# Active Learning Agents with Artificial Neural Brains

**Byoung-Tak Zhang**
Department of Computer Engineering
Seoul National University
Seoul 151-742, Korea
E-mail: btzhang@comp.snu.ac.kr

## Abstract

Two case studies on active learning agents are described. In the first case, the agent is a robot arm whose task is to grasp a rolling ball. It learns from self-generated examples by initiating conversations with the external teacher. In the second case, the agent is an autonomous mobile robot that works in cooperation with other mobile agents. The mobile robots learn group behavior by actively moving, evaluating their moves, and adapting their internal models. In both cases, each agent has a neural network as its brain. Simulation results are provided that support the evidence that active learning agents not only look more intelligent but they also acquire domain knowledge or adapt to their environment more efficiently than conventional agents.

## 1 Introduction

From the cognitive science point of view, conventional neural learning methods are not very interesting. They usually process a large set of training examples in a blind manner, assuming the training set to contain sufficient information [Hinton 1989]. In this passive learning paradigm (see Figure 1 in the next page) the network, i.e. the neural agent, can not learn much more across the training set provided by the external teacher. In addition, such a passive learning agent cannot effectively adapt to a changing environment.

To be more useful, a learning agent should be able to actively interact with its environment to acquire new knowledge by generating hypotheses based on its own knowledge and testing them on its environment (Figure 2). Such active learning ability, often referred to as exploration or reflection, is a fundamental feature of intelligent systems.

Research on exploration has begun early in symbolic machine learning (see [Michalski *et al.* 1983] and references therein), but remains relatively unexplored. It is only recently that this fact has attracted the interest of many researchers and formed a new research field, known as active learning [Zhang and Veenker 1991, Cohn 1994, Zhang 1994], query learning or reflective learning [Paaß and Kindermann 1995]. Exploration has been, in one or the other way, applied to problems involving adaptation, such as in robotics [Thrun and Möller 1992] and autonomous vehicles with a changing environment [Jochem and Pomerleau 1996].

In this paper we review two of our case studies on active learning agents. The first example involves active learning in individual agents. Here the agent is a robot arm and its brain is a neural network [Zhang 1993]. The objective is to learn to position the joint angles of the robot arm so that its hand reaches the target object. Instead of learning by given training examples, the agent itself explores the example space and actively constructs new examples for training the neural network.

The second example involves active learning in collective agents. Here the agents are small mobile robots whose job is to transport a large table in teamwork [Hong and Zhang 1997]. Each agent is controlled by a chaotic neural network. Starting from the same neural network, each agent learns to cooperate by acting on its environment and by seeing how good its move was. This is a reinforcement learning situation in a group. We use an evolutionary algorithm for evolving neural networks [Zhang *et al.* 1997] for group behavior.

The paper is organized as follows. Section 2 describes the active learning method for the robot arm. Sections 3 and 4 describe the neural network architecture and the evolutionary algorithm for active learning of cooperating mobile robotic agentsm, respectively. Section 5 discusses the implications of the experimental results from the cognitive science perspective.

## 2 Active Learning Agents

Assume that we want a robot arm to follow and grasp a rolling ping-pong ball. This task can be divided into two subtasks. One is to predict the trajectory of the moving ball. The second subtask is to move the robot arm to reach the ball in the predicted position. Here, we deal with the problem of learning the second task, known as the inverse kinematics problem.

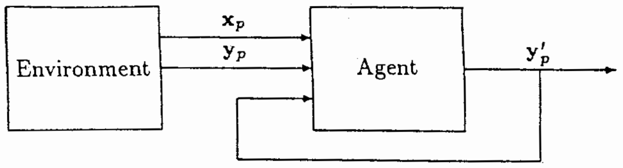


Figure 1: Passive learning agent.

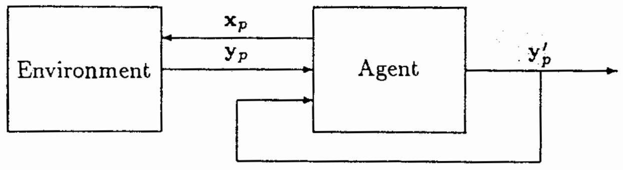


Figure 2: Active learning agent.

