IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 3, Ver. VII (May – Jun. 2015), PP 63-69
www.iosrjournals.org

A New Classifier Based onRecurrent Neural Network Using Multiple Binary-Output Networks

Mohammad Fazel Younessy[1] and Ehsanolah Kabir[2]
[1]*Department of Electrical and Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran*
[2]*Department of Electrical Engineering, TarbiatModarres University, Tehran, Iran*

Abstract:*Conventional classifiers have several outputs equal to the number of classes (N) which imposes so much complexityto the classifiers,both in training and testing stages. Instead of using one extra-large and complicated classifier, we created N number of simple binary Recurrent Neural Network with true/false outputs. Each network is responsible of recognizing its own trained class.We also proposed adecision layer added to each network, making a final decision from a sequence of outputs. We test our system on features extracted from Iranshahr database, which is a set of 17,000 black-and-white handwritten images of Iranian city names. Experimental results authenticate the effectiveness of the proposed method.*
Keywords: *Recurrent Neural Network,RNN, Classifier, Binary Network.*

I. Introduction

Classifiers are essential tools for the applications that need to find a specific type of result among a given set of results, such as offline/online handwritten/typewritten word recognition[1][2][3][4], speech recognition[5][6][7][8], and medical diagnosis systems[9][10][11]. Examples of such classifiers are Artificial Neural Network (ANN), Hidden Markov Models (HMM),support vector machine (SVM), and k-nearest neighbor (k-NN). Generally, a classifier has a training phase, in which some input samples with known results are fed into the classifier and the classifier internal weights would be trained by such data. Afterwards, in the testing phase, the input samples with unknown output results would be applied to the classifier, and the results would be evaluated.

Conventional classifiers are consisted of high-quantity outputs, which force the classifier to be internally complicated and huge. Such classifiers all suffer from poor results and slow timing, both in training stage and testing phase. In our paper, we have proposed a new approach to conquer such problems. Instead of such complicated networks, we make multiple number of networks, each only consisted of a True/False output. It means, instead of one classifier with N number of outputs, we use N classifiers with simple outputs.Each classifier would be trained to have true result only by one class. This leads to small and simple classifiers with faster response time.

One of the most common classifiers used in practice, is ANN. An ANN is consisted of units called neurons, which are arranged in layers. Typically, ANN converts an input vector (features) to outputs. The classifier type that we used in our work is a Recurrent Neural Network (RNN). A simple RNN is a special type of ANN, with oneor more memory layer(s) added to its feedforward layers.The memory layer is a feedback path, which makes the recurrence of the system.

A typical ANN has an input vector, which makes one output (vector), whereas, RNN has a sequence of input vectors, which makes series of outputs. Hence, we added a decision layer to the output of the network to make the final decision (true or false) out of the sequence of inputs. Some other researchers also tried to make a classifier by RNN system. However, they all seem to be some complex mathematical problem, which would be added to the inherent complicated nature of conventional networks. For example, in [12], the authors proposed an RNN classifier with several recurrent layers added to each other.A valuable system has been presented in[13][14][15]. This system has addeda decision layer called connectionist temporal classification (CTC) to the output of the RNN and tried to train the network by a differentiable function. A rapid temporal sampling combined by an RNN classifier has been used for measurement of areal snow coverage in [16]. The authors in [17] replaced the simple neurons in RNN with a more complicated neuron set, and called it long-short term memory (LSTM) system, and used it in manuscript recognition.

The rest of the paper would be organized as follows. In next section we will describe the proposed classification method in detail.In Section 3, we will show our experimental results on a real database. Finally we conclude our work in section four, and show a possible future work in that section.

II. Proposed ClassifierSystem

In this section we will focus on details of our proposed system. The task of the system is to make a classifier to classify N outputs. Block diagram of the system is shown inFig. 1. As this figure shows, the system is composed of a training phase and testing phase.

