Learning Belief Connections in a Model for Situation Awareness

Maria L. Gini[1], Mark Hoogendoorn[2], and Rianne van Lambalgen[2]

[1] University of Minnesota, Department of Computer Science and Engineering
200 Union Street SE, Minneapolis, MN 55455, United States of America
gini@cs.umn.edu
[2] VU University Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen,r.m.van.lambalgen}@few.vu.nl

Abstract. Situational awareness is critical in many human tasks, especially in cases where humans have to make decisions fast and where the result of their decisions might affect their life. This paper addresses the problem of learning optimal values for the parameters of a situational awareness model. The model is a complex network with nodes connected by links with weights, which connect observations to simple beliefs, such as "there is a contact", to complex belief, such as "the contact is hostile", and to future beliefs, such as "it is possible the pilot is being targeted". The model has been built and validated by human experts in the domain of F16 fighter pilots and is used to study human decision making. Given the complexity of the model, there is a need to learn appropriate weights for the connections, which, in turn, affect the activation levels of the beliefs. We propose the use of a genetic algorithm and of a sensitivity based approach to learn the weights in the model. Extensive experimental results are included.

1 Introduction

In order to create agents that exhibit human like intelligent behavior, one of the crucial elements to be addressed is the formation of an understanding of the current situation. In case an agent lacks such an understanding, it is very difficult to come to intelligent decisions. In psychological literature this process of becoming aware of the current situation is commonly referred to as Situation Awareness (see e.g. [1], [2]).

A variety of models for situation awareness have been proposed (see e.g. [3]). The main principle behind the majority of these models is that certain knowledge is present that expresses relationships between the various concepts in the world. The agent can utilize this knowledge by combining observation knowledge (which is most likely partial) with the knowledge about the aforementioned relationships. A complete picture of the situation can thus result. Such knowledge about relationships between concepts in the world is however mostly domain dependent, and for each new domain, a domain expert needs to specify the relationships that hold within the domain. Ideally, one would want an agent to learn these relationships based upon examples it has seen in the world. Since this is a challenging task, we leverage the

D. Kinny et al. (Eds.): PRIMA 2011, LNAI 7047, pp. 373–384, 2011.

availability of networks of the relevant concepts and their connections built for many domains by domain experts. We use such a network to simplify the learning task.

In this paper, an existing model for situation awareness (cf. [4]) is taken as a basis. The model basically consists of beliefs which have a certain activation value, and a network which expresses relationships between the beliefs in the form of connections with a certain strength (in this paper we will refer to the value of a connection strength as weight).These relationships are not only one-to-one, but can also be more complex whereby multiple beliefs are aggregated into more complex beliefs about the situation. In addition, the model also incorporates a time aspect, whereby influences of connections are calculated based upon their importance, and inference stops when the time limit has been reached. In order to learn the weights of the relationships, two algorithms are introduced, namely a genetic algorithm, and an algorithm which is based upon the importance of the weights. Both algorithms are compared based upon a case study from the domain of F16 fighter pilots.

This paper is organized as follows. First, related work is presented in Section 2. The model for situation awareness which is used in this paper is described in Section 3. Section 4 describes the learning algorithms that are utilized. The case study is in Section 5 and the results in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Many approaches exist to estimate parameters of a given model. Since many parameters of a system or model might not be known in advance and can only be determined by the observed behavior of a system, such techniques are essential in a variety of fields. Approaches that are based on analytical mathematical techniques can be used (see e.g. [5]), but especially when looking at highly complex models they can be difficult to apply. Other approaches include Genetic Algorithms (cf. [6]) but also algorithms such as Simulated Annealing (cf. [7]). In this paper, the main focus is on two approaches, namely Genetic Algorithms and a sensitivity-based approach.

Genetic Algorithms (GA) are a popular method to solve a variety of optimization problems (cf [6]; [8]) GAs have the advantage over other methods of being simple to use and capable of exploring a large part of the search space. GAs can converge to local minima, but appropriate choices for crossover points, methods to maintain the distribution of the population, and a higher probability of mutation typically enable converging to a good quality solution. GAs have been used, for instance, for mission planning ([9]) and situational awareness ([10]). Other machine learning methods have been used for situational awareness, such as particle filters that have been used for state estimation for Mars Rovers ([11]) and Adaptive Resonance Theory (ART) that has been used for information fusion ([12]).We use GAs for parameter optimization, specifically to find the weights of the links that connect simple beliefs, complex beliefs, and future beliefs. We have chosen to use GAs because there is a large number of links in the network and hence a large number of parameters we need to learn. Since we know from the experts the expected activation levels of the complex beliefs, we use those in the fitness function.

