MODIFIED BACKPROPAGATION TRAINING ALGORITHM WITH FUNCTIONAL CONSTRAINTS

T. Kathirvalavakumar¹ and S. Jeyaseeli Subavathi²

¹Department of Computer Science, V. H. N. S. N. College, Virudhunagar – 626001, Tamilnadu, India
E-mail: kathirvalavakumar@yahoo.com
²Department of Information Technology, Sri Kaliswari College, Sivakasi – 626130, Tamilnadu, India
E-mail: jsabavathi@yahoo.co.in

Abstract: In this paper, a modified backpropagation learning algorithm with functional constraints is proposed to increase the convergence speed and to have better performance. New cost function is proposed for output layer and hidden layer separately. In the proposed algorithm, the additional functional constraints namely, penalty term at hidden layer, weight decay term at output layer are incorporated into cost function having linear and nonlinear error terms. New cost function increases the convergence speed of the training of single hidden layer neural network. The efficiency of the algorithm is shown in terms of number of epochs and time by various simulation results.

Keywords: Modified standard back propagation – Penalty term – Weight decay term – Linear error – Nonlinear error – Functional constraints.

I. INTRODUCTION

In recent years, many neural network models have been proposed for pattern classification, function approximation and regression problems. But the most popular one is the class of multilayer feedforward networks [1-3]. Backpropagation (BP) learning algorithm is a supervised learning method extensively applied in the training of multilayered feedforward neural networks (FNN) [4,5]. This algorithm is widely used because of its simplicity and low computational complexity. But it has main two drawbacks. First, the learning takes long time to converge. Secondly, it falls into local minima which may lead to occurrence of premature saturation sometimes referred as “flat spot” problem [6,7].

Recently, many researchers have paid more attention to improve the performance of BP. Some focused on choosing better energy function and choosing suitable learning rate and momentum [8-12]. Modifications to BP algorithms are suggested to achieve global minimum and to increase convergence speed [10, 11, 13].

To design fast algorithm, Abid et al. proposed a new alternative algorithm by minimizing sum of squares of linear and nonlinear errors for all output units and for current pattern [14]. But in this algorithm the improper choice of design parameters may cause undesirable convergence behaviour. Kathirvalavakumar and Thangavel proposed a new efficient learning algorithm for training single hidden layer FNN [13]. The hidden layer and output layer was trained separately to speed up the convergence. In order to have better generalization capability, many constrained learning algorithms with functional constraints into neural networks have been proposed [15-20]. Jeong et al. proposed adaptive learning algorithms namely Hybrid I and Hybrid II with additional functionality based on the first and second order derivatives of neural activation at hidden layers [21]. The additional cost term penalizes the input-to-output mapping sensitivity in the course of training. This algorithm increases problem complexity and improves generalization capability. Moreover, two algorithms are offline learning and the computational requirements are large.

Han et al. proposed two improved constrained learning algorithms called Improved Hybrid I and Improved Hybrid II based on Hybrid I and Hybrid II. [22]. These two algorithms are online learning algorithms. The cost term of first improved algorithm is selected based on the first order derivatives of neural activation at hidden layers while second improved algorithm is selected based on second order derivatives of neural activation at
hidden layers and output layer. They also proposed two modified constrained learning algorithms First new LA and Second new LA to obtain faster convergence rate [23]. The additional cost terms of the first algorithm are selected based on the first order derivatives of the activation functions of the hidden neurons and second order derivatives of the activation functions of the output neurons while the additional cost terms of second one are selected based on the first order derivatives of activation functions of the output neurons and second order derivatives of the activation functions of the hidden neurons.

In this paper, a modified BP algorithm with additional functional constraints in cost function to train single hidden layer neural network is proposed. This algorithm is referred as MBFC. We have proposed new cost function by incorporating functional constraints proposed by Han et al into the cost function proposed by Abid et al [14,23]. The proposed algorithm incorporates the first order derivatives of the activation functions of the hidden neurons and the second order derivatives of the activation functions of the output neurons into sum of linear and nonlinear quadratic errors of output neurons. Since the proposed training algorithm do not depend on the nonlinearity approximation [14], it converges very faster than the original hybrid algorithms [14,21,23] which can be seen through the simulation results. The proposed training method is presented in section 2. Section 3 describes the simulation results of the selected examples.

