TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 60, Issue II June, 2017

GENERALIZED DELTA RULE WITH ENTROPY ERROR FUNCTION

Javier BILBAO, Imanol BILBAO, Cristina FENISER

Abstract: *Machine Learning can refer to a different and various algorithms, based on artificial intelligence, that are able to recognize data patterns through continuous and repeated learning techniques, and we do not have to assume any prior data distribution. In this way, artificial neural networks can provide an effective technology and methodology. Artificial neural networks need a rule, delta rule, in order to learn from the supplied data. In this paper a generalized delta rule with a new error function is presented.*
Key words: *neural networks, delta rule, error function, cost function.*

1. INTRODUCTION

Artificial Neural Networks have proven to be an effective technology in performing tasks in which the human brain provides good results. In addition, thanks to their parallel architecture, they are able to quickly handle the immense amount of data and variables that, today, are available thanks to the wide deployment of mobile technologies and, in general, the so-called Big Data. What's more, their capacity for adaptation and evolution is also desirable in systems that evolve and develop changing over time.

Feedforward Artificial Neural Networks using classical generalized delta rule have shown ability to classify the data in previously defined clusters, represent the information internally in a way that extracts de data's features and compresses it, and finally, generalize results to patterns not presented before.

The power of the idea of the generalized delta rule is that even the connections of the network's internal neurons can adapt to minimize the error between the target and the result of the output neurons. It allows somehow hidden layers' neurons to create an adequate representation of the information.

2. CONSIDERATIONS ABOUT ERROR FUNCTIONS

For a training data set $(\vec{x},\vec{t})_p$, where $\vec{t}$ is the desired response associated to the $\vec{x}$ network entry data of the pattern p, all the suggested error functions are accumulative and independent respect to their terms, this is, the total error will be a sum of the individual errors. Therefore, $E_{total}=\sum_{\forall p} E_p$, the total error is the sum of the error made with each pattern.

We will understand total error as a function of the network weights w_{ij}, and we will try to locate its minimum value using the gradient descent method.

The fact that $E_{total}=\sum_{\forall p} E_p$ allows us to write also the partial derivatives as an accumulation for each pattern:

$$\frac{\partial E_{total}}{\partial w_{ji}}=\frac{\partial \sum_{\forall p} E_p}{\partial w_{ji}}=\sum_{\forall p}\frac{\partial E_p}{\partial w_{ji}} \qquad (1)$$

So, following the gradient descent rule:

$$\Delta w_{ji} = -\frac{\partial E_{total}}{\partial w_{ji}} = -\sum_{\forall p} \frac{\partial E_p}{\partial w_{ji}} \qquad (2)$$

If we name $\Delta_p w_{ji}$ something that is proportional to each $-\frac{\partial E_p}{\partial w_{ji}}$ we are able to write $\Delta w_{ji} = \sum_{\forall p} \Delta_p w_{ji}$.

3. CLASSICAL GENERALIZED DELTA RULE

3.1 Used error function

The error function used in [1] is the usual sum-squared error function calculated through all the output layer's neurons:

$$E_p = \frac{1}{2} \sum_{\forall j} \left(t_{pj} - o_{pj}\right)^2 \qquad (3)$$

t_{pj} is the j-th component for the desired output $\vec{t}_p$. Therefore, it is the desired output for the j-th output neuron when the pattern p is applied.

o_{pj} is the output obtained in the output neuron j when $\vec{x}_p$ is presented to the network input layer.

3.2 Initial definitions and used output function

In a purely feedforward network architecture the i layer's neurons only are connected to the i+1 layer's neurons. The input layer receives the $\vec{x}_p$ data and transforms it in network signals. The output layer transmits its signals outside the network. The hidden layers are responsible of finding an appropriate representation of the info to achieve the desired result.