Sensitivity analysis can also be used to refine models. For instance, sensitivity analysis has been used to prune units in a feedforward neural network ([13]) or to update parameters in Bayesian networks ([14]). We use sensitivity analysis to estimate the weights on the links.

The model we use for situation awareness is discussed next. Other models, such as Fuzzy Cognitive Maps ([15]), could be refined in a similar way.

3 Model for Situation Awareness

The model for Situation Awareness which has been adopted (cf. [4]) consists of four main components. Three components are in line with the model of Endsley [1] which includes the perception of cues, the comprehension and integration of information and the projection of information for future events. The last component describes the mental model of the human which describes the connections between the various states in the situation awareness model. The model functions as follows. Initially, the agent starts to observe within the world, and obtains the results of these observations. Observations are obtained with a certain degree of certainty. Using these observations and the knowledge stored in the mental model, simple beliefs about the situation are derived. Simple beliefs concern simple statements about the current situation that have a one-to-one mapping to observations, or have a one to one mapping to another simple belief. An example of a mental model including such a mapping is shown in Figure 1 (whereby the lower part concerns the aforementioned mapping). This figure assumes a fighter pilot setting whereby the fighter pilot is trying to judge his current situation. In the figure, it can be seen that there are two observations: whether a plane is closing in (closing_in) and whether this plane comes from a hostile direction (from_hostile_direction). These are also present as simple beliefs within the mental model (with a direct connection between the observation and the belief). In addition, a simple belief is present that the plane is hostile (hostile_plane). Simple beliefs are represented by the following predicate:

simple_belief: INFO_ELEMENT x TIME x VALUE

In the predicate, the value presents the *activity* of the belief in the mind of the agent, which depends on a number of aspects, such as the certainty of the observation. In order to translate the certainty of an observation into an activation of a belief, the following rule is used (note that the elements that are part of the mental model are shown in gray). The arrow (↠) represents a temporal relationship, namely that the antecedent being true for 1 time point results in the consequent being true for 1 time point:

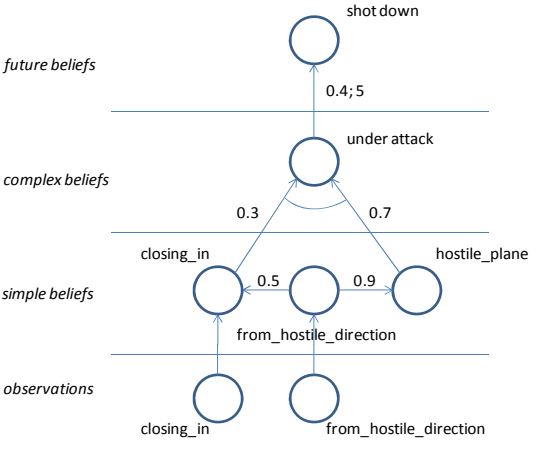

Fig. 1. Example mental model

LP1: Observations to simple beliefs

current_time(t) ∧ observation_result(I, t, V1) ∧ simple_belief(I, t-1, V2) ∧
simple_belief_decay(I, γ) ∧ steepness(I, σ) ∧ threshold_value(I, τ) ∧ recency_influence(I, α)
→ simple_belief(I, t, (1-α)·γ·V2 + α·th(σ, τ, V1))

This expresses that in the formation process of a simple belief, both the certainty of the observation and the old value of the belief are considered. Two parameters influence the value of a new observation compared to a previous belief value as well as how fast the belief decays (i.e., how fast the belief looses activation). Furthermore, a so-called threshold function is applied with parameters σ and τ, representing the steepness of the function and the threshold respectively. It enables the transformation of the certainty value of an observation to the activation value of a belief. Something might be observed with only a very low certainty but due to its importance, it might still result in a high activation. During times when no observations are made concerning a specific belief, the belief just decays.

