Customer Churn Prediction using Recurrent Neural Network with Reinforcement Learning Algorithm in Mobile Phone Users

Zolidah Kasiran ,Zaidah Ibrahim and Muhammad Syahir Mohd Ribuan
Faculty of Computer and Mathematical Sciences,
UiTM 40450 Shah Alam,
Selangor, Malaysia
zolidah@tmsk.uitm.edu.my; zaidah@tmsk.uitm.edu.my; syahir.ribuan@gmail.com

Abstract

The ability to retain customers is an issue in many of the service industries. Industries spend a lot of resources in gaining new customer and trying all their effort to maintain the existing customers and various churn predictor engine has been developed to fulfill this purpose. The objective of this study is to implement Elman Recurrent Neural Network and Jordan Recurrent Neural Network with Reinforcement Learning in predicting probability of mobile phone churning rates. The study had proved that Jordan Recurrent Neural Network gave better accuracy rate than Elman Recurrent Neural Network

Keywords*: Churn engine, Churn Predictor, Q-Learning, ERNN, JRNN*

1. Introduction

The technology in telecommunication has experience a rapid growth that cause the price of the services more affordable. This also had lead to the increase number of mobile user. Nevertheless, telecommunication companies is struggling to retain their current customers due the market liberation that open up more competitor. As business world becomes compact and saturated, gaining new customers is harder than maintaining existing customers, especially in the mobile phone services in which the new technology and services is experiencing a rapid growth.. In order to fulfill the need to retain the customer, various churning method has been developed as a marketing strategy to keep surviving in this volatile and rapid growing environment [1].

Churn is frequently spoken of in a communication context, where it refers to the tendency of cell-phone subscribers to switch providers. The most basic reasons for churn are dissatisfaction with an existing provider, the lure of a lower price from a different provider, a change in the subscriber's geographic location, the desire for increased connection speed, or a need for different or enhanced cell-phone coverage [2],[3].

A business have to spend a lot more resources when attempting to win new customers than to retain existing ones [4]. As a result, much research has been invested into new ways of identifying those customers who have a high risk of churning. However customer retaining efforts have also been costing organizations large amounts of resource [5]. In response to these issues, the next generation of churn prediction should focus on accuracy.

Variety of churn prediction techniques have been developed as a response to the above requirements. The focus of this study is to compare between two different supervised learning techniques which is Elman Recurrent Neural Network (ERNN) and Jordan Recurrent Neural Network (JRNN) with Reinforcement Learning. The study came to the conclusion on which Recurrent Neural Network could produce more accurate result.

This study basically is about comparing two different Supervised Recurrent Neural Networks with using Reinforcement Learning algorithm. The result of churning prediction will be beneficial to telecommunication companies as they can decide which type of Neural Network to formulate their business strategy for the purpose of retaining their customers. It consists of 3 main objectives to be achieved. There are; to acquire training and testing data through survey; to design and develop Elman and to Jordan Recurrent Neural Network by applying Reinforcement Learning Algorithm for both Networks.

2. Literature Review

The research of customer churn prediction has been carried out extensively with various technique been applied such as statistical[6],[7],[8] and machine learning[9] in order to help the organization to make the decision. The performance of this type of machine learning depends on the learning algorithm and the given application, the accuracy of the modeling and structure of each model. This study is focusing on the machine learning technique which is Recurrent Neural Networks with Reinforcement Learning.

2.1. Recurrent Neural Networks

Recurrent Neural Network (RNN) is one of the classes in neural network where its connections between units form a directed cycle where the outputs are recursively fed back to the input layer until stable output pattern results [10] as shown in Figure 1.

Unlike feed forward neural networks (FNNs), RNNs have an internal state which is very important for many temporal processing tasks. Those internal states can take on both discrete and continuous values. According to [11], recurrent networks can be very sensitive, and can adept to past inputs. This makes RNN a robust learning algorithm. One of RNN advantage as machine learning is that their potential for dealing with two sorts of temporal behaviors. Based on [12] statement, RNN is capable of settling to a solution, as in vision system which gradually solves a complex set of conflicting constraints to arrive at an interpretation.

