Determining the optimal percent of negative examples used in training the multilayer perceptron neural networks

COSMIN CERNAZANU-GLAVAN, STEFAN HOLBAN
Computer Science Department
Politehnica University of Timisoara
Bd. Vasile Parvan, Nr. 2, Cod 300223
ROMANIA
cosmin.cernazanu@ac.upt.ro, stefan.holban@ac.upt.ro

Abstract: - Neural networks prove their usefulness mainly in solving difficult problems of estimation, identification, prediction or complex optimization. For finding a correct solution for the problem, the network must be trained with a comprehensive dataset, able to completely cover all the possible situations that may appear when using it. Most often, the training set includes only positive examples which may change the neural network into an over confident network in solving the problems. A simple solution for this problem is the introduction of negative examples in the training set. Through this procedure, the network will be prepared for the cases it has not been trained for. The present article aims at finding the exact percentage of negative examples that can successfully be used in training any multilayer perceptron neural network. For this purpose, two databases completely different in content and uniformity between examples have been chosen. There have been made successive trainings for exactly determining the percentage. Moreover, there has been suggested a new calculus formula for calculating a network performance based on its impact on a complex production set.

Key-Words: - Neural networks, Training problems, Negative examples, Production set.

1 Neural Networks

The structure of the human brain consists of approximately 10^{11} neurons (nerve cells) interconnected through 10^{14} synapses [1]. For this reason, it is difficult to imagine an artificial device able to imitate completely this natural complexity.
Although the working speed of a biological neuron is highly reduced as compared to that of a computer, on account of the large number of neurons and connexions, the human brain is highly superiour to any super computer used nowadays.
Starting from the biological neural networks, people tried to use and imitate its features and structure in order to create a similar artificial model.
This attempt resulted in developping the Artificial neural networks [2][3], special devices projected to adapt the activity of the biological neural networks to a certain detail level in order to reach (several) human brain characteristics.
However, the main difference between the biological neural networks and computers consists in their structure. On the one hand, computers are made up of a high-speed main processor that performs the calculations, each component of the computer having a precise function. On the other hand, the human brain is made up of over 100 different types of special cells (neurons), their estimated number being between 50 billions and 100 billions (10^{11})[4].
An artificial neural network is a model that emulates a biological neural network. The nods from a neural network are based on a simplistic mathematic representation resembling to the real neurons.
The simulation of the cerebral action is eloquent for its two main actions:

- Knowledge is stored through a learning process
- Knowledge storage is performed by using the inter-neural connexions value, named synapse weights.

2 Training neural network

The neural networks essential utility consists in solving some difficult problems such as: estimation, identification and prediction or complex optimization. Their feasibility plan is very wide. They accompany us day-by-day either inbuilt in household appliances (cell phones, washing machines, TV sets, microwave ovens, etc.) or by interacting with them in our daily life (form recognition, speech recognition, automatic diagnosis, etc.).[5]
A network can be implemented by following several stages. In the first stage, we will select a certain type of neural network (feed forward, recursive,

Kohonen,etc.) and we will define its configuration (number of layers, number of neurons, activating function, etc.). In the second stage, after selecting the network for a certain application, we will start the neural network training process. In the third stage, the neural network will be tested on a broad dataset and, depending on the result obtained, the process will be either restarted from stage 1, or the network will be successfully used in the future.
The neural network *classic* training implies the use of a number of examples in the training set and their random introduction at the network input until the error will decrease under a certain threshold. (Figure 1)

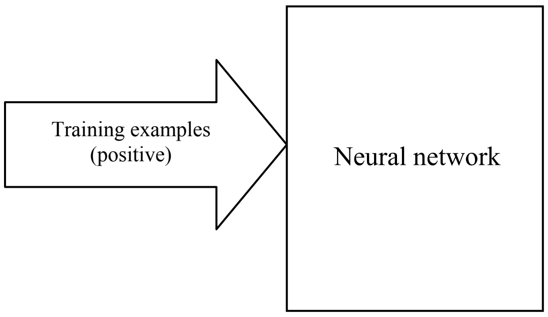

Figure 1. Classic training

After the training stage, an example that has not been previously used in the training will be introduced at the network input. If this example is similar to those previously used, it will successfully be recognized by the network. If an example totally different from those present in the training set is introduced at the network input, then the network will classify it as belonging to one of the categories of examples it has been trained with. In this case, we can define it as an over confident network. This type of errors is frequent in the classification problems that have an insufficient training set. [6]
For simply solving this type of problems, it is necessary to introduce some negative examples in the training set – examples that should not be recognized by the network as belonging to some of the already known categories. For example, if a network is trained to recognize figures, it is recommendable that examples of other characters to be introduced in the training set (letters, punctuation marks, other signs). (Figure 2)
In this case, the network is trained to recognize both examples from certain categories and examples that do not belong to the desired categories.

