Learning from Examples using Reflective Exploration

Uwe Beyer Frank Śmieja

October 11, 1993

Abstract

An important property of models constructed through the process of learning from examples is the manipulation and control of the data itself. When the data is actively selected or generated the process is known as *exploration*. Reflection about the internal model allows exploration to be more than just a random choice in the input space. In this paper we identify two basic forms of reflective exploration: density-based and error-based. We demonstrate the applicability of exploration processes and the advantages of using them in open systems using the task of learning a 2-dimensional continuous function.

Keywords: Open systems, reflection, density-based exploration, error-based exploration

1 Introduction

In Nature exploration is a universal process. Each species comprising the animal kingdom demonstrates at least some aspects of exploration in daily survival. A basic characteristic of animal life is locomotion, and this ability is not just used in achieving a translation from `A` to `B`, but also in improving the pay-off received on getting from `A` to `B`. Even plants show aspects of exploration, albeit through directed growth rather than locomotion.

Exploration is a natural feature of adaptive systems living in open worlds. It can be seen as a basic drive [4, 9] on a par with those of survival, food, and sex. For example, consider a cat after the owners have changed residence. Its immediate instinct is to wander around, most notably identifying all the escape routes from the various rooms. This instinct supports the survival drive. A dog is guided in its exploration by smells left by other dogs. This is a social behavior, related to the pack instinct, but also serves as information gathering for survival. In animal foraging the explore instinct functions to identify the choicest food sources. Such behavior guides the primary activities of insect colonies: ant foraging begins with multiple random exploration into the vast environment. Such forms of exploration all have a common ingredient: information is actively sought out and used to shape internal models of (a part of) the world.

Exploration has already been applied to problems involving adaptation, in robot control [12, 14], in systems with vehicles coming to terms with a changing environment [1], in automata employing behaviors [5, 7, 9], and in the learning of functions [16]. It has also been handled in [13] in the application of intelligent control. Our aim in this paper is to define exploration in a general way in the framework of learning from examples, and outline its possible strengths and limitations.

The idea of open and closed systems is explained in section 2. In section 3 we introduce the concept of exploration in the framework of agent models. The test domain chosen to illustrate exploration is defined in section 4, as is the non-reflective random exploration method used for comparisons. The test problem was chosen to be a continuous 2-dimensional function. The requirements for reflective exploration are explained in section 5 and three examples of it are tested in section 6. They are exploration methods that are density-based, error-based, and a combination of the two. They are reflective because they require information about the current predicted performance of the agent on a given (x, y) tuple. The relative performance of the methods in terms of model accuracy is discussed in section 7.

2 Open and closed systems

In the following a system is taken to be a world containing a variable space $\mathcal{X}$ and a function space $\mathcal{F}$ (dimension and definition ranges are not specified here for simplicity). All processes active in the system can be characterized as

$$y = f(x), \quad y, x \in \mathcal{X}, \quad f(\cdot) \in \mathcal{F}. \tag{1}$$

An agent is a *model* of a system (see [2] for a more detailed definition).

A *closed system* is one in which there are a finite number of examples (data-points) that define its behavior. For this to be the case the number of possible examples in the system must be finite. *Open system* is a term applied to those systems characterized by being *effectively* infinite in at least one dimension. A dimension defines the values of any of the variables (including time) that specify the examples, as well as the function space itself, which specifies the behavior of the system. An "effectively" infinite dimension is one that is finite when seen at an appropriately high level, but infinite from the viewpoint of an adaptive agent that has not seen enough examples. A result of this is that all systems are effectively open for agents new to the current problem domain.

In effect, when an agent attempts to model it, an open system differs from a closed system in the incrementally adaptive property of the agent itself. An agent that is not adaptive operates in a closed world. Consequently, once the training of an adaptive agent is over, it then operates deterministically in the closed world defined by the part of the system that it has seen. The rather academic differences between closed and open system tend to encourage the view that all problems are open so long as the agent continues to adapt to new instances of it. Such is the characteristic of robot worlds and real-world problems.

3 Learning and Exploration

The agent adaptively models open systems by *learning from examples* [2, 6]. The examples are instances of equation (1). The x–values are provided either by the environment or by the agent itself. The system reacts according to its current state, applying the appropriate function f to x and generating y. When the x–values are suggested by the agent itself the process is known as *exploration*.