We used the robot arm RV M1 with five degrees of freedom (DOF), of which three DOF were used. The rotation of the hand was not considered since it is irrelevant to the task. Figure 3 shows the kinematic parameters of the robot arm. The angles $\theta_i$ represent the joint angles between two adjacent arm sections, respectively. The angle $\theta_1$ denotes the rotation angle of the basis of the arm.

The inverse kinematics problem for this work consists of determining the joint angles of the robot arm to reach the desired position of the ball. This task can be described as a transformation

$$IK: \quad (p_x, p_y, p_z) \to (\theta_1, \theta_2, \theta_3, \theta_4) \qquad (1)$$

where $(p_x, p_y, p_z) \in \mathbb{R}^3$ is a point in the work space of the robot arm. The vector $(\theta_1, \theta_2, \theta_3, \theta_4) \in \mathbb{R}^4$ describes a point in the configuration space of the joint angles.

To learn the mapping, a multilayer feedforward neural network is used. A total of 30 input units of the network encode the spatial position $(p_x, p_y, p_z)$ of the target object, in this case a ball, in the work space of the robot arm. The four joint angles $\theta_1, \theta_2, \theta_3, \theta_4$ are represented on 24 output units of the network. Thus, each training example consists of an input vector of 30 bits and an output vector of 24 bits.

Figure 4 describes the learning procedure for the robot arm agent. The neural network is initialized with random weights. Learning starts with a small number of training examples. The weights of the network are trained using a connectionist learning method, such as backpropagation. If the training reduces the error to an acceptable level, then the agent learns more examples by selecting given examples if they are available. However, if there are no examples any more, the agent generates new examples to learn more about the environment. This means that the agent initiates the learning procedure, which is in contrast to conventional super-

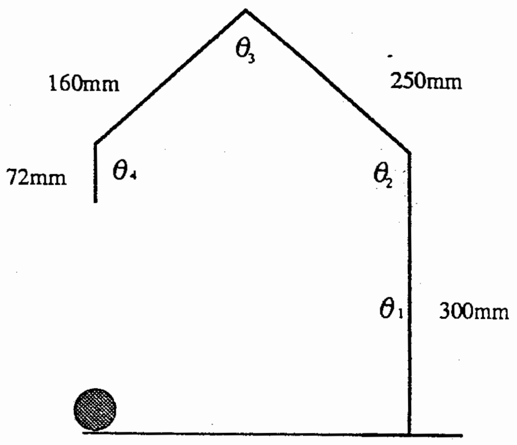


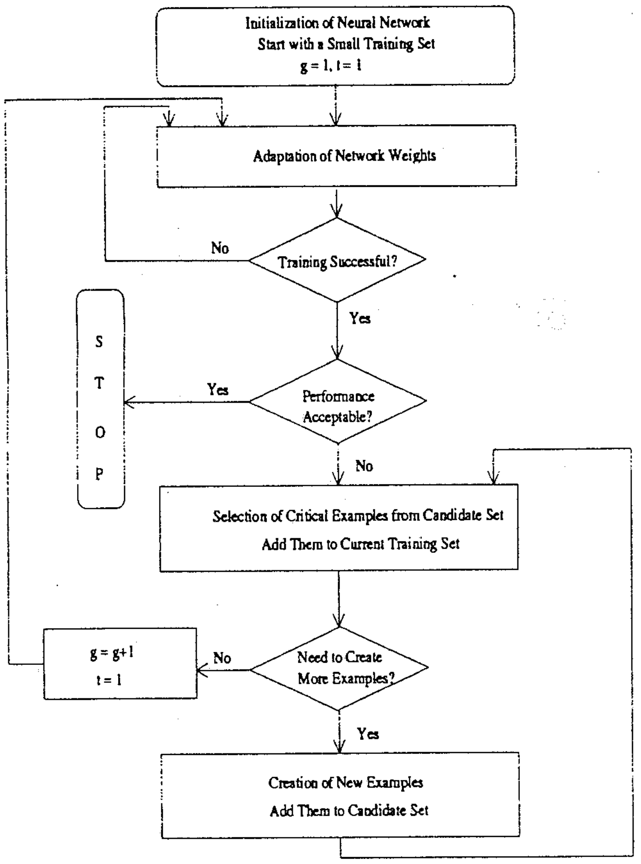
Figure 3: Kinematic parameters of the robot arm. The angles $\theta_i$ represent the joint angles between two adjacent arm links.