II. TRAINING METHOD
Consider a feedforward neural network with one input layer namely F, one hidden layer H and one output layer L as in Figure 1.

Let X=(x_i) be the input vector and Y=(y_j) be the output vector. w_{ij} is the weight connection between output neuron j and hidden neuron i. For all the neurons at output layer L and hidden layer H, the activation function used is

\[
f(x) = \frac{1-e^{-x}}{1+e^{-x}}
\] (1)

The function has the following property

\[
f''(x) = -f(x)f'(x)
\] (2)

where f'(x) is the derivative of the activation function f(x). The linear and nonlinear outputs for a neuron j at output layer L respectively given by Abid et al [14] are:

\[
u_j^L = \sum_{i=1}^{n_H} w_{ji} y_i^H
\] (3)

where n_H is number of neurons at hidden layer H.

\[
f(u_j^L) = \frac{(1-e^{-u_j^L})}{(1+e^{-u_j^L})} = y_j^L
\] (4)

The nonlinear and linear errors are respectively given by

\[e_j^L = d_j^L - y_j^L\] (5)
\[e_j^L = ld_j^L - u_j^L\] (6)

where

\[ld_j^L = f^{-1}(d_j^L)\] (7)

Here d_j^L and y_j^L respectively are desired and current output for j\(^{th}\) unit in the L\(^{th}\) layer. Linear output error is calculated by inverting the output layer nonlinearity which increases the convergence speed [14].

The cost function with linear and nonlinear error term [14] for the j\(^{th}\) neuron and for the current pattern is

\[
E_p = \frac{1}{2}(e_j^L)^2 + \lambda \frac{1}{2}(e_j^L)^2
\] (8)

where \(\lambda\) is a weighting coefficient.

One can improve the robustness of the algorithm by reducing input-to-output mapping values. If the output value is very low then it is advantageous for robust classification and...
approximations even for noisy input data [24]. When the value of \( u_j^i \) becomes larger, the value of its derivative \( f'(u_j^i) \) may become smaller sharply. As a result, the low input-to-output mapping will be achieved. The errors can be reduced by low input-to-output mapping and by introducing first order derivative into cost function [24]. So the proposed algorithm has penalty term defined by Jeong and Lee [21] at hidden layer in MBFC in addition with the cost function having linear and nonlinear term, which is defined as follows:

\[
Penalty_{term} = \frac{1}{n} \sum_{i=1}^{n} f'(u_j^i) \tag{9}
\]

where \( n \) represents number of neurons in that layer. In order to obtain good generalization capability, one can reduce the network complexity by incorporating the weight decay term \( E_{w}(w) = \frac{1}{2} \sum_{j} w_j^2 \) into sum-of-square error cost function [21]. This term based on second order derivative function penalizes large weights and high frequency noisy components and gives small weights [21]. The weight decay term [21] to be included at output layer in MBFC is defined as follows:

\[
Weight_{Decay_{term}} = \frac{f'}{2} \sum_{j} w_j^2 \tag{10}
\]

Upon saturation of activation, it produces only large weights and the derivative \( f' \) controls the weights during learning process [21].

A. MBFC

To obtain a more effective cost term to increase the convergence speed, a new cost function is proposed here based on linear and nonlinear error with additional weight decay term in the output layer and penalty term at hidden layer.

The new cost function \( E_p \) is defined as

\[
E_p = \sum_{i} \frac{1}{2} \lambda \sum_{j} (e_j^p)^2 + \sum_{j} \gamma_j f'(u_j^p) + \frac{1}{2} \sum_{j} f'(u_j^p) \left[ \frac{1}{2} \sum_{j} (w_j^p)^2 \right] \tag{11}
\]

where \( \lambda \) is weighting coefficient, \( \gamma_j \) and \( \gamma_j \) represents penalty constant at hidden layer and weight decay constant at output layer respectively. \( n_i \) denotes the number of neurons at output layer and \( n_H \) denotes the number of neurons at hidden layer.