3.1 Types of learning

An adaptive agent can be trained in a number of ways:

1. In the standard supervised learning method, the examples are all provided by the environment itself (the "teacher") and the agent tries to learn them (Figure 1a). The examples are determined before the learning and are not influenced by it. In practice the examples are fixed at the beginning of the learning session.

2. If the function to be learned is a mathematical or probabilistic function, the (x, y) pairs may be generated throughout the course of the adaptation (i.e. *incrementally*). The agent produces random x suggestions and the function is applied by the environment to generate the y value to be learned (Figure 1b).

3. The examples may be selected by the agent from the (fixed) learning set offered by the environment, in a batch manner (Figure 1c). Thus patterns are chosen for the entire learning set using a fixed agent state [15]. In further batch learning phases the learning set may be reselected in the same manner, using updated information about the agent's state.

4. The examples may be selected by the agent from the (fixed) learning set offered by the environment, in an incremental manner (Figure 1d). Thus a pattern is chosen singly and the agent is updated before the next pattern is seen.

5. New examples are incrementally generated in a non-random way by the agent, in the form of an x suggestion [16] (Figure 1e).

Of the five schemes the last four all involve exploration methods. Scheme 1 can be seen as a random (i.e. non-reflective) exploration method, since the example set is normally generated in a random order. Scheme 5 involves explorative generation of examples, while 3 and 4 involve explorative selection of examples. In our experiments three examples of scheme 5 were compared with an example of a random scheme 2.

3.2 Agent models

As indicated in the last section, there are two possible modes for for construction of an agent's learning set $\mathcal{L}$: generation and selection.

Figure 1: Five schemes in which the agent can interact with the environment in the adaptation process. (a) is non-exploratory adaptation, (b)–(e) all involve exploration of some kind.

1. *Generate* $\mathcal{L}$ using information from the world. The agent is free to suggest any x within the definition range of the input space. The corresponding y value is produced by observing the result of the application of the function active in the world (we will call this function "G"). The returned value is either a valid answer or the tag "don't know".

2. *Select* $\mathcal{L}$ as a subset of a finite data set. Their exists a finite set of available (x, y)–tuples in the world. The agent must choose its (x, y) from this set.

The construction of $\mathcal{L}$ is an iterative process, with discrete time step t incremented with each adaptation of the agent. Thus an input pattern $x(t+1)$ is chosen after adaptation step t. The existing $\mathcal{L}(t)$ may be used to determine $x(t+1)$.

An agent is graded by the *accuracy* of its predictions. This may be estimated statistically by comparing a set of its predictions with events in the real world. The accuracy $A_\alpha(t, \mathcal{M})$ of an agent α tested on a set $\mathcal{M}$ of (x, y) tuples is defined:

$$A_\alpha(t, \mathcal{M}) := \frac{1}{|\mathcal{M}|} \sum_i^{|\mathcal{M}|} \delta(f_\alpha(t, x_i), y_i) \tag{2}$$

where $\delta(a, b)$ is a similarity measure in $[0, 1]$ between a and b. In a classification problem the measure will produce 0 when an incorrect classification occurs, and 1 when a correct classification occurs. There are normally no half-measures for such problems. For a non-discrete problem the distance measure represents the closeness of the agent's suggestion with the correct answer, normalized using the output range of the function. The model built by the agent α at t is denoted by $f_\alpha(t, x)$. The learn accuracy $A_\alpha(t, \mathcal{L})$ is the degree to which an agent correctly reproduces the tuples it has already seen. The test accuracy ("generalization") $A_\alpha(t, \mathcal{T})$ is the degree to which the agent correctly models a new set of tuples $\mathcal{T}$.

3.3 The goal of exploration

Exploration has at least two clear goals:

1. Improve the accuracy of an agent as quickly as possible as each new example is chosen.

2. Minimize the number of examples necessary to be seen in order to reach the maximum attainable accuracy. For a problem defined by a finite learning set this is defined as the accuracy reached using the entire learning set. For a problem with infinite learning set it is defined asymptotically (see [2]).

Thus if two agents (α_1, α_2) use the same method to approximate the same function, it is desirable that the agent using exploration (α_1) should have at least as high an accuracy as the one not using exploration:

$$A_{\alpha_1}(t, \mathcal{T}) \geq A_{\alpha_2}(t, \mathcal{T}) \ \forall t > 0 \tag{3}$$

Figure 2: Linear triangle based approximation. The triangle is constructed using three stored points and the new value (arrowed) is approximated to lie on it.

