# A Study on the Scalability of Artificial Neural Networks Training Algorithms Using Multiple-Criteria Decision-Making Methods

Diego Peteiro-Barral[*] and Bertha Guijarro-Berdiñas

Dept. of Computer Science, University of A Coruña,
Campus de Elviña s/n, 15071, A Coruña, Spain
`{dpeteiro,cibertha}@udc.es`
`http://www.lidiagroup.org/`

**Abstract.** In recent years, the unrestrainable growth of the volume of data has raised new challenges in machine learning regarding scalability. Scalability comprises not simply accuracy but several other measures regarding computational resources. In order to compare the scalability of algorithms it is necessary to establish a method allowing integrating all these measures into a single rank. These methods should be able to i) merge results of algorithms to be compared from different benchmark data sets, ii) quantitatively measure the difference between algorithms, and iii) weight some measures against others if necessary. In order to manage these issues, in this research we propose the use of TOPSIS as multiple-criteria decision-making method to rank algorithms. The use of this method will be illustrated to obtain a study on the scalability of five of the most well-known training algorithms for artificial neural networks (ANNs).

**Keywords:** Machine learning, scalability, artificial neural networks, multiple-criteria decision-making methods.

## 1 Introduction

In machine learning, scalability is defined by the effect that an increase in the size of the training set has on the computational performance of an algorithm (accuracy, training time and allocated memory). So the challenge is to find a tradeoff among them or, in other words, getting “good enough” solutions as “fast” as possible and as “efficiently” as possible. This issue becomes critical in situations in which there exist temporal or spatial constraints like: real-time applications dealing with large data sets, unapproachable computational problems requiring learning, or initial prototyping requiring quickly-implemented solutions.

[*] This work was supported by Secretaría de Estado de Investigación of the Spanish Government under projects TIN 2009-02402 and TIN2012-37954, and by the Xunta de Galicia through projects CN2011/007 and CN2012/211, all partially supported by the European Union ERDF. Diego Peteiro-Barral acknowledges the support of Xunta de Galicia under *Plan I2C* Grant Program.

A sample of the interest generated by large-scale learning was revealed with the organization of the workshop PASCAL Large Scale Learning Challenge [1] at the 25th International Conference on Machine learning (ICML'08). It was concerned with the scalability and efficiency of machine learning algorithms with respect to computational, memory and communication resources. In order to deal with large data sets, it is essential to minimize training time and allocated memory while maintaining accuracy. However, up to now, most machine learning algorithms do not provide an appropriate balance among them. Most algorithms tend to look with favor on one of these variables against others. A more recent sample of the relevance of scalability in learning, also known as "big learning", was disclosed with the conference of the Neural Information Processing Systems Foundation (NIPS'2011), aiming to provide a forum for exchanging solutions that address big learning problems. The relevance of these conferences for researchers and practitioners is meaningful.

In recent years, several researchers have addressed the issue of scalability of machine learning algorithms [2–4]. Scalability is wider than simple evaluations of accuracy. Scalability involves many aspects such as error, training time and memory requirements that should be merged into a single evaluation framework in order to rank algorithms. Such framework should be able to handle three different aspects: i) merging results of algorithms to be compared from different benchmark data sets, ii) being able to quantitatively measure the difference between algorithms, and iii) being able to weight some measures against others if necessary. In order to manage these issues, we propose to use a multiple-criteria decision-making method to rank algorithms. In this research, this framework will be applied to experimentally assess the scalability of five of the most popular training algorithms for ANN: gradient descent, gradient descent with momentum and adaptive learning rate, scaled conjugate gradient, Levenberg-Marquardt and stochastic gradient descent. To the best knowledge of the authors, this is a novel research that will shed light on the scalability of ANN training algorithms.

The remainder of this paper is structured as follows: section 2 describes the training algorithms, section 3 presents the measures of scalability used in this research, section 4 describes the MCDM method used, section 5 presents the experimental procedure followed, and section 6 shows the results obtained, and section 7 shows the conclusions and future lines of research.

## 2 Training Algorithms for ANN

This section gives a brief overview of the five training algorithms for ANNs considered in this research: gradient descent, gradient descent with momentum and adaptive learning rate, scaled conjugate gradient, Levenberg-Marquardt and stochastic gradient descent.

### 2.1 Gradient Descent

Gradient descent is one of the simplest training algorithms for ANNs. In the batch version, the algorithm starts with a random weight vector denoted by $w^{(0)}$.