After the new activation value of simple beliefs has been calculated, the influence of the simple beliefs among each other is determined. Hereby, influence weights reside in the interval [-1,1]. Figure 1 also shows the weights, whereby there is a strong connection (0.9) between the belief that the plane comes from a hostile direction, and the belief that it concerns a hostile plane. Furthermore, a somewhat weaker connection (0.5) exists between a plane coming from a hostile direction, and the plane closing in. In domains where there are clear relationships between beliefs, experts typically have stronger connections than novices. In order to calculate the new values for simple beliefs due to mutual influences, an iterative form of updating is used. This is based upon calculating all the influences that originate from the simple belief with the highest activation value:

Method 1: Updating simple beliefs

1. Search for the simple belief with the highest value that has not been considered yet and whose value is above the threshold.
   - For all connections originating from the selected belief:
     a. Select the connection with the highest strength originating from the selected belief that has not been considered yet of which the absolute value is above the minimal connection threshold. In case none are left, go to (d). If none were present in the beginning, go to (e).
     b. Perform calculations (LP2 shown below)
     c. Mark the connection as considered and go to (a).
     d. Add 1 to the time used.
     e. Mark the selected belief as considered. In case the time has reached the maximum time the algorithm terminates, otherwise go to 1.

This algorithm has an anytime behavior, and stops when the available time has ended. The updating of the belief is expressed as follows:

LP2: From simple beliefs to simple beliefs

current_time(t) ∧ simple_belief(I1, t, V1) ∧ simple_belief(I2, t, V2) ∧ connection_strength(I1, I2, w1) →simple_belief(I2, t, V2 + γ·(Neg(V1·w1·V2) + Pos((1–V2)·(V1·w1)))

After simple beliefs are updated, complex beliefs are derived from them. Complex beliefs are aggregations of multiple beliefs (simple or complex) and describe the situation in a composed manner. Figure 1 shows an example, whereby the simple beliefs about a hostile plane and the fact that the plane is closing in results in the

complex belief that you are under attack. In the model, it is assumed that the complex beliefs are calculated by taking a weighed sum of the relevant simple beliefs. An iterative form is used to update the complex beliefs, identical to method 1 sketched above. The updating of the value itself is expressed in LP3:

LP3: From simple to complex beliefs
complex_belief(CI1, t, VI1) ∧ belief(I1, t, V1) ∧ ….∧ belief(In, t, Vn) ∧
in_same_group(I1, ... In, CI1) ∧ connection_strength(I1, CI1, w1) ∧ ….. ∧
connection_strength(In, CI1, wn) ∧ steepness(CI1, σ) ∧ threshold_value(CI1, τ) →
complex_belief(CI1, t, VI1 + γ_c·(f(w1V1, ... , wnVn) – VI1))

Here, the contributions of the beliefs that together form a connection to the complex belief are calculated using a combination function f (e.g. a logistic threshold function or weighted sum). In case no new information is present with respect to a complex belief, a simple decay of the activation value is assumed.

In order to project the complex beliefs to the future situation, they are forwarded to the component belief formation on future situation. To calculate activation values of future beliefs, time and delay parameters are an aspect of the connection strength used to derive the specific belief (see again Figure 1 for an example, shot down in this case refers to the pilot that is being modeled being shot down):

LP4: From complex to future beliefs
complex_belief(I1, t, V1) ∧ ..... ∧ complex_belief(In, t, Vn) ∧ future_belief(FI1, t+D, VI) ∧
in_same_group(I1, ... In, FI1) ∧ delay_parameter(I1, ... In, FI1, D) ∧
connection_strength(I1, FI1, D, w1) ∧ ..... ∧ connection_strength(In, FI1, D, wn) ∧
steepness(FI1, σ) ∧ threshold_value(FI1, τ)
→ future_belief (FI1, t+D, VI+γ_f·(f(w1V1, .. , wnVn) – VI))

Note that the future beliefs can be the same as the complex beliefs. An agent might for instance know that the belief refers to a state that will happen in 5 time points.

The judgment of the future situation that then follows is used to direct the observations of the agent, together with the goals of the agent at a specific point in time. Also, goals and complex beliefs are used by the agent to make decisions on the actions to take. However, these aspects are outside the scope of this paper.

4 Learning SA Model Parameters

The focus in this paper is to learn the weights of the mental model (i.e. not the other parameters part of the model) In order to learn these connections, two different approaches have been utilized, namely a genetic algorithm and a dedicated approach based upon measuring the importance of the weights. Before they are explained, the definition of the fitness function, which both approaches use, is addressed.

