat value given their randomized measurements. As stated above, this is a very simple method which is not to say that it is a good method to evaluate a target, but more to illustrate the ability of the method to predict the outcome using cue dependencies and conditional probabilities.

### 5.9 Optimization algorithm

SGD is applied to both the sigma parameters and the weight nodes so that an optimal structure for each IS can be generated. First, the cues are segmented into their base case information states as described in section 5.1. The structures are built and the SSF nodes are inserted to prepare each IS for training. These base cases are saved in a database as the reference decision structures for later comparison with the optimized versions. The second step is the optimization process. This process occurs in the following three phases: training data pre preparation, training, and evaluation.

#### 5.9.1 Training data preparation

SGD attempts to optimize a cost function with respect to some training parameters. If there is some underlying tendency of the function under evaluation to have a specific distribution, then SGD will recognize this and optimize with respect to that distribution. Given a large sample of target input cases, the target threat value calculation converges to the probability distribution shown in Figure 10.

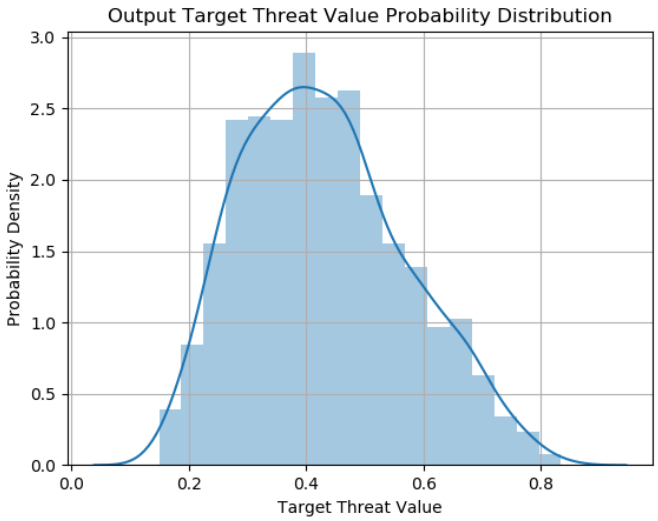



Figure 10 Probability distributions of target threat value

The threat value output approximately takes the form of a normal distribution with a mean of 0.43 and standard deviation of 0.14. If SGD were applied to the information states with a training batch that was randomly sampled, the optimization might learn to predict the distribution as opposed to the dependencies and relationships within the cues themselves since the probability that the correct classification will be within the standard deviation is so high. In other words, any underlying interdependencies of the threat calculation might get overpowered by the predisposition of the output to reflect a normal distribution.

To address this problem, a partitioning algorithm was implemented to create a uniform set of training targets from which to sample from during training. A random sample of 1000 reference targets were selected to mimic a uniform distribution of threat value. This is known as a training set. Using this set during training will not reveal the underlying

probability of the outcome to the decision structure and consequently requires that the prediction be based on cue interdependency and cue conditional probabilities.

#### 5.9.2 Training method

The training process is when the optimization occurs. A pseudo code for learning optimization is outlined below. The training set is described in section 5.9.1 and all the Greek variables have been defined in the sections about weight and sigma parameter optimization. The length of the batch is the number of cost gradients that are averaged per iteration. As mentioned before, this averaging promotes smoother learning due to reduced noise.

**WHILE (NOT** stop condition):

**FOR EACH (**information state **IN** information state space**):**

Average Cost Gradient$(\theta, \omega, \varphi, \delta, \rho, \beta) = [0, 0, 0, 0, 0, 0]$

**FOR (**length (batch)):

**FROM** uniform training set **SELECT** $\mu(x)$

Reference Threat = CIS Threat Value ( $\mu(x)$ )

**PARTITION** $\mu(x)$ into IS shape

**FROM** database **SELECT** $\theta, \omega, \varphi, \delta, \rho, \beta$ for IS

Calculate IS Threat Value: $T(\theta, \omega, \varphi, \delta, \rho, \beta, \mu(x))$