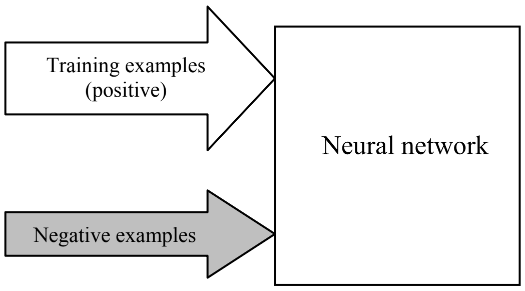

Figure 2. Training a neural network with positive and negative examples

There have been cases when the use of negative examples visibly overran the number of positive examples. Therefore, the importance of the negative examples compared to the positive ones was decreased by a certain value. [7] In this way, a penalty of negative example has been introduced, by which, each negative example is able to influence the result only through a fraction depending on the total number of negative examples.
Unfortunately, there are no clear specifications about the percentage of negative examples (out of the total number of examples) that must be used in training a neural network. (Figure 3) All the studies performed until the present moment are specific for a certain type of problem and the percentage of negative examples used in different applications vary between 5% and 80%. [7][8][9]
The value of the percentage is very important since its variation can significantly modify the recognition percentage of the neural network. For a training set with many negative examples, we will obtain low recognition rates for the desired patterns and high recognition rates for the undesired patterns.

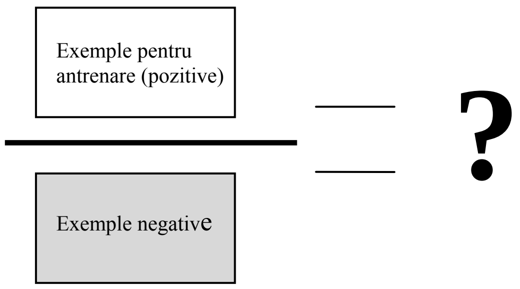

Figure 3. Which is the optimum percentage of negative examples that must be used for best training a neural network?

A low number of negative examples in the training set cannot modify the network weights so that it could successfully recognize the undesired patterns.
In order to solve the problem, we will start by detailing the problem of learning through examples which represents the basis of the artificial neural network supervised training.
The process of learning from examples applied to a neural network can be viewed as a problem of finding a function f modeled with a hypothetical class *H* and a training set of the type *D={ (x_i,y_i) , i = 1,..m}* where $x_i \in R^n$ is the n-dimensional example used for training, and y_i is the network output value.
The error resulted from this pattern-making process will have the following form:

$$J = Er_{aproximare}(n) + Er_{estimata}\left(\frac{VC(H_n)}{m}\right) \quad (1)$$

, where $VC(H_n)$ represent a measure of complexity for the hypothetical class. For minimizing this function, we must minimize the 2 components: the approximated error and the estimated error.
The problem that arises is that the approximation error decreases proportionally to the increase of the pattern complexity, and the estimation error decreases proportionally to the reduction of the pattern complexity.
For this reason, a compromise of complexity must be found for these types of problems in order to obtain a minimum value of the error.
For the problem of learning from examples, the pattern of the hypothetical class will have the following form: [10]

$$H[f] = \sum_{i=1}^{m}\left(f(x_i) - y_i\right)^2 + \lambda\|Pf\|^2 \quad (2)$$

, where λ is a positive regularization parameter, P is the differential operator and $\|Pf\|^2$ is a cost function that rapidly reduces the interval of possible solutions for any type of priority information.
According to the regularization theorem, the solution for this problem will have the following form:

$$f(x) = \sum_{i=1}^{m} a_i K(x, x_i) + b \quad (3)$$

, where α_i represent the kernel function coefficients and b represents the bias function values.
When creating a number of negative examples of the type $D'=\{ (x_i',y_i') , i = 1,..m\}$, the formula (2) will have the form [11]:

$$H[f] = \sum_{i=1}^{m}\left(f(x_i) - y_i\right)^2 - \sum_{i=1}^{m}\left(f\left(x_i'\right) - y_i'^{0}\right)^2 + \lambda\|Pf\|^2 \quad (4)$$