Exploration cannot produce higher accuracy than is possible with the given data set. Consider that the learning set $\mathcal{P}$ is finite, and that the agent using exploration (α_1) eventually gets to see the entire set. Then its accuracy after it has learned the entire set should be on average the same as that of a agent which has learned the whole set without exploration:

$$A_{\alpha_1}(t, \mathcal{T}) \approx A_{\alpha_2}(t, \mathcal{T}) \quad t = |\mathcal{P}|. \tag{4}$$

If the model's accuracy is dependent on the order in which the patterns in $\mathcal{L}$ are presented to the agent, α_1 should still be able to reach the final $A(\mathcal{T})$ quicker than α_2:

$$A_{\alpha_1}(t_1, \mathcal{T}) \approx A_{\alpha_2}(t_2, \mathcal{T}) \approx A(\mathcal{T}) \Rightarrow t_1 < t_2 \tag{5}$$

If $\mathcal{L}$ is infinite, then in real world problems we postulate that

$$\lim_{t \to \infty} \|A_\alpha(t+1, \mathcal{T}) - A_\alpha(t, \mathcal{T})\| \to 0, \tag{6}$$

which means it becomes ever more insignificant which of the examples is learned next, as $t \to \infty$.

3.4 k-NN agent

For the experiments in this paper the agents are based on linear triangle-based approximation (see Figure 2). This method has the advantage of working in a predictable way (as opposed to neural network agents). The examples patterns are all stored in the agent's memory. Given the input x the approximated y lies on a triangle constructed by the 3 nearest neighbors surrounding x. If there are less than 3 points within a distance ε of the x, then the next nearest neighbor is used as approximated value. Thus large planes (relative to ε) cannot be approximated with only 3 points.

4 The test problem

The quality [2] of a model is dependent on a large number of parameters. A model constructed using exploration depends on additional "exploration" parameters. Like any modeling

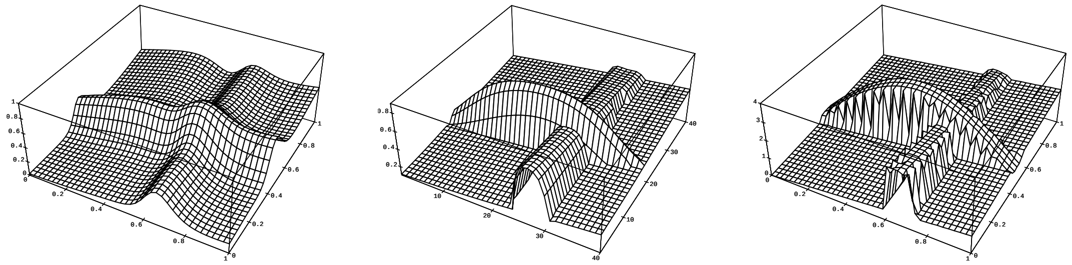
Figure 3: The function to be approximated (a). The plan (b) identifies regions for future reference.

procedure, it is unreasonable to expect a "general method" for exploration. Everything is highly dependent on the task to be performed, and the assumptions made in the modeling procedure [6, 3]. We therefore focus on a particular example to test exploration, and to illustrate the basic ideas.

The test function is plotted in Figure 3a. Figure 3b shows a plan view of the input space, identifying the different regions of the function by the labels (`a`–`f`) for future reference.

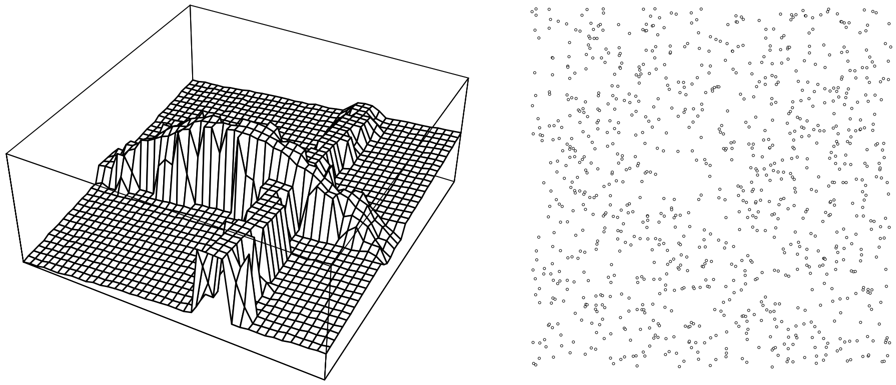
To give an idea of the effect of the approximation method chosen, we show in Figure 4 the function approximated by 3 different types of agents.

