Pergamon

Omega, Int. J. Mgmt Sci. Vol. 24, No. 4, pp. 385–397, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0305-0483/96 $15.00 + 0.00

S0305-0483(96)00010-2

Effect of Data Standardization on Neural Network Training

M SHANKER[1]
M Y HU
M S HUNG

Kent State University, Kent, OH 44242, USA

(Received 9 June 1995; accepted after revision 21 February 1996)

Data transformation is a popular option in training neural networks. This study evaluates the effectiveness of two well-known transformation methods: linear transformation and statistical standardization. These two are referred to as data standardization. A carefully designed experiment is used in which data from two-group classification problems were trained by feedforward networks. Different kinds of classification problems, from relatively simple to hard, were generated. Other experimental factors include network architecture, sample size, and sample proportion of group 1 members. Three performance measurements for the effect of data standardization are employed. The results suggest that networks trained on standardized data yield better results in general, but the advantage diminishes as network and sample size become large. In other words, neural networks exhibit a self-scaling capability. In addition, impact of data standardization on the performance of training algorithm in terms of computation time and number of iterations is evaluated. The results indicate that, overall, data standardization slows down training. Finally, these results are illustrated with a data set obtained from the American Telephone and Telegraph Company. Copyright © 1996 Elsevier Science Ltd

Key words—neural networks, modelling

1. INTRODUCTION

WHEN TRAINING a neural network, one can use either the original data or the transformed data. Many researchers routinely use transformed data—sometimes due to algorithm requirement [1]; sometimes for improved learning; and other times for no reported reasons [5]. Some researchers transform (normalized) data to the interval [0, 1], whereas others transform to [−1, 1]. Simple data transformation methods are provided as a feature in most popular neural network software; for example, NeuralWare [10] and Brainmaker [2]. In addition to being needed for algorithm requirements, data transformation may be related to network computational and classification performance. This paper reports on the results of an empirical study on the effect of data transformation on network performance.

There are many transformations that can be applied to a set of data. We evaluate two of the most popular ones: linear transformation and statistical standardization. The former uses the range to transform a set of values to the interval [0, 1], while the latter computes the deviation from the mean and divides it by the standard deviation. Since data transformation in general includes many more methods and is a well established area in statistics, we will use data standardization to refer to the above two simple schemes of data transformation. In order to evaluate the effectiveness of these standardization schemes, we design an extensive experiment to generate a wide range of training situations. Classification problems are chosen as the subject of our experiments. Three problem types, based on distributions of variables, are defined. These problems range from simple to complex. For each problem type, training samples of various sizes and various proportions are generated randomly. Each sample is then trained on neural networks of various architectures and with various bounds on the arc weights.

Three different measures of performance are used to compare the data standardization methods. These measures are classification rate, mean square error, and the percentage of local minima, which are also called the Kuhn–Tucker points. In addition, a smaller experiment gathered computational statistics such as computation time and number of iterations. Finally, data standardization is evaluated on a dataset obtained from American Telephone and Telegraph Company.

The next section presents a brief review of the computational aspects of neural network training. That is followed by the experimental design, and a discussion of performance measures and the reasons for including them. The main results are in Section 5. Section 6 presents an empirical example to illustrate the effects of data standardization in classification, and Section 7 summarizes the major findings of this project.

2. NEURAL NETWORKS TRAINING

Each neural network defined in this study is a fully connected multi-layer feedforward network. In other words, there are connections between every node of one layer and every node of the next layer. Every node beyond the input layer has a bias. The activation function at each node is logistic. The objective function for training is based on the sum of square errors; i.e., least squares. The following brief review is not intended as a full discussion of neural networks, and serves only for defining notation and terminology, particularly those related to the computational issues of neural network training.

Let y_i^p represent the activation value at node i corresponding to pattern p.

$$y_i^p = \begin{cases} x_i^p & \text{if } i \in N_I \\ F(x_i^p) & \text{if } i \in N_H \cup N_O \end{cases}$$

where x_i^p is the input value of pattern p at node i, and N_I, N_H, and N_O are respectively sets of input, hidden, and output nodes. F is the logistic activation function, i.e., $F(a) = (1 + e^{-a})^{-1}$. For a hidden or output node i, the input is defined as

$$x_i^p = \sum_j w_{ji} y_j^p + w_{0i}$$

where w_{ji} is the weight of arc (j,i) and w_{0i} is the bias.

Let t_i^p denote the target for pattern p at output node i. The objective function for training is typically

$$Minimize \quad \frac{1}{2} \sum_p \sum_{i \in N_O} (y_i^p - t_i^p)^2 \qquad (1)$$

Since network training is an unconstrained minimization problem, nonlinear optimization method based on well founded theory [4, 6] can be used. The successful and reliable nonlinear programming methods use a search strategy. Let w^k denote the vector of weights in iteration k. A descent direction d^k is determined. Then a step size α_k is calculated. The new search point is defined as

$$w^{k+1} = w^k + \alpha^k d^k$$

Algorithms differ in the choice of d^k and α^k. For example, d^k can be set to the negative gradient (this method is called the steepest descent), or the conjugate gradient, or the quasi-Newton direction, or even the Newton direction (see [4] and [6] for details). The step size can also be fixed, or dynamically determined by minimizing the objective function along d^k. The typical back propagation algorithm [12] uses the steepest descent direction with fixed step size. An algorithm stops if the gradient vanishes or an improved new point cannot be found. In the former, we have reached a local minimum, and in the latter we may be in an area where objective contours are too close together (i.e., the objective function is ill-conditioned). In practice, when each component of the gradient is smaller than a preset tolerance, the current search point is considered as a local minimum.