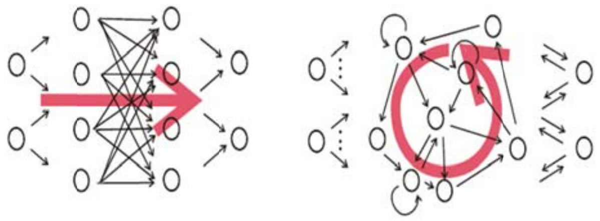

Figure 1: Typical structure of a feed forward network (left) and a recurrent network (right)

2.2. Elman Model

Elman Neural Network (ENN) is one type of the partial recurrent neural network, which consists of two-layer back propagation networks with an additional feedback connection from the output of the hidden layer to its input [13]. He then further states that feedback path represents a dynamic mapping between its outputs and inputs. So ENN is more suitable for processing temporal sequence data than multi-layer perception since it maintains state information between the input samples. Figure 2 shows the Elman RNN model with input layer, Hidden layer and output layer.

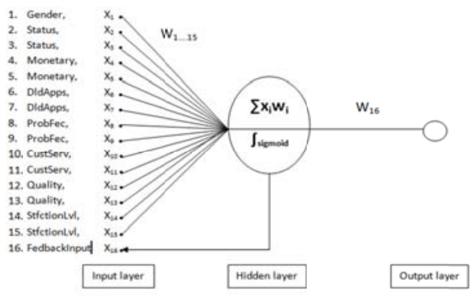

Figure 2. Elman RNN model

The recurrent connection which characterizes this network offers the advantage of storing values from the previous time to step to be used in the current time step, [14]. [15] state that 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.

2.3. Jordan Model

Similar to Elman model, Jordan model is another type of the partial recurrent neural network. The context units are however fed from the output layer instead of the hidden layer. The number of hidden layer and the number of neutron will have huge effect in predicting process as shown in Figure 3.

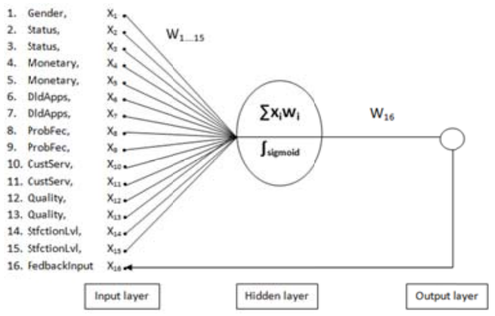

Figure 3. Jordan RNN Model

2.4. Analysis of Elman and Jordan Model

Basically, Elman Recurrent Neural Network (ERNN) and Jordan Recurrent Neural Network (JRNN) have almost the same structure. The only thing that differentiates between these two neural networks is the values of feedback input. Meanwhile, for ERNN, the value of feedback input which will be represented by input, X16 is obtained from the first output computed in the hidden layer. Meanwhile for JRNN, the feedback input is obtained from the actual output computed after the sigmoid function being applied.

Similar to ERNN, the JRNN input was processed twice except with extra weight (w16). Using this concept, we can presume that by applying the same value of data amount, reward, Epoch, learning rate, threshold value, and sigmoid bound and also discount factor as a constant for both ERNN and JRNN, we can compare the result obtained and thus decide which one is better in term of error prediction. Figure 4 shows the similarities of the ERNN and JRNN.

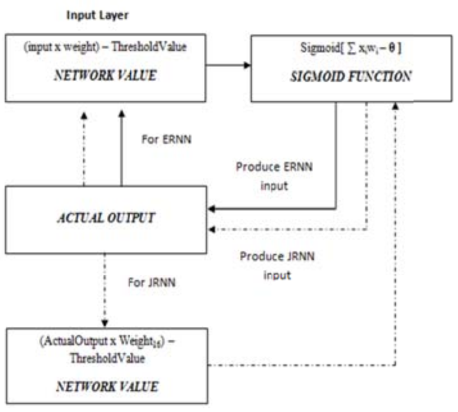
Input Layer
(input x weight) – ThresholdValue
NETWORK VALUE
Sigmoid[$\sum x_i w_i - \theta$]
SIGMOID FUNCTION
For ERNN
Produce ERNN input
ACTUAL OUTPUT
Produce JRNN input
For JRNN
(ActualOutput x Weight_{16}) – ThresholdValue
NETWORK VALUE

Figure 4. Recurrent Neural Networks similarity and basic concept

2.5. Reinforcement Learning Algorithm

Reinforcement learning is an approach to artificial intelligence that emphasizes learning by the individual from its interaction with its environment [15]. Reinforment learning, learns how to map situation into action, rather than depend on a complete environment model. Figure 5 shows the framework of the Reinforcement Learning with two actors, Agent and Environment.

