A Dynamic Tree Structure for Incremental Reinforcement Learning of Good Behavior

Tomas Landelius Hans Knutsson
Department of Electrical Engineering, Computer Vision Laboratory
Linköping University, 581 83 Linköping, Sweden
email: tc@isy.liu.se, knutte@isy.liu.se
Also published in Proceedings of the 2nd Swedish Conference on Connectionism as *Behaviorism and Reinforcement Learning.*

Abstract

This paper addresses the idea of learning by reinforcement, within the theory of behaviorism. The reason for this choice is its generality and especially that the reinforcement learning paradigm allows systems to be designed, which can improve their behavior beyond that of their teacher. The role of the teacher is to define the reinforcement function, which acts as a description of the problem the machine is to solve.

Gained knowledge is represented by a behavior probability density function which is approximated with a number of normal distributions, stored in the nodes of a binary tree. It is argued that a meaningful partitioning into local models can only be accomplished in a fused space consisting of both stimuli and responses.

Given a stimulus, the system searches for responses likely to result in highly reinforced decisions by treating the sum of the two normal distributions on each level in the tree as a distribution describing the system's behavior at that resolution. The resolution of the response, as well as the tree growing and pruning processes, are controlled by a random variable based on the difference in performance between two consecutive levels in the tree. This results in a system that will never be content but will indefinitely continue to search for better solutions.

1 Introduction

The quest for machines that learn from memorized experiences is one of the great challenges of modern science. The word machine origins from the Greek word *machina* related to an old word for power, or to make something possible, *mēchánē*. Machines, which today are claimed to possess an ability to learn show a remarkable rigid behavior, far behind that of the simplest biological organisms. However, a machine *capable* of learning would indeed possess the attributes here ascribed to the word machine. It would means be a very powerful tool, making things possible which are out of reach for any human programming effort.

In this paper a method of training machines in achieving the desired results will be studied. The method is based on the idea of learning by reinforcement, within the theory of behaviorism. Why reinforcement and

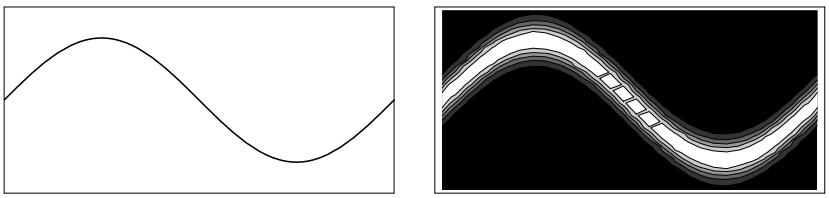

Figure 1: A distribution (left) representing the desired behavior (right).

behaviorism? These questions are now addressed in reverse order.

Behaviorism is all about the same issues as is machine learning - *prediction* and *control* of a system's behavior, *drilling*. The mission for a behaviorist is to make his subject *produce* the desired *results*. By no means should this choice of a behavioristic view on machine learning be seen as an advocacy for behaviorism in general. The behavioristic view taken here is first of all a technological one which can be useful when formulating control methods.

This paper focuses on what can be done with as few a priori assumptions about the system as possible, or stated in another way, with little system bias. J. B. Watson, the leader of the behavioristic revolution, explains learning as Pavlonian conditioning. A response is associated with a stimulus such that after the learning trial, the stimulus will result in the associated response with a certain probability. The learning system can be viewed as a black box which behavior is modeled only with a probability distribution, $p(\mathbf{d})$, over the product space of the stimulus and response spaces, $\mathbf{d} \in D = X \times Y$. Hence a stimulus, $\mathbf{x}$, and its associated response, $\mathbf{y}$, can be treated together as a *decision*, $\mathbf{d} = (\mathbf{x}, \mathbf{y})$. When the system has learned to solve a problem its decisions will be generated from an *optimal behavior distribution, OBD*.

The notion of stimulus should not be taken only to mean current sensory input. A key mechanism for learning is the system's ability to keep a memory of both past stimuli and responses. In fig. 1 the optimal system behavior is to produce a sine wave. This behavior can be described with a probability density function having its maximum along the decisions which constitute the optimal behavior. Probabilities are reflected in the grey levels of the graph to the right in fig. 1, the whiter the more probable a decision. Learning from the system's point of view is then equal to estimating the OBD.