Calculate $\nabla Cost(\theta, \omega, \varphi, \delta, \rho, \beta)$

Average Cost Gradient = Average Cost Gradient + $\frac{\nabla Cost(\theta,\omega,\varphi,\delta,\rho,\beta)}{Length\ Batch}$

$$[\theta, \omega, \varphi, \delta, \rho, \beta] = -\left[\alpha_\theta, \alpha_\omega, \alpha_\varphi, \alpha_\delta, \alpha_\rho, \alpha_\beta\right] - \nabla(\theta, \omega, \varphi, \delta, \rho, \beta)$$

**UPDATE** database with new $\theta, \omega, \varphi, \delta, \rho, \beta$ for current IS

A skeleton outline of the pseudo code is depicted in the information flow graphic in Figure 11. Below, the optimization of the weights is shown; however, the process for the sigma parameters is much the same.

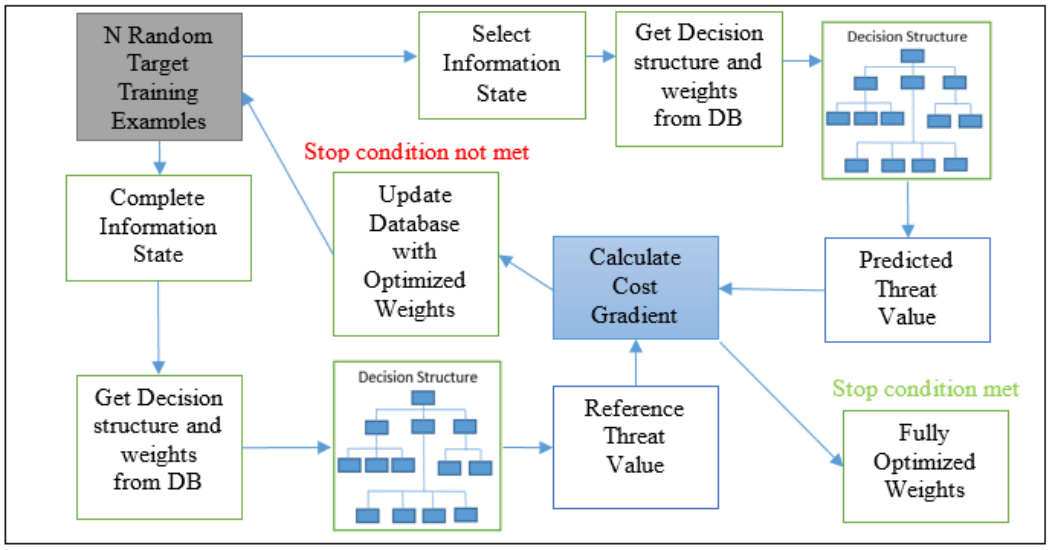



Figure 11 Graphic representation of information flow

For the example scenario in the following sections, the following hyper parameters were used. These might not be the optimal values; however, they are enough to illustrate the results in the following sections. There is a set of selected hyper parameters which were found using a grid search and a set of random parameters that were found using random generation shown in Table 5. Both sets produce similar results.

Table 5
Hyper parameters values

| ***Learning Rate*** | ***Best Selected Values*** | ***Best Random Values*** |
|---|---|---|
| $\theta$ | 0.005 | 0.00661 |
| $\omega$ | 0.01 | 0.000795 |
| $\delta$ | 0.05 | 0.11 |
| $\beta$ | 0.05 | 0.00661 |
| $\varphi$ | 0.5 | 1.645 |
| $\rho$ | 1.5 | 9.03 |
| Batch | 3 | 3 |
| Iterations | 800 | 1000 |

## 6. Experiment and results

### 6.1 Measures of performance

Three measures of performances were used to compare the base case to the trained structures. The first is range classification. A target threat prediction is considered correct if the difference between the reference threat value and the predicated threat value are within 0.1 of each other.

$$\varepsilon = |Reference - Threat| < 0.1$$

The second measure of performance is the bin classification method. The target threat values are partitioned into five bins of length 0.2.

$$Friendly \rightarrow Neutral \rightarrow Unknown \rightarrow Suspicious \rightarrow Threatening$$

If the reference threat value and the predicted threat value fall within the same bin, the prediction is considered correct. Finally, the mean error of prediction and the standard deviation of that error is calculated. For each of these measures of performance, the result will be considered as an average over all of the information states and for each individual state.