An example of a non-reflective exploration method is one using random choice. It can work



(a) **(b)** **(c)**

Figure 4: Approximations using (a) back-propagation neural network, (b) Pandemonium multiple neural networks [10, 11], (c) k–nearest neighbor.



(a) (b)

Figure 5: Approximation using non-reflective selection of examples. (b) shows the examples chosen.

in the following ways:

- If the learning set is finite, each pattern in the learning set has equal probability of being selected.
- If the learning set is infinite, every point in the input space has equal probability of being generated.

We use the second option for comparison with our reflective exploration.

Figure 5 shows the approximation (Figure 5a) of the test function and the chosen (Figure 5b) points produced using non-reflective selection, in which each x value is chosen with uniform probability from the input space.

5 Reflective exploration

In reflective exploration a training example is chosen on the basis of the current progress of the agent's adaptation.

To make exploration effective, the following criteria are applied:

(a) Accuracy should be a positively increasing function of t.

(b) The selection of the next example to be learned should be correlated with rate of increase of accuracy for all $0 < t < N$, where N is the maximum number of examples which can be learned in a given time interval.

In problems with real time demands (e.g. robotics), case (b) is more relevant, because the size of the learning set tends to be highly limited.

For the agent to choose an appropriate x its reflective function I must satisfy:

$$I: \;\; x(t+1) \rightarrow A_\alpha(t+1, \mathcal{T}) > A_\alpha(t, \mathcal{T}). \tag{7}$$

Such a function predicts the improvement to be gained by learning $(x, y = G(x))$. I is highly dependent on the model that α has of the world. The agent knows about its current modeling successes, and where it needs more information. Unfortunately there is no obvious general procedure for constructing I. At this point mathematics cannot help us further and the "engineering" must begin. This takes the form of making assumptions about the problem domain and the behavior of the function to be learned. For our experiments with exploration we make the following assumptions:

- There exists a metric $d(p_1, p_2) \rightarrow \Re$ measuring the distance between two points in the input space.
- There exists an error induction function E_{in}:

  $$E_{\mathrm{in}}(x, x') \rightarrow \Re, \tag{8}$$

  which measures the error that would be produced in approximating $G(x)$ if the point x' were part of the learning set.
- We assume that points lying close to the approximation point x result in better approximation than points lying far away. Thus on average and for some $\epsilon > 0$ the following applies:

  $$d(x, x') < d(x, x'') \Rightarrow P(E_{\mathrm{in}}(x, x') > \epsilon) < P(E_{\mathrm{in}}(x, x'') > \epsilon). \tag{9}$$

- If

  $$E_{\mathrm{app}}(x) \rightarrow \Re \tag{10}$$

  is the approximation error at point x, then for two points x_1 and x_2 the following equation is on average true:

  $$E_{\mathrm{app}}(x_1) > E_{\mathrm{app}}(x_2) \Rightarrow P(E_{\mathrm{app}}(x_1') > \delta) > P(E_{\mathrm{app}}(x_2') > \delta) \tag{11}$$
  $$\forall\, x_1' : d(x_1, x_1') < \epsilon, \forall\, x_2' : d(x_2, x_2') < \epsilon,$$

  for some δ. That is, in the neighborhood of a badly approximated point there is a higher possibility for errors than in the neighborhood of a well approximated point. "Geometrical reasoning" [8] is also based on such assumptions about the problem space.

6 Density–based vs. error–based exploration

Two basic methods of exploration are identified. One works in the x–space (density-based exploration) and the other in the y–space (error-based exploration). The combination of the two is also investigated.

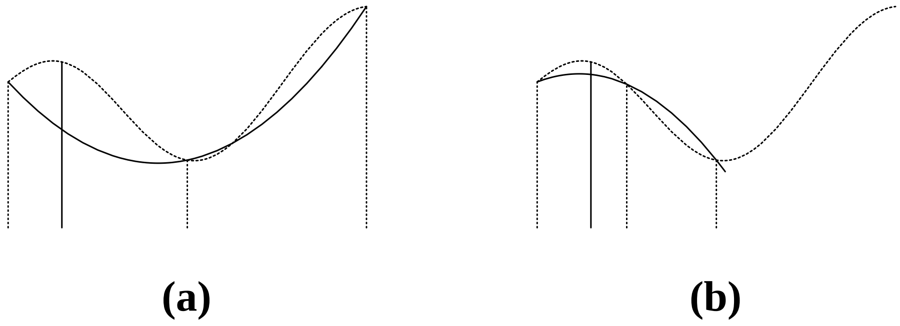


Figure 6: Spline approximation for a function G of high curvature, (a) with low density training points, and (b) higher density.