, where the second term corresponds to the negative examples and the final solution will have the form:

$$f(x) = \sum_{i=1}^{m} a_i K(x, x_i) + \sum_{i=1}^{m} a_i' K\left(x, x_i'\right) + b \quad (5)$$

, where α_i' represents the values of the negative functions.
The solution of the problem consists in finding all the values of the formula (5). In this way, the function by which the neural network is approximated will be clearly defined.
Starting from these theoretical grounds, we will experimentally try to find a calculus formula for the number of negative examples that will form the training set.
For each problem, we need a different number of negative examples. We will try to group these numbers and to find a formula able to interpolate these values.
The use of the negative examples in training the neural networks improves their recognition rate. Moreover, the negative examples are easier to built (virtually speaking all the other examples that are not positive will be negative) and there can be found in great number.

3 Experiments

For finding the accurate percentage of negative examples used for training, several experiments with two large databases have been performed. The first dataset used for training was NIST 19 – an international database comprising more than 800.000 letters written by different persons. [12] This database was especially chosen, as letter recognition with neural networks is an active research field, thoroughly explored and studied nowadays. In addition, we must say that the training of a neural network for letters recognition requires both a well-trained network and a large internal structure.
The second dataset used in the experiment was provided from a coast observatory. This database was used in a broad, important governmental program for monitoring and measuring various parameters of the ocean. [13] Data were collected every 20 minutes and then processed, for providing statistical values on the Martha's Vineyard Observatory web page.

With a view to obtaining an exact number of negative examples, large amounts of training datasets have been used for each experiment.

3.1 NIST experiments

Two experiments have been performed on this database for the purpose of finding the percent of negative examples. In both experiments, there have been made efforts to find the exact value of the desired percentage, the only difference consisting in the value of the increase step. Therefore, in experiment 1, we have used a training set to which large percentages of negative examples have been added at every step. The final result consisted in a training set containing more negative than positive examples.

In experiment 2 we used data from Experiment 1 and we have tried to apply a higher degree of digitization. Although the same number of trainings has been performed, the entire application interval for the second experiment consisted only in 2 digitization steps as compared to Experiment 1 (the increase step for the second experiment was 10 times smaller than the one used in example 1).

Experiments 1 and 2 consisted in training a neural network to recognize the letters “a”, “e”, “n” and “r”. For this purpose, we have used a multilayer perceptron neural network with one hidden layer. The network had the following configuration:

- The input layer with 1024 neurons – as inputs we have used 32x32 pixels drawings of the respective letters
- The hidden layer with 700 neurons
- The output layer with 4 neurons – each neuron being responsible for one letter recognition

The activating functions corresponding to each of the 3 layers are:

- The activating function for the input layer is the linear function $f(x) = x$.
- The activating function for the hidden layer is the logistic function
- The activating function for the output layer is the logistic function

3.1.1 Experiment 1 details

We started the training with a set of 4 letters (a, e, n, and r) with 5000 instances each 1000 new letters (mainly the letters b, d, h, l and t) – used as negative examples have been gradually added to each new training. For each network training, the testing procedure was performed on a set of 26.000 letters; consequently, each letter of the alphabet had 1000 instances.

The results obtained in the experiment are shown in Table 1.

Training no.	Negative examples %	Letter„a” %	Letter „e” %	Letter „n” %	Letter „r” %	The alphabet %	Production set %
1	0	91.00	94.40	96.30	96.10	31.95	78.94
2	4.76	90.80	92.60	95.20	95.80	54.23	82.06
3	9.09	91.30	91.90	95.30	95.20	58.90	82.96
4	13.05	90.30	92.80	94.80	95.50	62.31	83.33
5	16.67	88.40	91.20	94.30	95.20	63.19	82.68
6	20.00	88.70	90.80	94.20	95.00	66.19	82.76
7	23.08	88.60	89.50	93.70	95.00	67.34	82.54
8	25.93	88.60	90.80	93.10	94.90	69.00	82.81
9	28.58	89.20	90.20	93.00	94.70	68.42	82.79
10	31.04	88.10	88.90	92.60	94.50	72.03	82.28
11	33.33	89.20	88.70	93.30	94.00	70.12	82.50
12	35.49	87.30	88.80	92.90	94.70	70.53	82.23
13	37.50	88.70	88.50	92.40	93.70	71.17	82.20
14	39.40	88.20	87.90	92.40	93.40	72.58	81.95
15	41.18	87.80	88.60	92.20	93.60	72.53	82.02
16	42.86	88.10	89.20	92.20	93.70	73.34	82.24
17	44.44	89.20	89.30	91.20	94.40	71.07	82.38
18	45.95	88.00	87.30	90.30	93.90	74.07	81.53
19	47.37	87.60	87.70	91.00	92.90	74.43	81.54
20	48.72	86.90	88.30	91.40	92.90	75.21	81.60
21	50.00	88.10	88.70	91.40	93.10	75.44	81.98
22	51.22	86.70	86.80	90.50	92.60	75.20	81.04
23	52.39	86.80	88.10	90.50	92.60	75.56	81.32