4.1 Fitness Function

In order for the learning to take place, a fitness function needs to be defined that expresses how well a solution complies with the desired state. In this case, the fitness can be measured in terms of how much the activation levels differ from the ideal activation levels (i.e. the activation levels an expert considers appropriate) since we

want the model to exhibit a good performance on each state. The following algorithm can then be used to determine the fitness:

Algorithm 1. Calculate full fitness

```
fitness = 0;
for all time points t
  for all simple belief elements SB
    if the ideal simple belief value for SB is V1 at t
      and the current simple belief value for SB is V2 at t
        fitness = fitness + |V1-V2|
    end
  end
  for all complex belief elements CB
    if the ideal complex belief value for CB is V1 at t
      and the current complex belief value for CB is V2 at t
        fitness = fitness + |V1-V2|
    end
  end
  for all future belief elements FB
    if the ideal complex belief value for FB is V1 at t
      and the current complex belief value for FB is V2 at t
        fitness = fitness + |V1-V2|
    end
  end
end
fitness = fitness / (t * (|SB| + |CB| + |FB|))
```

In addition, a partial fitness function is also used, whereby more emphasis is placed on the formation of complex and future beliefs, as these are a true measure of the full understanding of the situation (since the complex and future beliefs are an aggregate of the simple beliefs). In this case, the approach as shown in Algorithm 1 can simply be reused except for the elements which concern the simple beliefs.

4.2 Genetic Algorithm

The genetic algorithm applied is a relatively standard GA from the Genetic Algorithm Toolbox[1]. Below, the most important aspects of the GA are explained.

- *Individual representation.* The population is composed of individuals that are represented by a binary string which represents real values (i.e. the weights in this case) with a certain precision.
- *Population initialization.* The population is in principle initialized randomly, but in some runs an individual with all zero weights has been explicitly added due to the fact that many connections between beliefs tend to be zero.
- *Selection.* The selection of individuals is performed by first ranking the individuals using linear ranking and then selecting individuals based upon stochastic universal sampling.
- *Mutation.* The mutation operator used is straightforward: each bit is simply mutated with a certain probability.
- *Crossover.* A single point crossover function is used to combine the individuals.

[1] Downloadable from `http://www.shef.ac.uk/acse/research/ecrg/gat.html`

4.3 Sensitivity Based Approach

The Sensitivity Based approach that is taken in this paper consists of two parts. In the first part sensitivities of all weights are calculated according to Algorithm 2. In the second part, weights are sorted from high to low sensitivity and the X most sensitive weights are adjusted (simulations can be performed with different values of X). The method to select and adjust the weights is shown in Algorithm 3. Each simulation cycle runs both parts, so that after adjustment of the weights the sensitivity of each weight is calculated again.

Algorithm 2. Determine Sensitivities

```
initialize weights
t = 0
sens_increment = 0.05

while t < end_time

run the SA model with current weights and
determine fitness value f_current

  for all weights W
    if the weight value for W is V1 at t
      V_old = V1
      V1 = V_old + sens_increment
      run the SA model with the current weights and the new V1
          and determine the fitness value f_add
      f_diff_add = f_add – f_current
      V1 = V_old – sens_increment
      run the SA model with the current weights and the new V1
          and determine the fitness value f_sub
      f_diff_sub = f_sub – f_current

      if f_diff_add > f_diff_sub,
        the value of upwards for W: U(W, t) = 1
        f_diff = f_add
      else
        the value of upwards for W: U(W, t) = -1
        f_diff=f_sub
      end
      the sensitivity S for W at t = f_diff / sens_increment
      V1=V_old
  end
 end
end
```

Algorithm 3. Adjust Parameters

```
for all weights W
   if W belongs to the X most sensitive weights
     if W has value V, sensitivity S and upwards value U
        if W should be adjusted upwards (U(W, t) == 1)
           V=V+tune_increment
        else if W should be adjusted downwards (U(W, t) == -1)
           V=V-tune_increment
        end
     end
  end
end
```

Simulations were performed using two different versions of Algorithm 3. In the first (Sensitivity_Based_fixed), a fixed number (fixed_increment) was added or subtracted to the dedicated weight. In the second (Sensitivity_Based_adjusted), this number was adjusted using the sensitivity according to the following formula:

$$\text{tune\_increment}=(\text{fixed\_increment}/(S*X))*\text{speed}$$

Here, S is the sensitivity and speed is the speed with which the weights are adjusted. X is the number of tuned weights.