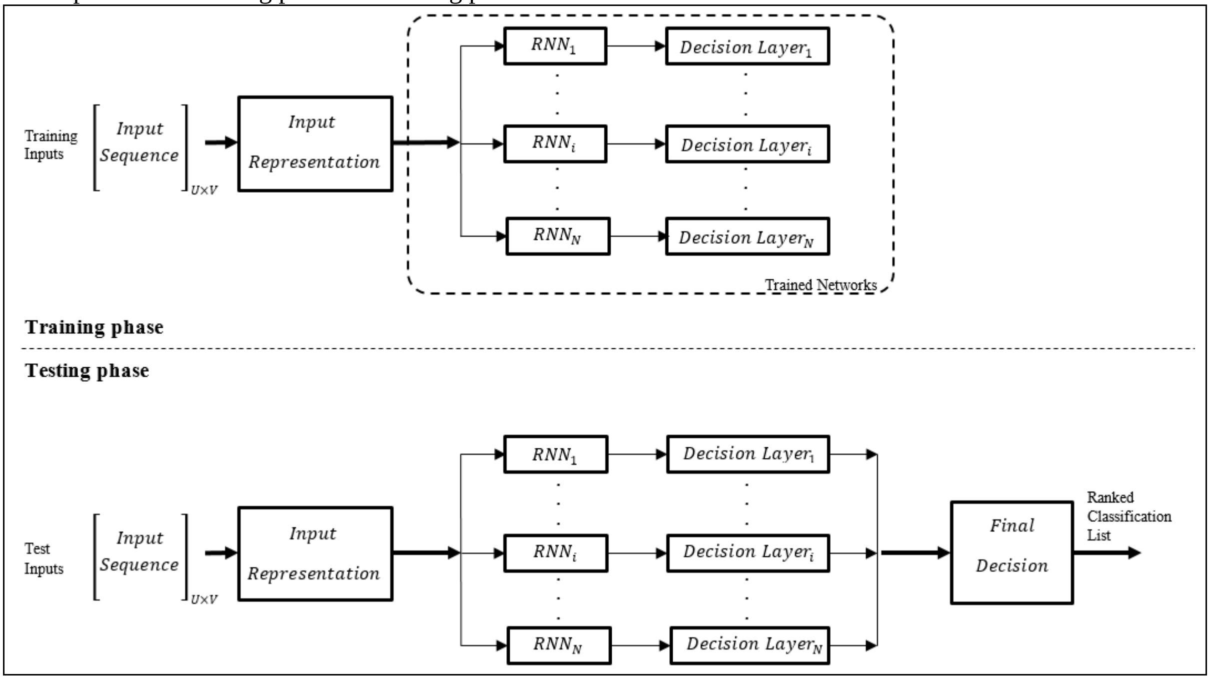

Fig. 1: Block diagram of proposed system.

In training stage, at first, input data vector would be represented to be fed properly to the network. As mentioned, in our system, instead of using one huge classifier with N outputs, N number of simple classifiers would be used. Each network is an RNN classifier with a decision layer added to the output layer. The RNN set would be expressed as below:

$$\text{RNNs}: \{\text{RNN}_1, \ldots, \text{RNN}_\text{N}\} \qquad (1)$$

Each RNN classifier is responsible for one class. Each classifier will have “True/False” outputs. This means in test phase, we expect that if samples of class i is applied to the RNN_i, then output should be true and if other samples from another classes are fed into the network, then output should be false. This is an interesting phenomenon, because each network would be simple, and only is trained for one class.

RNN networks normally need a sequence of inputs and make a sequence of outputs. Our task is to make a classifier, so the network should have one output (true or false), not a train of outputs. As a result, we made a decision layer added to input of each RNN. Decision layer would use the sequence of outputs from each RNN and finally make a final decision which might be true or false.

In the testing phase, similar to the training phase, we send the input vectors to the representation unit, and then to each RNN classifiers (Recall that each RNN has a decision layer at the output). After this stage, we have a list of outputs derived from the RNN classifiers. We sort the outputs in the final decision section and output of the system would be a sorted list of probabilities of each class.

In the following sub-sections we will discuss each part of the main block diagram in detail.

A. Input Representation

Input data to the RNN network is a sequence of vectors. Let’s define F_i as an input vector with the dimension of $\text{U} \times 1$. A sequence of F_i, with length of V would make the input data of the network. Note that, interestingly,V is dependent on the input vector sequence’s length, and may be different from sample to sample. Equation below shows the idea:

$$\text{F}_\text{i} = \begin{bmatrix} \text{f}_1 \\ \vdots \\ \text{f}_\text{U} \end{bmatrix} \qquad (2)$$

Before sending the input vectors to the networks, we standardized the inputs. We forced the inputs to have a mean (μ) of 0, and standard deviation (σ) of 1. This procedure would make data optimum for the activation functions inside the RNNs, and main information will not be changed, however, performance of system would improve significantly[18].