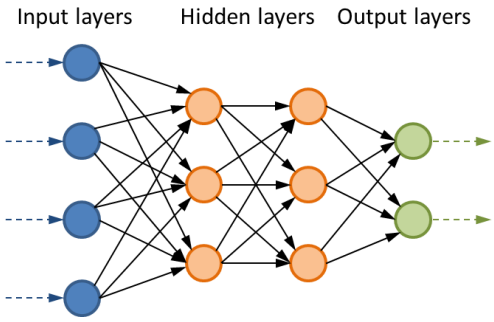

Fig. 1. Feedforward architecture.

In a feedforward architecture each neuron of the network receives signals from the neurons of the previous layer. The effect of these signals is combined in the neuron net total input:

$$net_{pj} = \sum_{\forall i} w_{ji} o_{pi} \qquad (4)$$

Here the net total input is the linear combination of the previous layer's neurons output with the corresponding connections weights. The subindex i indicates each neuron of that previous layer and w_{ji} is the weight of the connection from the neuron i to the neuron j that's being analyzed.

As usual in this kind of models, all the neuron's activation thresholds are θ_j treated as a weight of a fictitious connection of input 1. In this sense, $\theta_j = w_{j0}$.

The mathematical development of [1] takes into account that all the neurons in the network have their own generic output function that relates the output of the neuron with the net total input. Therefore,

$$o_{pj} = f_j\left(net_{pj}\right) \qquad (5)$$

The only condition they must fulfill is to be non-decreasing and differentiable.

3.3 Classical generalized delta rule summary

The modification in the weights that should be done after a pattern p is presented is

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \qquad (6)$$

The δ_{pj} for the neurons of the output layer is

$$\delta_{pj} = \left(t_{pj} - o_{pj}\right) f'_j\left(net_{pj}\right) \quad (7)$$

For the hidden layers' neurons:

$$\delta_{pj} = f'_j\left(net_{pj}\right) \sum_{\forall k} \delta_{pk} w_{kj} \quad (8)$$

Where k are all the neurons which the neuron j is connected to.

The calculations are divided into two phases. The first is a feedforward propagation of the entry data $\vec{x}_p$ through the network, from the input layer towards output layer. The objective is to generate each neuron's net total input and each neuron's output for the known weights. Once net_{pj} and o_{pj} have been calculated for all the neurons the second back propagation phase can be done. Starting in the last layer, with the known $\vec{t}_p$ for the used data entry, we calculate each layers neurons' δ_{pj}. Therefore, we calculate δ_{pj} for all the neurons of the output layer, and with all them together we continue calculating the previous layer neurons' δ_{pj} just as a linear combination of them. The process continues recursively layer by layer until the input layer is reached.

3.4 Classical generalized delta rule development

Following the general steps provided by [1] we will calculate $-\frac{\partial E_p}{\partial w_{ji}}$ for each w_{ji} and try to write it as the product $\delta_{pj} o_{pi}$. Thus, we can write:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \quad (9)$$

Where net_{pj} is the net total input of the neuron j where the connection w_{ji} finishes.

From the equation (4) it is easy to follow that

$$\frac{\partial net_{pj}}{\partial w_{ji}} = o_{pi} \quad (10)$$

Cause all the other weights act as constants for ∂w_{ji}.

To compute the other term of (9),

$$-\delta_{pj} = \frac{\partial E_p}{\partial net_{pj}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \quad (11)$$

Where o_{pj} is the output of the neuron where the connection w_{ji} finishes.

From the equation (5) it follows immediately that

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f'_j\left(net_{pj}\right) \quad (12)$$

The first term is more complicated to evaluate. If we suppose that j is a neuron from the output layer, then

$$-\frac{\partial E_p}{\partial o_{pj}} = -\frac{\partial}{\partial o_{pj}} \frac{1}{2} \sum_{\forall k} \left(t_{pk} - o_{pk}\right)^2 \quad (13)$$

$$-\frac{\partial E_p}{\partial o_{pj}} = t_{pj} - o_{pj} \quad (14)$$

Cause all the other o_{pk} distinct from o_{pj} are unrelated and act as constants.