The cost function \( E_p \) denotes the corresponding cost function for the \( p \)th stored pattern. First term on the right side of the (11) denotes nonlinear error term at output layer, second term denotes linear error term at output layer, third term denotes penalty term at hidden layer and fourth term denotes weight decay term at output layer.

1. Learning of the Output Layer

The network is trained by gradient descent method. In the output layer both linear and nonlinear errors are known. So the weight update rule for the output layer is

\[
\Delta w_j = -\mu_l \frac{\partial E_p}{\partial w_j} \tag{12}
\]

\[
\Delta w_j = \mu_l e^{l}_{ji} \frac{\partial y_j^l}{\partial w_j} + \mu_l \lambda e^{l}_{ji} \frac{\partial u_j^l}{\partial w_j} - \mu_l \gamma_j f''(u_j^l)(w_j^l)
\]

\[
\Delta w_j = \mu_l e^{l}_{ji} \frac{\partial y_j^l}{\partial w_j} + \mu_l \lambda e^{l}_{ji} y_j^l - \mu_l \gamma_j f''(u_j^l)(w_j^l)
\]

\[
\Delta w_j = \mu_l e^{l}_{ji} f'(u_j^l) y_j^l - \mu_l \gamma_j f''(u_j^l)(w_j^l)
\]

\[
\Delta w_j = \mu_l e^{l}_{ji} f'(u_j^l) y_j^l - \mu_l \gamma_j f''(u_j^l)(w_j^l)
\]

2. Learning of the hidden layer

In the hidden layer, both the linear and nonlinear errors are unknown and must be calculated. The weight updating equations for the hidden layer \( H \) is

\[
\Delta w_j = -\mu_H \frac{\partial E_p}{\partial w_j} \tag{13}
\]

\[
\Delta w_j = \mu_H e^{H}_{ji} \frac{\partial y_j^H}{\partial w_j} + \mu_H \lambda e^{H}_{ji} \frac{\partial u_j^H}{\partial w_j} - \mu_H \gamma_j f''(u_j^H)(w_j^H)
\]

\[
\Delta w_j = \mu_H e^{H}_{ji} f'(u_j^H)(w_j^H)
\]

\[
\Delta w_j = \mu_H e^{H}_{ji} f'(u_j^H)(w_j^H)
\]
\[ \Delta w_{ji} = \mu_i e_{ij}^H \frac{\partial y_i^H}{\partial w_{ji}} + \mu_i \lambda e_{ij}^H y_i^H - \mu_i \frac{y_i^H}{n_i} f''(u_{ij}^H) y_i^f \]

\[ \Delta w_{ji} = \mu_i e_{ij}^H f'(u_{ij}^H) y_i^f + \mu_i \lambda e_{ij}^H y_i^f - \mu_i \frac{y_i^H}{n_i} f''(u_{ij}^H) y_i^f \]

where

\[ e_{ij}^H = \sum_{r=1}^{n_i} f'(u_{ir}^H) e_{ir}^H w_{nj}^f \]

\[ e_{ij}^H = f'(u_{ij}^H) \sum_{r=1}^{n_i} e_{ij}^H w_{nj}^f \]

are nonlinear and linear terms.

B. Algorithm

In the proposed algorithm, single hidden layer network is considered. The network is trained using sequential mode. Network structure is defined first and the learning parameters \( \mu_H, \mu_L \) and \( \lambda \) are initialized according to the convergence of the problem. The new constants \( \gamma_L \) and \( \gamma_H \) are initialized to small values between 0 and 1 which play an important role in minimizing the error. Then the network is trained with input values and the corresponding change of weight for both hidden and output layer is determined by calculating linear error, nonlinear error, penalty term and decay term using new cost function for the selected pattern. Then the new weight for both output neurons and hidden neurons are calculated using the equation :

\[ w_{ji}^k (t + 1) = w_{ji}^k (t) + \Delta w_{ji}^k \]

where \( t \) represents iteration and \( k \) represents layer. This training process is repeated for all the patterns. Then the network error is calculated. The above procedure is repeated until desired accuracy is obtained.