B. Network Architecture

Normally, any arbitrary network can be made with any configuration. We have chosen a heuristic network topology which is presented in Fig. 2.As this figure shows, the network is consisted of two feedforward layers, hidden layer and output layer.The network also has a feedback (recurrent) layer, with delay of T_{delay} which has been fed back to the hidden layer (layer 1). As mentioned earlier, output layer of each RNN has two outputs, one showing the state of being true, and the other one the state of being false.

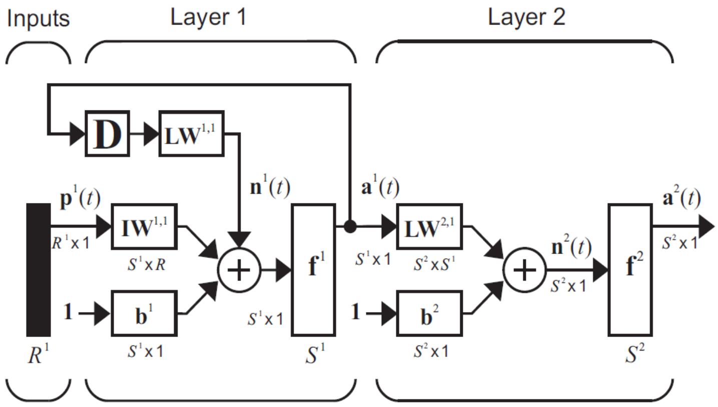

Fig. 2: Topology of the RNN (Figure from[19]).

For each feedforward layer we use Tanh-Sigmoid transfer function[20], which has a output between -1 and +1. This makes the final outputs of the network to be in the range of $[-1,+1]$, which gives us the probability of the true/false instead of a hard output of being discrete 0 or 1.

C. Decision Layer

Naturally, an RNN has a train of input vectors, and this leads to a sequence of 2-dimensional (2D) vectors on the output. As a classifier we need to concatenate the result sequence and make a decision at the end of output sequence. Hence we made a decision layer at the end of each RNN.As shown inFig. 3, a decision block is added to the RNN block.

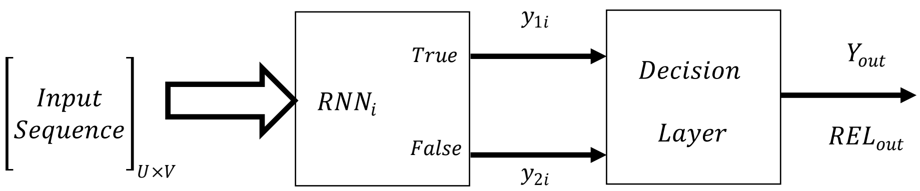

Fig. 3: Single RNN with decision layer added to the outputs.

Suppose V is the length of input sequence. We call the outputs of the RNN as $y_{1i}$and y_{2i}, which are the true and false outputs of the RNN respectively. We have:

$$Y_i = \begin{Bmatrix} 1 & ,\text{if } y_{1i} > y_{2i} \\ 0 & ,\text{else} \end{Bmatrix}$$

$$Y_{out} = \sum_{i=T_{delay}}^{V} \frac{Y_i}{V - T_{delay}} \cdot \frac{\omega_i}{\Omega} \times 100 \qquad (3)$$

$$\Omega = \sum_{i=T_{delay}}^{V} \omega_i$$

Looking at (3), Y_i is the decision made by a single vector at each output of the RNN.Y_{out} is the final decision of the network taken from train of 2D outputs. Please note that T_{delay} is the network recurrence time, and is a system parameter. Normally outputs before that time is of less value, so we don't consider them.As above formula shows, output ofdecision layer, Y_{out}, is in fact the ratio of the number of true outputs to the sequence length. We added a weight coefficient, ω_i, to the calculation to improve the results. These weights, should go from a small value to bigger values, as the network makes output through the time sequence. This phenomenon guarantees that the network's output at the end of the sequence has more influence on the final decision. This is due to the fact that, at the start of the network sequence evaluation, the recurrent characteristics of the network is not so strong, but, as more input vectors (from the input sequence) are fed into the network, network has more reliable outputs. At the end, to omit the effect of the weights, we have normalized the weights by summation of ω_i, Ω, to remove the effect of the weights.In our systems we used three functions for the weights: $1.\, \omega_i = \log(i)$, $2.\, \omega_i = \exp(i)$, and $3.\, \omega_i = \tanh(i/N)$

As a result of all mentioned, Y_{out} shows the percentage of similarity of each input sequence, to the class that RNN network is trained with. It means, ideally, if a sequence of inputs similar to the specific RNN's class is fed, output would be 100% and for the other sequences, output is 0%.