### 6.2 Results

Using the hyper parameters described in section 5.9, an experiment was run in which targets from the training set were randomly sampled to apply SGD optimization over the learning parameters to the information states. The progression of the average cost at every 10 cycles of the 800 iteration learning progression is depicted in Figure 12. Learning occurs very quickly at the start, followed by a gradual reduction in cost as learning continues. The cost used to create this plot is the average quadratic cost. The yellow line represents a moving average. It can also be seen that as learning progresses, the cost becomes smoother with fewer large upward spikes, indicating a smaller standard deviation on error.

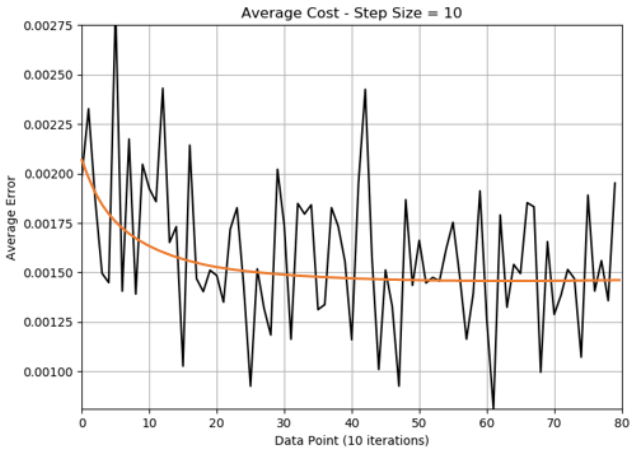



Figure 12 Learning progression in 800 iterations with 10 per cycle

The final optimized structures were saved in a database. Based on a trial of 1000 targets sampled randomly to form a validation set, the following results were generated for each measure of performance as shown in Table 6.

Table 6
Performance results

| ***Metric*** | ***Untrained*** | | | ***Trained*** | | |
|---|---|---|---|---|---|---|
| *Information State* | *1* | *Mean* | *6* | *1* | *Mean* | *6* |
| Mean Absolute Error | 0.285 | 0.152 | 0.073 | 0.121 | 0.096 | 0.066 |
| Error Standard Deviation | 0.161 | 0.120 | 0.053 | 0.082 | 0.069 | 0.044 |
| Range Classification Accuracy | 12 % | 37 % | 62 % | 46 % | 55 % | 79 % |
| Bin Classification Accuracy | 13 % | 33 % | 72 % | 45 % | 48% | 69 % |

Using range classification, there is an average improvement of 18%. Bin classification showed an improvement of 15%. The range classifications are higher which is expected since when using bin classification, there are cases when even if the prediction is very accurate, the deviation causes the prediction to fall within the wrong bin.

The mean error and standard deviation also give merit to the performance of the algorithm. The mean error is reduced by 37% from 0.152 to 0.096. Additionally, the average standard deviation is almost halved. This is an indication that the learned structure is not only more accurate, it also attests to the fact that when it is wrong, it is still more precise. The error is much more highly concentrated meaning that even an incorrect classification is likely not far from a correct one. The distribution of target threat values that are output during the validation set have the following distributions: green is the base case, red is the trained case and blue is the reference threat output.