The reason for a reinforcement learning paradigm is twofold. First, somehow the system must be told whether or not its responses are advisable. A natural way of doing this is to reward it, whatever this may mean, when it gives good responses, or at least when the task it has been facing is sat-

isfactory completed. Second, it allows systems to be designed which can improve their behavior beyond that of their teacher. Even the behavioristic theories that claim contiguity alone to suffice for generation of associations between stimuli and responses admit that rewards influence outcomes. The role of the teacher is to define the reinforcement function, which acts as a description of the problem the machine is to solve.

Hence, in reinforcement learning the teacher does not tell the system anything about the desired response but instead lets it know how good, or bad, it performed. This corresponds to the *value* of a performance function at a given point in the decision space, meaning that no directed information is given to the system. For a reinforcement learning system, appropriate actions can only be found by performing repeated trials and comparing the outcomes. However, having performed an action, the world may have responded and the system now views a different stimulus, making it impossible to try new responses to the previous stimulus. The previous remark makes it obvious that a system's abilities not only to sense, but also to respond, will be of great relevance to the learning capabilities of the system. These circumstances have been shown to be of major relevance to humans, illustrated in several experiments in perceptual psychology, which strongly suggest that closing the *system-environment loop* is necessary for a system to learn successfully, (Wilson and Knutsson, 1993). The idea that interaction with the world is necessary for knowledge acquisition has also been pointed out by empiricist philosophers throughout history. One example is given by the italian G. Vico, who claimed that the requirement for knowing something is that you have done it, and another by L. Wittgenstein who points out that the grammar of the word "knows" is closely related to that of "can", "is able to".

2 The Tree Representation

In the previous section it was suggested that an unbiased route towards learning machines would go via a representation of their behavior distributions. Any choice of approach will however inevitable introduce more or less bias. Even if the generality of a system is a desirable property it should of course be allowed to bias the system with knowledge and structures that are known to be of great value for speeding up the learning procedure. The environment of the system could provide essential clues and it must be acknowledged that evolutionary discoveries are not made in a day. But, care should be taken when introducing biological features as system bias, not to end up with "airplanes flapping their wings". The following two properties of the world seem harmless enough to be used to bias the system models and its architecture (Granlund and Knutsson, 1983).

- Continuity
- Locality

The continuity prior states that the world is geometric, yielding continuous models to be preferred before discontinuous ones, as long as data does not contradict it. The second prior states that models are preferred such that only a small portion of the experience base will be relevant to any specific situation, i.e. models should be localized in time and space.

A natural way to bootstrap a learning system would be to begin with an emphasis on single memories. When the system has no or few experiences in a domain, every single experience is critical and remembered more or less in detail. New responses are formed by generalizing from these stored experiences (Omohundro, 1992). Later on, when more data is available, models can be formed and there is no longer a need for using individual experiences to generate new responses. Instead the focus is on discovering regularities in the stored decisions and model them using methods for parameter fitting.

One of the major obstacles for any system engaged in realistic interaction with a natural environment is what has come to be known as the "curse of dimensionality" (Bellman, 1957). An approach to deal with the problems associated with the high dimensionality of the decision space is to fight them with a divide-and-conquer strategy. The strategy investigated in this paper is based on a partitioning of the decision space and the assignment of local models to the emerged partitions. This is in contrast with most standard neural networks which maintain a global model of the signal domain. It should be stressed that the space to be partitioned is a fusion of *both* the input and output spaces. Partitioning only the input space is meaningless since there is no way to decide whether a given subset of this space should be assigned a model or not. This is however possible in the decision space, where pairs of input and output which are worth representing are characterized by a high reward.

To build models the system will have to estimate parameters. Parameter fitting usually results in good generalization but has a fundamental problem in overfitting, i.e. having insufficient data to validate the model at hand as useful for generalization. A solution to the problem of overfitting is to start with a coarse model, when only a small amount of data is available. Then, as more data arrives, more complex models could be validated. Note that if some performance criterion is available it is possible to stop the model complexity from growing larger than necessary. Such a criterion makes it possible to vary the model complexity across the signal space. One of the more flexible approaches to model representation at various levels is a tree structure. This approach has been used for similar purposes under names such as regression and neural trees, see e.g. (Utgoff, 1989).