5 Case Study: Fighter Pilots

The case study used to evaluate the approach is a short version of the case study presented in [4]. The idea of applying the model in this context is to develop human-like agents against which human fighter pilots can practice in a simulator.

In this simplified case study it is assumed that a pilot performs observations through a radar warning receiver and forms beliefs regarding the behavior of the opponent. The opponent of the pilot uses the radar to search for the agent, to track the flying pattern of the agent, or (when the opponent has found the agent) to lock it and eventually shoot a missile. The radar warning receiver can generate an occasional tone, a frequent tone or a continuous tone, which are translated into simple beliefs. These beliefs can be derived into other simple beliefs, indicating that the pilot is respectively searched, tracked or locked by the opponent. Each simple belief is connected to another simple belief and the values of these connections need to be learned. Connection values from the simple beliefs to complex beliefs (*not detected*, *detected*, *tracked* and *locked*) are learned in this paper as well as connection values from complex beliefs to future beliefs (*detected*, *ownship tracked*, *ownship locked* and *opponent missile release*).

6 Experimental Analysis

In this section, the results of the various approaches that have been introduced in Section 3 are shown. First, the overall setup of the evaluation is discussed, followed by results with the full fitness function (based on an evaluation of all activation values) and the results with a partial fitness function.

6.1 Evaluation Approach

In order to investigate how well the two approaches are able to find parameters of the model that describe the desired behavior well, they have been tested on a specific case study (as described in Section 4). The relevant states, connections between the states, and the strengths of those connections have been determined by domain experts. Furthermore, a scenario in which observations get a certain value at specific time points has been defined by domain experts as well. After running the simulation with these settings, the domain experts evaluated the resulting activation values, and they

indicated the levels were indeed as they would expect. In the experiments, these activation levels have therefore been used as a golden standard. Hereby, it concerns a total of 23 states, 229 possible connections, and 50 time steps during which new activations come in. As a result, a total of 1150 activation values occur over the entire timeline. For the full fitness function, the average deviation of all these states is used as a measure to determine the fitness, whereas for the partial fitness function merely the activation level of the complex and future beliefs are used. In order to compare the activation values, a simulation of the model is performed with the current set of parameters and the resulting activation values are compared.

6.2 Results

Table 1. Algorithm settings

Genetic Algorithm		*Sensitivity-based Algorithm*	
Parameter	**Setting**	**Parameter**	**Setting**
population size	200	sens_increment	0.05
number of generations	1000	X	10
precision of variables (number of bits)	6	fixed_increment	0.1
generation gap	0.5	speed	0.1
crossover rate	0.7		
mutation rate	0.4		

The first series of experiments have been run based upon the *full fitness function*, in which all activations are used to determine the fitness of a certain set of parameters. For the Genetic Algorithm, two runs have been performed: one whereby the initial population explicitly includes an individual with all weights set to zero (and the rest initialized with random values), and a second whereby all individuals are initialized with random values. Both approaches have been included due to the fact that many of the weights to be learned are typically zero. The settings used for the parameters of the Genetic Algorithm in all experiments are shown in the first part of Table 1 and have been set to this value in accordance with the experiences obtained while running

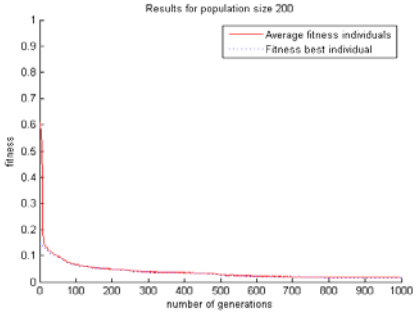

Fig. 2. GA with initial random population and full fitness function

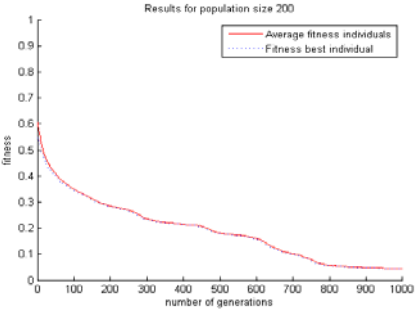

Fig. 3. GA with initial random population and zero and full fitness function

the algorithm. Considering the Sensitivity Based algorithm, simulations were done first to derive parameter settings for the optimal results. The parameter values are in the second part of Table 1. Both the Sensitivity_Based_fixed and the Sensitivity_ Based_adjusted approach were used with initial weights set either to 0.5 or to 0.1.

