# Individual Activity 2

**IA5008 Sistemas neuronales**

**Author:** Nisim Hurst

**Registry ID:** A01012491

**Due Date:** 24th of January 2018

## Abstract

It was demonstrated by Minsky and Papert [1] that one layer perceptron's fail to map to non-linearly separable functions, e.g. the XOR function. Other models were proposed with hidden layers, that were known to be more powerful.

A single hidden layer is able to effectively circumvent the non-linearly separable XOR problem. However, due to the lack of a desired output in the hidden layers there was no way to train them. A convergence procedure using the powerful delta rule was not discovered yet. There have been three responses proposed to compensate this lack:

1. Unsupervised learning rules to generate useful hidden units using competitive learning, however there is no guaranty that useful hidden layers will develop.
2. Assume internal representation using apriori knowledge. The downside is that it needs prior knowledge.
3. Use a learning procedure using stochastic units, e.g. Boltzman machines.

The chapter [2] presents a generalization of the delta rule that involves only local computation and uses deterministic units. It can be applied to feed forward and recurrent networks. The proposed backpropagation algorithm provides a mechanism for training the weights of any networks considering an arbitrary number of hidden layers and units. This algorithm was also proposed independently by Werbos (1974) however it was never published.

## Generalized delta rule

Changes in the weights of the connections are applied to produce the output vector, given by the following delta rule:

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

1. $t_{pj}$ is the target input for the jth component of the output pattern $o_{pj}$
2. $i_{pi}$ is thevalue of the ith element of the input pattern
3. $(t_{pj} - o_{pj})$ is $\delta_{pj}$
4. $\Delta_p w_{ji}$ is the change to make to the weight from in pattern $p$

The proposed backpropagation algorithm uses gradient decent to minimize of a loss function, in this case the square errors of a linear fit.

$$E_p = \frac{1}{2}\sum_j (t_{pj} - O_{pj})^2$$

We fit the model by following the steepest descent direction given by the derivative of the error by the weights:

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} \textbf{ by the chain rule equals } -\frac{\partial E_p}{\partial o_{pj}}\frac{\partial o_{pj}}{\partial net_{pj}}$$

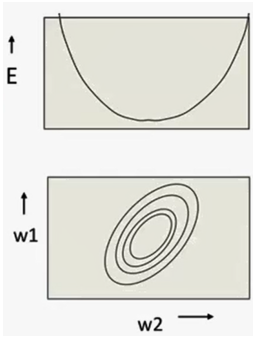


Figure 1: Error gradient as a function of the weights forming a 3D *bowl* [3]

By back propagating the deltas from the output layers to the input layers recursively we obtain a general rule for calculating all the deltas using the previous iteration weights and previous deltas.

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \sum_{\mu} \frac{\partial E}{\partial V_j^{\mu}} \frac{\partial V_j^{\mu}}{\partial w_{jk}} = -\eta \sum_{\mu i} [t_i^{\mu} - O_i^{\mu}] g'(h_i^{\mu}) W_{ij} g'(h_j^{\mu}) i_k^{\mu} = -\eta \sum_{\mu i} \delta_i^{\mu} W_{ij} g'(h_j^{\mu}) i_k^{\mu}$$

This boils down to an equation of the same form of the first equation:

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}$$

Thus, we have 2 phases in the generalized delta rule:

1. The input is presented and propagated forward to compute the output value o for each unit, thus the name feed forward.
2. Compare output with the targets and compute the error signal for each output unit.
3. Backward pass the error signal to each unit to change the weights, recursively.

Further remarks:

1. Arbitrary weights can be fixed.
2. There is no reason for avoid fully interconnectivity of previous layers.

## Simulation results

Local maxima was initially regarded a problem. Empirical evidence points out that saddle point usually have the same gradients. To test the algorithm, the authors used the non-linear logistic function which can take advantage of hidden layers and provides stability. 0.1 and 0.9 are usually the targets because the function is asymptotic. A momentum term can be added to the delta rule to increase the learning rate without oscillations. Weights need to be initialized randomly to break symmetry problems.