6.1 Density-based exploration

From equation (9) one deduces that if a point x has no stored examples in its vicinity, then the possibility of making a bad approximation here will be high.

This is clearly true for functions of high curvature, which require more fixed points at the areas where the function is rapidly changing. Figure 6 is an example of two spline approximations of the same function using fixed points of different densities.

In Figure 6a the fixed points for approximation (dashed lines) lie far away from the approximation point (solid line). The approximated curve (solid line) has a large deviation from that using the correct function G (dashed line). In Figure 6b the fixed points are closer to the approximation point and the result is better. This is also the case even for "mostly" flat functions, because it is always possible that a single sharp peak may occur somewhere in the x–space.

In the Figure 7a the fixed points are of low density, so the peak cannot be recognized by the approximation method. However, if a fixed point falls somewhere within the peak then the approximation is improved significantly. Using equation (9) as a guideline for approximation, the agent tries to place the next x–value at the place with lowest density of neighbors in x–space, resulting in the peak being recognized (Figure 7b).

Density-based exploration gives an equally distributed grid-like structure of exploration points (Figure 8b). The density of points is dependent on how many x–values may be chosen. In general this method does not rule out overlooking all peaks (a peak may be extremely small), but the *probability* of overlooking a peak decreases with increase of exploration point density.

Random generators are not perfect, and this becomes clear when one considers the distribution of points in Figure 5b. For this reason, compared with random choice, density exploration points are more uniformly distributed. This results in a better approximation in regions `a`, `b`, `c` and `d` of the function (see Figure 5b).

The density information has been used to estimate another reflective quantity, the *confidence*

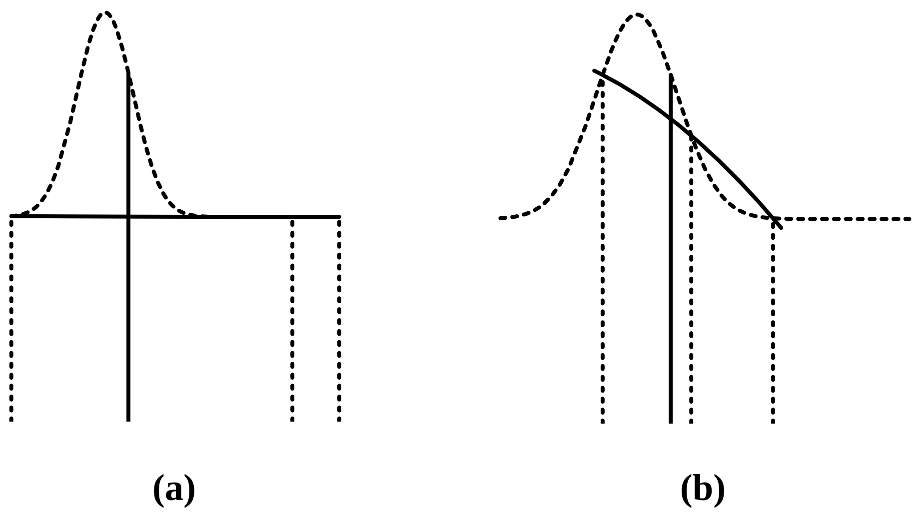


Figure 7: Spline approximation for a function G containing a peak, (a) for low density training points, and (b) for higher density.

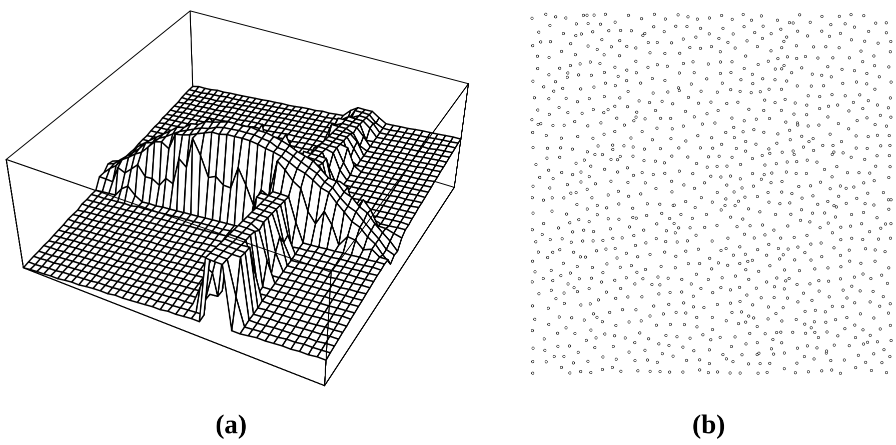


Figure 8: Density-based exploration. (a) shows the approximation and (b) the exploration points chosen.

