International Journal of Managerial Studies and Research (IJMSR)
Volume 4, Issue 2, February 2016, PP 65-69
ISSN 2349-0330 (Print) & ISSN 2349-0349 (Online)
www.arcjournals.org

Machine Learning Algorithm for Learning Natural Languages

Luminita Pistol[1], Radu Bucea-Manea-Tonis[2], Rocsana Bucea-Manea-Tonis[1,3]

[1]Spiru Haret University, Faculty of Economical Sciences, Str. Fabricii nr.46G, Sector 6, Bucharest, Romania.
[2]Stefanini ADC Romania, Bd. Dimitrie Pompeiu 10 A, Bucureşti, Romania.
[3]University Politehnica of Bucharest, Faculty of Engineering and Management of Technological Systems, Splaiul Independentei nr. 313, sector 6, Bucuresti, Romania.
[1]luminita_pistol@yahoo.com, [2]radub_m@yahoo.com, [3]rocsanamanea.mk@spiuharet.ro

Abstract: *In our paper we implement a lexical formal system linked to a neural network in order to build increasingly precise tokens starting with a randomly generated set of lexemes. After a literature review regarding neural networks, in our paper we described the formal system that is analphabet made of symbols and a set of rules concerning adding/removing lexemes, we described the neural network design, and the learning algorithm of solving the lexical puzzle.*

Keywords: *neural network, lexical formal system, tokens, lexemes*

1. LITERATURE REVIEW

Neural networks are widely used in classifications, because they examine a lot of information and sort it all out very efficient. They are applied in solving diverse problems on medical, linguistic, economic-financial, technical fields, and so on. The risk management can be better mitigated and the investments in innovation could turn more valuable using neural network. Neural networks might be used in character recognition – eg. recognition of handwritten characters, image compression, stock market prediction, security, and loan applications to decide whether or not to grant a loan.

Neural networks were successfully applied in diagnosis classification or for investigation of neural distinctions between inflectional and derivational morphology and their interaction with lexical frequency using the mismatch negativity [Leminen, 2013].

In linguistic analysis neural networks are applied in distinguishing between *micro-grammar* and *macrogrammar*, two cognitive serialization principles that allow speakers to build up a unit of talk in real-time. "*Macrogrammatical* elements serve different cognitive tasks that arise at particular points in the linear production of an utterance, such as fine-tuning epistemic value or modifying illocutionary force". [Haselow, 2016]

In the economic field [Vinodhini, 2016] shows that combining a PNN (Probabilistic Neural Network) and PCA (Principal Component Analysis) is reducing the training time. This result might be explained by the mixing the computational capability and flexibility, by retaining its simplicity.

The structure, the transfer function, and learning algorithm might influence the performance of a neural network. The best chosen structure is essential for a higher performance in classification using neural networks and it depends on the complexity of the relationship between the input and the output. [Siswantoro, 2016]

In this paper we model a neural network that will combine lexems into tokens according to a fixed set of rules in order to simulate the human child ability to learn a natural language.

2. CASE STUDY

2.1. The Formal System

The formal system is made of an alphabet made of symbols and a set of rules concerning adding/removing lexemes in order to reach a desired output, starting from a known input [Hofstadter, 2015]. For simplicity sake, our alphabet will contain only three symbols:

A={**..**;--;**..-**};

The input string of symbols will be **--..** and the output is expected to be **--..-** following the rules:

1. When the last symbol is **..**, we can add **..-**;
2. **–** followed by any symbol can be transformed in **–** followed by two symbols of that kind;
3. A sequence of three **..** symbols can be replaced by the **..-** symbol;
4. A double **..-** symbol can be removed by the original string.

We are not sure if our problem has any solutions at all but we shall find if so by implementing a machine learning algorithm in order to gradually refine our search for composing the final string.

For this purpose we should employ a neural network made of two perceptrons - one for storing the input string and one for the desired output sequence - and a variable number of hidden layers made up by four sigmoidal neurons to match our set of rules.

The output of a sigmoid neuron with inputs x1, x2,…x1, x2,…, weights w1, w2,…w1, w2,…, and bias b is, after [Nielsen,2015]

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

The sigmoid function plot is the following (Fig.1.):

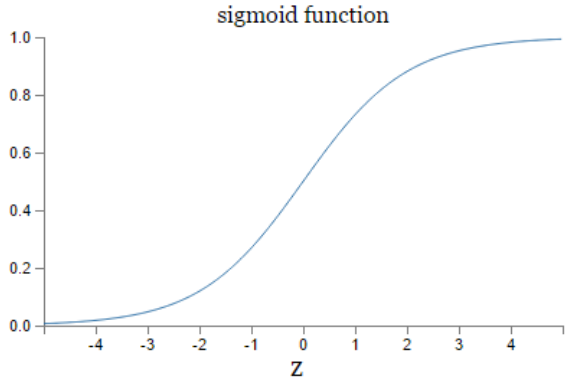

Fig1. *Sigmoid Function*

2.2. Neural Network Design, after [Miller, 2013]

The neural network will contain a topology (ex. 3, 2, 1) of layers and neurons (Fig.2)

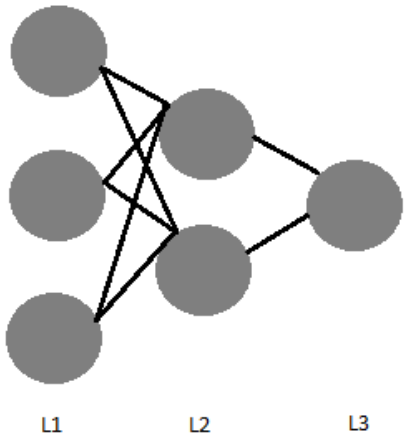

Fig2. *The neural network layers*

This means a network consists of an array of layers (ex. 3), which in they turn consists of arrays of neurons (3, 2, 1). In this, way the topology could be represented as a bi-dimensional array: vector<vector<Neurons>> where the second layer is a hidden one.

