Performance of Neural Network Algorithms during the prediction of yarn elongation

Josphat Igadwa

Fibers and Polymers 2008, Vol.9, No.1, 80-86

Performance of Neural Network Algorithms during the Prediction of Yarn Breaking Elongation

Josphat Igadwa Mwasiagi*, XiuBao Huang, and XinHou Wang

College of Textiles, Donghua University, Shanghai, 200051, China

(Received July 18, 2007; Revised November 21, 2007; Accepted November 21, 2007)

Abstract: Yarn breaking elongation is one of the most important yarn quality characteristics, since it affects the manufacture and usability of woven and knitted fabrics. One of the methods used to predict the breaking elongation of ring spun yarn is artificial neural network (NN). The design of an NN involves the choice of several parameters which include the network architecture, number of hidden layers, number of neurons in the hidden layers, training, learning and transfer functions. This paper endeavors to study the performance of NN as the design factors are varied during the prediction of cotton ring spun yarn breaking elongation. A study of the relative importance of the input parameters was also undertaken. The results indicated that there is a significant difference in the types of transfer and training functions used. Of the two transfer functions used, purelin performed far much better than logsig function. Among the five training functions, the best training functions in terms of performance was Levenberg-Marquardt. The study of the relative importance of input factors revealed that yarn twist, yarn count, fiber elongation, length, length uniformity and spindle speed, were the six most influential factors.

Keywords: Yarn breaking elongation, Artificial neural networks, Cotton fiber, Ring spinning

Introduction

Yarn breaking elongation is one of the most important yarn quality characteristics, which is defined as the percentage increase in length when the yarn breaks due to a tensile force applied along the main yarn axis [1]. Several researchers [2-5] have reported that yarn breaking elongation affect the manufacture, quality and usability of knitted and woven fabrics, with higher elongation giving better results. One area of the study of yarn breaking elongation is the prediction of yarn breaking elongation for cotton ring spun yarn. According to Majumdar and Majumdar [6] yarn breaking prediction Artificial neural networks (NN) models, are more efficient than mathematical and statistical models. The design of an NN model involves the selection of several parameters, which include the network architecture, number of hidden layers, number of neurons in the hidden layers, training, learning and transfer functions [7,8]. Other factors such as improving generalization methods and data pre and post processing must also be considered in the design process since they improve the performance of the NN. While there are many possible designs of an NN, the success of the design process can be judged based on the performance of the NN. The purpose of this research work is to study the performance of an NN prediction model and hence propose an optimum NN model for the prediction of yarn breaking elongation. The optimum model can also be used to study the effect of input factors on yarn breaking elongation.

Factors Affecting Yarn Breaking Elongation

The machine factors which affect the breaking elongation for ring spun yarns include spindle speed, traveler mass, machine draft, yarn count and twist [9,10]. Considering the effect of fiber properties on yarn breaking elongation, Douglas [11] reported that the most important factors which affect yarn breaking elongation are fiber elongation and length. Other factors which influence yarn breaking elongation are fiber micronaire, strength, color and trash content. According to Majumdar and Majumdar the impact of fiber length on yarn breaking elongation is listed well behind fiber elongation, length Uniformity index, yellowness, yarn count, reflectance, fiber strength and micronaire. While the effect of different fiber parameters and machine factors on yarn breaking elongation keep on varying from author to author [12-14], it is worthy noting that previous researchers [1,9,10,12-14] have used at most eight inputs for the reported yarn breaking elongation prediction models. This could have limited the study of the factors affecting yarn breaking elongation. For a more comprehensive understanding of the factors which affect yarn breaking elongation we selected the following 19 factors as inputs to the NN;

- Machine factors: spindle speed, ring diameter, traveler weight and ring spinning draft
- Fiber properties: micronaire, maturity, spinning consistency index, length, length uniformity, short fiber index, strength, elongation, reflectance, yellowness, trash cent, trash area and trash grade
- Yarn factors: count and twist.

The selected 19 factors are by far more than any set of inputs used by the previous researchers and will give a deeper understanding of the effect of different factors on yarn breaking elongation for ring spun cotton yarn.