vised learning methods in which external teacher initiates learning.

In the experiments, we started with just one training example and let the agent generate for itself the input vectors, i.e. the training points. A genetic algorithm was used to create new input vectors from existing ones by considering them as chromosomes. The corresponding joint angles for the training points, i.e. the target values, were produced by a simulated teacher which led the arm to the desired position and measured the joint angles.

Analysis of the learning curves has shown continuous improvement of generalization performance, suggesting the examples created are very useful. To see the effectiveness of the example generation and selection mechanism we examined the sequence of examples used during the learning. Figure 5 shows the learning points that were explored and used to train the network. Shown in the figure is the $xy$-plane projected from the original $xyz$-space of three dimensions. Learning started at the bottom left corner of the work space as shown in the figure. The darkness of the field indicates the generation in which the corresponding example was introduced to the training set. The figure clearly demonstrates the tendency of the agent to search for good examples first in the vicinity of the starting points for the training to be economic, but sometimes it makes some jumps to distant regions to learn more. Using about a quarter (250) of the all possible examples, the genetic search explored the work space of the robot arm very well.

The analysis of the behavior of active exploration of the learning points in the configuration space of the robot arm has shown that many of the angle combinations are used already in early generations. This implies the learning agent tends to generate the examples



Initialization of Neural Network
Start with a Small Training Set
g = 1, t = 1
Adaptation of Network Weights
Training Successful?
No
Yes
Performance Acceptable?
Yes
No
STOP
Selection of Critical Examples from Candidate Set
Add Them to Current Training Set
Need to Create More Examples?
No
Yes
g = g+1
t = 1
Creation of New Examples
Add Them to Candidate Set

Figure 4: The procedure for active learning of the robot arm.

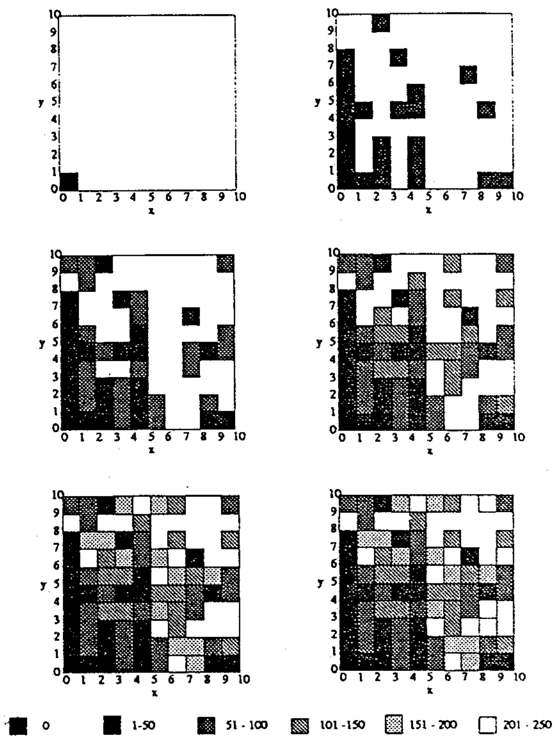


Figure 5: Active exploration of the training points in the work space of the robot arm. This figure shows the $xy$ plane projected from the three dimensional $xyz$ work space. The grey level of each field denotes approximately the time of the example (corresponding to the training point) added to the training set, darker ones meaning earlier selection. It can be seen that using only 250 examples (out of 1000 possible ones) created by genetic algorithm, almost all of the training points on this plane are explored.

which are involved with less frequently trained joints. In general, these examples will improve the performance of network maximally since they contain more information than others.