The networks in this study were trained with the GRG2-based system of Subramanian and Hung [13]. GRG2 is a widely distributed nonlinear programming software [7]. The default search direction, which is the one used in the study, is conjugate gradient. The step size α_k is determined by a series of probes of the objective function taking into account the bounds on variables. The tolerance for zero gradient is 10^{-4}. In other words, when each gradient component is within this tolerance, then we stop the algorithm and declare the solution a Kuhn–Tucker point (i.e., a local minimum). If the objective function does not improve by more than 10^{-4} after 6 iterations, we stop the algorithm with a non-Kuhn–Tucker solution. See [13] for details.

The neural networks used in this study have one hidden layer. The number of input nodes is two since every pattern has only two variables and there is one output node representing the group membership of the pattern. In other words, $|N_I| = 2$ and$|N_O| = 1$. The target value is 0 for members of group 1 and 1 for those of group 2. The hidden nodes are one of the design variables in this experiment.

3. DESIGN OF EXPERIMENT

A computer experiment was conducted to evaluate the effectiveness of three methods of data standardization. Transformation is performed on each input variable independently. In other words, for each training sample, statistics for variable i, such as mean $\bar{x}_i$, minimum l_i, maximum u_i, and variance s_i^2, are computed for each variable. Value p of variable i, x_i^p, is then scaled according to the following methods:

1. No transformation at all.
2. Linear transformation—

$$z_i^p = (x_i^p - l_i)/(u_i - l_i)$$

3. Statistical standardization—

$$z_i^p = (x_i^p - \bar{x}_i)/(s_i\sqrt{n-1})$$

where z_i^p is the transformed value of x_i^p. The sample variance is computed with a divisor $n - 1$.

Linear transformation scales each variable into the interval [0, 1]. Standardization 3 is widely used in statistics for controlling roundoff errors and making the units of variables comparable [9, p. 379].

The experimental subjects are 2-group 2-variable classification problems. Three types of problems are considered.

P1. Variables have a bivariate normal distribution with equal variance–covariance matrices across the groups.

P2. Variables have a bivariate normal distributions but with unequal variance–covariance matrices.

P3. Variables have a bi-exponential distribution.

Aside from differences in distribution and variance–covariance matrix, the three types of problems are made to be as much alike as possible. For example, variables of the same group have the same means across the three cases and the same correlation. If μ_1 and μ_2 are vectors of variable means in group 1 and group 2, respectively, and Σ_1 and Σ_2 are the respective variance–covariance matrices, then these parameters are chosen as follows:

$$\mu_1 = \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 15 \\ 5 \end{pmatrix},$$

$$\Sigma_1 = \begin{pmatrix} 25.0 & 7.5 \\ 7.5 & 25.0 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 225.0 & 22.5 \\ 22.5 & 25.0 \end{pmatrix}$$

For P1, mean vectors are as specified, but the variance–covariance matrix for group 2 is set to Σ_1. For P2 and P3, the mean vectors and the variance–covariance matrices are as specified. In both matrices, the off-diagonal elements are chosen so that the correlation between variables is 0.3. This coefficient of correlation is arbitrary. It is large enough to create irregular "overlaps" among the generated patterns and it is small enough that sample covariance matrices do not become ill-conditioned.

The bi-exponential distribution belongs to the bi-gamma distribution whose joint density function cannot be specified [8, p. 506]. What can be said is that each variable has an exponential distribution and the correlation between variables is known. Specifically, let x_{ij} denote variable i in group j with mean μ_{ij}, then the marginal density function is

$$f(x_{ij}) = \mu_{ij}^{-1} e^{-x_{ij}/\mu_{ij}} \; for \; x \geq 0$$

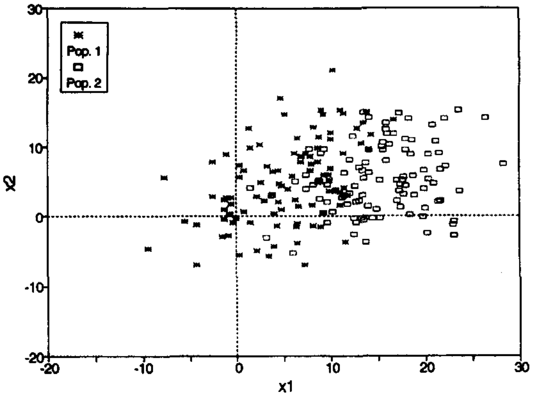

Fig. 1. Problem type 1 bivariate normal, equal covariance

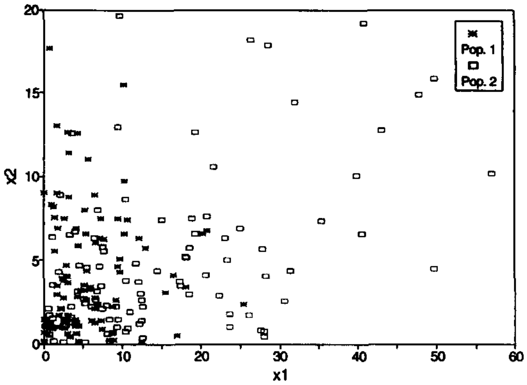

Fig. 3. Problem type 3 bivariate exponential.

Since the mean of an exponential distribution is equal to the standard deviation, it is not possible to generate exponential distributions with unequal means and equal variances. This explains the choice of the diagonal values for matrices Σ_1 and Σ_2. For each distribution, training samples of various sizes are generated using algorithms shown in [8, pp. 505–506].

Examples of these three types of problems are shown in Figs 1–3. The stars represent points from group 1 and the hollow squares represent points from group 2. Each graph shows 100 randomly generated points from each group. In P1, both groups are slanted because of the covariance between variables. In Figs 2 and 3, it can be seen that group 2 encloses group 1 because of the larger variance and covariance terms of group 2.

The other factors in this experiment are described in the following sections.