*Corresponding author: igadwa@yahoo.com

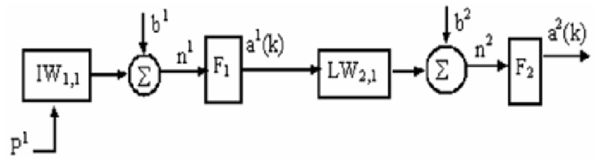

Figure 1. The architecture of a feedforward network.

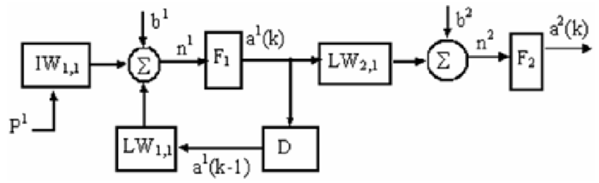

Figure 2. The architecture of Elman network.

Artificial Neural Network

Artificial neural network commonly referred to as neural networks (NN) is basically an information processing algorithm that is inspired by the way biological nervous systems, such as the brain, process information. NN are applicable in virtually every situation in which a relationship between the inputs and outputs exists. NN can be classified based on their attributes [7,8] such as: topology, applications, connection types, and learning methods. The basic unit of an NN is the neuron which can be described as an information processing unit with four basic components (Figure 1) namely weights ($IW_{1,1}$), summing device (Σ), bias (b^1), and activation function (F_1). Several neurons can be combined to form a layer. Similarly several layers can be combined to form a network. Information passes from one layer to another until the network produces an output.

Feedforward Network

The feedforward neural network is one of the simplest types of NN devised. In this NN, the information moves only in one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes [15]. There are no cycles or loops in the network. A feedforward NN will have one or more hidden layers of neurons followed by an output layer. Figure 1 shows the architecture of a feedforward network with one hidden layer which receives a set of inputs (P^1) together with the weights ($IW_{1,1}$) and biases (b^1) and processes it using the activation function F_1 to give an output $a^1(k)$.

The outputs from the hidden layer are fed to the output layer together with another set of weights ($LW_{2,1}$) and biases (b^2) which are then processed according to the activation function F_2 to give an output $a^2(k)$. The final output of the network is compared with the targeted output in what is normally referred to as a training process. The activation functions (F_1 and F_2) can be any of the differentiable transfer function such as tansig, logsig or purelin [7].

Elman Network

The Elman network is a type of a recurrent feed forward neural network, with a feedback connection from the output of the hidden layer neurons to the input of the network as shown in Figure 2. This feedback path allows Elman networks to learn how to recognize and generated temporal patterns, as well as spatial patterns [15,16]. The Elman network which was originally designed to learn time-varying patterns or temporal sequences has proven to be useful for the design of algorithms used in the control of manufacturing processes.

Backpropagation Training Algorithms

The training process in an NN involves adjusting the weights and biases so as to minimize the network's performance function. For training algorithms which use the gradient of the performance function to determine how to adjust the weights, the gradient can be determined by using a technique called backpropagation (BP), which involves performing computations backwards through the network. The simplest implementation of BP learning which has been adequately explained by several authors [7,8,15-17] updates the network weights and biases in the direction in which the performance function decreases most rapidly (the negative of the gradient). An iteration of the algorithm can be written as:

$$X_{k+1} = X_k - \alpha_k g_k$$

where X_k is a vector of current weights and biases, g_k is the current gradient, and α_k is the learning rate. At the end of the iteration, the outputs of the network are compared with the targeted values and the difference between the two is referred to as the error of the network. In BP algorithm, if the error of the algorithm is not within accepted pre-set values, the output is propagated back through the network and is re-trained with a different set of weights and biases. This is the process by which the algorithm is named. This process is repeated until the pre-set error is achieved. Standard BP algorithms suffer from many drawbacks, such as slow rate of convergence and are therefore unsuitable for practical problems.

Faster training algorithms have been designed and they fall into two classes, namely the heuristic techniques and the numerical optimization techniques. Examples of heuristic techniques are variable learning rate and resilient backpropagation training algorithms. Variable learning rate training algorithms were designed to improve the performance of the NN by using an adaptive learning rate which attempts to keep the learning step size as large as possible while keeping learning stable. This is done by choosing a learning rate that is responsive to the complexity of the error at any point during the training of the NN. Resilient Backpropagation (*rbp*) training algorithms on the other hand were designed to eliminate the harmful effects of the magnitudes of the partial derivatives and as such, *rbp* algorithms use the sign of the derivative to determine the direction of the weight update. The magnitude of the derivative has no effect on the weight update.