In summary, while the example creation component searches in the work space of the robot arm, the selection component seeks critical examples in the configuration space of the arm. The active learning by iteration of creation and selection of examples leads to an automatic correction of the correlation between the input space and the output space of the desired mapping. This results in fast improvement in the performance of the robot arm.

## 3 Collective Learning Agents

Some tasks can be done faster or more easily by dividing it up among many agents. Other tasks may not only be solved better by using multiple agents, but can only be effectively solved, by using teams of agents working together [Mataric 1993]. We have studied collective learning behavior of a group of mobile robots for the transport of a large table (Figure 6). The table transport problem cannot be performed by a single small robot; it requires multiple robots to cooperate.

Robots are cloned to form a homogeneous team. Cooperation strategies for a team of the robots are represented in a neural architecture consisting of multiple neural networks. Evolutionary algorithms [Zhang *et al.* 1997] are used to evolve the multinet architecture fitted for goal-directed group behavior.

The network consists of four layers (Figure 7). The bottom layer is the input layer that receives sensory inputs. The next three layers are the recurrent neural network (RNN) layer, the self-organizing map (SOM) layer, and the Cognitron output layer.

The recurrent neural network receives the environmental inputs and processes them to detect the spatio-temporal pattern of the inputs. It contains both excitatory and inhibitory neurons. The recurrent network used in the experiments contains 84 neurons in

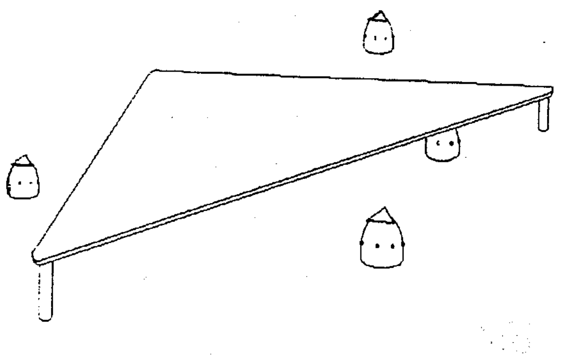


Figure 6: The robots in motion to transport the table.

a virtual cylinder. The neurons are connected with other neurons in the neighborhood more frequently than the neurons in the distant area within the cylinder. We adopted a neuron model with chaotic dynamics [Hong and Zhang 1997]. The activation is determined by:

$$\begin{aligned} x_i(t+1) &= f\left[\sum_{j=1}^{M} v_{ij} \sum_{d=0}^{t} k_e^d A_j(t-d) \right. \\ &\quad + \sum_{j=1}^{N} w_{ij} \sum_{d=0}^{t} k_f^d x_j(t-d) \qquad (2) \\ &\quad \left. - \alpha \sum_{d=0}^{t} k_r^d g\{x_i(t-d)\} - \Theta_i \right] \end{aligned}$$

where $v_{ij}$ and $w_{ij}$ are synaptic weights to the $i$th constituent neuron from the $j$th external input and from the $j$th constituent neuron, respectively, and $k_e$, $k_f$, and $k_r$ are the decay parameters for the external inputs, the feedback inputs, and the refractoriness, respectively. We set $k_e = k_f = k_r$ and $g\{x_i(t-d)\} = x_i(t-d)$. The recurrent network considers the spatio-temporal summation of both external inputs and feedback inputs from other chaotic neurons.

The neurons in the recurrent network learn their weights for each input data. The modified Hebbian learning rule is used to adapt the weights:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) \\ &\quad + r\{1 - w_{ij}(t)\}h\{x_i(t), x_j(t)\} \\ &\quad - K\{w_{ij}(t)\} \qquad (3) \end{aligned}$$

where $K\{w_{ij}(t)\}$ is the weight decay function. The parameters are determined by an evolutionary method. For each input data, the network is activated iteratively to converge to a pattern or oscillate between two or more patterns. For our application, it is useful if the network produces similar output patterns for similar input patterns.