Then, it iteratively updates the weight vector such that, at step $\tau$, it moves a short distance in the direction of the greatest rate of decrease of the error, that is in the direction of the negative gradient, evaluated at $w^{(\tau)}$ [5]:

$$\Delta w^{(\tau)} = -\eta \nabla E_{w^{(\tau)}}$$

where the parameter $\nabla$ is the learning rate and $E$ is the error function evaluated at $w^{(\tau)}$. Note that the gradient is re-evaluated at each step. It is expected that the value of $E$ will decrease at each step.

## 2.2 Gradient Descent with Momentum and Adaptive Learning Rate

The performance of the gradient descent algorithm is very sensitive to the proper setting of the learning rate $\eta$. If the learning rate is set too high, the algorithm can oscillate and become unstable. If the learning rate is too small, the algorithm takes too long to converge. Note that it is not practical to determine the optimal setting for the learning rate before training. With standard gradient descent, the learning rate is held constant throughout training. However, the performance of the gradient descent algorithm can be improved if it allows the learning rate to change during the training process. An adaptive learning rate attempts to keep the learning step size as large as possible while keeping learning stable.

Another very simple technique for improving the performance of the gradient descent algorithm is to add a momentum term [6]. The modified gradient descent formula is given by:

$$\Delta w^{(\tau)} = -\eta \nabla E_{w^{(\tau)}} + \mu \Delta w^{(\tau-1)}$$

This term adds inertia to the motion through weight space smoothing out the oscillations of the algorithm whilst speeding up the convergence. Moreover, the momentum term can be helpful in reducing the likelihood of finding a local minima.

## 2.3 Scaled Conjugate Gradient

With simple gradient descent, the direction of each step is given by the local negative gradient of the error function, and the step size is determined by an arbitrary learning rate parameter. A better procedure would be to consider some search direction in weight space, and then find the minimum of the error function along that direction. The minimum along the search direction $d^{(\tau)}$ then gives the next value for the weight vector:

$$w^{(\tau+1)} = w^{(\tau)} + \delta^{(\tau)} d^{(\tau)}$$

where the parameter $\lambda^{((\tau))}$ is chosen to minimize:

$$E(\lambda) = E(w^{(\tau)} + \lambda d^{(\tau)})$$

This gives an automatic procedure for setting the step length [7].

## 2.4 Levenberg-Marquardt

The Levenberg-Marquardt [8, 9] algorithm was designed specifically for minimizing a sum-of-squares error ($E = \frac{1}{2}\sum_n \epsilon_n^2 = \frac{1}{2}\|\epsilon\|^2$). Suppose we are currently at point $w_{old}$ in weight space and we move to a point $w_{new}$. If the displacement $w_{new} w_{old}$ is small then the error vector $\epsilon$ can be expanded to first order in Taylor series:

$$\epsilon(w_{new}) = \epsilon(w_{old}) + Z(w_{new} - w_{old})$$

where the matrix Z is defined with elements

$$(Z)_{ni} = \frac{\partial \epsilon^n}{\partial w_i}$$

If the error is minimized with respect to the new weights $w_{new}$ then:

$$w_{new} = w_{old} - (Z^T Z)^{-1} Z^T \epsilon(w_{old})$$

where the Hessian can be written in the form $H = Z^T Z$. Since there is no guarantee that the Hessian $H$ is positive definite, a correction term can be introduced to cover this problem by

$$H = H + \delta I$$

where $I$ is the identity matrix and $\delta$ is a parameter which value changes during training to guarantee the positive definiteness of the Hessian matrix.

The weight update formula could be applied iteratively in order to try to minimize the error function. The problem is that the update term could turn out to be relatively large. This problem is addressed by seeking to minimize the error function whilst at the same time trying to keep the step size small so as to ensure that the linear approximation remains valid.

## 2.5 Stochastic Gradient Descent

In the stochastic version of gradient descent, the error function is evaluated for just one sample at a time. The weights update rule is:

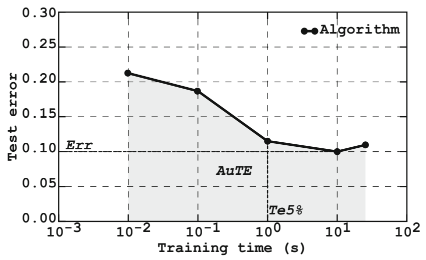
$$\Delta w^{(\tau)} = -\eta \nabla E^n_{w^{(\tau)}}$$

where the different samples $n$ in the training set are selected at random order. It is expected a steady reduction in error since the average direction of motion in weight space should approximate the negative of the local gradient [5]. An important advantage of the stochastic approach over batch methods arises if there is a high degree of redundant information in the data set. Another potential advantage of the stochastic approach is that it has the possibility of escape from local minima.

