# Machine Learning

# Assignment 1

Literature review: LEARNING INTERNAL REPRESENTATIONS BY ERROR PROPAGATION

AUGUST 27, 2019

Name: Liang Ou
Student ID: 13060835

# Table of Contents

# Introduction

Minsky and Papert (1969) proposed the problem of the many-layered version neural network lacks powerful convergence theorem. To solve this problem, the authors of this book have found a learning result sufficiently powerful to demonstrate that their pessimism about learning in multi-layer machines was misplaced.
The authors point out that without hidden layers in a neural network, it is unable to learn certain mappings between similar inputs and different outputs using the similarity of patterns. However, by adding internal representation units to augment the original input pattern, the network can perform any mapping from input to output. However, optimizing weights between units of such network is a complicated job. This book propose a so called “backpropagation” method to address this optimization problem.

# Content

The main challenge of the book is defining a generalized learning procedure for multi-layer perceptron network. This is because multi-layer neural networks have hidden units between input layer and output layer, thus change of weights between layers influence the inputs of the units in higher layers, which bring a difficulty to optimize these weights. Although there are several methods for updating weights, they all have limitations. Therefore, network with hidden layers units cannot use a simple rule, such as "delta rule”, for all problem, 3 response is proposed:

1. Competitive Learning: hidden units develop by simple unsupervised learning rules.
    a) The disadvantage of the response is that there is no guarantee that hidden units appropriate for the required mapping are developed.
2. assume an internal representation:
    a) appropriate for verb learning and word perception
3. develop a **learning procedure** which is adjustable for task variations.
    a) Boltzmann machines:
        i. Uses stochastic units
        ii. Reach equilibrium in two different phases
        iii. limited to symmetric networks
    b) stochastic units by Barto (1985)
    c) generalized delta rule.
        i. deterministic units
        ii. involves only local computations
        iii. a clear generalization of the delta rule
    d) learning-logic (Parker (1985))
        i. a similar generalization with “generalized delta rule”

e) Le Cun (1985) has also studied a roughly similar learning scheme.

This book proposed a generalized delta rule by backpropagation of the error signal. This approach is tested in several kind of problems and yields significant success. The detail of this approach and its results are discussed in the following section.

# Innovation

The learning procedure this book proposed is called "The Generalized Delta Rule".

The three steps of delta rule are:

1. uses the input vector to produce its output vector
2. compares this with the desired output or target vector.
3. the difference is reduced by change weights

**The standard delta rule is given by the following formula:**

$$\Delta_p w_{ji} = \eta(t_{pj} - o_{pj}) i_{pi} = \eta \delta_{pj} i_{pi}$$

For $w_{ji}$, j refers to the jth perceptron in the output layer, i refers to the ith perceptron in the input layer. This formula explains weights will change by measuring the difference between target output and actual output. $\eta$ is the learning rate. This can be derived by taking the partial derivative of Error (defined by $E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2$) with respect to $w_{ji}$. It is given by: ( $-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_{pi}$).

**Error signal:**

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj}$$

This function means the "error signal" of a perceptron is calculated by its firing strength's derivative multiplies the sum weighted "error signal" of its connected upper perceptron. If the perceptron is output perceptron, the last term is $(t_{pj} - o_{pj})$. Error signal means the derivative of the error with respect to net input, defined by the formula $\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}$

**The delta rule for semilinear activation functions in feedforward networks:**

By adding hidden units w may converge at local minima.

The activation function is defined as $o_{pj} = \frac{1}{1+e^{-(\sum_i w_{ji} o_{pi} + \theta_j)}}$, its derivative is calculated through $\frac{\partial o_{pj}}{\partial net_{pj}} = o_{pj}(1 - o_{pj})$.

**SIMULATION RESULTS**

After several simulations, the authors found there are two major local minima issues involved in their optimization procedure:

1. **Symmetry breaking:**

If weights are initiated equally, the error signal could be the same, because it is calculated by weight multiplies output error. Then the changes for all neurons are the same, which again results in the same weights. To solve its local maximum risk, small random weights is initiated.

