# Rosetta: Toward A Model of Learning Problems

**Stephen J. Smith**
Thinking Machines Corporation
245 First Street, Cambridge, MA 02142–1214
smith@think.com

**Stewart W. Wilson**
The Rowland Institute for Science
100 Cambridge Parkway, Cambridge, MA 02142
wilson@think.com

## Abstract

This paper proposes a canonical model of a machine learning problem. The language and structure of the model characterize learning problems independently from the learning system that solves them. To facilitate this decoupling of the problem and the learner the concept of information pathways or "channels" is introduced. Explicitly specifying the bandwidth and content of these channels can more completely define the learning problem. In addition, a small vocabulary of problem descriptors is introduced. It is hoped that these descriptors and the learning problem model will allow for a more precise definition of learning problems in the future and will provide a common language of comparison among researchers.

## Motivation

Because of the current diversity of approaches to machine learning it is often difficult to compare or to even accurately reconstruct an experiment. One reason for this is that for most learning systems it is not possible to determine where the problem itself ends and the learner begins. This makes it difficult to evaluate the performance of the learner and it also makes it difficult for other researchers to reproduce the problem and hence verify the results. There is a need for a model of a learning problem that decouples it from the implementation of the learner and that can be used to produce a common language for the definition of learning problems.

Once such a model is defined it is also important to be able to talk about it in as high a level language as is possible, a language that still allows for the description of significant differences between problems. Currently such descriptions are ad hoc and not well defined. One reason for this is that the descriptors are often dependent on the learning system and are not built explicitly for the description of the problem.

## Components Of The Rosetta Model

The Rosetta model is an abstract description of a general learning problem. As the name "Rosetta" implies it is hoped that such a model can be used as a translation point among a variety of problems. To illustrate the application of the model, four different, previously described, learning problems are used as examples. They are Wilson's (1985) Animat, the letter sequence prediction task of Robertson and Riolo (1988), the boolean multiplexer task from Wilson (1987), and the Pole Balancing task described by Sutton, Barto, and Anderson (1984).

The problems can be stated briefly as follows:

*Animat*: A simple organism must find its way through a predefined environment of trees and pieces of food. Reward is given to the organism when trees are avoided and food is found. The organism's sensory channel is limited to its immediate neighborhood.

*Letter Sequence Prediction*: A predefined sequence of letters is presented one letter at a time. Reward is based on the ability to predict the next letter.

*Boole*: Eleven–bit binary strings are generated where three bits encode a lookup position in the remaining eight bits. Performance is based on the percentage of correct lookups that the system can perform on randomly generated strings.

*Pole Balancer*: A vertical pole is attached by a pivot to a movable cart. The physical state of the pole is available, forces can be applied to the cart, and the system is evaluated by the amount of time until the pole falls over.

These brief statements informally separate out the learning problems from the systems in which they were embedded. However, we would like to go further. The first step in the Rosetta model is to think of the relation between learner and learning problem in terms of the diagram of figure 1. There, the learner is seen interacting with a *problem environment* through *stimulus*, *action*, and *payoff channels*. These, plus the *learning objective*, form the components that the model uses to define a learning problem. Figure 2 schematically shows how the components could be filled in for the four example learning problems.

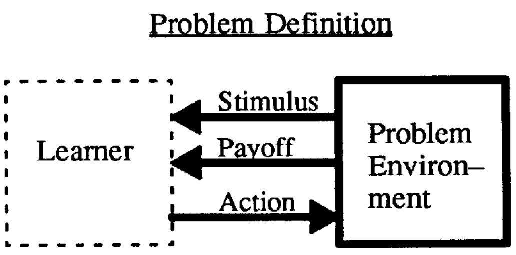

figure 1.

The "problem environment" is like a transition table that defines the problem's outputs (stimuli, payoffs) in terms of its inputs (learner actions) and the current environment state. Less formally, the environment contains all of the information about "running" the problem, including state information (e.g., location in the Animat's Woods), determination of the outgoing stimulus (e.g., a description of the Animat's local surroundings), and processing of action inputs that either affect the state or require payoff (e.g., when food is found).

Because the problem environment is essentially omniscient, it is important to separately define just what it "tells" the learner, and how the learner can act on it. A complete definition of communication channels is thus an important part of the learning model although it is treated lightly in many current problem descriptions. The fact is that defining the channel bandwidth can make a problem either hard or easy, independent of the problem environment. For example we could define a machine learning problem whose problem environment was very large, say a Woods environment on a very large two dimensional surface. If, however, all of this information and an appropriate evaluator were given to the learner the determination of the solution would be greatly simplified. On the other hand the woods problem can become very difficult if only local neighborhood information is allowed to flow out of the problem environment.

The stimulus channel is in most cases a restrictor of information flow from problem environment to learner. Its information quanta are often limited views of the total environment (e.g., a single letter in the letter sequence task). The action channel carries information to the problem environment that can have two possible consequences. It can either change the state of the problem environment (as a change in position would for the Woods domain) or it can be evaluated to determine whether a payoff quantum should be placed on the payoff channel. In most cases information on the action channel will be evaluated but the payoff channel is not always activated. The bandwidths of the action and payoff channels are also important to the problem definition. For example consider a problem that is defined to have a payoff channel that is active after every action quantum is received. This would correspond to a problem that evaluates every step taken by a learning system. This problem would be much

<u>Components of a Learning Problem</u>

| *Problem Name* | *Problem Environment* | *Stimulus Channel* | *Action Channel* | *Payoff Channel* | *Learning Objective* |
|---|---|---|---|---|---|
| Animat | Woods | Current View | One Step Moves | On Contact With Food | Get Food Expeditiously |
| Letter Sequence Predictor | Letter Sequence | Current Letter | Name a Letter | On Correct Prediction | High Percent Correct |
| Boole | Binary Strings | Random String | 1 or 0 | On Correct 1 or 0 | High Percent Correct |
| Pole Balancer | Pole Physics | Pole State | Applied Force | When Pole Falls Over | Long Time Till Fall |

figure 2.

easier than a similar problem that sent payoff quanta only after many stimulus quanta. An example can be seen in the pole balancing problem where it becomes a relatively simple problem if feedback is given after every small movement made by the learner. Certainly the learning problem is much harder when payoff is made only after the pole falls over.