Gained knowledge, i.e. experienced stimulus-response pairs associated with a high reward, is represented by a behavior probability density function. This distribution is estimated by partitioning the set of experienced decisions, which are seen as samples from the behavior distribution, and storing local models of the decision distribution in the nodes of a binary tree struc-

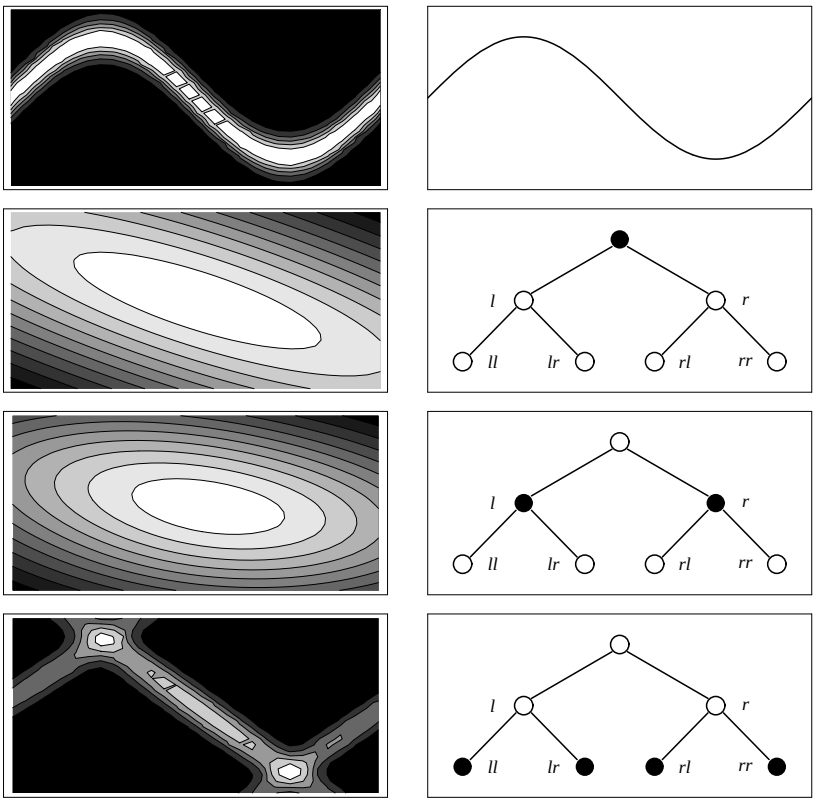

Figure 2: Hierarchical decomposition of the behavior distribution.

ture. Within each partition the local decision distribution is modeled with a normal distribution, which parameters are stored in the corresponding node in the tree. The idea behind the approach to employ local normal distributions in the approximation originated from the successful use of tensors as local signal descriptors (Knutsson, 1989). By summation of the distributions at a given level in the tree, an approximation of the behavior distribution at the corresponding resolution is obtained. The approximation hence goes from coarse at the root node, to fine towards the leaves of the tree. This is illustrated in figure 2 where a behavior distribution, describing a system exhibiting a sinusoidal behavior, is represented with models distributed in a binary tree with three levels. At the root node the distribution is approximated with one normal distribution only. On the second and third level the approximating sum contains two and four terms respectively.

When traversing the tree downwards from the root, one coarse node model is replaced with a sum of two finer ones at every new level. This means that the finest approximation of the behavior distribution can be seen to

originate from a stepwise, or recursive, refinement of the root distribution, p_0. This relationship which was illustrated in figure 2 can also be expressed as follows, where α are coefficients in the linear combination:

$$\begin{aligned} p(\mathbf{x}) &\approx p_0(\mathbf{x}) \\ p(\mathbf{x}) &\approx \alpha_l p_l(\mathbf{x}) + \alpha_r p_r(\mathbf{x}) \\ p(\mathbf{x}) &\approx (\alpha_{ll} p_{ll}(\mathbf{x}) + \alpha_{lr} p_{lr}(\mathbf{x})) + (\alpha_{rl} p_{rl}(\mathbf{x}) + \alpha_{rr} p_{rr}(\mathbf{x})). \end{aligned} \quad (1)$$