Table 1. The results obtained on NIST 19 database in experiment 1

A small decrease in the learning rate for the letters „a”, „e”, „n” and „r” can be noticed when introducing the negative examples in the training set. This is quite normal and natural as the neural network had not been previously trained with negative examples and it was searching to include any letter from the input into one of the classes. As only few letters are “left out”, the success rate is very high. This approach is somewhat similar to the idea of training a neural network to recognize one single letter. The training and testing sets should include only that letter. This would lead to a recognition rate of that letter of approximately 100%.

In this case, problems may arise when in the testing set there are letters different from those which the network has been trained with. We do not refer here to different types of writing a letter, but to different classes of letters. As shown in the table, the neural network that has been trained only with the letters „a”, „e”, „n” and „r” is facing great difficulties when at the input may appear other classes of letters, too. Although the network provides recognition rates between 91% and 96.3% for the letters „a”, „e”, „n” and „r”, it will only provide a rate of 31.95% for the total set of letters in the alphabet. In conclusion we must add that many of the letters unknown for the network have been “recognized” as belonging to the already known classes.

The situation radically changes when in the training set are introduced negative examples, classes of letters that must be recognized by the network as not belonging to the classes „a”, „e”, „n” and „r”. Moreover, for a testing set where we have included all the letters of the alphabet, the recognition rate will increase proportionally to the number of negative examples already known by the network.

In addition, we can notice a decrease in the recognition percentages of the main letters („a”, „e”, „n” and „r”), decrease variable between 4 and 6 percents.

Consequently, we are in a situation in which the increase of a variable leads to a decrease of another variable. It is thus necessary to create a formula that could best reflect a real situation.

For finding the optimum percent of negative examples, we have designed a production set where the number of negative examples is a fraction equal to each class that must be recognized by the neural network.

For example, if we consider example 1, we have 4 classes to be recognized by the network and as a result, the percentage of negative examples must be equal to the percentage of positive examples for each of the 4 classes. In this case, each of the 4 classes and the total number of negative examples will hold 1/5 of the training set.

The general arithmetic formula for calculating the total percentage of the production set is:

$$P_{prod} = \frac{\sum\limits_{rec.classes} P_{rec.} + P_{rec.neg.classes}}{No.poz.classes + 1} \qquad (6)$$

The graphic representation for the recognition percentage of the production set can be seen in Figure 4. At a close investigation of this graph, we will notice its maximum point for a training consisting in 13.04% of the negative examples (or 15% of the number of positive examples).

This will be the maximum percentage for a set of negative examples that increases at every step with 5% of the initial set of positive examples. For finding a more accurate value of this percentage, a second experiment using a finer degree of increase is imposed.

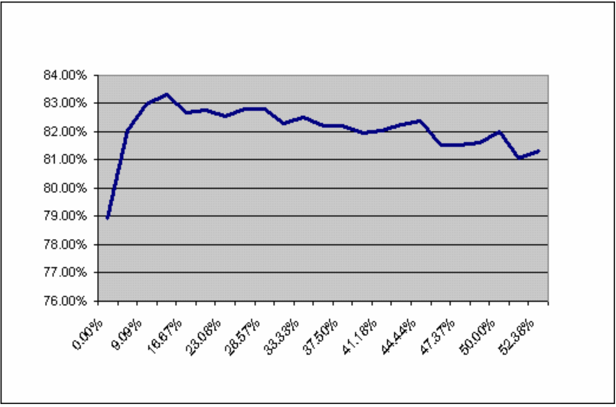

Figure 4. The graphic representation of the recognition rates for the production set of the neural network used in experiment 1

3.1.2 Experiment 2 details

The neural network is identical to that of example 1. Both the testing set with all the alphabet letters (26,000 characters) and the formula for calculating the percentage for a production set will be the same, too. (see formula 5)

The increase step used represents 0.5%, 10 times lower than the step used in experiment 1.

