Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Active Training of Backpropagation Neural Networks Using the Learning by Experimentation Methodology

Fu-Ren Lin
Department of Information Management
National Sun Yat-sen University
Kaohsiung, Taiwan,
R.O.C.

Michael J. Shaw
The Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 North Mathews Avenue
Urbana, IL 61801
U.S.A.

Abstract

This paper proposes the learning by experimentation methodology (LEM) to facilitate the active training of neural networks. In an active learning paradigm, a learning mechanism can actively interact with its environment to acquire new knowledge and revise it. The learning by experimentation is an active learning strategy. Experiments are conducted to form the hypotheses, and the performance of those hypotheses feeds back to the learning mechanism to revise knowledge. We use a backpropagation neural network as the learning mechanism. We also adopt a weight space analysis method and a heuristic to select salient attributes to perform new experiments in order to revise the network.

1. Introduction

The *Learning by Experimentation Methodology* (LEM) is an *active* learning strategy. The LEM consists of two main tasks: *hypothesis formation* and *hypothesis revision.* The learning by experimentation methodology is proposed and applied in various areas, such as AM [3], BACON [2], LEX [4], IDS [5], KEKADA [1], and PDS [7][6]. These works are done in the symbolic learning approach, but works on using neural networks to accomplish the LEM are still rare [9]. However, we found some advantages of using neural networks for learning by experimentation, and erected our motivations of pursuing this research from them. First, learning by experimentation is an incremental learning process, and learning based on neural networks is essentially an incremental learning. Therefore, using neural networks is a straightforward option for inductive learning mechanism. Second, learning in neural networks is more robust in dealing with noisy data than some symbolic learning algorithms. So, the concepts learned using neural networks is more accurate. Third, because the learned concepts are embedded in the connection of neural networks, once the concepts are learned it is very efficient to generate the output given the input variables. Therefore, this efficiency makes neural networks suitable for the real time dynamic control systems. Finally, the property of the incremental learning of neural networks is very suitable for the knowledge revision in the LEM.

This research takes the advantages of the characteristics of neural networks in the inductive learning mechanism and applies the learning by experimentation methodology to perform active learning. Therefore, the linkage of neural networks with the LEM is one of the main contributions. The other contributions of this work include methods for the salient attribute detection and the informative example selection for the active learning.

2. Learning by Experimentation Methodology

The Learning by experimentation methodology (LEM) is an active learning paradigm which the learner should actively select examples from the environment as the input patterns for induction. The framework of the LEM consists of two main tasks: *hypothesis formation* and *hypothesis revision.* During the hypothesis formation, first, a learner should define the problem domain, determine the variables, and build the model. Second, a learner can decide the experimentation goal, experimental design, and then perform experiments. Finally, from the experiments' results which are distinct instances, an inductive learning mechanism can be applied to form hypotheses. The generated hypotheses have to be tested before they are viewed as theories. We can either test them on seen events or use them to predict future events. The results of testing, either supporting or denying hypotheses, feed back for the hypothesis revision. During the hypothesis revision, first, a learner should detect the defective hypotheses, locate the important variables, and then concentrate on the informative values. Second, experiments are performed by inputting these informative values as the independent variables, and training examples are collected for the further learning. Finally, the hypotheses are refined and then tested. These stages iteratively continue until the learner is satisfied with the learned hypotheses.

Figure 1 shows the spiral model of life cycle for learning by experimentation. The curve extends outward while the quality of hypotheses (theory) improved by theory revision. The curve stops growing when the performance of the learned knowledge exceeds the threshold of acceptance. This spiral model demonstrates LEM's *dynamic*, *continuous*, and *self-adaptable* properties. It is dynamic because the monitoring of the system performance will trigger the theory revision process when the prediction accuracy of theory is below the revision threshold. It is continuous because the reach of outside circle is not a direct leap from inside core, but a continuous movement consisting of the iteration of hypothesis formation and revision stages. It is self-adaptable because the learning agent itself controls the learning process, including to investigate the outside world using its own experimental design, to obtain various quality of theory, and to generate heuristics for refining the existing theory by monitoring the results of problem solving. The iteration of hypotheses formation and revision in the LEM can be further developed into four iterative phases: *experimentation, hypotheses formation, problem solving* and *performance evaluation,* and *hypotheses revision*, (Figure 2) and is discussed in the followings.