**2. A rare local minima:**

If two opposite pattern's (like 0 and 1) net input for the output unit is 0 (the output is defined to be 0 if net input is negative, and 1 if net input is positive), the outputs for both cases are 0.5, and errors are 0.5 and -0.5, so the sum of the error is 0. And the weight will not change.

To further discuss the effectiveness and issues of the learning procedure, this book elaborates several problems.

**Problem 1: XOR problem**

An experiment of XOR problem with only one hidden layer shown that $\mathrm{P} = 280 - 33\log_2 H$, in which P is the average number of presentations to solve the problem, H is the number of hidden units employed. The formula implies that as the number of hidden units increases, the solving time reduces. Another finding it the within the range from 0.1 to 0.75, the larger the learning rate, the faster the converging speed. For the learning rate, beyond 0.75, the predictor will be unstable.

As it is shown in **Figure 1,** biases of all nodes are written inside the circles and weights are presented beside connection lines. For example, the left hidden unit will be activated if the left input is off or the right input is on.

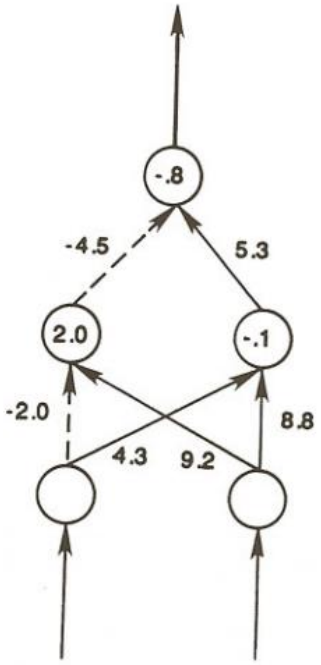



Figure 1, Network structure for solving XOR problem.

**Problem 2: Parity**

If the answer is different for similar input patterns, the hidden layer is needed to interpret the problem.

As Figure 2 shows an converged network structure for solving the odd-even classification problem. The number of hidden units needed is equal to the number of input units. Therefore, they simply count how many input number is activated.

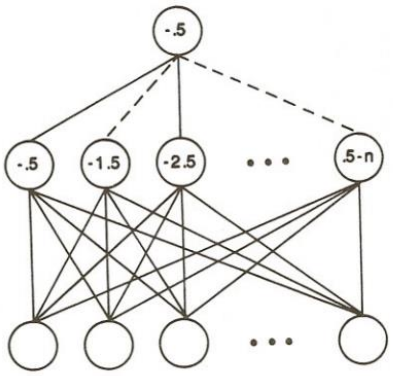



Figure 2. Network structure for discriminating odd and even numbers

**Problem 3: The Encoding Problem**

Using intermediate values other than only 0 (fully turned off) and 1 (fully turned on) as output values increase the flexibility of the learning system.

**Table 1** shows how inputs are transformed into values in the range of [0,1] for a single unit. The benefit of using continuous values is the system can reduce the number of hidden units employed.

| Input Patterns | Signleton Hidden Unit | Remaining Hidden Units | Output Patterns |
|---|---|---|---|
| 10 | 0 | 1/1/1/0 | 0010 |
| 11 | .2 | 1/1/0/0 | 0001 |
| 00 | .6 | .5/0/0/.3 | 1000 |
| 01 | 1 | 0/0/0/1 | 0100 |

Table 1. pattern transformation with only one unit in a intermediate layer.

**Problem 4: Symmetry**

Only 2 hidden units are enough for classifying whether an input pattern is symmetric or not. The network does that by applying symmetry weights for all input neurons with opposite signs, such as 1, -2, 4, -4, 2, -1 for one neuron and -1, 2, -4, 4, -2, 1 for the other. The negative biases on hidden units and positive bias on the output units insure that the output only turns on for symmetric input values.

**Problem 5: Addition**

For two “m” length of binary bits, a minimal network needs 2m inputs units, m carry (hidden) units and m+1 output units. Because the lower carry unit should be considered as one input units of a higher carry unit, an appropriate connection of hidden layer is necessary. This local minima problem can be solved by adding one more hidden units.