The numerical optimization techniques can be subdivided into three categories; conjugate gradient, quasi-Newton, and Levenberg-Marquardt algorithms. Examples of conjugate gradient algorithms are Fletcher-Reeves Update, Polak-Ribiere Update, Powell-Beale Restarts, and Scaled Conjugate Gradient. As was previously mentioned, the basic BP algorithm adjusts the weights in the steepest descent direction (negative of the gradient), which is the direction in which the performance function is decreasing most rapidly. Although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms, a search is performed along the conjugate direction which produces faster convergence than steepest descent directions. The conjugate gradient Polak-Ribiere (P-R) Update training algorithm, like other conjugate gradient algorithms, start out by searching in the steepest descent direction (negative of the gradient) on the first iteration as shown below, where p_0 is the initial search and g_0 is the gradient of the learning function.

$$p_0 = -g_0$$

A line search is then performed to determine the optimal distance to move along the current search direction:

$$X_{k+1} = X_k + \alpha_k P_k$$

The next search direction is determined so that it is conjugated to previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction:

$$p_k = -g_k + \beta_k p_{k-1}$$

The various versions of conjugate gradient are distinguished by the manner in which the constant β_k is computed. For the P-R update algorithm, β_k is computed as:

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}$$

This is the ratio of the norm squared of the current gradient (g_k) to the norm squared of the previous gradient (g_{k-1}). Newton's method is an alternative to the conjugate gradient methods for fast optimization, but it requires the calculation of the second derivatives (Hessian matrix) of the performance index, hence it is complex and expensive to compute for feedforward neural networks. The Quasi-Newton (or secant) methods do not calculate the second derivate, but updates an approximate Hessian matrix at the algorithm iteration which in effect, reduces the computational requirement for the algorithm. Levenberg-Marquardt (*lma*) backpropagation is another algorithm that uses a simplified version of Newton's method of training, and like the Quasi-Newton methods, the *lma* algorithm was designed to approach second-order training without having to compute the Hessian matrix. Since the performance function of a feedforward network has the form of the sum of squares, the Hessian matrix can be approximated as:

$$H = J^T J$$

and the gradient can be computed as:

$$g = J^T e$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. Each term in the matrix has the form:

$$J_{i,j} = \frac{\partial e_i}{\partial w_j}$$

The simplest approach to compute the derivates is to use the approximation:

$$J_{i,j} \approx \frac{\Delta e_i}{\Delta w_j}$$

where Δe_i represents the change in the output error due to the small perturbation of the weight Δw_j. After the Jacobian matrix is computed the weight updates can be performed and the performance of the algorithm computed as in the standard BP algorithms.

Learning and Transfer Functions

Apart from the training algorithms, learning and transfer functions are other important parameters to be considered during the design of an NN algorithm. In backpropagation training algorithms two weight and bias learning functions are used. These are gradient descent and gradient descent with momentum. The gradient descent method calculates the weight change Δw from the weight (or bias) learning rate lr and the gradient descent gW as follows [15]:

$$\Delta w = lr \times gW$$

while gradient descent with momentum calculates the weight change according to the following equation:

$$\Delta w = mc \times \Delta w_{prev} + (1 - mc) \times lr \times gW$$

where mc is the momentum constant and the previous weight change Δw_{prev} is stored and read from the learning state. The behavior of an NN depends on both the weight and the input-output function (transfer function) that is specified for the units. The transfer functions can be classified into three main groups [7]: linear (or ramp), threshold (or hard limit), and sigmoid. For linear functions, the output is proportional to the total weighted input. For threshold functions the output is set at one of the two levels depending on whether the total input is greater than or less than some threshold value. For sigmoid function, the output varies continuously but not linearly as the input changes.

Materials and Methods

Materials

Cotton lint and carded ring spun yarn samples were collected from textile factories in Kenya. For every yarn sample collected, a sample of the corresponding cotton lint mixture used to spin the yarn was also collected. Table 1 gives the details of the cotton and yarn samples collected.