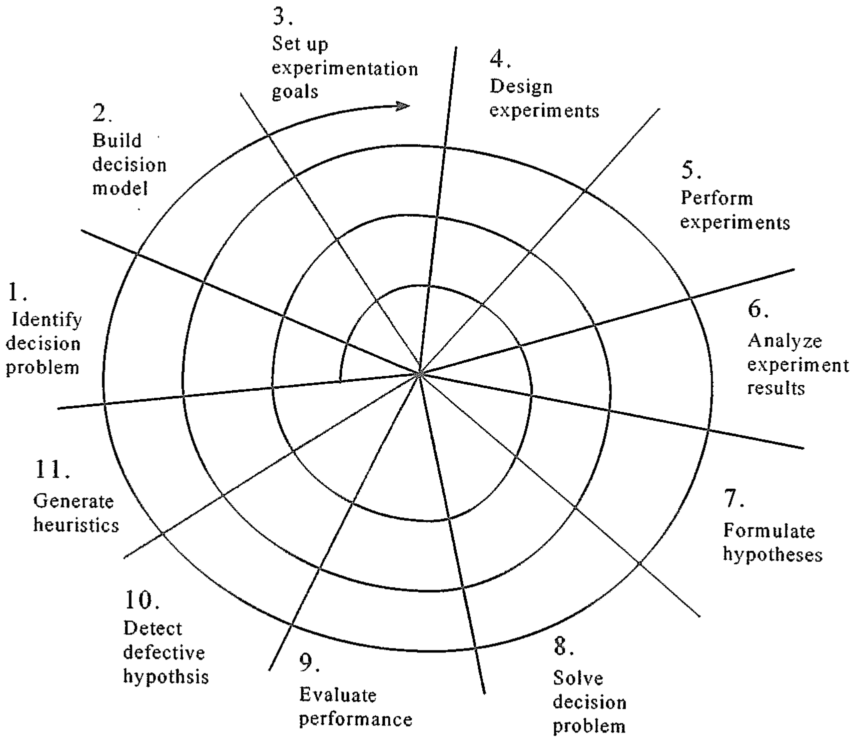
3. Set up experimentation goals
4. Design experiments
2. Build decision model
5. Perform experiments
1. Identify decision problem
6. Analyze experiment results
11. Generate heuristics
7. Formulate hypotheses
10. Detect defective hypothsis
9. Evaluate performance
8. Solve decision problem

Figure 1. The spiral model of the LEM

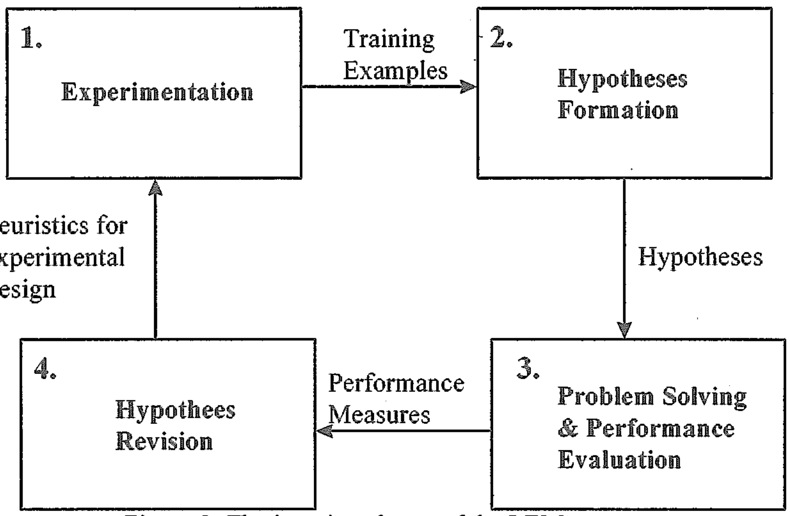

Figure 2. The iterative phases of the LEM

Phase 1: Experimentation

Experimentation plays a fundamental role in scientific discovery. Scientists use experiments to investigate phenomena, gather data, and verify theories. In an experiment, there are *factors* which are observable and controllable and *responses* which are effects caused by the manipulation of factors. Therefore, instead of passively receiving instances from outside world, experimentation serves as an active information collector. The goal of experimentation is to investigate the relationship between actions and responses of the examining system in various system settings. Experimental design completely specifies the experimental test runs. In the LEM, the experimentation is a mechanism that generates the training examples for hypothesis formation. The experimental design also takes into account the heuristics generated from the hypothesis revision.