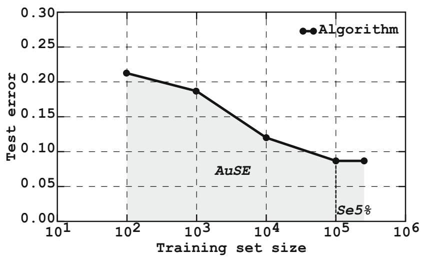
## 3 Scalability Measures

Performance measures such as mean squared error or class accuracy are inadequate to evaluate the performance of learning algorithms in large data sets since they do not take into account all aspects involved in scalability.

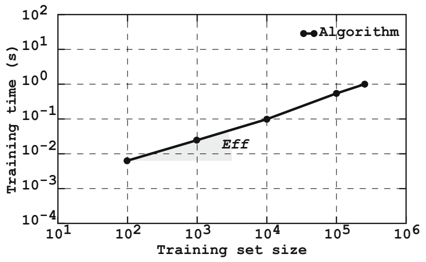
Scalability measures must take into consideration error, time and memory constraints. Thus, the goal is to find a learning algorithm that obtains the lowest error in the shortest time using the smaller number of samples. However, there are no standard measures of scalability. In order to overcome this issue, those measures defined in the PASCAL large scale learning challenge [1] will be used in this research. In this challenge, six scalar measures were defined based on three figures (see Figure 1).



(a) Training time vs Test error.



(b) Training set size vs Test error.



(c) Training set size vs Training time.

**Fig. 1.** Performance measures in terms of scalability, where the six scalar measures are marked in italic on the three figures.

- Figure 1(a) shows *Training time* vs *Test error*. It is obtained by displaying the evolution of the test error along certain time budgets and employing the *largest dataset* the algorithm can deal with. We compute the following scalar measures based on this figure:
  - *Err*: minimum test error (standard class error [10] for classification and MSE [10] for regression tasks).
  - *AuTE*: area under Training time vs Test error curve.
  - *Te5%*: the time $t$ for which the test error $e$ falls below a threshold $\frac{e-Err}{e} < 0.05$.

– Figure 1(b) shows *Training set size* vs *Test error*. It is obtained by displaying the different training set sizes, $10^{[2,3,4\cdots]}$, and the corresponding test errors achieved. Based on this curve, we compute:
  - *AuSE*: area under Training set size vs Test error curve.
  - *Se5%*: the size $s$ for which the test error $e$ falls below a threshold $\frac{e-Err}{e} < 0.05$
– Figure 1(c) shows *Training set size* vs *Training time*. It is obtained by displaying the different training set sizes and the corresponding training time needed by the algorithm. We compute the following scalar measure based on this curve:
  - *Eff*: slope $b$ of the curve by using a least squares fit to $ax^b$.

Following PASCAL, algorithms should be ranked for each of these six measures and compute the score of each algorithm as its average position with regard to the six rankings. For example, an algorithm that ranks first in three measures and second in the remaining three will obtain a final score of $\frac{1+1+1+2+2+2}{6} = 1.5$. Note, however, that this procedure do not take into consideration the magnitude of the measures but simply the ranking. This may lead to unfair results, mostly if some algorithms perform notably good or bad. In order to overcome this issue, the use of a multiple-criteria decision-making method is proposed.

## 4 Multiple-Criteria Decision-Making

Classification algorithms are normally evaluated in terms of multiple criteria. But how can multiple criteria be handle into a single evaluation model? Multiple-criteria decision-making [11] (MCDM) is focused on addressing the aforementioned issue. MCDM methods evaluate classifiers from different aspects and produce rankings of classifiers [12]. A multi-criteria problem is formulated using a set of alternatives $\{a_1, a_2, \ldots, a_m\}$ and criteria $\{k_1, k_2, \ldots, k_n\}$. The decision matrix is formulated as

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$$

where $x_{ij}$ represents the performance measure of the $i$th alternative in the $j$th criterion.