Results for the GA are presented in Figure 2 and 3. Results for the Sensitivity_Based are presented in Figure 4 and 5. Overall results are given in Table 2. Figure 2 shows the fitness over the number of generations given a completely random initial population. The fitness gradually decreases as the number of generations goes up, and stabilizes at a fitness value around 0.044, which means that only a very small deviation of the activation values remains. Hence, the learning process has been very successful. Figure 3 shows the results with the full fitness function, and when an individual with all weights set to zero is included in the initial population. Hereby, the results are immediately a lot better, and also converge to an even higher precision: 0.017. The weights also deviate a bit less from the golden standard, but the standard deviation is a bit higher. Another interesting aspect is how close the learned weights are compared with the expert weights that have formed the basis of the golden standard. These results are shown in Table 2 (along with the detailed results of all experiments).

Table 2. Detailed results of parameter estimations

Setting	Full fitness	Partial fitness	Average weight dev.	Standard deviation weights
Training on full fitness				
GA	0.043956	0.030761	0.230297	0.316123
GA with zero	0.016647	0.004239	0.216074	0.343887
SBfixed	0.3723	0.5860	0.4592	0.3093
SBfixed 0.1	0.1582	0.0330	0.2948	0.3632
SBadjusted	0.3893	0.5772	0.4662	0.2802
SBadjusted 0.1	0.0624	0.0162	0.2459	0.3607
Training on partial fitness				
GA	0.220711	0.006288	0.406425	0.398423
GA with zero	0.149290	0.000428	0.303008	0.387562
SBfixed	0.3486	0.2460	0.3633	0.4198
SBfixed 0.1	0.1669	0.0115	0.1926	0.3055
SBadjusted	0.3253	0.2334	0.4070	0.4592
SBadjusted 0.1	0.1909	0.0178	0.2325	0.3328

The weights are shown to deviate quite a bit (certainly considering the fact that the majority of the weights cannot deviate more than 1 due to the chosen range), and the standard deviation is also quite high. This shows that despite the weights not being exactly as in the golden standard, the behavior of the model is still satisfactory. This is expected when using such a complex model.

Figure 4 and 5 show how the fitness decreases when using either the Sensitivity_Based_fixed or the Sensitivity_Based_adjusted method (final results are shown in Table 2). It can be seen from Figure 4 that the fitness is similar in the adjusted method as compared to the fixed method (around 0.375). When starting at 0.1 (Figure 6), the final fitness is much lower. The adjusted method performs better (fitness around 0.06) as compared to the fixed method (fitness around 0.15).

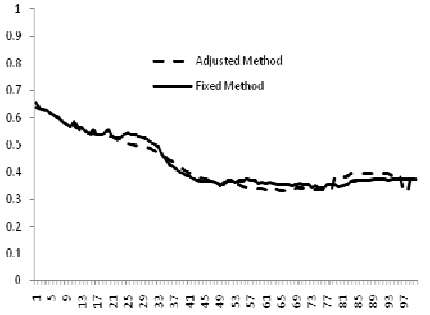

Fig. 4. Sensitivity based analysis using the full fitness function and initial weights set to 0.5

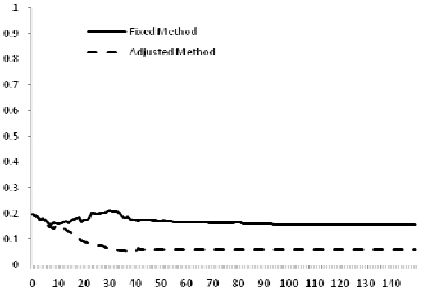

Fig. 5. Sensitivity based analysis using the full fitness function and initial weights set to 0.1

Besides the full fitness function, the algorithm has also been run with the *partial fitness function*. The results hereof are shown in Table 2. When tuning is performed on the partial fitness function, the performance of the different varieties of the algorithms is very good. The results generalize quite well for the full fitness function as well (going up to an accuracy of about 0.15). When tuning is performed on the full fitness function the performance on the partial fitness function is shown to vary a lot.