Because of comparison in the first line of (3), decision made by above formula, is in fact a hard decision. To preserve the soft information of RNN outputs, we created a reliability function, REL_{out}. This functions shows how much valid the output of the decision layer is. We formulate REL_{out} as below:

$$REL_{out} = \frac{1}{2} \sum_{i=T_{delay}}^{V} \frac{|y_{1i} - y_{2i}|}{V - T_{delay}} \cdot \frac{\omega_i}{\Omega} \times 100 \qquad (4)$$

As the above formula shows, REL_{out} is, in fact, the average of distance of true outputs and false outputs, multiplied by the weights. Please note that the coefficient of $\frac{1}{2}$ is due to the fact that y_{1i} and y_{2i} are in the range of $[-1, +1]$.

D. RNN Training

Some special attention should be given to the training method of the RNNs. Generally, in a database, there are a few samples for each class. In our case, each RNN is responsible for a specific class. If we give the few related samples to this RNN, forcing the outputs to true (for backpropagation training), and force the outputs to be false for the many unrelated samples, the network would be finally trained to false results. For examples, 3% of the input samples would train the network to true, and 97% of the other samples would train the network to false outputs. This means network will generate false results at the test time. Hence, we have added the quantity of true samples virtually, by adding some Gaussian noise to the true samples, making them about one-third of the total samples.

To initialize the weights of the network, we used a Gaussian distribution with $(\mu, \sigma) = (0, 0.1)$. Furthermore, we can retrain the network after one successful training phase. Our simulation showed this technic may increase total result by 2-3%, and also retraining more than 3 retraining has no sensible effect. The negative point about the retraining is the time it consumes because of repeating the training phase.

Several algorithms for network training has been proposed in scientific texts, such as Levenberg-Marquardt[21], Bayesian Regularization[22],[23], BFGS Quasi-Newton[24], Gradient Descent[25] and Resilient Backpropagation(Rprop)[26]. Among those, the Rprop seems to work better for us.Rprop algorithm normally uses relatively small amount of memory and is faster than most of the algorithms.

E. RNN Testing

To test the Network, we divided the sample data to 3 sets, 70% for training stage, 15% forvalidation of training phase, and 15% for final test and report.

II. Experimental Results

A. Test Database

To test our system, we have used a real database called Iranshahr. This database is consisted of more than 17,000 samples of binary images of handwritten Iranian city names, which has been also used in [27]. The number of classes (cities) in this database is more than 500, which means each class has about 30 samples. We have used the method in [28] to extract the feature vectors. This method uses the sliding window technic to extract image features inside a frame. Then by shifting the window from right to left of the image, makes a sequence of vectors for each sample image. These features are input vector sequences to our system.

B. Classification Results

Table 1 shows the best classification result obtained by the system. The second row in this table shows the classification rate (the percentage of recognizing correct city classes in the database) and the third row shows the average reliability of the correct results. As this table shows, the recognition rate and reliability of our system is relatively high and acceptable.

Table 1: Classification results

	Top 1	Top 2	Top 5	Top 10
Classification rate	83.9%	84.7%	87.5%	91.1%
Averagereliability	72.3%	73.1%	78.5%	80.5%

C. Effect of Recurrence Delay

A system parameter that has been taken into consideration is the T_{delay} , which is the network recurrence delay. The effect of this parameter has been shown in Fig. 4. This figure shows that for$T_{delay} = 3$, the rate is at its maximum. Please note that, obviously, if we increase T_{delay} , we will miss some recurrence statesand information of the RNN, and thus decreasing the classification rate.