This Addition problem demonstrates that if the number of hidden units is more than minimal requirement, it enhances the interpretability and avoids localist (stuck in local minima). However, hidden units become hard to be interpreted, and their importance is the same.

**Problem 6: The Negation Problem**

This is the problem in which one input is considered as a "sign" to control whether the n outputs should be exactly the same as the rest of n inputs or the complement of those inputs. In this case, n hidden units are needed to detect the combination of the "sign" and every input unit.

**Problem 7: The T-C Problem**

A system is designed to discriminate the shape of “T” and “C”, which consist of 5 squares. Each hidden units measures the inputs' shape by projecting the inputs into a square 3 x 3 region. Feature detector of all hidden units is the same, how due to the location and rotation of the inputs is uncertain. A two-dimensional grid of hidden units is required to scan the input space for pattern recognition. Features detected of the hidden units “includes on-center-off-surround”, “vertical bar”, “diagonal bar” and “compactness”.

One conclusion from the solutions of this problem is that inhibit the hidden units at the beginning of the learning can avoid correct answer by random connections. That means without turning on by inputs, the hidden units should be on.

**Generalization:**

This book than generalize the generalized delta rule to sigma-pi units and recurrent networks. For sigma-pi units with conjunction less than two, the error signal is given by

$$\delta_j = f'_j(net_j) \sum_{j,k} \delta_k w_{kij} o_j$$

For recurrent networks, they can be transformed into multiple layered feedforward network with same weights for every iteration. The experiment for “**shift register**” shows that the system will set all weights to be 0, but the one connects to its left to be within

200 sweeps and with learning rate η= 0.25. Another experiment of "**complete sequences**" let errors are injected at each time-step by comparing the remembered actual states of the output units with their desired states.

# Technical quality

In the aspects of theory inference and formula derivation, this book has a high quality. This is because it derives its formulas step by step with an explicit explanation of all variables and notations. A total 17 equations, 12 tables and 17 figures are introduced for supporting its arguments. Moreover, limitation and issues of backpropagation are widely discussed, which facilities further relative research. For example, although this book identifies 2 local minima problems as they are discussed, they point out that more studies need to be conducted under different conditions. Another example is that they state the limitations when testing the generalized delta rule on Recurrent Nets and sigma-pi units.
In the aspect of simulations, although it explores many problems and evaluates the learning procedure by the time complexity and accuracy, it lacks the explanation for how the backpropagation algorithm is applied to each problem. Therefore, although the results look pretty neat, it cannot be replicated by readers, or at least hard to be re-implemented.

# Application and X-factor

Apart from backpropagation, there are numbers of optimization algorithms have developed for a neural network. One type of popular algorithms are Evolutionary Algorithms (EAs) which mimic natural evolutionary principles. EAs can be further divided into Evolutionary strategies (ES), Evolutionary Programming (EP), Genetic Algorithms (GAs), and Genetic Programming (GP). Take GAs as an example, it optimizes all weights in a network with the following process:

1. reproduction:
   Different combinations of weights are evaluated and the best of them are selected
2. recombination (crossover):
   interchange some weights in two sets of weights
3. mutation:
   randomly change some weights

The merit of GAs is that it can easily escape from local minima.
Another type of optimization algorithm is Swarm Intelligence (SI), by which individual solution communicate with each other for finding the optima in the solution space.

One example of SI is Particle Swarm Optimization (PSO), which is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995. In PSO, every particle (solution) have its position ($\overrightarrow{x_i}$) and moving speed ($\overrightarrow{v_i}$). For each iteration, the moving speed is updated by the following formula (Kennedy 2010):

$$\overrightarrow{v_i}(t+1) = \overrightarrow{v_i}(t) + c_1\varphi_1\left(\overrightarrow{p_i}(t) - \overrightarrow{x_i}(t)\right) + c_2\varphi_2\left(\overrightarrow{p_g}(t) - \overrightarrow{x_i}(t)\right)$$