Step 1: Initialization

Define network structure and assign initial weights randomly. Assign values randomly for \( \mu_H, \mu_L \) and \( \lambda \). Choose small values randomly for \( \gamma_L \) and \( \gamma_H \).

Step 2: Training pattern

Select a pattern to be processed into the network.

Step 3: Find the network output

Step 4: Error signals and Functional constraints

For the output layer \( L \) calculate

- nonlinear and linear errors using (5) and (6).
- change of weight using (13).

For the hidden layer \( H \), compute

- nonlinear error and linear error using equations (15) and (16).
- change of weight using (14).

Step 5: Weight updation

Update weights of output and hidden layer neurons using (17).

Step 6: Repeat step 2 to 5 for all the patterns.

Step 7: Evaluate network error with new weights.

Step 8: Repeat the steps 2 to 8 until termination condition is reached.

III. SIMULATION RESULTS AND DISCUSSIONS

Effectiveness of the proposed algorithms are shown by simulating the problems namely Noisy sin function approximation and Sin function problem. using a language C on a Pentium IV with 2.40 GHz. The results of the selected problems are compared with the results obtained by executing the algorithms BP, Modified Backpropagation algorithm (MBP) proposed by Abid et al, Levenberg-Marquardt algorithm [25,26], Quickprop algorithm [27], First new LA and Second new LA proposed by Han et al. [14,23] on the same system. Selected problems were executed with 50 different initial weights for different network structure. Then the average training mean squared error (MSE), testing MSE and time for 200 epochs are taken for comparison. To statistically compare the prediction accuracies standard deviation for mean squared error of testing data (SDMSETD) is calculated and summarized in the tables. For all the problems the weights are initialized from the range [-1, 1]. The parameters \( \mu_H, \mu_L, \lambda, \gamma_L \) and \( \gamma_H \) are chosen by trial and error to get good convergence results and these are kept constant for all the problems which is shown in Table I. In all the problems random values are generated using uniform distribution.

B. Noisy sin function Approximation

The function to be approximated has the form of harmonic function \( z = \sin(xy) \). 1000 points are generated in the range [-2,2] and the corresponding
z value is calculated. Then gaussian noise is added into this desired function. 3-5-1, 3-8-1 and 3-10-1 sized networks are trained 50 times with different initial weights.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Parameters Values for all the Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm</strong></td>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>BP</td>
<td>µ = 0.15</td>
</tr>
<tr>
<td>MBP</td>
<td>µ = 0.25, λ = 0.01</td>
</tr>
<tr>
<td>LM</td>
<td>λ = 0.001, β = 0.1</td>
</tr>
<tr>
<td>QuickProp</td>
<td>µ = 1.75, poffset = 0.1, epsilon = 0.55, momentum = 0.9, decay = -0.0001</td>
</tr>
<tr>
<td>First newLA</td>
<td>µH = 0.02, µL = 0.15, γH = 0.01, γL = 0.00001</td>
</tr>
<tr>
<td>Second new LA</td>
<td>µH = 0.01, µL = 0.15, γH = 0.0001, γL = 0.0001</td>
</tr>
<tr>
<td>MBFC</td>
<td>µH = 0.12, µL = 0.14, λ = 0.01, γH = 0.00001, γL = 0.00001</td>
</tr>
</tbody>
</table>

The results obtained are compared with other algorithms and tabulated in Table II. From the table it has been observed that the MSE and time at training and testing are less for the proposed algorithm than the other algorithms as the proposed algorithm filtered the high frequency components. The effects of parameters for different cases are discussed in the Table III. Figure 2 shows the learning curve obtained for the above considered network structures respectively for the proposed algorithm.