If the neuron j is a neuron from the previous layers, we should follow the chain rule, just as indicated in [1]:

$$-\frac{\partial E_p}{\partial o_{pj}} = -\sum_{\forall k} \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} \quad (15)$$

$$-\frac{\partial E_p}{\partial o_{pj}} = -\sum_{\forall k} \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_{\forall i} w_{ji} o_{pi} \quad (16)$$

$$-\frac{\partial E_p}{\partial o_{pj}} = \sum_{\forall k} -\frac{\partial E_p}{\partial net_{pk}} w_{kj} = \sum_{\forall k} \delta_{pk} w_{kj} \quad (17)$$

4. GENERALIZED DELTA RULE WITH ENTROPY ERROR FUNCTION

4.1 Desired values and possible output values

As in the case of the logistic regression explained in [2] there are some cases, especially for digital data of input and output, where the provided targets t_{pj} are always 0 or 1

and the desired outputs of the network should be in the same range to be interpreted. In classification networks, for example, each output neuron could reflect the probability of a data to belong to a determined predefined class being desirable this property.

We can assure, in accordance with the digital nature of the data collected, that all the target values will be 0 or 1.

On the other hand, we will interpret the output values of each output neuron as the probability of that output to be a 1. To be so it's neurons output range should be $[0,1]$. We will later choose an appropriate output function that fulfils this condition.

4.2 Used error function

The error function is that used in [2] for the logistic regression. For a given pattern the error on each neuron of the output layer is:

$$E_p(t_{pj}, o_{pj}) = \begin{cases} -\ln(o_{pj}) & t_{pj} = 1 \\ -\ln(1-o_{pj}) & t_{pj} = 0 \end{cases} \quad (18)$$

This function is a known function in the statistical field making use of the maximum likelihood principle. It's a concave function where the errors are nearly 0 if the estimation is correct, or near correct, and where the errors tend to infinity if the estimations are incorrect. It takes intermediate values for values that are in an uncertain probability.

If the desired output is $t_{pj} = 1$ and the actual output is near 1 the error would be very small. As o_{pj} tends to 0 the error would increase slowly, and for values of o_{pj} near to 0 the error would be very big (it tends to infinity as the output tends to 0).

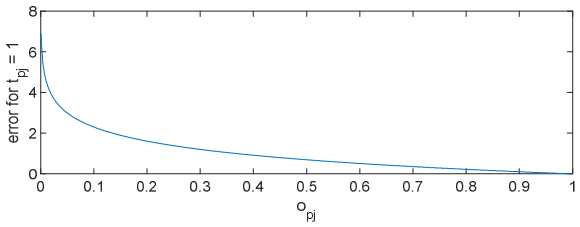

Fig. 2. Error for t_{pj}=1.

If the desired output is $t_{pj} = 0$ and the actual output is near 0 the error would be very small. As o_{pj} tends to 1 the error would increase slowly, and for values of o_{pj} near to 1 the error would be very big (it tends to infinity as the output tends to 1).

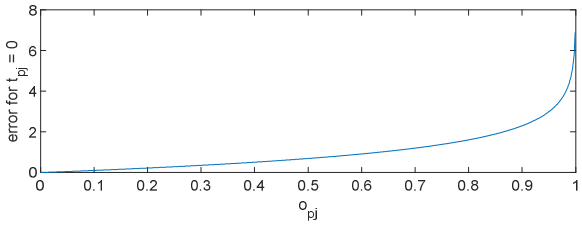

Fig. 3. Error for t_{pj}=0.