Experimentation is an active approach for decision makers or problem solvers to explore the environment by changing some situations in the given model and then observing the results of those changes. These changes can be systematic such as the experimental design mentioned above, or dynamically controlled by some heuristic criteria such as the *interestingness* used in AM system [4]. In the LEM, the experimentation works as an example generator. It is designed to demonstrate the performance of problem solving and then the situations, actions and results are stored as training examples for inductive learning. The design of experimentation is guided both by the systematic factor combination and heuristics.

Phase 2: Hypothesis Formation

Hypothesis formation, in essence, is a process of inductive learning, especially *learning from examples*. The tasks of inductive learning is to generalize the relationship among examples and represent them in a concise and consistent form (depending on the knowledge representation) to predict the unseen events. The performance of learning from examples is heavily influenced by the training examples. Therefore, the distribution of training examples, the noise, and the bias of sampling data are major considerations in inductive learning. Since learning by experimentation is an active learning that a learning agent actively selects training examples for its own goal, the experimentation and hypothesis formation work in a supplier-consumer relationship. The examples generated from experimentation are the source for the learning from examples. On the other hand, the obtained hypotheses will be used to solve problems. This chain effect shows that the performance of learning process is decided by the quality of training examples, and the success of problem solving is decided by the quality of learned hypotheses.

Phase 3: Problem Solving and Performance Evaluation

The learned hypotheses can be tested by generated testing examples or can be used to solve problems in order to evaluate their effectiveness or efficiency. Due to the incompleteness or incorrectness of the existing hypotheses, the knowledge base is demanding for revision. The results of problem solving or testing are the source of feedback to the whole system of learning by experimentation. This feedback provides the information for the theory revision mechanism to refine the existing theory in the knowledge base.

Phase 4: Hypothesis Revision

The feedback of performance measures from the hypothesis evaluation is the motivation of revision. The hypothesis revision mechanism consists of the following functions: (1) detecting the defective hypotheses, (2) extracting the salient attributes, and (3) generating the heuristics of experimental design for the next run of experimentation.

The performance of hypothesis is measured by comparing the predicted value with the actual output value. The detection of defective hypotheses is to search for hypotheses that have substantial differences between expectation and reality. Those hypotheses performing poorly (under given threshold) in solving problems will be the focus of revision. From the collection of defective hypotheses, we can extract the attributes that makes the effective hypotheses different from the defective hypotheses. These attributes will be the focus of experimental design for following experiments. This hypothesis revision will generate the heuristics of experimental design for the next run of experimentation.

3. Active Training of Backpropagation Neural Networks

The active training of backpropagation neural networks using the LEM *iteratively* invokes the following procedures:

1. **Experimentation:**
   - performing experiments,
2. **Hypothesis formation:**
   - preparing examples, and
   - training the neural network,
3. **Problem solving and performance evaluation:**
   - using the trained network to test on holdout examples, and
   - if the performance is satisfied then stop,
4. **Hypothesis revision:**
   - analyzing the weight space of learned networks, and
   - selecting salient attributes and effective value ranges, go to step 1.

The following subsections will describe the analysis of network and the selection of informative attributes in order to revise the networks.

3.1 The Analysis of Neural Networks

Since the architecture of neural network is a sub-symbolic structure, we cannot straightforwardly convert it into explicit rules. The knowledge refinement for this kind of sub-symbolic representation mainly lies in the exploration of the relationship between input pattern and output values through the connection. The analysis of relative strength for input variables with their output variables provides valuable information for experimental design to select informative examples for the continuous improvement.