Among many MCDM methods that have been developed up to now, Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [13] is a well-known method that will be used in this research. TOPSIS finds the best algorithms by minimizing the distance to the ideal solution whilst maximizing the distance to the anti-ideal one. The extension of TOPSIS proposed by Opricovic and Tzeng [14] and Olson [15] is used in this research. The steps of the method are described as follows:

1. Compute the decision matrix consisting of $m$ alternatives and $n$ criteria. For alternative $A_i$, $i = 1, \ldots, m$, the performance measure of the $j$th criterion $C_j$, $j = 1, \ldots, n$, is represented by $x_{ij}$.
2. Compute the normalized decision matrix. The normalized value $r_{ij}$ is calculated as
$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$$
3. Establish a set of weights $w$, where $w_j$ is the weight of the $j$th criterion and $\sum_{j=1}^{n} w_j = 1$, and compute the weighted normalized decision matrix. The weighted normalized value $v_{ij}$ is computed as
$$v_{ij} = r_{ij} w_j$$
4. Find the ideal alternative solution $S^+$ and the anti-ideal alternative solution $S^-$, which are computed as,
$$\begin{aligned} S^+ &= \{v_1^+, \ldots, v_n^+\} = \\ &= \left\{ \left( \max_i v_{ij} | i \in I' \right), \left( \min_i v_{ij} | i \in I'' \right) \right\} \end{aligned}$$
and
$$\begin{aligned} S^- &= \{v_1^-, \ldots, v_n^-\} = \\ &= \left\{ \left( \min_i v_{ij} | i \in I' \right), \left( \max_i v_{ij} | i \in I'' \right) \right\} \end{aligned}$$
respectively, where $I'$ is associated with benefit criteria and $I''$ is associated with cost criteria.
5. Compute the distance of each alternative from the ideal solution and from the anti-ideal solution, using the Euclidean distance,
$$D_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^+)^2}$$
and
$$D_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2}$$
respectively
6. Compute the ratio $R_i^+$ equal to the relative closeness to the ideal solution,
$$R_i^+ = \frac{D_i^-}{D_i^+ + D_i^-}$$
7. Rank alternatives by maximizing the ratio $R_i^+$

### 4.1 Combining Divergent Rankings

While the rankings of alternatives on several data sets may agree, it is common the case in which they disagree. Thus, the problem of handling multiple criteria is translated into a problem of handling multiple rankings.

In this research, we propose to rank the alternatives in a secondary ranking that combines divergent rankings by re-applying the MCDM method using as inputs the values of the MCDM on the primary rankings. In this manner, the MCDM method is arranged in a two-step pipeline in which the output values of the primary rankings are used as inputs in the secondary ranking (see Figure 2).

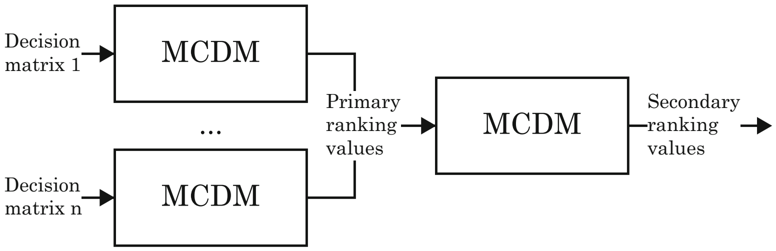


**Fig. 2.** Combination of divergent rankings obtained on different data sets

## 5 Experimental Study

The aim of this research is to experimentally evaluate the scalability of five of the most popular training algorithms for ANNs using a MCDM method.

### 5.1 Data Sets

Training algorithms were applied to two common tasks in machine learning: classification and regression. Table 1 shows the data sets used in this research with a brief description of their features: number of inputs, classes, training samples and test samples, and learning task.