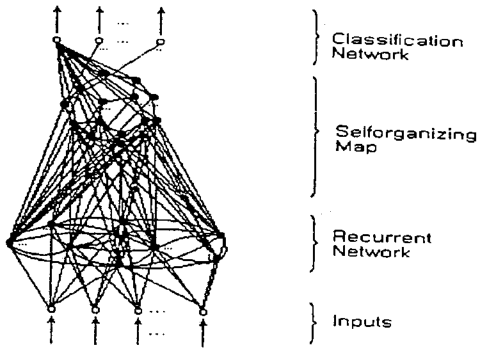


Figure 7: The neural architecture used for the mobile robotic agent.

The next layer is the self-organizing map (SOM). Its objective is to recognize the activation pattern of the recurrent network and classify it into a class. The SOM learns for every update of activation at the recurrent network layer:

$$w_{ij}(t+1) = w_{ij}(t) + r_{SOM}\{a_j - w_{ij}(t)\} \qquad (4)$$

This implies that the SOM layer learns to recognize the activation patterns of the recurrent network. If the RNN converges to a single pattern, then SOM recognizes it. The ultimate winner of the SOM is the unit whose weight vector has the smallest distance to one of the activation patterns at RNN.

The output layer determines the output of the network based on the winner of the SOM network. This layer is trained

$$y(s) = w_{cij}(t) + \Delta w_{cij}(t) \qquad (5)$$

using reinforcement learning as follows:

$$\Delta w_{cij}(t) = \begin{cases} r_c \cdot (1 - w_{cij}(t)) \cdot s \cdot a_j & \text{if } s \geq 0 \\ r_c \cdot w_{cij}(t) \cdot s \cdot a_j & \text{otherwise} \end{cases} \qquad (6)$$

where $s, -1 \leq s \leq 1$, is the normalized difference between the present penalty and the previous one. $a_j$ is the activation value of the winner at the SOM layer.

## 4 Learning Group Behavior by Doing

Each of the mobile robotic agents has the multinet as its brain. The cooperative behavior of the mobile agents is learned by simultaneously moving and evaluating their moves in the group. At the outset, the agents are implanted with the same neural network which evolves as learning proceeds.

At the start of learning, each robot is given a bucket of points. The robots are allowed to move a fixed maximum

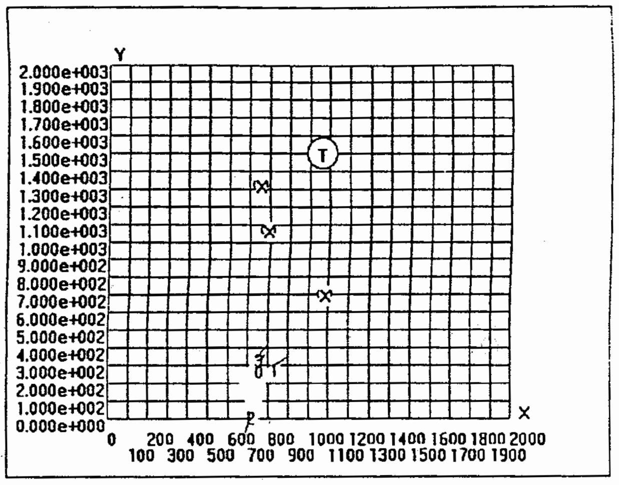


Figure 8: The robots moving in a group toward the table.