The effects of parameters are shown in Table III. Case I : µH = 0.12, µL = 0.14, λ = 0.01 and γH = 0.00001 are kept constant. γL is selected as 0.0001, 0.001, 0.01 and 0.1. It can be seen that the bigger the γH is worse the generalization performance. But when the λ is changed as 1.25 we get the minimum MSE. Case III : µH = 0.12, λ = 0.01, γL = 0.00001 and 001 are kept constant. µL is selected as 1.14, 2.14, 3.14 and 4.14 respectively. It can be seen that the bigger the µL is, the worse the generalization performance. But when the λ is changed as 0.85 we get the minimum MSE. Case IV : µL = 0.14, λ = 0.01, γH = 0.00001 and γL = 0.00001 are kept constant. µH is selected as 1.12, 2.12, 3.12 and 4.12 respectively. It can be seen that the bigger the µH is e worse the generalization performance. But when the λ is changed as 0.9 we get the minimum MSE. Case V : µH = 0.14, µL = 0.12, γH = 0.00001, γL = 0.00001 are kept constant. µL is selected as 0.1, 1.1, 2.1 and 3.1 respectively. It can be seen that the results obtained are more are less same.

**B. Sin Function Problem**

The following sin function is considered for approximation,

\[ z(x, y) = \sin[0.5(x + 5)^2 + (y - 5)^2] + \sin[0.5(x - 5)^2 + (y - 3)^2] / (x + 5)^2 + (y - 5)^2 + (x - 5)^2 + (y - 3)^2 \]

where \((x, y) \in [-20, 20]\). The network structures considered for training this problem is 3-5-1, 3-8-1 and 3-10-1 for all the algorithms. The number of input patterns selected to train and test the network is 300 for all the algorithms. The experiment is carried out 50 times for each algorithm and then its average value is calculated. The results obtained are summarized in the Table IV. It can be seen that the MSE of the proposed learning algorithm for training and testing data set are smaller than the other algorithms.

The time required for the proposed algorithm is less than the other algorithms. Therefore the modified algorithm converge faster than the other algorithms. Figure 3 shows the learning curve obtained for one trial for the proposed algorithm of network structure 3-5-1, 3-8-1 and 3-10-1 respectively.
Table II
Comparison Table for Noisy Sin Function Problem