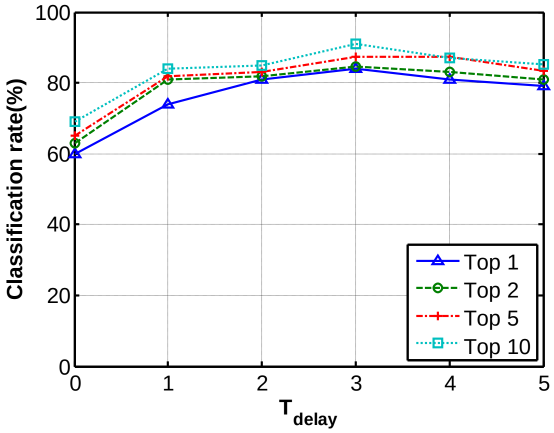

Fig. 4: Effect of T_{delay} on classification rate.

D. Effect of Retraining

Figure below shows the effect of retraining on our system rate. We can see retraining of the network up to 3 times is a good practice, and more than that has almost no effect on system.

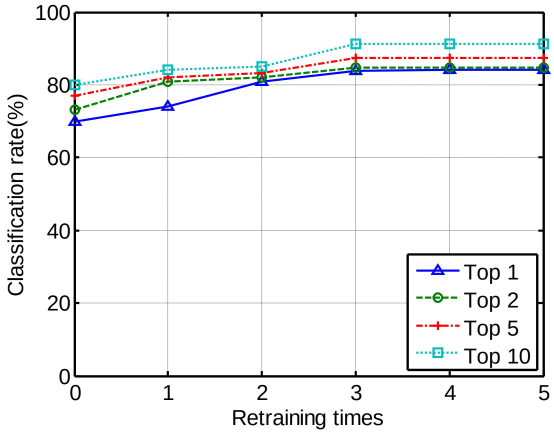

Fig. 5: Effect of retraining times on classification rate.

E. Effect of Weights

We have tested our system by 3 weight types. Table below shows the effect of each weight type on our system. As the table shows, the logarithm function works the best for us and exponential the worst.

Table 2: Effect of weight function on classification rate.

		Classification rate			
		Top 1	Top 2	Top 5	Top 10
Weight type	$\omega_i = \exp(i)$	75.3%	76.0%	79.6%	80.5%
	$\omega_i = \tanh(i/N)$	80.2%	81.1%	84.1%	85.5%
	$\omega_i = \log(i)$	83.9%	84.7%	87.5%	91.1%

F. Comparison with conventional RNN Classifier

To compare our method with the conventional classification method, i.e. using one huge classifier with so many outputs, we trained an RNN network with output layer of 500 neurons (equal to the number of classes in database), and added a decision layer to make the decision. The classification rate was very poor (less than 50%), and shows that our method is strongly better than the conventional classification method which uses RNNs.

III. Conclusion and Future Work

In this paper we proposed a good method of reducing the network complexity both in training and testing stages by making several binary networks instead of one complicated network. We used a Recurrent Neural Network as a classifier and added a decision layer to make the decision. We conducted our test in a real database to show the power of the method and a good recognition rate has been derived from the system.

As a future work, researchers could use our method in the field of handwritten/typewritten/speech recognition systems. Even some extra information, such as length of the inputs,can prune some RNN classifiers, so the final result would be chosen from a reduced set of the networks.

References

[1] Kumawat, P.; Khatri, A.; Nagaria, B., "Comparative Analysis of Offline Handwriting Recognition Using Invariant Moments with HMM and Combined SVM-HMM Classifier," Communication Systems and Network Technologies (CSNT), 2013 International Conference on , vol., no., pp.140,143, 6-8 April 2013.

[2] Maqqor, A.; Halli, A.; Satori, K.; Tairi, H., "Off-line recognition handwriting Arabic words using combination of multiple classifiers," Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in , vol., no., pp.260,265, 20-22 Oct. 2014

[3] Wu Wei; Gao Guanglai, "Online Handwriting Mongolia words recognition based on HMM classifier," Control and Decision Conference, 2009. CCDC '09. Chinese , vol., no., pp.3912,3914, 17-19 June 2009.

[4] Al-Hajj Mohamad, R.; Likforman-Sulem, L.; Mokbel, C., "Combining Slanted-Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.31, no.7, pp.1165,1177, July 2009

[5] Chung-Hsien Wu; Wei-Bin Liang, "Emotion Recognition of Affective Speech Based on Multiple Classifiers Using Acoustic-Prosodic Information and Semantic Labels," Affective Computing, IEEE Transactions on , vol.2, no.1, pp.10,21, Jan.-June 2011.