3 Tree Responses

In the previous section a behavior representation, suitable for machine learning, was discussed. An important aspect of such a representation was however left out. It must be useful to the system in its struggle to fulfill its mission. Given the tree representation and new sensory data, what is the system to do next? The tree is the system's only guide to response generation and must be asked for an output likely to receive a high reward. In order to answer the question not only should the decision probability function $p(\mathbf{x}, \mathbf{y})$ be represented but also, at least indirectly, the conditioned distribution $p(\mathbf{y} \mid \mathbf{x})$. This distribution is proportional to the distribution of decisions with a fix input, $\mathbf{x} = \mathbf{x}_0$:

$$p(\mathbf{y} \mid \mathbf{x}_0) = \frac{p(\mathbf{x}_0, \mathbf{y})}{p(\mathbf{x}_0)} \propto p(\mathbf{x}_0, \mathbf{y}). \quad (2)$$

Hence representing $p(\mathbf{x}, \mathbf{y})$, will allow for an output to be generated to the current input $\mathbf{x}_0$, by selecting an output at random from the distribution $p(\mathbf{x}_0, \mathbf{y})$. Note that this procedure makes it possible for the system to have several equally plausible responses to one stimulus.

One of the major properties in favor of the normal model is that it can be shown that the projection of a normal distribution onto the hyperplane $\mathbf{x} = \mathbf{x}_0$, specified by the current input, is a new normal distribution, $p(\mathbf{y})$, times a constant, c:

$$p(\mathbf{y} \mid \mathbf{x}_0) \propto p(\mathbf{x}_0, \mathbf{y}) = c(\mathbf{x}_0, \mathbf{m}, \mathbf{B})\ p(\mathbf{y}). \quad (3)$$

Here $\mathbf{m}$ and $\mathbf{B} = \mathbf{C}^{-1}$ are the mean vector and the inverse of the covariance matrix respectively. This means that it is also possible to use a gaussian model, describing the system's behavior, in order to generate proper responses. But each of the nodes in the tree contains a normal distribution estimating the behavior distribution in *its part* of the decision space. The finest approximation of the *global* behavior distribution, $p(\mathbf{d})$, is found by a top-down expansion of the *sum* of local distributions from the root down to the leaves:

$$p(\mathbf{x}, \mathbf{y}) \approx p_0(\mathbf{x}, \mathbf{y}) \approx \alpha_l\ p_l(\mathbf{x}, \mathbf{y}) + \alpha_r\ p_r(\mathbf{x}, \mathbf{y}) \approx \ldots \approx \sum \alpha_i\ p_i(\mathbf{x}, \mathbf{y}). \quad (4)$$

Here the indices l and r refer to the distributions in the left and right child of the root node. These two distributions are themselves approximations

of the distributions in their children, and so on. The recursion stops with the sum of the leaf distributions as seen in equation 4. This was illustrated in equation 1 and figure 2 for a tree of depth three.

The projection property of the normal distribution also allows the global conditioned distribution $p(\mathbf{y} \mid \mathbf{x}_0)$ to be calculated as a linear combination of the conditioned leaf distributions, yielding the distribution of possible responses as a new sum of normal distributions:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}_0) &\propto p_0(\mathbf{x}_0, \mathbf{y}) \\ p(\mathbf{y} \mid \mathbf{x}_0) &\approx \alpha_l \; p_l(\mathbf{x}_0, \mathbf{y}) + \alpha_r \; p_r(\mathbf{x}_0, \mathbf{y}) \\ p(\mathbf{y} \mid \mathbf{x}_0) &\approx \alpha_l \; c_l \; p_l(\mathbf{y}) + \alpha_r \; c_r \; p_r(\mathbf{y}) \;\approx\; \ldots \;\approx \sum \alpha_i \; c_i \; p_i(\mathbf{y}). \end{aligned} \quad (5)$$

Generating a random output from the conditioned distribution in equation 5 is quite easy. When a probability density function can be expressed as a linear combination

$$p(x) = \beta_1 p_1(x) + \beta_2 p_2(x) + \ldots + \beta_n p_n(x) \;, \quad \sum \beta_i = 1 \;, \quad \beta_i > 0 \;, \quad (6)$$

where $p_1(x), \ldots, p_n(x)$ are probability density functions, the following two step procedure can be applied to produce a sample from the total distribution $p(x)$ (Mitrani, 1982):

1. Generate a random integer, m, being 1 with probability β_1, 2 with probability β_2 and so on.
2. Generate a random variable from the probability density function $p_m(x)$ and let it be the output.