Knowing that the unique values that t_{pj} can have are 0 or 1, the error for the pattern among all the neurons in the output layer may be rewritten as:

$$E_p = -\sum_j t_{pj} \ln(o_{pj}) + (1 - t_{pj}) \ln(1 - o_{pj}) \quad (19)$$

4.3 Used output function

The used output function is the sigmoidal function. It's defined by:

$$o_{pj} = f_j(net_{pj}) = \frac{1}{1 + e^{-net_{pj}}} \quad (20)$$

It's a non-decreasing and differentiable function in the range $[0,1]$ that may be interpreted as the probability of the neuron j to be activated. For low values of net_{pj} it produces outputs that are almost 0. For high values of net_{pj} it results in values nearly 1. For values near to the threshold of activation of the neuron θ_j it produces a progressive change.

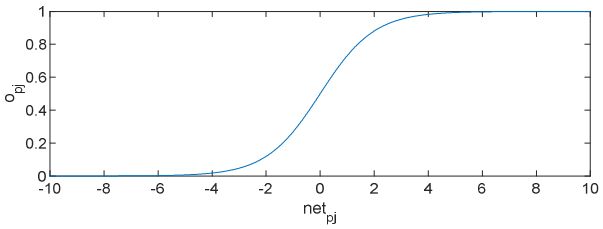

Fig. 4. Sigmoidal output function

It's derivative can be written as:

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f'_j\left(net_{pj}\right) = \frac{e^{-net_{pj}}}{\left(1+e^{-net_{pj}}\right)^2} \quad (21)$$

$$f'_j\left(net_{pj}\right) = \frac{1}{1+e^{-net_{pj}}}\frac{e^{-net_{pj}}}{1+e^{-net_{pj}}} \quad (22)$$

$$f'_j\left(net_{pj}\right) = \frac{1}{1+e^{-net_{pj}}}\frac{1-1+e^{-net_{pj}}}{1+e^{-net_{pj}}} \quad (23)$$

$$f'_j\left(net_{pj}\right) = o_{pj}\left(\frac{1+e^{-net_{pj}}}{1+e^{-net_{pj}}} - \frac{1}{1+e^{-net_{pj}}}\right) \quad (24)$$

$$f'_j\left(net_{pj}\right) = o_{pj}\left(1 - \frac{1}{1+e^{-net_{pj}}}\right) \quad (25)$$

$$f'_j\left(net_{pj}\right) = o_{pj}\left(1-o_{pj}\right) \quad (26)$$

Its derivative is nearly 0 if the output rounds 0 or 1, and has it maximum if net_{pj} is around the threshold value.

4.4 Generalized delta rule

We will follow the same steps we have done in the case of the classical generalized delta rule of the section 3.4. We start applying the chain rule introducing the net_{pj} of the neuron j where the connection w_{ji} finishes:

$$-\frac{\partial E_p}{\partial w_{ji}} = -\frac{\partial E_p}{\partial net_{pj}}\frac{\partial net_{pj}}{\partial w_{ji}} \quad (27)$$

The second term is identical to that of equation (10), so we still can write that

$$\frac{\partial net_{pj}}{\partial w_{ji}} = o_{pi} \quad (28)$$

Just as we did before in (11) and (12) for the recurrent term δ_{pj}, we begin with its definition and apply the chain rule again, introducing o_{pj}, the output of the neuron where the connection w_{ji} finishes:

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}}\frac{\partial o_{pj}}{\partial net_{pj}} \quad (29)$$

$$\delta_{pj} = -\frac{\partial E_p}{\partial o_{pj}} f'_j\left(net_{pj}\right) \quad (30)$$

Fort the output layer neurons the first term of the derivative is a direct application of the partial derivative. It is analogous to that done in (13).