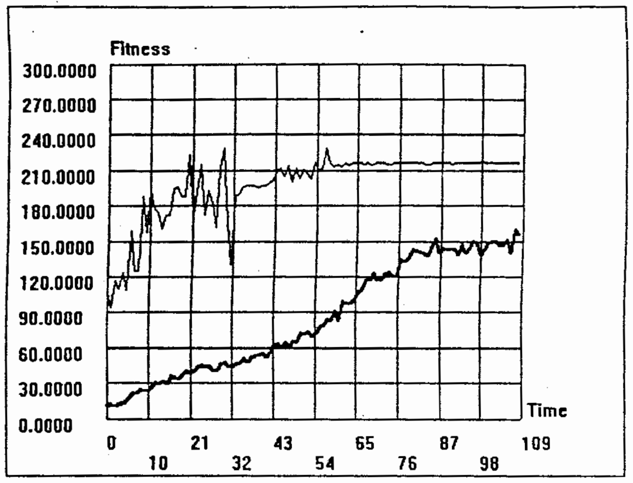


Figure 9: Change of the best and average fitness values.

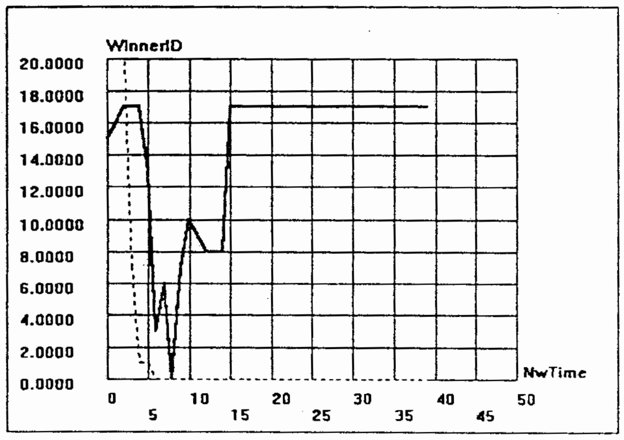


Figure 10: Change of the winner at the SOM layer while the RNN updates its activation.

number of steps. At each move, the badness of their behavior is evaluated and penalized. When it collides, for example, with its colleagues, it gets $K_{coll}$ points subtracted from the bucket. An amount of $K_{away}$ points is also subtracted when it moves too far away from its colleagues. This factor encourages herding behavior. When the table is out of sight of the robot on each move action, the robot gets $K_{sight}$ points decreased. In effect, the sum of the penalties on each movement is

$$\begin{aligned} A &= K_{away} \cdot \text{NumAways} \\ &+ K_{coll} \cdot \text{NumCollisions} \\ &+ K_{sight} \cdot \text{NumOutsights} \end{aligned} \tag{7}$$

where NumAways is a count for the number of being far away from the colleagues, and NumCollisions and NumOutsights are counts for collisions and being too far away from the table.

To encourage the movements of the robots, the resulting bucket of points is multiplied by the following factor

$$S = K_S \cdot \text{NumStepsMoved} \tag{8}$$

where $K_S$ is a constant amd NumStepsMoved is the total number of steps moved. This term penalizes the robots that stay at the same location or move seldom.

The bucket of remaining points is subtracted again by a fractional amount of the distance from the table:

$$D = K_D \cdot \text{FinalDisplacement} \tag{9}$$

where $K_D$ is a constant. This is to promote moving toward the table.

Overall, the fitness of a robot or its neural network is:

$$F_i = (\text{Bucket} - A) \times S - D \tag{10}$$

where $A$, $S$, and $D$ are defined as above.

After being evaluated their fitness, the individuals are selected to be parents for recombination. The selection probability of each individual is given as:

$$R_i = \frac{F_i - F_{min}}{F_{max} - F_{min}} \tag{11}$$

where $F_{max}$ and $F_{min}$ are, respectively, the maximum and minimum fitness values of the individuals in the current population. We used steady-state selection, i.e. 5% of the population are replaced after each evaluation. Thus the population evolves more slowly but continuously than in generational selection.

Figure 8 demonstrates the behavior of robots whose neural network was evolved after 200 generations. The environment contains three obstacles which are placed randomly between the robots and the table. The inputs to the neural network of each robotic agent are the detectors for the table to transport, colleagues, and obstacles. The network of each agent has 9 outputs, i.e. 8 for determining the direction and the rest one for non-moving.