Note that the time and space complexity of the algorithms are not described for the sake of brevity. The complexity is mainly located in calculating the fitness of the parameter setting as this requires running the model itself.

7 Conclusions and Future Work

In this paper, an approach has been presented that enables the learning of parameters of a model for Situation Awareness. Models for Situation Awareness are crucial to enable agents to have a good understanding of the current state of the world and undertake appropriate actions. The models frequently have a large set of parameters and setting them manually is time consuming. Therefore learning these parameters is essential. Two learning approaches have been adopted, namely genetic algorithms and a sensitivity-based approach. Both showed very good results, with a very high accuracy of reproducing the judgment of the situation given certain observations in the world (although there is quite some deviation between the weights used to generate the test cases and the learned weights). The genetic algorithm performs significantly better than the sensitivity-based approach.

In the current research the connections strengths were learned through a model with ideal connections that was available beforehand and ideal results that were judged by experts as realistic. However, for future work we would like to investigate online learning of the mental model weights based upon the success of actions that are performed using the current judgment of the situation. This would mean that the entire input for learning the connections between states in the world would be feasible just by looking at the appropriateness of the actions. Furthermore, we would like to investigate the suitability of other learning approaches as well, such as Bayesian learning.

Acknowledgments. This research has been conducted and partially funded in the SmartBandits project in cooperation with the National Aerospace Laboratory in the Netherlands and the Royal Netherlands Air Force. We would like to thank the other participants of this project for their valuable input and Jan Treur for the fruitful discussions. Support for Maria Gini was provided by a KNAW Visiting Professorship.

References

1. Endsley, M.R.: Toward a theory of Situation Awareness in dynamic systems. Human Factors 37(1), 32–64 (1995)
2. Endsley, M.R.: Theoretical underpinnings of situation awareness: a critical review. In: Endsley, M.R., Garland, D.J. (eds.) Situation Awareness Analysis and Measurement. Lawrence Erlbaum Associates, Mahway (2000)
3. So, R., Sonenberg, L.: Situation Awareness in intelligent agents: foundations for a theory of proactive agent behavior. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), pp. 86–92 (2004)
4. Hoogendoorn, M., van Lambalgen, R., Treur, J.: Modeling Situation Awareness in human-like agents using mental models. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (2011)
5. Koch, K.R.: Parameter Estimation and Hypothesis Testing in Linear Models. Springer, Heidelberg (1999)
6. Kirkpatrick, S., Gelatt, C.D., Vechhi, M.P.: Optimization by simulated annealing. Science 220, 4598, 671–680 (1983)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)
8. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Trans. on Evolutionary Computation, 9(3), 303–317 (2005)
9. Rosenberg, B., Richards, M., Langton, J.T., Tenenbaum, S., Stouch, D.W.: Applications of multi-objective evolutionary algorithms to air operations mission planning. In: Proceedings GECCO (2008)
10. Salerno, J., Hinman, M., Boulware, D., Bello, P.: Information fusion for Situational Awareness. In: Proc. Int'l Conf. on International Fusion Cairns, Australia, (2003)
11. Dearden, R., Willeke, T., Simmons, R., Verma, V., Hutter, F., Thrun, S.: Real-time fault detection and situational awareness for rovers: report on the Mars technology program task. In: Proc. Aerospace Conference, vol. 2, pp. 826–840 (March 2004)
12. Brannon, N., Conrad, G., Draelos, T., Seiffertt, J., Wunsch, D., Zhang, P.: Coordinated machine learning and decision dupport for Situation Awareness. Neural Networks 22(3), 316–325 (2009)
13. Engelbrecht, A.P., Fletcher, L., Cloete, I.: Variance analysis of sensitivity information for pruning multilayer feedforward neural networks. In: Int'l Joint Conf. on Neural Networks, vol. 3, pp. 1829–1833 (1999)
14. Wang, H., Rish, I., Ma, S.: Using sensitivity analysis for selective sarameter update in Bayesian network learning. AAAI Spring Symposia, Tech. report SS-02-03 (2002)
15. Kosko, B.: Fuzzy Cognitive Maps. International Journal of Man-Machine Studies 4, 65–75 (1986)