Where $c_1$ and $c_2$ are acceleration coefficients, often positive constants, $\varphi_1$ and $\varphi_2$ are random numbers in [0,1], $\overrightarrow{p_i}(t)$ is the best position of particle $\overrightarrow{x_i}$ and $\overrightarrow{p_g}(t)$ is the best position of all particles, t is the iteration. By updating the position through $\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t+1)$, all particles move toward the best one. **Figure 3** shows how a particle moving toward the best position in the solution space

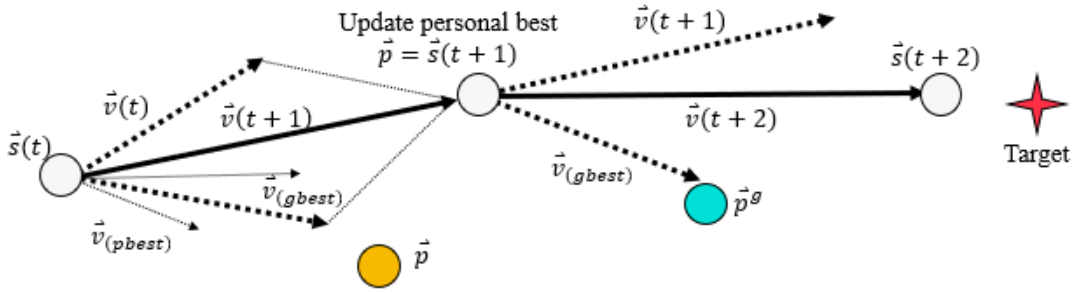
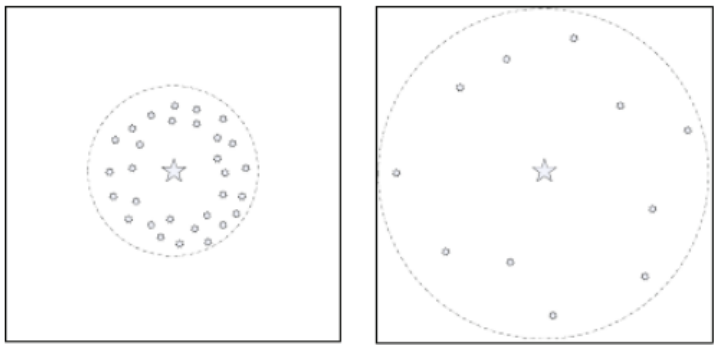



Figure 3. The updating strategy of a particle moving toward the best position in the solution space

Although PSO is very efficient in converging to the best solution, it is likely to trap in certain local minima.



(a) Good explosion (b) Bad explosion

Figure 4. The search strategy of FA. (a) is an explosion by a high fitness firework, whereas (b) is an explosion by a low fitness firework.

Another SI algorithm, called Firework Algorithm (FA) provides both efficiency and diversity. The framework of FA is generating new solutions (sparks) by old solutions (fireworks), the explosion (search) radius and density are defined by fireworks' performances, as it shows in **Figure 4** (Tan & Zhu 2010),. Selection strategy in FA is based on the distance (similarity) of particles so that the diversity of particles of the next generation is guaranteed.

# Presentation

If the lowest rate the quality is 0 and the highest rate of quality is 5, I will rate this book with a rate of 4. Overall, the organization of this book well-aligned, terminologies in this book is well- explained. Therefore, one can easily grasp the main concept of "backpropagation" even as a beginner for machine learning. However, there are still some bad presentation approaches make me feel hard to follow the book. First, when it uses some definition discussed in other chapters, such as “semilinear”, “sigma-pi units”, there is no brief description at all. This explanation style makes those concepts impossible to understand due to other chapters are not available to readers. Second, it wastes length very much on calculating simple

numerical additions for inputs and outputs rather than explaining why such additions should be happening. Third, many experiments on the research of this book are based on Minsky and Papert (1969)'s book. However, it assumes that readers should understand the research problems proposed by Minsky and Papert (1969), while in most cases they probably don't.

# References

Kennedy, J. 2010, 'Particle swarm optimization', *Encyclopedia of machine learning*, pp. 760-6.

Tan, Y. & Zhu, Y. 2010, 'Fireworks Algorithm for Optimization', eds Y. Tan, Y. Shi & K.C. Tan, Springer Berlin Heidelberg, pp. 355-64.