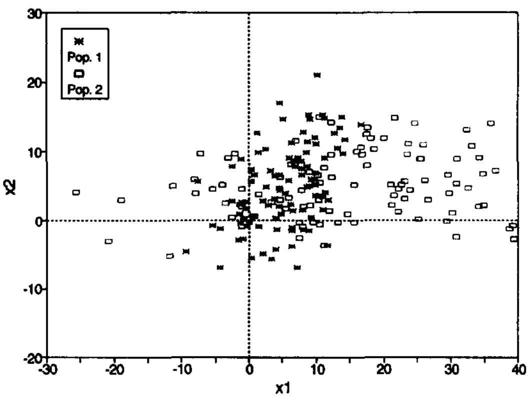

Fig. 2. Problem type 2 bivariate normal, unequal covariance matrix.

3.1. Network architecture

Network architecture determines the number of weights to be estimated by the optimizer. As a network has more arcs, it is expected that computation will be more difficult. Since we use fully connected networks with only one hidden layer, the differences in architecture are the differences in the number of hidden nodes. We use three levels here: 0(H0), 2(H2), and 5(H5). With 0 hidden nodes, the network reduces to one without a hidden layer. The other values are chosen to provide a wide variety of problem sizes for the optimizer. Since the biases are defined for all hidden and output nodes, the number of weights to be computed for is as follows:

Architecture	Number of arcs	Number of biases	Number of weights
H0	2	1	3
H2	6	3	9
H5	15	6	21

3.2. Sample size

Three levels of training sample sizes are selected—30 (S1), 50 (S2), and 100 (S3). Sample size refers to the total number of patterns from both group 1 and group 2. Since every iteration in the nonlinear optimizer requires computation of the gradient, which is accumulated from all the patterns in the training set, sample size directly affects the amount of work in the iteration.

3.3 Sample ratio

The other factor is the ratio of group 1 members in the training sample. There are 3 levels: 0.5 (R1), 0.7 (R2), 0.9 (R3). The combination of sample size and sample ratio determines the number of patterns from either group of our classification problem. For example, with sample size 30 (S1) and ratio of 0.7 (R2), the number of patterns from group 1 is 21, whereas the number of patterns from group 2 is 9. The reason to include this factor is that as proportion increases, we expect the training to be increasingly difficult, by which we mean an increase in both computational time and number of iterations.

3.4. Bounds on arc weights

With our training system, it is very simple to put bounds on the weight w_{ji} of arc (j,i). While each weight in principle is without bound, it tends to be small in practice. In previous studies [11, 14], we used bounds of ± 100 on each arc weight (but not the biases). We found that some of the solutions had some weights at the bounds. So we decided to include this as an experimental factor and added two other bounds. The three levels are ± 100 (B1), ± 500 (B2), and ± 1000 (B3). Our assumption is that as bounds increase, the likelihood of finding Kuhn–Tucker points should increase too. The reason is simply that with larger bounds, we increase the solution space and therefore give the algorithm more chances to find a local minimum.

The next section presents the performance measures used in our study.

4. PERFORMANCE MEASURES

The measures for comparing the effectiveness of data standardization methods are as follows.

4.1. Classification rate

At the termination of training, a pattern is correctly classified if the network output is within 0.5 of a pattern's target value. Classification rate is simply the percentage of the correct classifications in the training sample. This criterion is an important summary statistic of a classifier. A better classifier is associated with a higher classification rate.

4.2. Mean square errors (MSEs)

Mean square errors (MSEs) is equal to sum of square errors defined in equation (1) divided by $(n - w)$ where n is the sample size and w is the number of weights to be estimated. w is determined by the network architecture and is given in an earlier table. MSE is used so that the objective function value in equation (1) can be compared across training samples of different sizes and different architectures. This criterion is related to the quality of least squares solution. In other words, a better solution is one with smaller MSE.

4.3. Percentage of Kuhn–Tucker points

The minimization problem equation (1) contains many local minima, flat areas, and areas where contour lines are very close together (called ill-conditioned points). Depending on the starting solution, the choice of search direction, and the tolerance for stopping, an algorithm may either end up at a local minimum or a flat area, or may fail to find a minimum (due to ill-conditioning). Everything else being equal, one would want a model to end up at a local minimum more often. For this experiment, every training sample is solved with 10 different starting solutions. The percentage of final solutions being Kuhn–Tucker points is reported.

4.4. Training time

This measures the computational time taken by the optimization algorithm for training the neural network.

These performance measures depend not only on the data but also on the algorithm used. Specifically, computation time depends on the choice of search direction and step size, and the termination criteria. Classification rate and MSE depend on the quality of the solution found by the algorithm. We mention this point to stress that the results reported below may be applicable only to the GRG2 based algorithm used in our experiments [13]. For researchers using back-propagation or similar algorithms, the effect of data standardization may be different.

5. RESULTS

Within each problem type, 10 different training samples from each combination of sample size, sample ratio, and weight bounds are generated. Thus, there are 270 different training samples in this experiment. Each training set is subject to all three standardizations; hence there are 810 different transformed samples. Each is then trained on networks with three different architectures. In addition, training is repeated with 10 different starting solutions. Therefore, for each problem type, we have a total of 24,300 observations.

The starting solutions are generated from a uniform distribution [− 3, 3] for each arc weight and node bias. Different starting solutions are the consequence of 10 different seeds for the random number generator. The same 10 seeds are used for all the networks. Therefore, networks of the same architecture have the same 10 starting solutions.

5.1. Summary results

Since there is a voluminous amount of output, we decided to report only the differences between standardization methods. For the first two measures of performance, classification rate and MSE, we take the difference between two standardizations and compute the mean difference across the 10 starting solutions. For example, for problem P1, a sample of size 50 with sample ratio of 0.5, and a network with 2 hidden nodes, we trained the sample with 10 starting solutions. At the end of each training, we have two performance measures based on the network outputs. We then compute the difference for each measure between every pair of standardizations. Since there are three pairs, there are three differences for each performance measure. For each pair of standardizations, we compute the mean difference across the 10 starting solutions. These means are then used for statistical analysis. For each problem type, there are 810 mean differences for each performance measure.