<table>
<thead>
<tr>
<th>NW Structure</th>
<th>Algorithm</th>
<th>Training MSE</th>
<th>Testing MSE</th>
<th>Time in Secs</th>
<th>SDMSE/ETD</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-5-1</td>
<td>BP</td>
<td>0.046211</td>
<td>0.046211</td>
<td>3.8</td>
<td>0.005145</td>
</tr>
<tr>
<td></td>
<td>MBP</td>
<td>0.046765</td>
<td>0.046765</td>
<td>3.88</td>
<td>0.004341</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0.231688</td>
<td>0.229686</td>
<td>2.72</td>
<td>0.027378</td>
</tr>
<tr>
<td></td>
<td>QuikProp</td>
<td>0.33168</td>
<td>0.330463</td>
<td>3.9</td>
<td>0.003627</td>
</tr>
<tr>
<td></td>
<td>First NewLA</td>
<td>0.050211</td>
<td>0.050211</td>
<td>4.56</td>
<td>0.004916</td>
</tr>
<tr>
<td></td>
<td>Second New A</td>
<td>0.041362</td>
<td>0.041362</td>
<td>2.98</td>
<td>0.004128</td>
</tr>
<tr>
<td>3-8-1</td>
<td>BP</td>
<td>0.030954</td>
<td>0.030954</td>
<td>5.74</td>
<td>0.005804</td>
</tr>
<tr>
<td></td>
<td>MBP</td>
<td>0.032261</td>
<td>0.032261</td>
<td>5.92</td>
<td>0.005695</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0.229601</td>
<td>0.235773</td>
<td>10</td>
<td>0.036397</td>
</tr>
<tr>
<td></td>
<td>QuikProp</td>
<td>0.049404</td>
<td>0.049403</td>
<td>4.46</td>
<td>0.003823</td>
</tr>
<tr>
<td></td>
<td>First NewLA</td>
<td>0.034954</td>
<td>0.034954</td>
<td>5.84</td>
<td>0.0048</td>
</tr>
<tr>
<td></td>
<td>Second New A</td>
<td>0.041054</td>
<td>0.041054</td>
<td>6.02</td>
<td>0.006221</td>
</tr>
<tr>
<td></td>
<td>MBFC</td>
<td>0.028637</td>
<td>0.028637</td>
<td>4.2</td>
<td>0.003022</td>
</tr>
<tr>
<td>3-10-1</td>
<td>BP</td>
<td>0.024701</td>
<td>0.024701</td>
<td>7.14</td>
<td>0.004168</td>
</tr>
<tr>
<td></td>
<td>MBP</td>
<td>0.023072</td>
<td>0.023072</td>
<td>7.2</td>
<td>0.004694</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0.239</td>
<td>0.215736</td>
<td>9.08</td>
<td>0.014553</td>
</tr>
<tr>
<td></td>
<td>QuikProp</td>
<td>0.312004</td>
<td>0.30115</td>
<td>5.38</td>
<td>0.007504</td>
</tr>
<tr>
<td></td>
<td>First NewLA</td>
<td>0.028091</td>
<td>0.028091</td>
<td>7.4</td>
<td>0.005129</td>
</tr>
<tr>
<td></td>
<td>Second New A</td>
<td>0.029456</td>
<td>0.029456</td>
<td>7.5</td>
<td>0.005007</td>
</tr>
<tr>
<td></td>
<td>MBFC</td>
<td>0.021354</td>
<td>0.014354</td>
<td>6.36</td>
<td>0.004019</td>
</tr>
</tbody>
</table>

Table III
The Effects of Parameters for Noisy Sin Function Approximation

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Mean squared error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_H = 0.12$, $\mu_L = 0.14$</td>
<td>$\gamma_H = 0.0001$ $\gamma_L = 0.0001$</td>
</tr>
<tr>
<td>$\lambda = 0.01$, $\gamma = 0.00001$</td>
<td>MSE = 0.023513 MSE = 0.021112 MSE = 0.031444 MSE = 0.048740 MSE = 0.024970</td>
</tr>
<tr>
<td>$\mu_H = 0.12$, $\mu_L = 0.14$</td>
<td>$\gamma_H = 0.0001$ $\gamma_L = 0.0001$</td>
</tr>
<tr>
<td>$\lambda = 0.01$, $\gamma = 0.00001$</td>
<td>MSE = 0.027385 MSE = 0.044985 MSE = 0.070261 MSE = 0.129631 MSE = 0.076976</td>
</tr>
<tr>
<td>$\mu_H = 0.12$, $\lambda = 0.14$</td>
<td>$\mu_L = 0.01$ $\mu_L = 0.14$</td>
</tr>
<tr>
<td>$\gamma_H = 0.00001$, $\gamma_L = 0.00001$</td>
<td>MSE = 0.023541 MSE = 0.027854 MSE = 0.026540 MSE = 0.027419</td>
</tr>
</tbody>
</table>
IV. CONCLUSION

A modified backpropagation algorithm MBFC with functional constraints is proposed. The functional constraints weight decay term and penalty term are included in the cost function with the linear and nonlinear error terms. The new cost function included in the algorithm increases the convergence speed and provides better generalization capability. The proposed algorithm MBFC converged in minimum number of time compared with cost function with linear and nonlinear error terms. Also it has been shown that the result of the proposed algorithm is better than First new LA and Second new LA. The learning parameters $\mu^L$, $\mu^H$, $\lambda$ and new constants $\gamma_i$ and $\gamma_o$ played an important role in reaching minimum convergence. It has been observed that when the penalty constant and decay constant is bigger, we get the worse generalization performance. But it is controlled by the weighting coefficient $\lambda$. The value of $\lambda$ must be selected with great care. Because larger $\lambda$ value leads to divergence of problem. The value of the parameters are selected through trial and error and kept constant for better comparison. Simulation of the selected problems show the efficiency of the proposed algorithm in terms of MSE, SDMSETD and time.