In a neural network, the weight space provides the connection strength between two units. For each attribute i (unit i of input layer) and classification j (unit j of output layer), we can compute the *relative salience* between this attribute and classification. Assume that a neural network has one input layer with m units, one hidden layer with n units, and one output layer with p units. The relative salience is computed as

$$RS_{ij} = \frac{\sum_{k=1}^{n}\left(w_{ki} \bullet w_{jk}\right)}{\sum_{i=1}^{m}\left|\sum_{k=1}^{n}\left(w_{ki} \bullet w_{jk}\right)\right|},$$

where RS_{ij} is the relative salience between the i^{th} input unit and the j^{th} output unit, w_{ki} is the weight between i^{th} input unit and k^{th} hidden unit, w_{jk} is the weight between k^{th} hidden unit and j^{th} output unit. RS_{ij} is the ratio of the strength between i^{th} input unit and j^{th} output unit over the total strength of all of the input units and j^{th} output unit.

We adapt the idea of weighted average to compute the *relative effect* of each input attribute. For the i^{th} input value I_i, and the j^{th} output value O_j, RS_{ij}, the effect of I_i upon O_j, is obtained by

$$RE_i = \frac{I_i \bullet RS_{ij}}{\sum_{k=1}^{m} I_k \bullet RS_{kj}}$$

RE_{ij} provides us the relative importance of attribute toward the output classification. An attribute with higher RE value contributes more information toward the classification. We can set up a threshold value β as the lower bound for us to select relative effective attributes. Yoon, et al (1994) propose this approach to extract explanation rules from networks [8]. In statistics jargon, this relative effects RE is equivalent to *main effects.*

3.2 Selection of Training Examples

In the learning by experimentation environment, the classification of examples is assigned after experiments, and we also have no pre-determined boundary about the target knowledge. The performance of neural network by testing examples is the only information fed back to the learning agent. To analyze those examples which are classified incorrectly (i.e., false examples) is the focus of our heuristic.

The heuristic used here we call the *nearest neighbor strategy*. For those false examples, we focus on the neighbor of each attribute value. The range of the nearest neighbor is defined by the discretization process which clusters contiguous values of an attribute which have the same classification into a cluster. For example, a false example consists of salient and un-salient attribute $\{a_s\}$ and $\{a_{\neg s}\}$. The value of a_s is v_s, and it belongs to a cluster between lb_k and rb_k; that is, $lb_k \le v_s \le rb_k$. v_s's nearest neighbor is defined between lb_k and rb_k. The value in the next experiment will be selected randomly from range $[lb_k .. rb_k]$ of a_s. The values for those un-salient attributes will be selected randomly from range of the possible smallest and largest values of that attribute.

Definitions:

- *BPN*: the backpropagation neural network.
- *S*: the example space, $S = S_{train} \cup S_{test}$, and $S_{train} \cap S_{test}, = \varnothing$.
- S_{test}: the testing examples, $S_{test} \subseteq S$.
- S_{false}: the testing examples which are classified incorrectly, $S_{false} \subseteq S_{test}$.
- S_{train}: the training examples, $S_{train} \subseteq S$.
- $[a_1, a_2, ..., a_n]$: a sequence of attributes of the example.
- *A*: the attribute space. i.e., $A = \{ a_1, a_2, ..., a_n \}$.
- $[v_1, v_2, ... , v_n, c_i]$: a sequence of values for the $[a_1, a_2, ..., a_n]$ of the example.
- c_i is the classification.
- RE_{ij}: the effect of the i^{th} attribute upon j^{th} class.
- β: the threshold value for RE_{ij}.
- A_{ef}: set of attributes whose *RE* value $\ge \beta$.
- S_{output}: the attribute : value pair for those salient attributes. i.e., $S_{output} = \{a_{ef}: d_{ef}\}$.

Heuristic:

Input: *S*, *BPN*.

Process:

```
REPEAT
  1. Choose $s = [v_1, v_2, ... ,v_n, c_i]$ from $S_{false}$.
  2. Compute $RE_{ij}$ for $s$, where $i = 1$ to $n$.
  3. Assign attributes which $RE \ge \beta$ to $A_{ef}$.
  4. $A_{\neg ef} = A - A_{ef}$.
  5. Discretize attributes in $A_{ef}$. $\forall a_{ef} \in A_{ef}$,
     $a_i$ clusters into intervals $[lb_1 .. rb_1]$, $[lb_2 .. rb_2]$, ..., $[lb_m, ...; rb_m]$.
  6. $a_{ef} \in A_{ef}$, $lb_i \le v_{ef} \le rb_i$, randomly
     generate a value $d_{ef}$, where $lb_i \le d_{ef} \le rb_i$.
  7. $S_{output} = S_{output} \cup \{d_{ef}\}$.
  8. $S_{false} = S_{false} - s$.
UNTIL $S_{false} = \varnothing$.
```

Output: S_{output}.