For the third measure, the percentage of solutions being Kuhn–Tucker points among 10 starting solutions is found for each training sample. The difference between every pair of standardization methods is then computed and used for further analyses.

Table 1 shows the mean differences between standardization methods. Standardization method 1 denotes no standardization, 2 linear transformation, and 3 statistical standardization. The other notations are as follows:

Ratexy = Classification rate (%) of standardization method x − that of method y
MSExy = MSE of standardization method x − that of method y
KTxy = Percentage of Kuhn–Tucker solutions of method x − that of method y

The classification rates show that standardization method 1 is inferior to method 2 by 2.85% for problem P1, by 1.86% for P2, and by 1% for P3. Standardization method 1 is also inferior to method 3 by 2.93% for P1, 2.07% for P2, and 0.76% for P3. All these differences are statistically significant at α of 5%. Therefore, one can conclude that based on overall classification rates, data standardization is beneficial to classification problems.

Standardization method 2 is inferior to method 3 for problems P1 and P2, but superior for P3. Again, these differences are statistically significant.

Based on MSE, standardization method 1 is inferior to method 2 and method 3 in all problem types. These differences are statistically

Table 1. Difference in performance measures of standardization methods

Measure	Problem P1		Problem P2		Problem P3	
difference	Mean[a]	*P* value[b]	Mean[a]	*P* value[b]	Mean[a]	*P* value[b]
Rate12	−2.852	0.0001	−1.865	0.0001	−0.999	0.0001
Rate13	−2.932	0.0001	−2.067	0.0001	−0.755	0.0001
Rate23	−0.080	0.0021	−0.202	0.0001	0.244	0.0001
MSE12	0.0164	0.0001	0.007	0.0001	0.005	0.0001
MSE13	0.0166	0.0001	0.009	0.0001	0.004	0.0001
MSE23	0.0002	0.3115	0.002	0.0001	−0.002	0.0001
KT12	−0.223	0.0001	−0.307	0.0001	−0.219	0.0001
KT13	−0.234	0.0001	−0.308	0.0001	−0.223	0.0001
KT23	−0.011	0.0224	−0.002	0.7426	−0.003	0.4265

[a] Sample size 810.

[b] Based on testing mean difference = 0.

Table 2. Mean differences in performance measures by network architecture

	Problem P1			Problem P2			Problem P3		
Measure	H0	H2	H5	H0	H2	H5	H0	H2	H5
Rate12	−6.584[a]	−1.341	−0.631	−4.236	−2.318	0.960	−1.998	−0.958	−0.042[b]
Rate13	−6.616	−1.495	−0.685	−4.268	−2.537	0.605	−2.001	−0.783	0.518
Rate23	−0.032[b]	−0.154	−0.054[b]	−0.032[b]	−0.219	−0.356	−0.003[b]	0.175	0.560
MSE12	0.038	0.006	0.005	0.026	0.005	−0.009	0.015	0.003	−0.002[+]
MSE13	0.038	0.006	0.006	0.026	0.006	−0.006	0.015	0.002	−0.006
MSE23	−0.0001	0.0002[b]	0.0005[b]	0.0001[b]	0.001	0.003	0.0000[b]	−0.001	−0.004
KT12	−0.421	−0.150	−0.099	−0.499	−0.318	−0.103	−0.413	−0.151	−0.093
KT13	−0.422	−0.187	−0.095	−0.505	−0.291	−0.129	−0.411	−0.161	−0.095
KT23	−0.001[b]	−0.037	0.004[b]	−0.006[b]	0.026	−0.026	0.001[b]	−0.009[b]	−0.002[b]

[a] Mean difference from a sample of size 270.

[b] Mean difference not significant at 5% level.

significant at $\alpha = 5\%$. Standardization method 2 is inferior to method 3 for problems P1 and P2, but superior for P3. But the difference for P1 is not significant.

The above two measures offer consistent comparisons between standardization methods. Namely, data standardization is beneficial. The difference between linear transformation or statistical standardization depends on the problem type. For P1 and P2 where each variable is normally distributed, statistical standardization is better. For P3, linear transformation is better.

On the percentage of finding Kuhn–Tucker points, standardization method 1 is worse than method 2 by 0.22% for P1, by 0.31% for P2, and by 0.22% for P3. It is inferior to method 3 by 0.23% for P1, by 0.31% for P2, and by 0.22% for P3.

These illustrate that data standardization does help the optimization algorithm in finding local minima. However, for both classification rates and MSE, the advantage of data standardization decreases as we move from problem type P1 to P2 and then to P3. It indicates that for difficult problems, data standardization loses its beneficial effects. Standardization method 3 is better than method 2 for three problem types, but the difference is only significant for P1.

Note that Rate 23 = Rate13 − Rate12, and similar equations hold for MSE23 and KT23. The reported numbers may differ from these equations because of rounding.

5.2. *Effect of network architecture*

The summary data in Table 1 are further broken up by network architecture and the results are reported in Table 2. The number of observations for each performance measure is 270. Additional clear patterns emerge from this table. One, the advantage of data standardization decreases, when architecture is held constant, as we move from problem P1 to P2, and then to P3; exactly as we have already observed in Table 1. For example, for architecture H0, Rate12 goes from − 6.584 to − 4.236 to − 1.998. Recall that a negative value means standardization method 2 (linear transform) is better than no standardization. This same pattern holds for MSE as well. This pattern says that as classification problem becomes more difficult, the advantage of data standardization diminishes. The only exception is the percentage for finding Kuhn–Tucker solutions where the advantage peaks at problem P2. This phenomenon was also seen in Table 1.