$$-\frac{\partial E_p}{\partial o_{pj}} = \frac{\partial}{\partial o_{pj}}\sum_k t_{pk}\ln\left(o_{pk}\right) + \left(1-t_{pk}\right)\ln\left(1-o_{pk}\right) \quad (31)$$

k is the set of neurons of the output layer. The output layer doesn't have connections between neurons of the same layer, thus, we know that each o_{pk} distinct from o_{pj} is unrelated to ∂o_{pj} and that acts as a constant in the derivative. The sum can then be eliminated and write simpler:

$$-\frac{\partial E_p}{\partial o_{pj}} = t_{pj}\frac{\partial \ln\left(o_{pj}\right)}{\partial o_{pj}} + \left(1-t_{pj}\right)\frac{\partial \ln\left(1-t_{pj}\right)}{\partial o_{pj}} \quad (32)$$

$$-\frac{\partial E_p}{\partial o_{pj}} = \frac{t_{pj}}{o_{pj}} - \frac{1-t_{pj}}{1-o_{pj}} = \frac{t_{pj}-o_{pj}}{o_{pj}\left(1-o_{pj}\right)} \quad (33)$$

When both equations (30) and (33) match together the expression obtained is:

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = t_{pj} - o_{pj} \quad (34)$$

It is more simple than that of the classical generalized delta rule in (7) because we have chosen the error function in a way that it's derivative simplifies with the output function derivative.

For the weights connected to the previous layers' neurons the procedure is the same we have followed with the classical generalized delta rule in (15), (16) and (17). As the idea is exactly the same they remain unchanged and we still can summarize that for hidden neurons:

$$\delta_{pj} = o_{pj}\left(1-o_{pj}\right)\sum_{\forall k}\delta_{pk}w_{kj} \quad (35)$$

5. CONCLUSION

Artificial neural networks have taken a new impulse due to different reasons, but mainly because the prominence of Big Data and Machine Learning in the last time, and because their capacity for adaptation and evolution in systems that evolve and develop changing over time. Delta rule is the core of the systems that use artificial neural networks, and although the practical totality of their users take software or libraries of software to use them, it is necessary a mathematical development to create these software.

A generalized delta rule is presented with a new error function. It maintains the same idea of the original rule but with fewer calculations. It's results can be summarized in three main equations.

The modification in the weights that should be done after a pattern p is presented is

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \qquad (36)$$

The δ_{pj} for the neurons of the output layer is

$$\delta_{pj} = \left(t_{pj} - o_{pj}\right) \qquad (37)$$

For the hidden layers' neurons:

$$\delta_{pj} = o_{pj}\left(1 - o_{pj}\right)\sum_{\forall k} \delta_{pk} w_{kj} \qquad (38)$$

6. REFERENCES

[1] Rumelhart, D.E., McClelland, J.L., *Paralell Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1*, Cambridge, MA: MIT Press, ISBN 0-262-18120-7, 1986.

[2] Ng, A. *CS229 Lecture notes. Machine Learning. Supervised Learning, Discriminative Algorithms* http://cs229.stanford.edu/notes/cs229-notes1.pdf

REGULĂ DELTA GENERALIZATĂ CU FUNCȚIE DE EROARE BAZATĂ PE ENTROPIE

Rezumat: Învățarea automată se poate referi la algoritmi diferiți și variați, bazați pe inteligență artificială, capabili să recunoască modele de date prin tehnici de învățare continuă și repetată, fără a fi nevoie de o distribuție de date anterioară. Astfel, rețelele neuronale artificiale pot contribui la asigurarea unor tehnologii și metodologii eficiente. Rețelele neuronale artificiale au nevoie de o regulă, o regulă Delta, pentru a învăța din datele furnizate. În această lucrare este prezentată o regulă Delta generalizată cu o nouă funcție de eroare.

Javier BILBAO, Professor, University of the Basque Country, Engineering School of Bilbao, Applied Mathematics Department, javier.bilbao@ehu.eus, 0034 94 601 4151.

Imanol BILBAO, PhD Student, University of the Basque Country, Engineering School of Bilbao, Electrical Engineering Department, imanol.bilbao@ehu.eus, 0034 94 601 4971, C/ Belostikale 4, 1, 0034 94 415 5896.

Cristina FENISER, Lecturer, Technical University of Cluj Napoca (Romania), Faculty of Machine Building, Department of Management and Economic Engineering, Cristina.FENISER@mis.utcluj.ro, 0040 752 105 451