Methods

An NN algorithm for predicting the yarn breaking elongation was designed and its performance studied. The inputs of the NN were ringframe processing parameters (spindle speed (ss), ring diameter (di), traveler weight (tw) and spinning draft (dt)), cotton fiber HVI characteristics, yarn twist (tp) and yarn count (tx). Yarn breaking elongation was set as the output of the network. The cotton fiber HVI characteristics used as part of the input parameters were micronaire (mi), maturity (mt), spinning consistency index (sc), fiber length (le), length uniformity (uf), short fiber index (sf), strength (st), elongation (ef), reflectance (rd), yellowness (+b), trash cent (tc), trash area (ta) and trash grade (tg). Using the following design factors the performance of an NN algorithm with one hidden layer was studied:

- Network architecture: two types used - Elman and feedforward networks
- Backpropagation training algorithms: five algorithms were used - variable learning rate (*vlr*), Levenberg-Marquardt (*lma*), resilient backpropagation (*rbp*), Quasi-Newton algorithm (*qna*) and Polak-Ribiere conjugate gradient (*prc*)
- No. of neurons in the hidden layers: varied from 2 to 20 in steps of 2s
- Learning functions: two functions were used - gradient descent and gradient descent with momentum
- Transfer functions: two functions were used - logsig and purelin functions

By using different combinations of the above factors a total of 400 (2×2×2×5×10) performance algorithms were designed. The main features of the algorithm involved data acquisition, data pre-processing, network training and data post processing. The acquired data (inputs and targets) were normalized so that they had zero mean and unity variance. The data was divided into training, validation, and test subsets in the ratio of 4:1:1 respectively, as equally spaced points. The performance of the NN was measured using mean squared error (mse).

Results and Discussion

The Effect of the Factors on the Performance of the Algorithms

The analysis of the performance of the 400 elongation prediction algorithms yielded the result given in Table 2. From the factors point of view, the performance of the algorithms showed no significant difference as the network architectures, weight/bias learning functions and number of neurons in the hidden layer were changed since their respective p-values are high (more than 0.05), and the F values are less than the corresponding F_{crit} values.

The type of training and transfer functions used indicated a significant change in the performance of the algorithms (p-value values more than 0.05 and F values more than the corresponding F_{crit} values). This implies that one out of the two transfer functions used performed better than the other. Similarly, at least one of the training functions out of the five used performed better than the others. This calls for further investigations.

The Influence of Transfer Functions on the Performance of the Algorithms

To further investigate the impact of the transfer functions on the performance of the algorithms, the purelin and logsig algorithms were analyzed to establish the relationship if any between the transfer functions and the performance of the algorithms.

Table 1. Details of cotton lint and yarn samples

Cotton lint	Mill code	Yarn Ne	No. of cops	Spindle speed (rpm)
Voi AR	B	30	20	11,000
Voi AR	B	20	20	10,000
WT AR	A	30	20	12,000
Kitui AR	A	30	20	12,000
Kitui AR	A	24	20	11,000
Kitui AR	C	24	20	8,000

Table 2. Analysis of the performance of the 400 algorithms

Factors	No. of factors	F	P-value	F_{crit}
Network architectures	2	0.6834	0.4088	3.8649
Transfer function	2	4799.755	3.5×10^{-24}	3.8649
Learning functions	2	0.0896	0.7649	3.8649
BP training functions	5	3.5279	0.00762	2.3945
No. of neurons	10	0.4131	0.9281	1.9039

Table 3. Analysis of the performance of the purelin algorithms

Factors	No. of factors	F	P-value	F_{crit}
Network architectures	2	0.8567	0.3558	3.889
Learning functions	2	1.5780	0.2105	3.889
BP training functions	5	60.7911	2.88×10^{-13}	2.418
No. of neurons	10	3.1425	0.0015	1.9294

Table 4. Analysis of the performance of the logsig algorithms

Factors	No. of factors	F	P-value	F_{crit}
Network architectures	2	11.8440	0.0007	3.8889
Learning functions	2	0.0637	0.8009	3.8889
BP training functions	5	29.32	4.32×10^{-19}	2.4180
No. of neurons	10	2.8112	0.0040	1.9294

The Performance of the Purelin Algorithms

In Table 3 the performance of the algorithms using purelin training function showed no significant difference for the type of network architecture and learning function. However, in the same table it is clear that there is a significant difference in the type of transfer function and number of neurons used. This is the same kind of relationship which the performance of the algorithm showed when compared with the combined set of algorithms (those using purelin and logsig functions). While the combined set of algorithms showed no significant difference for the number of neurons in the hidden layer, the purelin algorithms however showed a significant difference in the performance of the algorithms as the number of neurons in the hidden layer were varied.