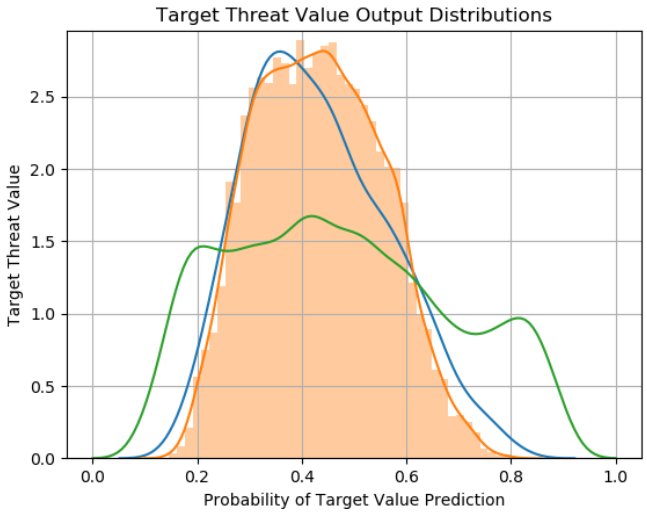



Figure 13 Target threat value output distributions

Figure 13 graphically shows the reduction of not only the error of classification but also the standard deviation. It is important to remember that the trained case was trained on a training set that had a uniform distribution. It was not given any information about what the probability distribution might look like.

### 6.3 Analysis of the information states

One of the other goals of this study was to prove the ability to predict dependent cues. As shown in Figure 14, the base case accuracy for each information state is represented in red and the optimized value is in green. Organizing the information states by the number of known independent cues produces the results shown in Figure 14.

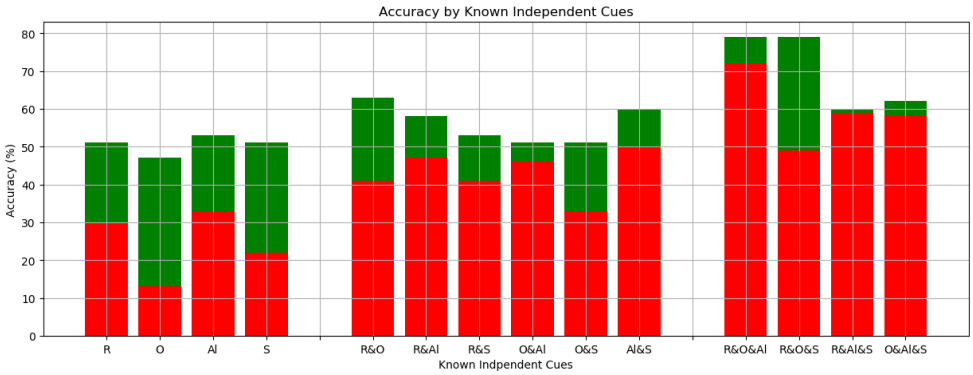



Figure 14 Independent cues accuracy

## 7. Conclusions

The AHP is a useful tool for synthesizing a decision based on the opinions of a SME. Supplementation of SSF nodes into the standard decision structure preserves the decisions that would have been made before the supplementation. The goal is to be able to predict the decision of a complete SSF supplemented decision structure with weights generated through the AHP when not all the information is available. This paper has shown that by applying the AHP to derive the weights and using machine learning to adjust the parameters of the sigmoid nodes in the deficient decision structures, the predictive capabilities can be vastly improved. This method is useful for all states of information availability and shows promise for use in practice.

## REFERENCES

Johansson, F. & Falkman,G. (2008). A Bayesian network approach to threat evaluation with application to an air defense scenario. *11th International Conference on Information Fusion,* 1-7. Cologne.

Liebhaber, M.J., Kobus, D.A., & Feher, B.A. (2002). Studies of U.S. navy cues, information order, and impact of conflicting data. *Technical Report 1888*. San Diego: SPAWAR Systems Center. Doi: https://doi.org/10.21236/ada406334

Menon, A., Mehrotra, K., Mohan, C.K., & Ranka, S. (1996). Characterization of a class of sigmoid functions with applications to neural networks. *Neural Networks,* 9*(5),* 819-835. Doi: https://doi.org/10.1016/0893-6080(95)00107-7

Mu, E. (2006). A unified framework for site selection and business forecasting using ANP. *Journal of Systems Science and Systems Engineering,* 15*(2)*, 178–188. Doi: https://doi.org/10.1007/s11518-006-5006-6

Ruder, S. (2017). An overview of gradient descent optimization algorithms. *Computer Research Repository (CoRR)*, 1609.04747.

Saaty, T.L. (1987). *The Analytical Hierarchy Process – What it is and how it is used.* Pittsburgh, PA: Pergamon Journals Ltd.