REFERENCES


<table>
<thead>
<tr>
<th>NW Structure</th>
<th>Algorithm</th>
<th>Training MSE</th>
<th>Testing MSE</th>
<th>Time in Secs</th>
<th>SDMSETD</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-5-1</td>
<td>BP</td>
<td>0.001424</td>
<td>0.00242</td>
<td>1.8</td>
<td>0.001427</td>
</tr>
<tr>
<td></td>
<td>MBP</td>
<td>0.001349</td>
<td>0.002164</td>
<td>1.34</td>
<td>0.001072</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0.0257</td>
<td>0.026154</td>
<td>2.78</td>
<td>0.025359</td>
</tr>
<tr>
<td></td>
<td>QuikProp</td>
<td>0.036521</td>
<td>0.036547</td>
<td>4.21</td>
<td>0.045871</td>
</tr>
<tr>
<td></td>
<td>First New LA</td>
<td>0.001348</td>
<td>0.002001</td>
<td>1.26</td>
<td>0.001134</td>
</tr>
<tr>
<td></td>
<td>Second New LA</td>
<td>0.001347</td>
<td>0.002013</td>
<td>1.3</td>
<td>0.001086</td>
</tr>
<tr>
<td></td>
<td>MBFC</td>
<td>0.001124</td>
<td>0.001104</td>
<td>1.14</td>
<td>0.000981</td>
</tr>
<tr>
<td>3-8-1</td>
<td>BP</td>
<td>0.001462</td>
<td>0.002481</td>
<td>1.9</td>
<td>0.001345</td>
</tr>
<tr>
<td></td>
<td>MBP</td>
<td>0.001395</td>
<td>0.001937</td>
<td>1.72</td>
<td>0.000896</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0.032735</td>
<td>0.033197</td>
<td>10.18</td>
<td>0.035653</td>
</tr>
<tr>
<td></td>
<td>QuikProp</td>
<td>0.045789</td>
<td>0.049654</td>
<td>11.25</td>
<td>0.045587</td>
</tr>
<tr>
<td></td>
<td>First New LA</td>
<td>0.001352</td>
<td>0.001645</td>
<td>1.74</td>
<td>0.001048</td>
</tr>
<tr>
<td></td>
<td>Second New LA</td>
<td>0.001341</td>
<td>0.001889</td>
<td>1.8</td>
<td>0.000967</td>
</tr>
<tr>
<td></td>
<td>MBFC</td>
<td>0.001115</td>
<td>0.001176</td>
<td>1.6</td>
<td>0.000138</td>
</tr>
<tr>
<td>3-10-1</td>
<td>BP</td>
<td>0.001467</td>
<td>0.003137</td>
<td>2.56</td>
<td>0.001714</td>
</tr>
<tr>
<td></td>
<td>MBP</td>
<td>0.001466</td>
<td>0.002007</td>
<td>2.44</td>
<td>0.000975</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0.023541</td>
<td>0.024386</td>
<td>19.1</td>
<td>0.030394</td>
</tr>
<tr>
<td></td>
<td>QuikProp</td>
<td>0.044112</td>
<td>0.045621</td>
<td>20.15</td>
<td>0.051478</td>
</tr>
<tr>
<td></td>
<td>First New LA</td>
<td>0.001336</td>
<td>0.001849</td>
<td>2.42</td>
<td>0.00108</td>
</tr>
<tr>
<td></td>
<td>Second New LA</td>
<td>0.00134</td>
<td>0.001867</td>
<td>2.45</td>
<td>0.001208</td>
</tr>
<tr>
<td></td>
<td>MBFC</td>
<td>0.001113</td>
<td>0.001137</td>
<td>2.1</td>
<td>0.00101</td>
</tr>
</tbody>
</table>