The Performance of the Logsig Algorithms

The performance of the algorithms using logsig transfer functions is given in Table 4, which indicate that there is no significant difference for the type of learning functions used. There is however a significant difference in the performance of the algorithms as the type of architectures, training functions and number of neurons in the hidden layer are varied. The means of the mse values for all the algorithms using Elman and feedforward architecture were 0.507 and 0.479, which are significantly different, indicating that Elman is more sensitive to the type of the transfer function used in the output layer when compared to the feedforward architecture.

Comparison of the Purelin and Logsig Algorithms as the Training Functions are varied

Table 5 gives the performances (mse) of the purelin and logsig algorithms as the training functions are varied. Purelin as a transfer function performed far much better than logsig for all the five training functions used. It can also be concluded that *vlr* performed significantly lower than the other four training algorithms.

Table 5. Performance (mse) of purelin and logsig algorithms

Training function	Purelin	Logsig
vlr	0.1645	0.5637
lma	0.0389	0.4887
rbp	0.0548	0.4685
qna	0.0530	0.4708
prc	0.0554	0.4722

The Optimum Elongation Prediction Algorithm

Having established that *vlr* training and logsig transfer functions performed significantly lower than the other functions, a study was conducted to establish how the remaining algorithms compared with each other. In this study the performance of the algorithms were monitored under the following design conditions:

- Weight/bias learning function: 2 used-gradient descent and gradient descent with momentum
- Transfer Function: one used - Purelin transfer function
- No. of neurons in the hidden layer varied from 2 to 20 in steps of 2s
- Network architecture: 2 used - Elman and Feedforward
- BP training algorithms 4 used - *lma*, *rbp*, *qna* and *prc*

A total of 160 (2×1×10×4×2) algorithms using purelin transfer function were analyzed and the results (Table 6) indicated that there is no significant difference for the type

Table 6. Analysis of the performance of the 160 algorithms with purelin

Factors	No. of factors	F	P-value	F_{crit}
Network architectures	2	1.4100	0.2368	3.901
Learning functions	2	0.1234	0.7259	3.601
BP training functions	4	27.32	3.44×10^{-14}	2.664
No. of neurons	10	21.95	2.12×10^{-23}	1.9428

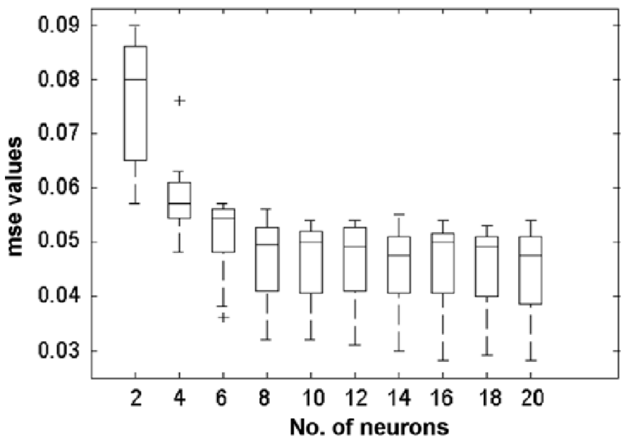

Figure 3. The performance of algorithms with different number of neurons.

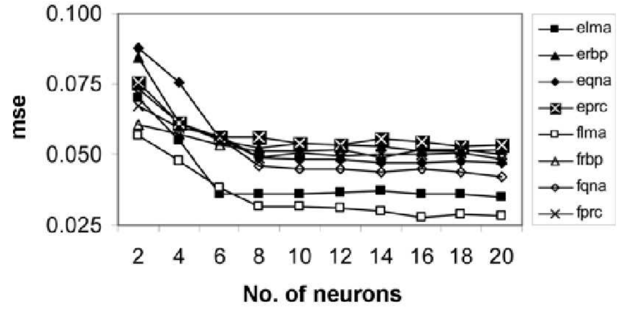

Figure 4. Performance of the best algorithms.

of network architecture and learning functions used.