It may be interesting to notice that in a paper proposing to define a canonical learning problem no attempt at a definition of learning itself has been proposed. The reason for this is that, in the model, the definition of success or failure of "learning" is defined by the problem. This definition of success or failure comprises the *learning objective* component of the Rosetta model. The learning objective can be viewed as the goal or solution to the defined learning problem, or it can be some measure of solution coupled with other measures.

## Problem Descriptors

The previous section showed how the Rosetta model separates a learning problem from an overall system in which it may happen to reside, and analyses the problem into five components. In this section we introduce four *descriptors* whose values further define the nature of a problem and are useful in comparing one problem with another. The descriptor values are derivable from the values of the five components. The descriptors are: *payoff frequency*, *history dependence of the optimal action*, *intra–step mapping*, and *noise*. Figure 3 shows the descriptor values for our example problems.

One characteristic of learning problems that has been implicitly assumed so far is that stimulus information is presented in some discrete way. One can think of this activation of the stimulus channel and the subsequent activations of the action and payoff channels as a single time "step". In the Rosetta model a time step is defined as the sequence consisting of activations of the stimulus channel, the action channel and then perhaps the payoff channel. This concept of a time step is important because it allows one to distinguish between actions that occur at different times. It also gives us a formal model for determining how previous channel activations affect current activations.

The descriptor called payoff frequency is now easy to define. If the learner receives payoff or reinforcement on every time step, then the payoff frequency is noted as "every step". If not, the payoff is infrequent to the degree that time steps on which payoff occurs are rare. In general, problems are harder if payoff frequency is low, since it is difficult to determine the worth of actions that did not receive payoff (the temporal credit assignment problem).

Our second descriptor depends on the notion of "optimal action". By this we mean the best action that could be taken with respect to the learning objective, given the current state of the problem environment. Once this concept is defined it is possible to notice the dependence of this action on previous stimuli. For example, the optimal action in a letter sequence task can depend on the stimuli from previous time steps as well as the letter currently in the stimulus channel. In the sequence "whyhwhyh...", prediction of the letter to follow "h" depends on whether "w" or "y" occurred on the previous step.

In the Rosetta model, the descriptor "history dependence of the optimal action" has the value "no" if the optimal action can be determined strictly from knowledge of the present stimulus. If, on the other hand, previous stimuli are required, the value is "yes". A more precise valuation would state the number of steps back in time that are necessary to determine the optimal action. Obviously, the greater the history dependence, the more the learner needs in some manner to remember prior events.

## Applying Descriptors to Learning Problems

| *Problem Name* | *Payoff Frequency* | *History Dependence of Optimal Action* | *Intra–Step Mapping* | *Noise* |
|---|---|---|---|---|
| Animat | Infrequent | Yes | Discontinuous | No |
| Letter Sequence Predictor | Every Step | Yes | Discontinuous | No |
| Boole | Every step | No | Discontinuous | No |
| Pole Balancer | Infrequent | No | Continuous | No |

figure 3.

Our third descriptor concerns the nature of the mapping from stimulus to optimal action within a time step. Without loss of generality, we can consider this mapping always to be from one binary string to another. However, the mapping can be either many–to–few or many–to–many. The first case is characteristic of classification problems in which large numbers of possible inputs are placed in a relatively small number of categories. We term this "discontinuous" because the input can change gradually without change in the output until suddenly a new, in general very different, output is called for. The second case is characteristic of sensory–motor problems in which for every possible input there is a relatively distinct optimal output. In this case the term "continuous" is appropriate since a small input change calls in general for a small output change. The two types of mappings present quite different problems and imply different learning approaches.

Last we can classify problems in terms of noise. Since by definition the problem environment contains all that is required to evaluate and define the problem we can not introduce noise at that point or we will have changed the problem. Instead noise can be introduced on the communication channels. There is perhaps no good example where corrupting the action channel would be relevant to real world learning systems; however, corruption of the information found in the stimulus channel and the payoff channel is of considerable interest. Corruption of the stimulus channel would, for example, correspond to the woods problem domain being covered in a dense fog bank where stimulus information was almost correct but could be sometimes corrupted. Corruption of the payoff channel would correspond to a preoccupied teacher where payoff was sometimes given when it should not have been and withheld sometimes when it should have been distributed.

## Summary

This paper has introduced a model of learning problems and a limited vocabulary with which to describe some of their higher level attributes. The main contribution of the model is its ability to separate the learner from the problem itself by explicitly defining the communication paths between these two entities. If this model is used in the future it should aid the researcher in more completely defining a learning problem so that other researchers can accurately understand and use it.

## References

Barto, A.G., Sutton, R.S., and Anderson, C.W. (1984). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC–14**, 834–846.

Robertson, G. G. and Riolo, R. L. (1988). A tale of two classifier systems. *Machine Learning* **3**, 139–159.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications, (pp. 16–23).* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning* **2,** 199–228.