The method for random number generation described above lends itself to a branch and bound implementation if applied recursively from the top node and downwards through the tree. This results in a stochastic traversal of the tree in search for the leaf model to be used for output generation. In each node one of the children will be selected in the first step of the procedure, but since it too will be a sum of two distributions the process will continue, recursively, until a leaf is reached and the output is generated.

If a performance measure is assigned to each node model, the response generating procedure could be terminated at any level in the tree. When traversing the tree a comparison is made between the performance of the current node model and that of its children suggested for the next recursion level. In case there is a big enough difference in their performance in favor of the father, the traversal is terminated and the father is allowed to generate the response. A smoother way of implementing this idea is to make the decision stochastic and let the father generate the response with a probability that depends on the difference in performance.

A natural way of measuring the performance is to compute the mean and the standard deviation of the reward collected by a model. Modeling this

measure as a normal distribution leads to a test on differences between means. Assuming n_f and n_c samples from two different normal distributions, yielding mean values μ_f and μ_c, along with standard deviations σ_f and σ_c, the test variable t has a Student t distribution with $n_f + n_c$ degrees of freedom:

$$t = \frac{\mu_f - \mu_c}{\sqrt{\frac{\sigma_f^2}{n_f} + \frac{\sigma_c^2}{n_c}}}. \tag{7}$$

Now if the distributions refer to the reward of the father, f, and a child, c, under consideration for response generation, it is possible to find a probability, given a threshold μ_t, for halting the recursion at the father node as:

$$P_h = P[\mu_f - \mu_c > \mu_t] = P[t > h] = 1 - P[t < h]. \tag{8}$$

This means that the choice at each node is between halting, with probability P_h, and continuing the recursion, with probability $1 - P_h$.

4 Growing and Pruning

Experienced decisions, i.e. input-output combinations, provide the raw material from which the tree representation of the decision distribution is grown in this approach. A decision is viewed as a sample from the behavior distribution and its received reward as the probability mass in that point of the decision space. Efforts are made to consider the experiences to be the single most important source of information available to the system in its struggle for reward, in line with the behavioristic view.

Estimation of the behavior distribution can be performed in two distinct ways. Either by collecting the samples and then run an off-line estimation procedure, or using an on-line strategy where each new sample is allowed to update the current model of the distribution. An off-line like procedure for this purpose has been investigated earlier (Landelius, 1993), and what follows is an outline of an on-line estimation procedure. Two arguments in favor of an on-line approach are its superior adaptability and its lower storage requirements.

When the system has generated a response to a presented stimulus it is rewarded. This reward, or probability mass, is used to modify both the performance measure of the responsible model as well as the coefficients, α, in the sum of distributions describing the systems behavior found in equation 4. Note that the performance measure is the mean reward received by the model, whereas the coefficient is the total probability mass that the model represents, which is the reward accumulated by the node.

When a new performance measure is available it is possible to compare the node with its parent. The entity to compare is, as stated before, the mean reward collected by the model. This test may come out in three different ways:

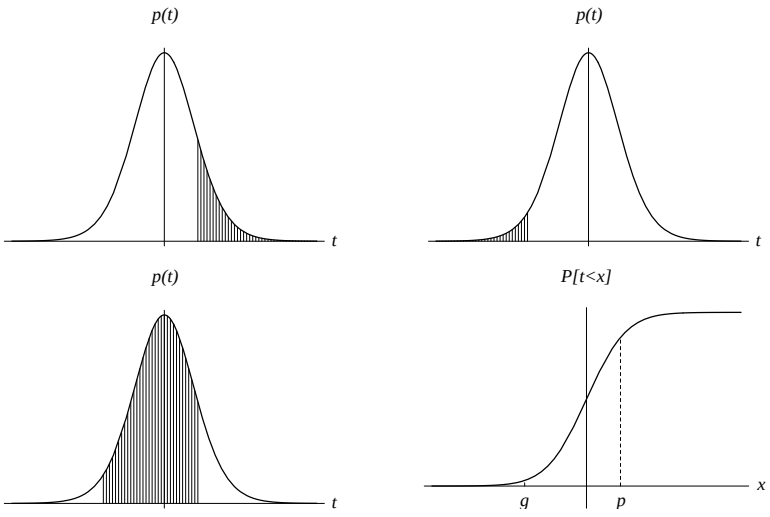

Figure 3: Decision bounds for growing, pruning, and leaving unchanged.