We will not make a thorough presentation of the values obtained in experiment 2; we should nevertheless conclude that the optimum percentage of negative examples varies between 16% and 18%.

3.2 Ocean Data experiments

As mentioned in the previous chapter, our main goal was to find the optimum percentage of negative examples that must be used for solving a problem with neural networks. In this experiment we tried to change the nature of the problem; instead of letter recognition, we tried to perform size estimation by using previously collected data.

The same as previously, two experiments have been performed. The first experiment aimed at locating the optimum percentage interval for the negative examples and the second experiment tried to establish the exact calculation of this percentage.

The results of the experiment showed that the optimum percentage varied between 17% and 18%. The graph obtained for the production set in this experiment was similar to the previous graph in Figure 4.

4 Conclusion

The present article dealt exclusively with the problems of training neural networks with negative examples.

A neural network trained only with positive examples may change into an overconfident network which could lead to high recognition rate of any example present at the input (even if the example is wrong).

The solution for this problem was the use of negative examples in the training process. The procedure is not new. However, it has not been sufficiently exploited until now, being both minimized and not currently used.

For successfully using the negative examples in the training stage, we suggest a method in which the negative examples are present in the training set in a percentage that has been calculated with the help of 4 experiments.

This method does not require changes in the architecture of the neural network and it is very convenient to use and more effective.

The optimum percentage of negative examples is situated between 16% and 18% of the number of positive examples.

As in the experiments there have been used two datasets completely different from the point of view of content and cohesion, we conclude that this percentage is relevant for all the problems that use neural networks.

We also suggested an approximation formula of the recognition percent for a viable production set. This is very important as during the testing process there may appear various problems that do not have an adequate testing set for real situation; in this case, a set adequate to a real situation must be approximated.

References:

[1] Haykin, S., *Neural Networks. A comprehensive Foundation*, Macmillan College Publishing, New York, 1994

[2] Rosenblatt, N., *Principles of Neurodynamics,* Spartan Books, Washington, D.C., 1962

[3] Rumelhart, D.E., McClelland, J.L. & PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* Volume 1: Foundations, The MIT Press, Cambridge, Massachusetts, 1986.

[4] Schwartz, J.T., The New Connectionism: Developing Relationships Between Neuroscience and Artificial Intelligence, *Proceedings of the American Academy of Arts and Science*, Vol. 117, No. 1, p. 123 – 141, Daedalus, 1988

[5] Rabunal, J. R., *Artificial Neural Networks in Real-life Applications*, Idea Group Pub, 2005

[6] Raudys, S; Somorjai, R; Baumgartner, R., Reducing the overconfidence of base classifiers when combining their decisions, *4th International Workshop on Multiple Clasifier Systems* (MCS 2003), vol. 2709, pag. 65-73, 2003

[7] Hosom, J.P., Villiers, J., s.a.l, Training Hidden Markov Model/Artificial Neural Network (HMM/ANN) Hybrids for Automatic Speech Recognition (ASR), *Center for Spoken Language Understanding (CSLU)*, 2006

[8] Qun, Z., Principe, J. C., Improve ATR performance by incorporating virtual negative examples, *Proceedings of the International Joint Conference on Neural Networks*, pag. 3198-3203, 1999

[9] Debevec, P., A Neural Network for Facial Feature Location, *CS283 Course Project*, UC Berkeley, 2001

[101] Tikhonov, A., Arsenin, V., *Solutions of ill-posed problems*, W.H. Winston, 1977

[11] Girosi, F. & Poggio, B., T. & Caprile, Extensions of a theory of networks for approximation and learning:outliers and negative examples., *Advances in Neural Information Processing Systems 3*, R.P. Lippmann, J.E.Moody and D. S. Touretzky, pag. 750-756, San Mateo, CA, 1991

[12] National Institute of Standards and Technology, *NIST Handprinted Forms and Characters Database* ,http://www.nist.gov/srd/nistsd19.htm, 2007

[13] Martha's Vineyard Coastal Observatory: „Ocean Data", http://www.whoi.edu/mvco/data/oceandata.html