Figure 9 plots the best and average fitness at each generation during evolution. As generation goes on, we could observe improvement in robots' herding behavior compared to that of the robots at early generations. Figure 10 shows the change of winners at the SOM layer as the recurrent neural network updates its activation. The tendency of convergence can be observed; after a period of oscillations for 15 iterations, the winner finally converged to unit 17. This suggests that, though the agents make some chaotic moves to explore their environment, some regularity (if it exists) can be detected and learned by the neural network using the reinforcement learning scheme.

## 5 Concluding Remarks

We have presented two examples of active learning agents. In both cases, the brain of agents consists of a neural network. Learning of the agents proceeds not in a passive manner, but in an active fashion. In the first case, the agent learns by generating the input part of learning examples in a supervised learning mode, while in the second case the agent learns from self-generated reinforcement signals in a collective learning environment.

From the cognitive science point of view, active learning agents are more interesting since they have the ability to evaluate their own knowledge and explore the environment to complement their weakness. Active learning agents are also very interesting from the practical standpoint. Our experimental results show that the active learning capability makes the agents more autonomous, more efficient, and thus more useful than conventional passive learning agents.

## Acknowledgments

This research was supported in part by grants from the Korea Science and Engineering Foundation (KOSEF) under contract numbers 961-0901-001-2 and 96-0102-13-01-3.

## References

[Cohn 1994] Cohn, D. A., Neural network exploration using optimal experiment design, AI Memo No. 1491, MIT, Cambridge, MA, June 1994.

[Hinton 1989] Hinton, J. E., Connectionist learning procedures, *Artificial Intelligence*, 40:185–234, 1989.

[Hong and Zhang 1997] Y.J. Hong and B.T. Zhang, A multinet neural architecture for evolving collective robotic intelligence, submitted to *Proc. ICONIP-97*, November 1997.

[Jochem and Pomerleau 1996] Jochem, T. and Pomerleau, D., Life in the fast lane: the evolution of an adaptive vehicle control system, *AI Magazine*, 17(2):11-49, 1996.

[Linden and Weber 1993] Linden, A. and Weber, F., Implementing inner drive by competence reflection, In *Proc. 2nd Int. Conf. on Simulation of Adaptive Behavior*, MIT Press, 1993.

[Mataric 1993] M.J. Mataric, Designing emergent behaviors: From local interactions to collective intelligence, *Proc. Second Int. Conf. on Simulation of Adaptive Behavior*, pp. 432-442, 1993.

[Michalski *et al.* 1983] Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (eds.) *Machine Learning: An Artificial Intelligence Approach*, Tioga, 1983.

[Paaß and Kindermann 1995] Paaß, G. and Kindermann, J., Bayesian query construction for neural network models, In *Neural Information Processing Systems 7*, MIT Press, 1995, pp. 443-450.

[Thrun and Möller 1992] Thrun, S. and Möller, K., Active exploration in dynamic environments, In Moody, J. E., Hanson, S. J., and Lippmann, R. P. (eds.) *Neural Information Processing Systems 4*, MIT Press, 1992.

[Zhang and Veenker 1991] Zhang, B.T. and Veenker, G. Neural networks that teach themselves through genetic discovery of novel examples, *Proc. Int. Joint Conf. on Neural Networks*, IEEE Press, pp. 690-695, 1991.

[Zhang 1993] Zhang, B. T., Teaching neural networks by genetic exploration, *Arbeitspapiere der GMD*, No. 805, German National Research Center for Computer Science (GMD), St. Augustin, November 1993.

[Zhang 1994] Zhang, B. T., Accelerated learning by active example selection, *International Journal of Neural Systems*, 5(1):67-75, 1994.

[Zhang *et al.* 1997] Zhang, B. T., Ohm, P. and Mühlenbein, H., Evolutionary induction of sparse neural trees, *Evolutionary Computation* 5(2), 1997.

[Zhang and Kim 1997] Zhang, B. T. and Kim, S. H., An evolutionary method for active learning of mobile robot path planning, *Proc. IEEE Symp. on Computational Intelligence in Robotics and Automation*, Monterey, CA, July 1997.