1.	Superior parent	,	$\mu_f - \mu_c$	$>$	μ_t	: Prune.
2.	Superior child	,	$\mu_c - \mu_f$	$>$	μ_t	: Grow.
3.	Small difference	,	$\lvert\mu_f - \mu_c\rvert$	$\leq$	μ_t	: No action.

A superior parent suggests that the finer representation in terms of two child submodels may be of limited use. This means that the probability for pruning away the subtrees beneath the parent should be high. When the child is a superior leaf it is the other way round, with a high probability for splitting the node and growing two new children in order to improve even more on the performance. Superior non-leaf nodes only justify their existence and cause no further actions, as is the case when the difference in node performance is to low to be under consideration. These situations leads to a high probability for status quo.

The choice between these three actions should not be based on hard decision boundaries, but rather depend on a random variable which points out each case in proportion to its validity. This is done in the very same way as was the decision whether or not to halt the traversal of the tree in the response generation situation. Again the test variable from equation 7 is used. Since the three probabilities for pruning, growing and doing nothing should add to one it is only necessary to compute two of them explicitly:

$$\begin{aligned} P_p &= P[\mu_f - \mu_c > \mu_t] = P[t > p] = 1 - P[t < p] \\ P_g &= P[\mu_c - \mu_f > \mu_t] = P[t < g] \\ P_n &= 1 - (P_p + P_g) = P[t < p] - P[t < g]. \end{aligned} \tag{9}$$

In figure 3 these three probabilities are illustrated. The probabilities can be found using the cumulative distribution function $P[t < x]$, shown in the lower right of the same figure.

What parts of the tree are affected in these different update situations? The three cases are treated in reverse order. Suppose no modification of the tree structure is suggested, which models should then be subjects to an update? If the response was generated from a non-leaf node, the idea is to update the nodes in straight descending order from the root down to the node under consideration. Also the nodes belonging to the child subtrees of the node are updated. This means that the coarser nodes which took part in the generation of a response are adapted to the new knowledge. In this way their partitioning of the decision space will be kept up to date. Child subtrees are modified to make use of the finer partitions they already represent in the tree. In case the response came from a leaf-node, only the former part of the update procedure, involving the coarser models come into question.

If the tree is suggested for pruning, this means that the node responsible for the produced response is probably still less effective than its parent in producing good responses. This also implies that the subdivision of the parent model is of little use and hence both the child subtrees can be pruned away. When this is done the nodes in straight descending order from the root to the parent are updated as in the previous case.

In case a leaf node is significantly better than its parent it should try to improve even more on its performance by modeling its part of the decision space with two distributions instead of one. For a non-leaf node two finer models already exist, and the update procedure follow that outlined for the no-modification case. If two new leaf nodes are grown they need a mean decision vector and a covariance matrix each. The mean vectors are placed on a line through the parent's mean vector along the direction of maximal data variation in the parent model, i.e. on the line directed along the eigenvector corresponding to the largest eigenvalue of the parent's covariance matrix. All but the largest eigenvalue of the parent's covariance matrices are inherited by the new models. In the direction of maximal data variation the standard deviation of a child distribution is suggested to be half that of its parent. Denoting the parent's parameters without any indices, the two new sets of child parameters, satisfying their parent's model, then become:

$$\begin{aligned} \mathbf{m}_l &= \mathbf{m} - \frac{\sqrt{3}}{2}\sigma\ \mathbf{e}_1 \\ \mathbf{m}_r &= \mathbf{m} + \frac{\sqrt{3}}{2}\sigma\ \mathbf{e}_1 \\ \mathbf{C}_l = \mathbf{C}_r &= \mathbf{C} - \frac{3}{4}\lambda_1\ \mathbf{e}_1\mathbf{e}_1^T. \end{aligned} \tag{10}$$

Again the ancestors in straight descending order are updated as in all the

previous cases. The performance measures of the children are initialized to be identical to that of their parent, in order to make it possible for them to participate in future response generation. Initializing them to zero would make it very unlikely to traverse down to them in the response generation recursion, since the difference in mean performance would be as large as is the performance measure of their parent.