The second pattern is that as the network becomes larger, the advantage of data standardization decreases. And this holds for all three measures. Take problem P1. As the number of hidden nodes goes from 0 to 2 and then to 5, Rate12 goes from − 6.584 to − 1.341 to − 0.631. For P2, it is even more dramatic. Rate12 goes from − 4.236 to − 2.318 to 0.960; the last shows that no standardization is better. What this implies is that the neural networks have the capability of self scaling. And the self scaling capability increases with increasing network size.

Another interesting phenomenon is that as network architecture changes, the difference between standardization method 2 and method 3 becomes important. For example, method 3 is increasingly better than method 2 according to Rate23 for problem P2 as architecture increases. The reverse holds for P3; method 2 is increasingly better than method 3. In fact, for P3 Rate23 is larger than either Rate12 or Rate13 under H5.

Table 3. Mean differences in performance measures by training sample size

	Problem P1			Problem P2			Problem P3		
Measure	S1	S2	S3	S1	S2	S3	S1	S2	S3
Rate12	−4.072[a]	−2.588	−1.897	−3.167	−1.487	−0.940	−1.936	−0.867	−0.195[b]
Rate13	−4.169	−2.721	−1.906	−3.459	−1.679	−1.063	−1.768	−0.501	0.004[b]
Rate23	−0.098	−0.133	−0.009[b]	−0.293	−0.191	−0.123	0.168	0.366	0.199
MSE12	0.022	0.015	0.011	0.013	0.005	0.004	0.009	0.006	0.002
MSE13	0.023	0.015	0.011	0.016	0.006	0.004	0.006	0.004	0.0009[b]
MSE23	0.0006[b]	0.0001[b]	−0.0002[b]	0.003	0.001	0.0007	−0.002	−0.002	−0.001
KT12	−0.313	−0.210	−0.148	−0.316	−0.328	−0.277	−0.287	−0.219	−0.151
KT13	−0.324	−0.223	−0.156	−0.313	−0.334	−0.279	−0.282	−0.224	−0.161
KT23	−0.011[b]	−0.014[b]	0.008[b]	0.003[b]	−0.006[b]	−0.002[b]	0.004[b]	−0.005[b]	−0.010[b]

[a] Mean difference from a sample of size 270.
[b] Mean difference not significant at 5% level.

5.3. *Effect of training sample size*

The summary data in Table 1 are broken up by sample size and the results are shown in Table 3. The patterns observed in Table 2 are also very visible here. For example, the advantage of data standardization decreases as we move from problem P1 to P2 and to P3, when sample size is held constant. Take Rate12. With sample size 30 (S1), the difference goes from − 4.072 to − 3.167 and then to − 1.936 as we move from P1 to P3. The only exception, also seen in Table 2, is the percentage of finding Kuhn–Tucker solutions. Measures KT12 and KT13 rise from P1 to P2 with most sample sizes, and drop from P2 to P3. Entries in P3 under S2 and S3 are higher than the corresponding ones in P1.

Secondly, as sample size increases, the advantage of data standardization decreases. This holds for measures Rate and MSE, and most of KT. For example, by measure Rate12 or Rate13, data standardization shows no significant effect for problem P3 when sample size goes up to S3. It shows that the self scaling capability of neural networks is helped by large samples.

The relative advantage of standardization method 2 and method 3 is mixed. Either Rate or MSE shows that method 3 seems to be better for problem P1 and P2, but worse for P3. The differences for P1, however, are not significant.

5.4. *Effect of sample proportion*

The summary data of Table 1 are broken up by factor sample proportion. The results in Table 4 do not exhibit the same patterns we saw earlier. The advantage of data standardization is still monotonic as we go from P1 to P3. This time, the trend is downward, however. Take ratio R1. Measure Rate12 goes from − 2.868 to − 1.214 and then to − 0.232, as we go from P1 to P2 and then P3. But within a problem type, the pattern is not clear. While the advantage of data standardization generally increases with sample proportion, measures Rate and MSE have dips in P1, but peaks in P3, at ratio R2. It is difficult to understand why sample proportion 0.7 (R2) causes the pattern to change. Measure KT, however, is consistent across both problem types and sample proportions.

The overall conclusion one can draw from these discussions is that neural networks seem to be able to adjust themselves to data when sample and architecture are large enough; and thus rendering data standardization less useful.

Table 4. Mean differences in performance measures by sample proportion

	Problem P1			Problem P2			Problem P3		
Measure	R1	R2	R3	R1	R2	R3	R1	R2	R3
Rate12	−2.868[a]	−2.463	−3.225	−1.214	−2.034	−2.345	−0.232[b]	−1.546	−1.221
Rate13	−2.947	−2.568	−3.280	−1.623	−2.131	−2.445	0.127[b]	−1.268	−1.124
Rate23	−0.079[b]	−0.105	−0.055[b]	−0.409	−0.097[b]	−0.100	0.359	0.278	0.096
MSE12	0.016	0.015	0.018	0.004[b]	0.005	0.013	0.002	0.007	0.007
MSE13	0.016	0.015	0.019	0.007	0.006	0.013	0.0007[b]	0.005	0.005
MSE23	−0.0003[b]	0.0006	0.0002[b]	0.003	0.0008	0.0007	−0.002	−0.002	−0.001
KT12	−0.172	−0.211	−0.286	−0.225	−0.269	−0.427	−0.086	−0.176	−0.396
KT13	−0.187	−0.222	−0.295	−0.216	−0.265	−0.444	−0.092	−0.179	−0.397
KT23	−0.015[b]	−0.010[b]	−0.009[b]	−0.009[b]	0.004[b]	−0.017[b]	−0.007[b]	−0.003[b]	−0.0004[b]

[a] Mean difference from a sample of size 270.
[b] Mean difference not significant at 5% level.