The types of training functions and number of neurons in the hidden layer showed a significant difference in the performance of the algorithms. In Figure 3, the changes of the performance for the algorithms as the number of neurons in the hidden layer are varied indicated that after the first 8 neurons, the change in the performance of the other neurons (10 to 20) was not significant. Therefore, 8 was considered as the optimum number of neurons in the hidden layer.

A final comparison of the algorithms was done by considering the performance of the algorithms using gradient decent with momentum as the learning function, and the results are given in Figure 4, where letters e or f have been prefixed before the BP training algorithms to denote that the network architecture used is Elman or feedforward network respectively. From Figure 4, *lma* algorithms which used the feedforward network architecture gave the best performance for elongation prediction algorithm, and was therefore considered as the optimum NN algorithm for the prediction of yarn breaking elongation.

The Relative Importance of Input Factors

The optimum NN algorithm was used to analyze the relative importance of various input factors. We conducted an input saliency test by eliminating one designated input from the NN at a time. The increase in mse value when compared to the mse of the optimum NN was considered as a measure of the relative importance of the eliminated input. The results of the above mentioned saliency test are given in Figure 5.

The most important factors which affected the prediction

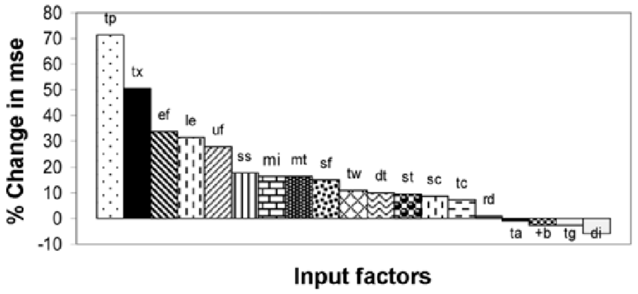

Figure 5. Relative importance of input factors.

of yarn breaking elongation were yarn twist (tp), count (tx) and fiber elongation (ef). Other important factors which showed a strong influence on the prediction of yarn breaking elongation were fiber length (le), length uniformity (uf) and spindle speed (ss). This agrees with previous results obtained using regression models [10]. Douglas [11] also reported that fiber elongation and length are highly correlated with yarn breaking strength for cotton ring spun yarn.

Cotton ring spun yarns are twisted to induce lateral forces which act by means of friction to prevent fibers from slipping over one another. The yarn twist causes the fibers to be rotated so that they make an angle with the yarn axis. This angle increases as the twist is increased. As discussed by Lawrence [18], yarn breakage in short staple ring spun yarn occurs either due to fiber slippage or fiber breakage. When yarn breaks due to fiber slippage, the hitherto twisted fibers have to be straightened. The straightening action has an effect of increasing the effective length of the fiber which in turn increases yarn length. This could be the reason yarn twist showed a strong impact during the prediction of yarn breaking elongation. Since yarn count is highly correlated with yarn twist it therefore came as no surprise that yarn count also showed a strong impact during the prediction of yarn breaking elongation. As mentioned earlier when yarn breaks due to fiber slippage, there is need for the straightening of the twisted fibers before they can slip over one another. This process of straightening the fibers is affected by other factors which include fiber length, length uniformity, short fiber index, traveler weight, ring spinning draft and Spinning consistency index. A longer fiber with better length uniformity, lower short fiber index and a higher spinning consistency index will give a higher increase in length when stretched. Proper combination of ringframe draft and traveler weight will lead to a better utilization of fiber length which will in turn lead to higher yarn breaking elongation.

When yarn breaks due to fiber breakage, the individual fibers are first stretched before breaking. The process of stretching the fiber as discussed above will be affected by other machine parameters and fiber properties. When the straightened fiber is further stretched until it breaks, the change in its length is defined as fiber elongation. The change in fiber length will cause a corresponding increase in yarn length. This could be the reason fiber elongation showed a strong impact during the prediction of yarn breaking elongation. It is noteworthy that trash grade, trash area, fiber yellowness and ring diameter showed a negative impact on the prediction of yarn breaking elongation. This finding calls for further investigation.