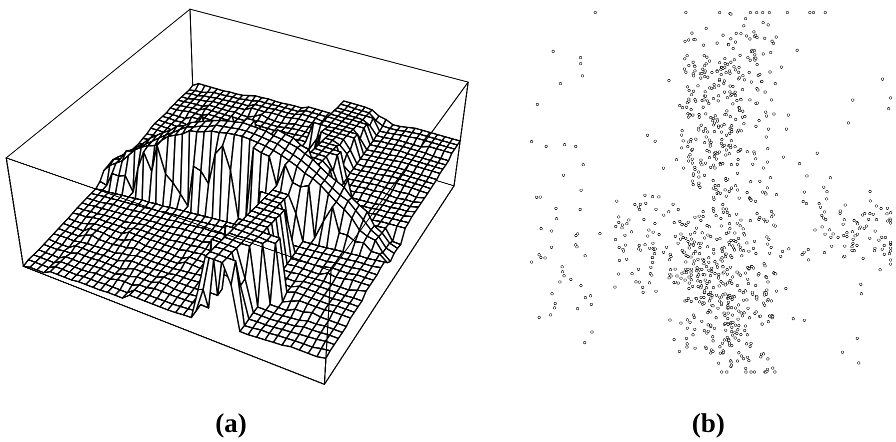


Figure 9: Error-based exploration. (a) shows the approximation and (b) the exploration points chosen

of a particular approximation (see [2, 11, 10]). The confidence of a prediction for point x would then be a function of the density of learned points about x.

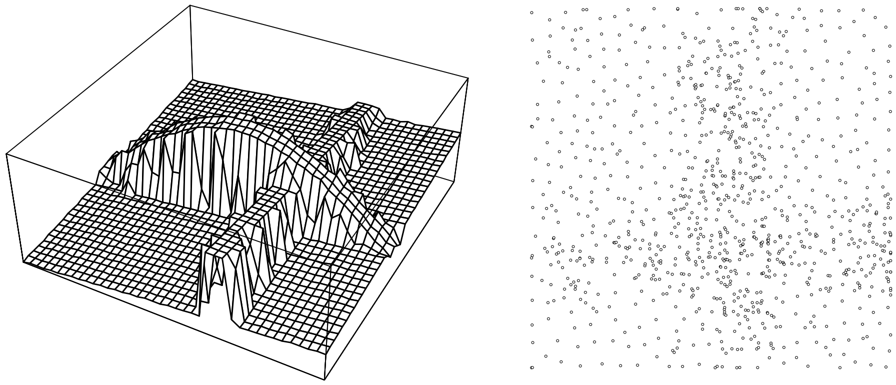
6.2 Error-based exploration

In order to use equation (11) for exploration, the agent must calculate $E_{\text{app}}(x)$ before x is learned (after learning x, $E_{\text{app}}(x)$ should be zero). The next x to be explored is chosen in the neighborhood of known x–values where $E_{\text{app}}(x)$ is highest. The size of the neighborhood and the method for choosing a candidate out of this range are parameters of the exploration algorithm.

The exploration points are distributed around regions `e` and `f` of the function (see Figure 9). They are better explored than in the density-based method, but regions `a–d` are less well approximated, because the agent first explores areas having higher $E_{\text{app}}(x)$. This kind of error-based exploration seeks out values in x–space where the model is currently difficult to learn.

6.3 Combination of error– and density–based methods

Density-based and error-based exploration methods can be applied complementarily. Thus, first the function is explored using density-based exploration until a certain density is reached. Then error-based exploration is used to improve approximation at those areas where more fixed points are needed by the agent to improve the approximation accuracy. It is however not a trivial matter to decide when to change the exploration model. One simple method would be to apply them alternately: first density, then error, then density, etc. With the latter method, we get the result shown in Figure 10.