Table 5. Mean differences in performance measures by bounds on arc weights

	Problem P1			Problem P2			Problem P3		
Measure	B1	B2	B3	B1	B2	B3	B1	B2	B3
Rate12	−2.797[a]	−2.891	−2.868	−1.949	−1.829	−1.816	−1.197	−0.899	−0.902
Rate13	−2.907	−2.948	−2.940	−2.153	−2.043	−2.005	−0.882	−0.705	−0.679
Rate23	−0.110	−0.057[b]	−0.072[b]	−0.204	−0.214	−0.189	0.316	0.194	0.223
MSE12	0.016	0.016	0.016	0.008	0.007	0.007	0.007	0.005	0.005
MSE13	0.016	0.017	0.017	0.009	0.009	0.008	0.005	0.003	0.003
MSE23	0.0003[b]	0.0001[b]	0.0001[b]	0.001	0.002	0.002	−0.002	−0.002	−0.002
KT12	−0.254	−0.214	−0.203	−0.332	−0.299	−0.289	−0.238	−0.214	−0.206
KT13	−0.257	−0.228	−0.218	−0.336	−0.297	−0.293	−0.250	−0.209	−0.209
KT23	−0.04[b]	−0.014[b]	−0.016[b]	−0.004[b]	0.002[b]	−0.004[b]	−0.012[b]	0.005[b]	−0.003[b]

[a] Mean difference from a sample of size 270.

[b] Mean difference not significant at 5% level.

5.5. *Effect of bounds on arc weights*

Table 5 shows the effect of bounds on arc weights. By measure Rate, it seems to indicate that as bounds increase, the advantage of data standardization decreases. But measure MSE shows little difference among various bounds. Measure KT exhibits an expected pattern. We had assumed that as bounds increase, the chances for finding Kuhn–Tucker solutions would also increase. Therefore, KT should tend to zero as bounds increase. The results are consistent with this assumption.

We examined several samples where local minima were not found in all bounds and several samples where local minima were found in bounds ± 100 (B1) but not in larger bounds. For the former cases, the final solution is typically within bounds B1. Since for the same network architecture, the starting points are all the same for the same data set, the final solutions are the same. For data sets where the solution is at bounds B1 for some arc weights, enlarging the bounds often resulted in non-Kuhn–Tucker points again. It seems to indicate that there is a large flat area extending outward from the origin of the solution space. In such cases, larger bounds seemed to allow line search (for step size α) to skip over local minima as it takes into account the bounds on arc weights. A different algorithm may thus exhibit a different pattern.

5.6. *Effect on computation*

A separate run was made in order to gather statistics on the performance of our training algorithm as a consequence of data standardization. For this run, only a subset of the training samples was used. There are still 3 methods of data standardization, 3 problem types, 3 network architectures, 3 sample sizes, and 3 sample proportions. However, due to the previous result indicating that arc bounds do not affect the classification rates or MSE, the level of bounds on arc weights is reduced to one: ± 500 (B2). The number of data sets for each combination of factor levels is reduced from 10 to 2, and for each data set, the number of starting solutions is reduced from 10 to 5. For each of these 2430 training sessions, the computation time (milliseconds on an IBM RS 6000 model 530) and the number of iterations were gathered. The computation time includes a little time for output of summary statistics but excludes all other input/output operations. The number of iterations is equal to the number of times gradient is computed.

Table 6 shows that the average time and iterations over all 2430 training sessions, broken up by standardization methods. The average time for untransformed data (method 1) was 82.72 ms, and it was 96.98 ms for method 2 and 106.70 ms for method 3. The average number of iterations per training session was 38.15 for method 1, 44.53 for method 2, and 48.65 for method 3. This was the second surprise finding from our experiments. We had expected data standardization to help in both computation time and the number of iterations. The results clearly indicate that just the opposite is true.

The differences in computation times and in iterations between standardization methods were used to create Tables 7–10. For each training sample, the average difference among the 5 starting solutions is computed and then

Table 6. Summary results on computation time and number of iterations

Standardization method	Mean time	Mean iterations
1	82.72[a]	38.15
2	96.98	44.53
3	106.70	48.65

[a] Based on sample size 162.

Table 7. Mean differences in algorithmic performance by problem type

Measure	P1	P2	P3	Total
TM12	−10.50[a,b]	−10.43[b]	−21.84	−14.26[c]
TM13	−20.87	−17.38	−33.69	−23.98
IT12	−4.98[b]	−4.40[b]	−9.74	−6.37
IT13	−8.25	−7.38[b]	−15.86	−10.50

[a] Mean difference from a sample of size 54.
[b] Mean difference not significant at 5% level.
[c] Mean difference from a sample of size 162.

summarized by problem type and by the other factors. The notation is as follows:

TMxy = Average time of standardization method x − that of method y

ITxy = Average iterations of method x − that of method y

From Table 7, we see that method 1 (no standardization) gave rise to faster computation time and fewer iterations than either standardization in all three problem types. In addition, TM13 − TM12 (not shown in Table 7) indicates that method 2 is better than method 3 by almost the same amounts. However, the progression from P1 to P3 as we have seen previously does not hold here. The differences are greatest in P3, then in P1, and finally in P2. Entries in the overall column are the averages of the three problem types and are equal to the differences in mean values from Table 6.

The results in Table 7 are broken up by network architecture and these are shown in Table 8. As architecture increases, which makes the optimization problem larger and more difficult to compute, the penalties of data standardization increase too. By both measures TM and IT, the differences become large, in absolute values, as we move from H0 to H5 in all problem types. However, statistical significance is detected mostly in P3 and mostly between methods 1 and 3. It is interesting to note that there are no significant differences for H0 in all problem types.