Producing sound criteria for when to halt the tree growing process is known to be a problem. Even though a node may seem not to benefit from being split once, it may benefit from further splits as illustrated in figure 2. The suggested solution to this problem is to let splits occur in a probabilistic manner, where the probability for a split depends on the estimated usefulness of the split. Even if another split does not seem to be very useful, the system will eventually try it out and thus discover if two or more splits are beneficial. This results in a system that will never be content, but will indefinitely continue to search for better solutions.

5 Discussion

The two main routes towards intelligent behavior are based on simulation of biological systems which possess this desired quality. One course is that of *artificial intelligence* which make use of introspection to find out how humans process information in an intelligent way. This perceived, conscious process is then imitated in a computer simulation. But in the same way as it is possible to build machines that fly but do not flutter their wings, it is possible to design machines that show intelligent behavior but lack the perceived conscious process found by human introspection.

Another course in a somewhat opposite direction is the approach called *connectionism*. Driven to extremes, this approach states that the lack of intelligent behavior in today's learning machines is due to the difference in the underlying computational structure between present machines and biological systems. Intelligent behavior will emerge if the underlying computational structure is the right one. However, the widely accepted Church-Turing thesis denies the importance of any specific underlying computational structure (Minsky, 1967).

The *functionality* of biological systems may be copied but need not be implemented literally. Instead, the route seems to go through the implementation of good algorithms, rather than through literal simulation. Learning systems must be able to build up their knowledge from experiences and generalize this information to novel situations. Choosing one way of generalization before another is to introduce bias. For a general purpose system it is impossible to determine how to bias its behavior towards important properties of a problem, and at the same time avoid minor details. However, geometric domains unlike the symbolic ones studied in artificial intelligence, incorporate a natural bias, that of *continuity*.

The neural network structures, suggested by the connectionists, work in a

geometric domain and make use of this bias to perform a simple form of generalization, that of interpolation and extrapolation. Two major drawbacks with main stream neural network structures are the lack of design techniques, and that they maintain a global model of the problem domain. A solution to the second problem is to introduce another bias, that of *locality*.

An alternative to the methods described above, both of which are of introspective nature, either with respect to human psychology or physiology, is the behavioristic approach. With this view as little as possible is assumed about the inside of the system and the interest is instead focused on the relations between the system's stimuli and responses. Analysis of stimuli which do not result in good responses is considered meaningless. The system outlined in this paper is based on such a behavioristic view and makes use only of the two priors of continuity and locality. Together with its dynamic data structure and the reinforcement learning paradigm, this approach could prove to be an interesting alternative to present efforts towards learning machines.

References

Bellman, R. E. (1957). *Dynamic Programming.* Princeton University Press, Princeton, NJ.

Granlund, G. H. and Knutsson, H. (1983). Contrast of structured and homogenous representations. In Braddick, O. J. and Sleigh, A. C., editors, *Physical and Biological Processing of Images*, pages 282–303. Springer Verlag, Berlin.

Knutsson, H. (1989). Representing local structure using tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, Oulu, Finland. Report LiTH–ISY–I–1019, Computer Vision Laboratory, Linköping University, Sweden, 1989.

Landelius, T. (1993). Behavior representation by growing a learning tree. Thesis No. 397, ISBN 91–7871–166–5.

Minsky, M. L. (1967). *Computation: Finite and Infinite Machines*, page 108. Prentice Hall Inc.

Mitrani, I. (1982). *Simulation techniques for discrete event systems.* Cambridge University Press.

Omohundro, S. M. (1992). Best-first model merging for dynamic learning and recognition. Technical report, International Computer Science Institute, Berkeley, California, USA.

Utgoff, P. E. (1989). Perceptron trees: A case study in hybrid concept representations. *Connection Science*, 1:377–391.

Wilson, R. and Knutsson, H. (1993). Seeing things. Report LiTH-ISY-R-1468, Computer Vision Laboratory, S–581 83 Linköping, Sweden.