(a) **(b)**

Figure 10: Combined density-based and error-based exploration. (a) shows the approximation and (b) the exploration points chosen

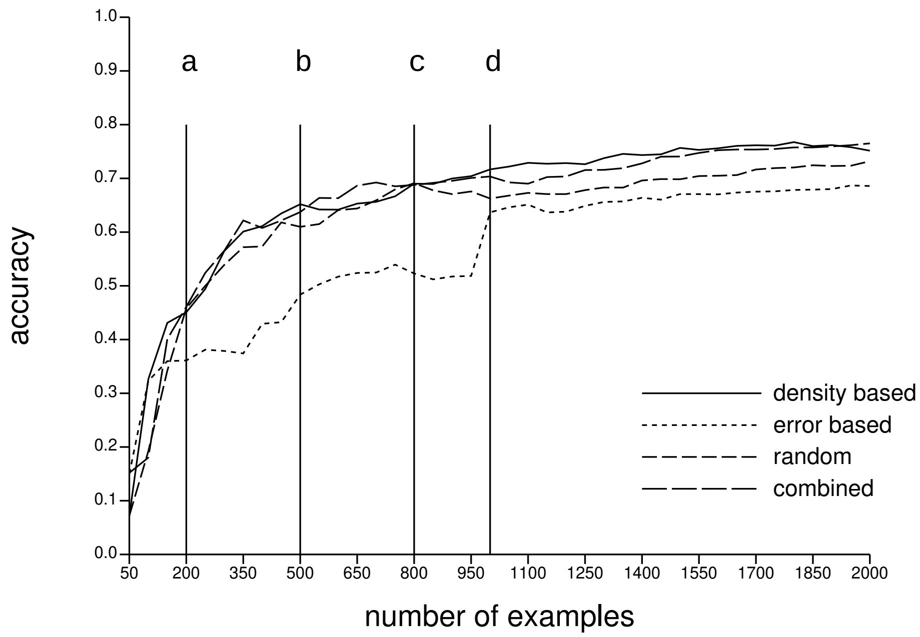


Figure 11: Accuracy on the test set for the four exploration methods as a function of number of examples chosen. Areas are marked with the lines *a*–*d* for reference. All runs were performed with agents using the triangular approximation method described in the text.

The exploration points cover the whole x–space, but they are more dense around regions `e` and `f` of the function.

7 Effect of exploration on model accuracy

Figure 11 shows the accuracy as a function of number of examples learned, for the four exploration methods: random, density-based, error-based and combination.

The following can be deduced from the curves:

- In the interval $(0, a)$ the agents have not yet gathered enough knowledge to show marked differences. Equations (9) and (11) do apply at this stage, but only after a certain amount of data has been seen.
- In the interval (a, ∞), error-based exploration gives rise to stepwise improvements. It explores locally around the points with the highest error. A step occurs when a new feature of the function is isolated for exploration. The improvement after a new feature is found gradually decreases until another feature becomes more profitable to explore.
- In the interval (b, c) the combined method is better than density-based. In this interval the additional use of error-based exploration makes sense, because there are enough data available to find interesting regions to be explored.
- In the interval (d, ∞) the problem is transforming into a "closed world" problem. Exploration becomes increasingly less fruitful. However there is still a difference to be seen between the methods. The asymptotic case where equation (4) applies requires a large number of steps ($N \gg 5000$).

The inferior performance of the error-based method can be explained by the fact that it attempts to reduce the maximum error, rather than the average error of the learning set. Thus since the points lying on the discontinuous regions are relatively infrequent, their contribution to overall inaccuracy for the other learning methods is not so large. This leads to the poor overall accuracy for the error-based method, although its maximum error for a pattern is the lowest. It indeed makes sense to combine the density– and error-based methods, since the relative frequency of discontinuous areas will not be known at the beginning of the adaptation.

8 Conclusion

In this paper we presented the reflective exploration learning method, in which the agent itself actively seeks out examples to learn. We have identified different classes of refelctive exploration: density-based and error-based. The former works on the information in the input space and the latter on the information in the output space. A combination of the two is also possible and it was shown to be more reliable for unknown domains.

The effect of exploration depends very strongly on the model used by the agent. As pointed out in [2] and implied in [6], choice of a "correct" model is highly problem dependent. The correct exploration method is dependent on the approximation model and furthermore introduces its own model assumptions and parameters to be tuned. Our simulation results demonstrate that it is difficult to choose the best exploration method. As the adaptation of the agent progresses different exploration methods may be necessary. Sometimes random exploration may be better, in that it has similar performance but at lower cost.