Table 9 show the differences by sample size. The patterns revealed in this table are not as clear as those in Table 8. As sample size increases, computation becomes more difficult because there is more work per iteration to evaluate both the objective function and the gradient. However, the significant differences only occur in sample size 50 (S2). In this table, measure IT does not seem to be a good predictor for measure TM. Consider problem P2 and sample size S3, where IT indicates standardization methods 2 and 3 are better than 1 but TM shows just the opposite, although the differences in TM are not statistically significant.

Similarly a confusing picture is seen in Table 10. There is no consistent progression in either TM or IT when sample ratio moves from 0.5 (R1) to 0.9 (R3). Most of the significant differences occur in R2. There are no significant differences in either TM or IT in R3.

The next section presents an empirical example to illustrate the effect of data standardization on an non-simulated dataset.

6. AN ILLUSTRATIVE EXAMPLE

The American Telephone and Telegraph Company conducted a survey to investigate the relationship between the perceived level of long distance telephone usage and some socioeconomic characteristics. Approximately 1,400 consumers responded to a questionnaire. Perceived level of usage is measured as either heavy/medium or light/non. Among the multitude of socioeconomic variables, two were selected for the purposes of this study—household income, and number of friends and relatives (see [3] for further details on this data set). Both are understandably related to the perceived level of usage and are employed to classify consumers into any of the two usage groups.

Table 8. Mean differences in algorithmic performance by network architecture

	Problem P1			Problem P2			Problem P3		
Measure	H0	H2	H5	H0	H2	H5	H0	H2	H5
TM12	−0.33[a,b]	−4.89[b]	−26.29[b]	0.33[b]	−3.67[b]	−27.96[b]	−0.08[b]	−13.67	−51.79
TM13	−1.98[b]	−11.19[b]	−49.46	0.56[b]	−6.52[b]	−46.17	−0.89[b]	−24.39	−75.78
IT12	−1.31[b]	−5.18[b]	−8.44[b]	0.10[b]	−3.22[b]	−10.09[b]	−0.13[b]	−8.30[b]	−20.79
IT13	−2.03	−7.48[b]	−15.24[b]	0.22[b]	−2.61[b]	−19.76[b]	−0.97[b]	−17.46	−29.14

[a] Mean difference from a sample of size 18.
[b] Mean difference not significant at 5% level.

Table 9. Mean differences in algorithmic performance by sample size

	Problem P1			Problem P2			Problem P3		
Measure	S1	S2	S3	S1	S2	S3	S1	S2	S3
TM12	−0.39[a,b]	−22.47[b]	−8.66[b]	−12.16[b]	−9.03[b]	−10.10[b]	−3.67[b]	−36.06	−25.81[b]
TM13	−2.38[b]	−20.17	−40.07[b]	−11.03[b]	−27.28[b]	−13.82[b]	−0.89[b]	−57.58	−42.59
IT12	−0.54[b]	−13.91	−0.48[b]	−9.56[b]	−4.47[b]	0.81[b]	−1.03[b]	−21.83	−6.36[b]
IT13	−2.58[b]	−10.73	−11.44[b]	−7.81[b]	−15.26[b]	0.92[b]	−0.79[b]	−35.17	−11.61

[a] Mean difference from a sample of size 18.
[b] Mean difference not significant at 5% level.

Three random samples of size 30, 50, and 100 were selected from this dataset. For this illustration, we considered only a single proportion, equal to 0.64, of the heavy/medium usage group. Each sample was trained on three different architectures, with 0, 2 and 5 hidden nodes. However, due to previous results (Section 5 indicating that arc bounds do not affect classification rates, here, we consider only a single level for arc bounds: ± 500. Ten different starting seeds were used for each network and were the same for the corresponding network across the three samples. As in the case of our simulated experiment, the three standardization methods were evaluated on classification rates, MSE, percentage of Kuhn–Tucker points, and computational time. Since only one sample is used for each of the three levels of sample size, statistical significance is no longer a meaningful measure of the differences across the various experimental conditions. Thus, the results presented below are only descriptive in nature.

6.1. Results

Results reported here largely support that of our simulated experiment. As shown in Table 11, standardization method 3 (statistical standardization) has the highest classification rate and the lowest average MSE of all three methods. Method 3 is better than method 2 by 3.281% and is better than method 1 by 16.699% in classification rate. Overall, it is more likely for method 3 to have Kuhn–Tucker condition satisfied as compared to the other two methods, and similar to previous results, method 3 is more computationally time consuming compared to the other methods.

For this data set, with zero hidden nodes (H0), method 2 and 3 yield similar results. This is reflected in the classification rate and MSE (see Table 11). In addition, all 3 methods have the same percentage of Kuhn–Tucker solution satisfied. Under H0, method 2 and 3 utilize about the same amount of computer time, which is substantially more than that of method 1. As the size of the architecture increases, the advantage of method 2 over method 1 in classification rate diminishes. The difference decreases from 33.110% to 2.367% when H5 is used. Yet a curvilinear relationship is indicated in the difference in classification rates between method 1 and 3 as the number of hidden nodes increases. A similar pattern is captured in the MSE differences. As network architecture increases, methods 2 and 3 require more computing time than method 1, with method 3 requiring the most computing time of all methods. As sample size increases, the advantage of data standardization decreases (Table 11). This holds for Rate, MSE and most of KT. Again, this shows the self scaling property of neural networks as sample size goes up. As sample size increases, as expected, methods 2 and 3 require proportionally more computing time than method 1.