1. **The XOR problem.** With two hidden units it is possible to get stuck in local minima. Each pattern was trained with 6587 samples and it was found that more samples only increased the magnitude of the weights but not the performance. Hyperparameter and time to solution experiments were done by Ives Chauvin.
2. **Parity.** If the input contains an odd number of 1s it should output one, zero otherwise. It is a general form of the XOR problem. The same number of hidden units as inputs are required. The solution required 2825 samples for each of 16 patterns with a $\eta = 0.5$.
3. **The encoding problem.** $N$ orthogonal input patterns are mapped to $N$ orthogonal output patterns through $\log_2 N$ units. The hidden units learns to encode the input through binary values. Distributed representations can fail to convert values into local representation when the precision is not enough in higher layers. There is a trick of mapping patterns into activation values and then converting them back for any number of patterns. The sigmoid pushes extreme values on the hidden units and linear activation functions can be applied selectively to those units.

4. **Symmetry.** If an input string is symmetric around its center outputs 1, otherwise 0. The problem can always be solved with two hidden units. The patterns on each side of the midpoint send unique activation sum that cancels out. This elegant solution wasn't envisioned previously.
5. **Addition.** Local minima can be found consistently. Hidden units are the carry bits. An exclusive or works on the two lower input bits. This can be used to inhibit or activate the outputs. It requires 2m input units, m hidden units and m+1 output units. Connecting the hidden units between them can lead to a local minima due to the XOR not converging. The learning must be done in order from the lowest order to the highest unit (the hidden units are not equi-potential), one half of the time it fails.
6. **The negation problem.** when one bit of the input is on the network must negate the pattern. It can be views as a composition of XOR's. It was solved in less than 5000 passes with $N = 3$.
7. **The T-C Problem.** For a given grid of inputs, it outputs 1 when a T is detected and 0 when a C is detected. Translation and rotation invariance is expected, a total of 8 patterns. There is only one bit of difference of the 5 squares and distance between pairs is maintain in rotations. Each input unit are connected to a 2D hidden units grid called the receptive field. The receptive fields show the empirical solutions found. Compactness and protudness were found to be useful. The solution was reached after 5000 presentations of each pattern.
8. **Further generalizations.**
    1. **Generalized delta rule for the Sigma-Pi units.** The products of activation values of individual units activate output units.
    2. **Recurrent nets.** Recurrent nets allow hidden units to be connected to themselves. A sequence is presented for some number of iterations. Future states must not affect past ones. Minsky and Papert pointed out that for every recurrent network there is a feed forward network (assuming finite time). The downside is hardware and memory duplication proportional to the lag time. The idea is to keep track of the weight changes, determining it by multiplying the error and the input along the relevant line. The weight can change when it passes also to itself. The backpropagation is repeated the same number of iterations than the forward pass. Memory for recording the activation values and sequence is required for each unit. The network prove useful to learn to be a circular shift register. Negative biases assure successful learning. 200 sweeps for each pattern were sufficient for the network to learn. Other mechanisms have been tested however parity in the number of shifts can affect accuracy. Other uses of the net is to learn to complete sequences, that is complete the state of the output units at a given time. All hidden units are interconnected. The learning required 260 sweeps on the 20 training sequences.

## Conclusions

Linear single-layer perceptrons tantalized interest in multilayer networks. It was shown that the backward error propagation (a generalization of the delta rule) leads to unsupervised solutions in every case and it can be parallelized. The proposed algorithm successfully unsupervisedly learns useful internal representations for many problems with the exception of very specific cases of the *adding* problem that fall on local minima. Generalization capacity must be tested with problem subsets on the training phase.

## References

[1] M. Minsky and S. Papert, *Perceptrons : an introduction to computational geometry.* MIT Press, 1988, p. 292.

[2] D. E. Rumelhart, J. L. McClelland, and S. D. P. R. G. University of California, *Parallel distributed processing : explorations in the microstructure of cognition.* MIT Press, 1986, pp. 318–362.

[3] G. Hinton, "Neural Networks for Machine Learning | Coursera.".