**Table 1.** Characteristics of each dataset

| Dataset | Inputs | Classes | Training | Test | Task |
|---|---|---|---|---|---|
| Connect-4 | 42 | 3 | 60,000 | 7,557 | Classification |
| Covertype | 54 | 2 | 100,000 | 50,620 | Classification |
| KDDCup99 | 42 | 2 | 494,021 | 311,029 | Classification |
| Friedman | 10 | 1 | 1,000,000 | 100,000 | Regression |
| Lorenz | 8 | 1 | 1,000,000 | 100,000 | Regression |

Connect-4, Covertype and KDD Cup 99 data set are available at the UCI Machine Learning Repository [16]. On the other hand, Friedman and Lorenz are artificial datasets. The former is defined by the equation $y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1)$ where the input attributes $x_1, \ldots, x_{10}$ are generated independently from a uniform distribution on the interval $[0,1]$. On the other hand, Lorenz is defined by the simultaneous solution of three equations $\frac{dX}{dt} = \delta Y - \delta X$, $\frac{dY}{dt} = -XZ + rX - Y$, $\frac{dZ}{dt} = XY - bZ$, where the systems exhibits chaotic behavior for $\delta = 10$, $r = 28$ and $b = \frac{8}{3}$.

## 5.2 Experimental Procedure

The following procedure was done in order to evaluate the scalability of the training algorithms:

- Divide the data set using holdout validation. This kind of validation is suitable because the size of the data sets is very large.
- Set the number of hidden units of the ANNs to twice plus one the number of inputs. Following [17], going beyond this number should not make any difference. Also, it is important to remark that the aim here is not to investigate the optimal topology but to evaluate the scalability of training algorithms and thus, it is interesting to use networks as large as possible. Set the parameters of the ANNs to default values (learning rate 0.001, goal accuracy 0.01, maximum number of epochs 1000, etc.).
- For each data set, train the ANNs and compute the six scalar measures of scalability defined in Section 3: Err, AuTE, Te5%, AuSE, Se5% and Eff.
- Rank the algorithms using TOPSIS method. The values of the weights corresponding with each criterion are assigned equally. Note also that all measures are cost criteria, i.e. the smaller the better.
- Combine the primary ranking results of the different data sets (see Table 1) in a secondary, final ranking using TOPSIS.

## 6 Results and Discussion

Table 2 shows the results of the five training algorithms on the five data sets using the six scalar measures of scalability. Based on these six measures, TOPSIS obtains a ranking value also shown in Table 2 (the larger the value the better).

The results showed in this table demonstrate the change in approach when the learning algorithms are evaluated in terms of scalability. Notice that in many cases algorithms with lower test error rank worse than others. For example, Table 2(a) shows that LM obtains a much lower error than GD (-15%) and GDX (-8%). However, LM ranks worse than GD and GDX. In spite of its good accuracy, the long training time of LM makes this algorithm worse in terms of scalability.

Table 3 summarizes the TOPSIS values of each algorithm on each data set. As can be seen, there is no agreement on the ranking among the different data sets. In order to provide a single answer a secondary ranking is applied using as inputs the TOPSIS values of the five algorithms on the five data sets.

**Table 2.** Performance results of Connect-4, Covertype, KDD Cup 99, Frieadman and Lorenz data sets and primary TOPSIS values

(a) Connect-4 data set.

| Name | *TOPSIS* | Err | AuTE | Te5% | AuSE | Se5% | Eff |
|---|---|---|---|---|---|---|---|
| GD | *0.9057* | 0.38 | 5.16e1 | 1.08e2 | 0.97 | 1.00e2 | 0.43 |
| GDX | *0.7189* | 0.31 | 3.71e1 | 7.98e1 | 0.92 | 6.00e4 | 0.40 |
| LM | *0.2110* | 0.23 | 3.79e2 | 7.80e2 | 0.77 | 1.00e4 | 0.77 |
| SCG | *0.9389* | 0.21 | 7.01e1 | 2.62e2 | 0.77 | 1.00e4 | 0.50 |
| SGD | *0.7099* | 0.16 | 5.32e1 | 2.36e2 | 0.54 | 6.00e4 | 0.54 |

(b) Covertype data set.

| Name | *TOPSIS* | Err | AuTE | Te5% | AuSE | Se5% | Eff |
|---|---|---|---|---|---|---|---|
| GD | *0.8765* | 0.38 | 1.24e2 | 2.78e2 | 1.20 | 1.00e3 | 0.49 |
| GDX | *0.8674* | 0.42 | 4.74e1 | 1.01e2 | 1.32 | 1.00e4 | 0.41 |
| LM | *0.2431* | 0.24 | 6.41e2 | 1.74e3 | 0.94 | 1.00e4 | 0.84 |
| SCG | *0.6308* | 0.20 | 1.64e2 | 5.80e2 | 0.81 | 1.00e5 | 0.55 |
| SGD | *0.6426* | 0.13 | 1.21e2 | 7.83e2 | 0.62 | 1.00e5 | 0.58 |