Network	*1* *3..**	Layer	*1* *1..**	Neuron

The Network class will have two public methods, one for feed forward information, the other for back propagation the weights adjustments and one constructor that will take the network topology as a parameter:

Network
layers[]
feedForward() backPropagation() Network()

The Layer class will have an array of neurons property and a constructor that take the number of neurons the layer will contain:

Layer
Neurons[]
Layer()

The Neuron class will contain a reference property to the layer it resides in, output weights property and back propagation weight values based on gradient descent delta learning rule:

Neuron
layer outWeights backWeights
Neuron()

Delta rule is based on Least Means Square (LMS) algorithm that adjusts weights when the output vector is compared with the correct answer and the difference is not 0 so as to decrease the cost [Russell]. We shall construct a cost function E that measures how well the network has learned [Orr, 2011]:

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - y_i)^2$$

Where:

n – number of examples;

t_i – desired target value associated with the i^{th} example;

y_i – output of the network when the i^{th} input pattern is presented to network.

The change in weight is given by:

$$\Delta_{w_{ij}} = -r\,\frac{\partial E}{\partial W}$$

where:

r – learning rate;

$\frac{\partial E}{\partial W}$ – derivative for cost function in w_i point.

From relations 1 and 2 results that:

$$\Delta_{w_{ij}} = -r\sum_{i=1}^{n}(t_i - y_i)\,x_i$$

For linear output units, delta is the difference between the desired and computed output values of the n^{th} neuron. We shall initialize the weights with random values, then update them with small values until E is within desired tolerance.

2.3. Learning how to Solve the Lexical Puzzle

We figure out to parse the four rules in two hidden layers each made up by four neurons, in order to reach all the possible permutations of the rules:

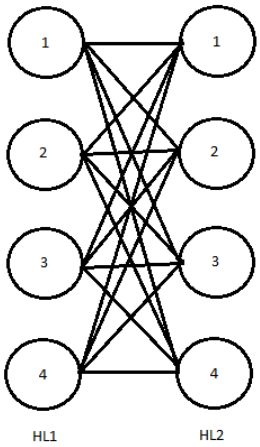

Fig3. *Permutations of the four rules in two hidden layers each made up by four neurons*

For each passing neuron will trigger a method (ex. applyRule) that will add/extract lexemes to/from the original string of symbols. The rule applied will be current neuron order number so there is no need to store such information in the neuron itself:

Neuron
layer outWeights backWeights
applyRule() Neuron()

3. RESULTS AND DISCUSSION

In this paper we tried to apply a machine learning algorithm to a formal system in order to test the machine capability of learning the simple rules of an artificial language. In a future paper, we shall test a stochastic algorithm for the same formal system and compare the results of these two approaches. Only then will be able to choose and implement an appropriate algorithm destined for natural language processing.

ACKNOWLEDGEMENT

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/ 1.5/S/134398.

REFERENCES

[Hofstadter, 2015] Hofstadter D.R., Godel, Escher, Bach: Brilianta Ghirlanda Eterna, Ed. Humanitas, Bucuresti, 2015, ISBN: 978-973-50-4422-0

[Nielsen, 2015] Nielsen M. A., Neural Networking and Deep Learning - Using neural nets to recognize handwritten digits, Determination Press, 2015, http://neuralnetworksanddeeplearning.com/chap1.html

[Miller, 2013] Miller D., Buiding a Neural Net Simulator in C++, Video tutorial, https://www.youtube.com/watch?v=KkwX7FkLfug

[Russell] Russell I., The Delta Rule, Collegiate Microcomputer Journal, University of Hartford, http://uhavax.hartford.edu/compsci/neural-networks-delta-rule.html

[Orr, 2011] Orr G. J. B., The Delta Rule, Willamette University, https://www.willamette.edu/~gorr/classes/cs449/Classification/delta.html

[Siswantoro, 2016] Siswantoro J., Prabuwono A. S., Abdullah A., Idrus B., A linear model based on Kalman filter for improving neural network classification performance, Expert Systems with Applications, Volume 49, 1 May 2016, Pages 112-122

[Leminen, 2013] Leminen A., Leminen M., Kujala T., Shtyrov Y., Neural dynamics of inflectional and derivational morphology processing in the human brain, Cortex, Volume 49, Issue 10, November–December 2013, Pages 2758-2771, ISSN 0010-9452,

[Haselow, 2016] Haselow A., A processual view on grammar: macrogrammar and the final field in spoken syntax, Language Sciences, In Press, Corrected Proof, Available online 18 January 2016, (http://www.sciencedirect.com/science/article/pii/S0893608015002609

[Vinodhini, 2016] Vinodhini G., Chandrasekaran R.M., A comparative performance evaluation of neural network based approach for sentiment classification of online reviews, Journal of King Saud University - Computer and Information Sciences, Volume 28, Issue 1, January 2016, Pages 2-12, ISSN 1319-1578, http://dx.doi.org/10.1016/j.jksuci.2014.03.024.