7. CONCLUSIONS

We have reported an experiment designed to evaluate the effectiveness of data standardiz-

Table 10. Mean differences in algorithmic performance by sample proportion

	Problem P1			Problem P2			Problem P3		
Measure	R1	R2	R3	R1	R2	R3	R1	R2	R3
TM12	2.7[a,b]	−22.03[b]	−12.17[b]	−24.46[b]	−6.28[b]	−0.56[b]	−39.49	−24.46	−1.59[b]
TM13	−23.67[b]	−18.94	−20.01[b]	−42.37	−19.29[b]	9.52[b]	−36.24	−49.63	−15.18[b]
IT12	3.09[b]	−10.19	−7.83[b]	−9.92[b]	−4.36[b]	1.07[b]	−17.80	−9.80	−1.62[b]
IT13	−7.42[b]	−10.02	−7.31[b]	−20.29[b]	−9.61[b]	7.76[b]	−15.90	−21.48	−10.19[b]

[a] Mean difference from a sample of size 18.
[b] Mean difference not significant at 5% level.

Table 11. Mean differences in performance measures of standardization methods for AT&T data

		By Network Architecture			By Sample Size		
Measure	Overall	H0	H2	H5	S1	S2	S3
Rate12	−13.418	−33.110	−4.777	−2.367	−14.554	−13.533	−12.166
Rate13	−16.699	−33.110	−2.289	−14.699	−18.331	−15.933	−15.833
Rate23	−3.281	0	2.488	−12.332	−3.777	−2.400	−3.666
MSE12	0.171	0.444	0.027	0.042	0.198	0.167	0.148
MSE13	0.284	0.444	0.018	0.390	0.384	0.266	0.203
MSE23	0.113	0	−0.009	0.348	0.186	0.098	0.054
KT12	0.077	0	0.233	0	0.200	0.100	−0.066
KT13	0.211	0	0.233	0.400	0.233	0.300	0.100
KT23	0.133	0	0	0.400	0.033	0.200	0.166
TM12	−14.144	−8.800	−19.633	−14.000	−16.733	−19.233	−6.466
TM13	−47.277	−8.933	−38.700	−94.200	−30.833	−47.100	−63.900
TM23	−33.133	−0.133	−19.066	−80.200	−14.100	−27.866	−57.433

ation on neural network training. Three transformation methods were compared, including the base case, which is no transformation. The subject is two-group classification problems, ranging from simple to complex. The major results are as follows.

- Overall, data standardization is beneficial, as measured by classification rate, MSE, and percentage of finding Kuhn–Tucker solutions.
- The advantage of data standardization diminishes as network becomes large.
- The advantage of data standardization decreases as sample size increases.
- The advantage of data standardization is evident in all sample proportions but there is no simple pattern observable.
- The advantage of data standardization is not affected by bounds on arc weights.
- Data standardization gives rise to more computation time and number of iterations required by the training algorithm.
- The disadvantage of data standardization in terms of computation time and number of iterations increases with network size but exhibits no consistent pattern otherwise.
- Neural networks in this study exhibit self scaling capability.

Again, we would like to mention that some of these results may not be applicable when other training algorithms are used. The algorithm we use has been proven to be very robust (see [15] for a survey of comparative studies), in that it has solved many difficult nonlinear programs. As mentioned earlier, the computational issues are affected by the algorithmic decisions in determining search direction and step size. The effect of data standardization on computation time and number of iterations may thus be different for other algorithms. Performance measures such as classification rate and MSE are dependent on the quality of solution. Given that neural network training problem is known to have many local minima, these measures are also affected by algorithm decisions. However, we believe that results pertaining to neural networks will stand when other algorithms are used. Results like the self-scalability and the combined effects of sample size and network architecture should be generally true.

REFERENCES

1. Atlas L, Cole R, Connor J, El-Sharkawi M, Marks R, Muthusamy Y and Barnard E (1990) *Performance Comparisons Between Backpropagation Networks and Classification Trees on Three Real-world Applications. Advances in Neural Information Processing Systems*, Vol 2, (Edited by Touretzky D and Kaufmann Organ).
2. Brainmaker (1990) California Scientific Software, Grass Valley, CA.
3. Bruning ER and Hu MY (1989) The role of demographic factors in the analysis of survey versus diary purchase reporting accuracy. *Survey Methodology*, **15**, 59–70.
4. Fletcher R (1987) *Practical Methods of Optimization*, 2nd edition. Wiley and Sons, New York.
5. Gallinari P, Thiria S, Badran F and Fogelman-Soule F (1991) On the relations between discriminant analysis and multilayer perceptrons. *Neural Networks*, **4**, 349–360.
6. Gill PE, Murray W and Wright M (1981) *Practical Optimization*. Academic Press, Boston.
7. Lasdon LS and Waren AD (1986) *GRG2 User's Guide*. School of Business Administration, University of Texas at Austin, TX.
8. Law AV and Kelton WD (1991) *Simulation Modeling and Analysis*, 2nd edition. McGraw-Hill, New York.
9. Neter J, Wasserman W and Kutner M (1992) *Applied Linear Statistical Models*, 3rd edition. Irwin.
10. NeuralWare (1993) NeuralWare, Inc, Pittsburgh, PA.

11. Patuwo E, Hu MY and Hung MS (1993) Two-group classification using neural networks. *Decision Sciences*, **24**, 825–845.
12. Rumelhart DE, Hinton GE and Williams RJ (1986) Learning internal representations by error propagation. In *Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition* (Edited by Rumelhart DE and Williams JL). MIT Press, Cambridge, MA.
13. Subramanian V and Hung MS (1993) A GRG2-based system for training neural networks: design and computational experience. *ORSA Journal on Computing*, **5**, 386–394.
14. Subramanian V, Hung MS and Hu MY (1993) An experimental evaluation of neural networks for classification. *Computers and Operations Research*, **20**, 769–782.
15. Waren A, Hung MS and Lasdon L (1987) The status of nonlinear programming software—an update. *Operations Research*, **35**, 489–503.

ADDRESS FOR CORRESPONDENCE: Professor Murali S Shanker, Department of Administrative Sciences, College of Business, Kent State University, Kent, OH 44242-0001, USA.