(c) KDDCup99 data set.

| Name | *TOPSIS* | Err | AuTE | Te5% | AuSE | Se5% | Eff |
|---|---|---|---|---|---|---|---|
| GD | *0.8341* | 0.13 | 4.29e1 | 5.53e1 | 0.43 | 1.00e2 | 0.50 |
| GDX | *0.8130* | 0.15 | 2.55e1 | 5.93e1 | 0.46 | 1.00e3 | 0.44 |
| LM | *0.3923* | 0.11 | 2.21e2 | 1.24e3 | 0.46 | 1.00e4 | 0.80 |
| SCG | *0.6808* | 0.14 | 1.10e2 | 3.54e2 | 0.51 | 1.00e4 | 0.55 |
| SGD | *0.4603* | 0.00 | 8.85e0 | 1.35e3 | 0.07 | 4.94e5 | 0.59 |

(d) Friedman data set.

| Name | *TOPSIS* | Err | AuTE | Te5% | AuSE | Se5% | Eff |
|---|---|---|---|---|---|---|---|
| GD | *0.6424* | 8.33 | 2.19e3 | 7.51e1 | 36.77 | 1.00e3 | 0.37 |
| GDX | *0.7805* | 4.41 | 1.83e3 | 7.20e1 | 24.57 | 1.00e5 | 0.37 |
| LM | *0.9079* | 0.11 | 1.11e3 | 8.74e2 | 8.57 | 1.00e5 | 0.59 |
| SCG | *0.9150* | 0.79 | 1.67e3 | 1.71e2 | 10.33 | 1.00e5 | 0.44 |
| SGD | *0.3442* | 0.21 | 2.24e4 | 1.12e4 | 6.88 | 1.00e5 | 0.68 |

(e) Lorenz data set.

| Name | *TOPSIS* | Err | AuTE | Te5% | AuSE | Se5% | Eff |
|---|---|---|---|---|---|---|---|
| GD | *0.9676* | 0.74 | 4.82e2 | 6.17e1 | 2.98 | 1.00e2 | 0.36 |
| GDX | *0.6022* | 2.66 | 2.45e2 | 2.04e1 | 13.63 | 1.00e4 | 0.26 |
| LM | *0.9380* | 0.00 | 3.26e3 | 5.19e2 | 0.00 | 1.00e5 | 0.54 |
| SCG | *0.9939* | 0.01 | 5.61e2 | 1.38e2 | 0.05 | 1.00e4 | 0.43 |
| SGD | *0.3828* | 0.01 | 6.54e3 | 9.96e3 | 0.69 | 1.00e6 | 0.67 |

**Table 3.** Summary of the primary TOPSIS rankings

| Name | Connect-4 | | Covertype | | KDDCup99 | | Friedman | | Lorenz | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Value | Rank | Value | Rank | Value | Rank | Value | Rank | Value | Rank |
| GD | 0.9057 | 2 | 0.8765 | 1 | 0.8341 | 1 | 0.6424 | 4 | 0.9676 | 2 |
| GDX | 0.7189 | 3 | 0.8674 | 2 | 0.8130 | 2 | 0.7805 | 3 | 0.6022 | 4 |
| LM | 0.2110 | 5 | 0.2431 | 5 | 0.3923 | 5 | 0.9079 | 2 | 0.9380 | 3 |
| SCG | 0.9389 | 1 | 0.6308 | 4 | 0.6808 | 3 | 0.9150 | 1 | 0.9939 | 1 |
| SGD | 0.7099 | 4 | 0.6426 | 3 | 0.4603 | 4 | 0.3442 | 5 | 0.3828 | 5 |

**Table 4.** Secondary TOPSIS ranking

| Name | Rank | *TOPSIS* | Connect-4 | Covertype | KDDCup99 | Friedman | Lorenz |
|---|---|---|---|---|---|---|---|
| GD | 1 | *0.9543* | 0.9057 | 0.8765 | 0.8341 | 0.6424 | 0.9676 |
| GDX | 3 | *0.8562* | 0.7189 | 0.8674 | 0.8130 | 0.7805 | 0.6022 |
| LM | 5 | *0.3147* | 0.2110 | 0.2431 | 0.3923 | 0.9079 | 0.9380 |
| SCG | 2 | *0.9354* | 0.9389 | 0.6308 | 0.6808 | 0.9150 | 0.9939 |
| SGD | 4 | *0.3218* | 0.7099 | 0.6426 | 0.4603 | 0.3442 | 0.3828 |