Conclusion

The design of yarn breaking elongation prediction algorithms was studied where the inputs were four spinning process parameters (spindle speed, ring diameter, traveler weight and spinning draft), HVI cotton characteristics, yarn twist and count. The output of the network was yarn breaking elongation. The performance of the algorithms was studied under the following design conditions: network architecture (two types), weight/bias learning functions (two types), transfer functions (two types), backpropagation training functions (five types) and the number of neurons in the hidden layer (10 levels). The algorithms performance indicated that there was a significant difference in the type of transfer and training functions used. For the two transfer functions used, purelin performed far much better than logsig function. Among the five BP training functions, Variable learning rate showed significantly lower performance while Levenberg-Marquardt showed the best performance. The other three (resilient backpropagation, Quasi-Newton and Polak-Ribiere conjugate gradient Update) were in between. The best performing algorithm for the prediction of yarn elongation had feedforward network architecture, Levenberg-Marquardt training function, purelin transfer function and 8 neurons in the hidden layer. The study of the relative importance of input factors revealed that yarn twist, yarn count, fiber elongation, length, length uniformity and spindle speed, were the six most influential factors.

Acknowledgement

The financial assistance received from China Scholarship Council (CSC), Beijing and University Research Fund (URF), Moi University, Eldoret is hereby acknowledged. We also wish to thank all the textile firms which provided the cotton lint and yarn samples used in this research work.

References

1. D. J. McCreight, R. W. Feil, J. H. Booterbaugh, and E. E. Backe, "Short Staple Yarn Manufacturing", pp.458-462, Carolina Academic Press, Durham, North Carolina, 1997.
2. S. Doonmez and A. Marmarali, *Text. Res. J.*, **74**(12), 1049 (2004).
3. R. D. Anandjiwala and B. C. Goswami, *Text. Res. J.*, **63**(7), 392 (1993).
4. S. Kovacevic, K. Hajdarovic, and A. M. Grancaric, *Text. Res. J.*, **70**(7), 603 (2000).
5. S. D. Kretzschmar, A. T. Ozguney, G. Ozcelik, and A. Ozerdem, *Text. Res. J.,* **77**(4), 233 (2007).
6. P. K. Majumdar and A. Majumdar, *Text. Res. J.,* **74**(7), 652 (2004).
7. F. M. Ham and I. Kostanic, "Principles of Neurocomputing for Science & Engineering", pp.132-135, 222-226, China Machine Press, Beijing, 2003.
8. M. T. Hagan, H. B. Demuth, and M. Beale, "Neural Network Design", China Machine Press, Beijing, 2002.
9. S. M. Ishtiaque, R. S. Rengasamy, and A. Ghosh, *Indian J. Fibre Text. Res*., **29**, 190 (2004).
10. E. Mustafa and U. H. Kadoglu, *Text. Res. J.*, **76**(5), 360 (2006).
11. K. Douglas, *Uster News Bulletin*, **38**, 23 (1991).
12. W. Zurek, I. Frydrych, and S. Zakrzewski, *Text. Res. J.*, **57**(8), 439 (1987).
13. L. A. Fiori, J. E. Sands, H. W. Little, and J. N. Grant, *Text. Res. J.*, **26**(7), 553 (1956).
14. W. P. Virgin and H. Wakeham, *Text. Res. J.*, **26**(3), 177 (1956).
15. H. B. Demuth, M. Beale, and M. T. Hagan, "Neural Network Toolbox, For Use with MATLAB", User's Guide Version 4, The MathWorks Inc., Natick, 2005.
16. J. L. Elman, *Cognitive Science*, **14**, 179 (1990).
17. E. Mizutani and J. S. R. Jang in "In Neuro-Fuzzy and Soft Computing", (J. S. R Jang, C. T. Sun, and E. Mizutani Eds.), pp.129-172, Prentice Hall, Upper Saddle River, 1997.
18. C. A. Lawrence, "Fundamentals of Spun Yarn Technology", pp.359-406, CRC Press LLC, Florida, 2003.