3.3 Knowledge Refinement on Neural Networks

Based on the feedforward neural network architecture, we propose a knowledge refinement algorithm which can analyze the weight space of learned network, locate the salient variables, and suggest the range of possible values for experimental design. We describe this knowledge refinement algorithm as follows.

Definitions:

- *BPN*: the backpropagation neural network.
- *S*: the example space, $S = S_{train} \cup S_{test}$, and $S_{train} \cap S_{test}, = \varnothing$.
- S_{test}: the testing examples, $S_{test} \subseteq S$.
- S_{train}: the training examples, $S_{train} \subseteq S$.
- *e*: the prediction error rate of *BPN*.
- *h*: the error rate threshold given by the user.

Algorithm:

Input: S_{init}, BPN_{init}, *h*.

Process:

```
$S = S_{init}$.
$BPN = BPN_{init}$.
$S = S_{train} \cup S_{test}$, and $S_{train} \cup S_{test}, = \varnothing$.
Test the accuracy of current BPN using $S_{test}$.
WHILE $e > h$
  1. Apply Heuristic (Subsection 3.2) to select attributes and values for experiments.
  2. Generate new examples $S_{new}$ from experimentation.
  3. $S = S \cup S_{new}$.
  4. $S = S_{train} \cup S_{test}$, and $S_{train} \cap S_{test}, = \varnothing$.
  5. Modify BPN using $S_{train}$.
  6. Test the accuracy of current BPN using $S_{test}$.
END of WHILE
```

Output: *BPN*.

4. Conclusions

In this paper, we propose a learning by experimentation methodology (LEM) to perform active learning. A backpropagation neural network is used as the learning mechanism. In order to embed the network with active learning capability, we adopt a self-refinement method to revise knowledge. This self-refinement method consists of analyzing knowledge embedded in the connection weights, selecting salient attributes and effective value range for performing the experiments. In order to speedup the consequent training on neural network, we saved the learned weights, and then loaded into the network for the next round of training. The stored weight space works as a *memory* which avoids training from scratch in each round of learning, and increases the learning process too.

Reference

[1] Kulkarni and H.A. Simon. Experimentation in machine discovery. In J. Shranger and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formulation*, pages 255-273. Morgan Kaufmann, San Mateo, CA, 1990.

[2] Langley, G.L. Bradshaw, and H.~A. Simon. Rediscovering chemistry with the bacon system. In R.S. Michalski, J.G Carbonell, and T.M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach*, pages 307-330. Morgan Kaufmann, Los Atlos, CA, 1983.

[3] Lenat. The role of heuristics in learning by discovery: three case studies. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach*, pages 243-306. Morgan Kaufmann, Los Atlos, CA, 1983.

[4] Mitchell, P.E. Utgoff, and R.Banerji. Learning by experimentation: acquiring and refining problem-solving heuristics. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach*, pages 163-190. Morgan Kaufmann, Los Atlos, CA, 1983.

[5] Nordhausen and P. Langley. An integrated approach to empirical discovery. In J. Shranger and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formulation*, pages 97-128. Morgan Kaufmann, San Mateo, CA, 1990.

[6] Piramuthu, N. Raman, and M.J. Shaw. Learning-based scheduling in a manufacturing flexible flow line. *Technical report*, Beckman Institute, University of Illinois at Urbana-Champaign, 1993.

[7] Park S. C., Raman N and Shaw, M.J. Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IIE Transaction*, 24(2):156-168, 1992.

[8] Yoon, T. Fuimaraes, and G. Swales. Integrating artificial networks with rule-based expert systems. *Decision Support Systems*, 11:497-507, 1994.

[9] Zhang and G. Veenker. Neural networks that teach themselves through genetic discovery of novel examples. In *1991 IEEE International Joint Conference on Neural Networks*, volume 1, pages 690-695. IEEE, New York, NY.