Table 4 shows the results of the secondary ranking. Note that this method not only provides a ranking but it give information about how close are algorithms one each other. As can be seen, GD is ranked first but TOPSIS also indicates that it is *closely* followed by SCG. These two algorithms show a good tradeoff between accuracy and training time. On the other hand, SGD and LM are ranked fourth and fifth, respectively. Despite usually obtaining the best accuracy in classification and regression tasks, their long training time has a negative impact on their performance. Halfway, GDX is ranked third. It usually obtains a worse performance than SGD and LM but in a much shorter lapse of time.

Finally, we established in TOPSIS the set of weights equally, i.e. the importance of every criterion is considered to be the same. Note that this procedure can be easily adapted to other sort of problems in which one or several criteria may be more relevant than others. This is also true in the second step of the methodology in which the rankings obtained on each data set are merged in a single ranking. In this case, for example, we may be interested in promote classification problems rather than regression problems.

## 7 Conclusions

Most published researches concerning learning algorithms simply assess their performance in terms of accuracy. In this paper, the scalability of five of the most popular training algorithms for ANNs has been evaluated: GD, GDX, LM, SGD and SCG. Since there are no standard measures of scalability, those defined in the PASCAL Large Scale Learning Challenge were used. These measures assess the scalability of algorithms in terms of error, computational effort, allocated memory and training time.

The evaluation of the algorithms in terms of multiple criteria led us to apply a MCDM method. In particular, TOPSIS was used in this research. Moreover, we proposed a two-step approach to combine divergent rankings coming from the evaluation of the training algorithms on different data sets. Moreover, the use of a MCDM method allows to measure the distance among algorithms whilst easily use the weights to enhance some criteria against the others.

For future work, we plan to extend this research to different MCDM methods. In this case, we would have to face the combination of different rankings obtained by different methods on different data sets.

## References

1. Sonnenburg, S., Franc, V., Yom-Tov, E., Sebag, M.: PASCAL Large Scale Learning Challenge. Journal of Machine Learning Research (2009)
2. Strigl, D., Kofler, K., Podlipnig, S.: Performance and scalability of gpu-based convolutional neural networks. In: 18th Parallel, Distributed and Network-Based Processing (PDP), pp. 317–324. IEEE (2010)
3. Casey, K., Garrett, A., Gay, J., Montgomery, L., Dozier, G.: An evolutionary approach for achieving scalability with general regression neural networks. Natural Computing 8(1), 133–148 (2009)
4. Peteiro-Barral, D., Bolón-Canedo, V., Alonso-Betanzos, A., Guijarro-Berdiñas, B., Sánchez-Maroño, N.: Toward the scalability of neural networks through feature selection. Expert Systems with Applications 40(8), 2807–2816 (2013)
5. Bishop, C.M.: Neural networks for pattern recognition (1995)
6. Plaut, D.C., Nowlan, S.J., Hinton, G.E.: Experiments on learning by back propagation (1986)
7. Møller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6(4), 525–533 (1993)
8. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly Journal of Applied Mathmatics 2(2), 164–168 (1944)
9. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Society for Industrial & Applied Mathematics 11(2), 431–441 (1963)
10. Weiss, S.M., Kulikowski, C.A.: Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems. Morgan Kaufmann, San Francisco (1991)
11. Zeleny, M.: Multiple criteria decision making, vol. 25. McGraw-Hill, NY (1982)
12. Gang, K., Lu, Y., Peng, Y., Yong, S.: Evaluation of classification algorithms using mcdm and rank correlation. International Journal of Information Technology & Decision Making 11(01), 197–225 (2012)
13. Hwang, C.L., Yoon, K.: Multiple attribute decision making: methods and applications: a state-of-the-art survey, vol. 13. Springer, New York (1981)
14. Opricovic, S., Tzeng, G.H.: Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. European Journal of Operational Research 156(2), 445–455 (2004)
15. Olson, D.L.: Comparison of weights in TOPSIS models. Mathematical and Computer Modelling 40(7-8), 721–727 (2004)
16. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
17. Hecht-Nielsen, R.: Neurocomputing. Addison-Wesley (1990)