[6] Tsang-Long Pao; Wen-Yuan Liao; Yu-Te Chen; Jun-Heng Yeh; Yun-Maw Cheng; Chien, C.S., "Comparison of Several Classifiers for Emotion Recognition from Noisy Mandarin Speech," Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on , vol.1, no., pp.23,26, 26-28 Nov. 2007.

[7] Chandrakala, S.; Sekhar, C.C., "Combination of generative models and SVM based classifier for speech emotion recognition," Neural Networks, 2009. IJCNN 2009. International Joint Conference on , vol., no., pp.497,502, 14-19 June 2009.

[8] Kucukbay, S.E.; Sert, M., "Audio-based event detection in office live environments using optimized MFCC-SVM approach," Semantic Computing (ICSC), 2015 IEEE International Conference on , vol., no., pp.475,480, 7-9 Feb. 2015

[9] Deng, Z.; Cao, L.; Jiang, Y.; Wang, S., "Minimax Probability TSK Fuzzy System Classifier: A More Transparent and Highly Interpretable Classification Model," Fuzzy Systems, IEEE Transactions on , vol.PP, no.99, pp.1,1.

[10] Gladence, L.M.; Ravi, T.; Karthi, M., "An enhanced method for detecting congestive heart failure - Automatic Classifier," Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on , vol., no., pp.586,590, 8-10 May 2014.

[11] AlObaidi, A.T.S.; Mahmood, N.T., "Modified Full Bayesian Networks Classifiers for Medical Diagnosis," Advanced Computer Science Applications and Technologies (ACSAT), 2013 International Conference on , vol., no., pp.5,12, 23-24 Dec. 2013

[12] Seong-Whan Lee; Hee-Heon Song, "A new recurrent neural-network architecture for visual pattern recognition," Neural Networks, IEEE Transactions on , vol.8, no.2, pp.331,340, Mar 1997.

[13] Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In Proceedings of the 23rd international conference on Machine learning, pp. 369-376. ACM, 2006.

[14] Graves, Alex, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. "A novel connectionist system for unconstrained handwriting recognition." Pattern Analysis and Machine Intelligence, IEEE Transactions on 31, no. 5 (2009): 855-868.

[15] A. Graves. Supervised Sequence Labelling with Recurrent Neural Networks. Dissertation, Technische Universität München, München, July 2008.

[16] Simpson, J.J.; McIntire, T.J., "A recurrent neural network classifier for improved retrievals of areal extent of snow cover," Geoscience and Remote Sensing, IEEE Transactions on , vol.39, no.10, pp.2135,2147, Oct 2001.

[17] Frinken, Volkmar, Francisco Zamora-Martinez, Salvador Espana-Boquera, Maria Jose Castro-Bleda, Andreas Fischer, and Horst Bunke. "Long-short term memory neural networks language modeling for handwriting recognition." In Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 701-704. IEEE, 2012.

[18] LeCun, L. Bottou, G. Orr, and K. Muller. Ecient backprop. In G. Orrand M. K., editors, Neural Networks: Tricks of the trade. Springer, 1998b.

[19] MathWorks™, (2012). Neural Network Toolbox7User's Guide (R2012b).
[20] Vogl, T.P., J.K. Mangis, A.K. Rigler, W.T. Zink, and D.L. Alkon, "Accelerating the convergence of the backpropagation method," Biological Cybernetics, Vol. 59, 1988, pp. 257–263
[21] Kenneth Levenberg (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares". Quarterly of Applied Mathematics 2: 164–168
[22] MacKay, Neural Computation, Vol. 4, No. 3, 1992, pp. 415–447
[23] Foresee and Hagan, Proceedings of the International Joint Conference on Neural Networks, June, 1997
[24] Gill, Murray, & Wright, Practical Optimization, 1981.
[25] Jan A. Snyman (2005). Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Springer Publishing.
[26] Riedmiller, M., and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", Proceedings of the IEEE International Conference on Neural Networks, 1993.
[27] Bayesteh, E.; Ahmadifard, A.; Khosravi, Hossein, "A Lexicon Reduction Method Based on Clustering Word Images in Offline Farsi Handwritten Word Recognition Systems," Machine Vision and Image Processing (MVIP), 2011 7th Iranian , vol., no., pp.1,5, 16-17 Nov. 2011.
[28] M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Handwritten Farsi (Arabic) Word Recognition: A Holistic Approach Using Discrete HMM," Pattern Recognition, vol. 34, no. 5, pp. 1057 1065, 2001.