There is a high dependence on the parameters chosen by some external source (e.g. the experimenter). To find optimal parameters, the function to be explored must either be well known by the experimenter, which is a contradiction to the idea of exploring an "unknown" function, or they too could be optimized by another agent.

In our test domain we observed three phases with smooth boundaries. At the beginning of the adaptation (a fully open system) there is not enough knowledge to make exploration decisions, so random selection is as good as reflective exploration. In the next phase the accuracy can be significantly improved by making good exploration decisions. In the third phase, so much is known about the system, that it becomes closed, and new examples don't have much influence on accuracy. In this phase exploration becomes less effective. The boundaries between such phases are not obvious.

We summarize conditions when exploration would *not* be advisable:

1. When the agent has learned enough about the system so that it is no longer an effectively open system.
2. When the overall learning set is going to be completely learned anyway.

In these cases the extra cost of performing exploration is not justified.

In conclusion, exploration may not make sense for every problem: it is as much dependent on the problem as the agent's approximation model itself. However, if appropriately applied it can result in significant improvement of both adaptation speed and agent generalization accuracy.

Acknowledgements

The authors thank Heinz Mühlenbein, Gernot Richter, Po Hsiang Chu, Dirk Schlierkamp-Voosen and Byoung-Tak Zhang for reading and correcting the draft, and for useful comments. A special thanks to Dirk Schlierkamp-Voosen for his help in generating some of the figures. This work is part of the GMD Adaptive Systems REFLEX project, supported by BMFT grant number 01IN111A/4.

References

[1] P. Bessiére, J-M Ahuactzin, E-G Albi, and E. Mazer. The Ariadne's clew algorithm: Global planning with local methods. In *IEEE-IROS'93 Conference (Intelligent Robots and Systems)*, Yokohama, Japan, 1993.

[2] U. Beyer and F. J. Śmieja. Learning from examples, agent teams and the concept of reflection. Technical Report 766, Gesellschaft für Mathematik und Datenverarbeitung, St Augustin, Germany, July 1993. submitted to "International Journal of Pattern Recognition and Artificial Intelligence".

[3] U. Beyer and F. J. Śmieja. Quantitative aspects of data-driven information processing. Technical Report 732, Gesellschaft für Mathematik und Datenverarbeitung, St Augustin, Germany, March 1993.

[4] N. Bischof. *The Oedipus Riddle*. Serie Piper 989, Munich, 1989.

[5] L. M. Gabora and P. W. Colgan. A model of the mechanisms underlying exploratory behaviour. In Jean-Arcady Meyer and Stewart W. Wilson, editors, *From animals to animats*, pages 475–484. First International Conference on Simulation of Adaptive Behavior, 1990.

[6] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[7] M. C. Mozer and J. Bachrach. Discovering the structure of a reactive environment by exploration. Technical Report CU-CS-451-89, Dept. of Computer Science, University of Colorado, Boulder, November 1989.

[8] Stephen Omohundro. Geometric learning algorithms. Technical Report TR-89-041, Berkeley, 1989.

[9] Bernd Schlüter. A cybernetic control model from ethology for adaptive coordination of robot behaviors. In *Proceedings of the SPIE Adaptive and Learning Systems Conference, 20–24 April 1992*, Orlando, Fl, 1992. SPIE, ??

[10] F. J. Śmieja. Multiple network systems (MINOS) modules: Task division and module discrimination. In *Proceedings of the 8th AISB conference on Artificial Intelligence, Leeds, 16–19 April, 1991*, 1991. Also available as GMD technical report 638.

[11] F. J. Śmieja and H. Mühlenbein. Reflective modular neural network systems. Technical Report 633, GMD, Sankt Augustin, Germany, February 1992.

[12] S. Thrun and K. Möller. Active exploration in dynamic environments. In *Proceedings of NIPS-4*, Denver, Colorado, 1992. to appear.

[13] S. B. Thrun. The role of exploration in learning control with neural networks. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky, March 1992.

[14] F. Weber and A. Linden. Neural networks for reflective exploration. In *Proceedings of the 2nd International Conference on Automation, Robotics and Computer Vision, Singapore*, pages INV–8.1.1–INV–8.1.4, 1992.

[15] Byoung-Tak Zhang. Learnign by incremental selection of critical examples. Technical report, St Augustin, Germany, March 1993.

[16] Byoung-Tak Zhang and Gerd Veenker. Neural networks that teach themselves through genetic discovery of novel examples. In *International Joint Conference on Neural Networks (IJCNN-91), Vol 1*, pages 690–695. IEEE and INNS, 1991.