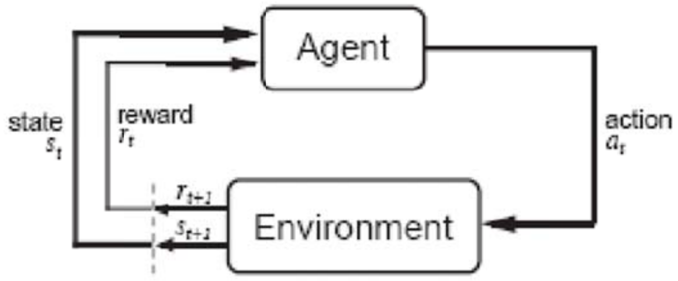

Figure 5. Reinforcement Learning Framework

3. Methodology

In this study, Reinforcement Learning (RL) will be used with Recurrent Neural Network as learning algorithm. The basic concept of RL is that whenever Recurrent Neural Network produces result that is not equal to expected result, RL system will be receiving negative reinforcement [15]. The RL system has just learned not to produce the same result, thus minimizing the number of error that will be produced.

Tabular 1-step Q-learning is one of the simplest reinforcement learning. It uses the experience of each state transition to update one element of a table. This will directly apply to agent's experience. The table, denoted by Q with entry Q(*s, a*) for each state, *s*, and action, *a*. Upon the transition st; *st+1*, having taken action at and receive reward rt+1. The algorithm shown in Figure 6 was used to perform the update. In the algorithm, ***a*** is a positive step-size parameter. Under appropriate conditions (ensuring sufficient exploration and reduction of ɑ over time), this process converge such that the greedy policy with respect to Q is optimal. The greedy policy is to select in each state***, s***, the action***, a***, for which G(***s, a***) is the largest. Thus, this algorithm provides a way of finding an optimal policy purely from experience, with no model of the environment's dynamics [15].

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t,a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q(s_{t+1},a)}_{\text{max future value}}}^{\text{expected discounted reward}} - \overbrace{Q(s_t,a_t)}^{\text{old value}} \right]$$

Figure 6. Update Algorithm

The study had gathered data from mobile phone user with 326 sample questionnaire had been distributed and collected from mobile phone users' around university students. The questionnaire had nine attributes to understand the users behaviors and satisfaction to the service provider. Based on the questionnaire result, the information was subtracted and filled into Microsoft Excel tables. Answers for each attributes were represented by binary number. This is because neural network input training and testing phase required binary number.

4. Training Result

In order to get the best combination of variables that will generate the smallest number of errors, firstly, the default values are declared. This default values is defined by any logical value that suited the condition either by the recurrent neural network method, activation function or the processor of the computing machine. Next, the experiments were conducted where each default values were altered and the changes in error produced were observed.

This trial and error process (various experiment) are repeated as many times as possible with the purpose of deciding the variables combination that would produce optimal solution. Table 1 shows the values that are possible to be used in training process for data amount, reward, epoch, threshold, discount factor and sigmoid bound.

Table 1 : Initial value for the training

DATA AMOUNT	REWARD	EPOCH, E	THRESHOLD VALUE, θ	DISCOUNT FACTOR, DF	LEARNING RATE, LR	SIGMOID BOUND, SB
140	0.9	50	0.9	0.9	0.9	0.9
120	0.7	40	0.7	0.7	0.7	0.7
100	0.5	30	0.5	0.5	0.5	0.5
80	0.3	20	0.3	0.3	0.3	0.3
60		10	0.1	0.1	0.1	0.1

4.1. Data Amount

As shown in both graphs in Figure 7 and Figure 8, data amount with value of 40 – 50 produce up to 96% - 100% errors. This is possible because when consistent default value is used, the system tend to predict almost all churn customer as not churn even does it is churn, thus producing large error number. Bear in mind that top 50 data are arranged as churn customers. In simpler word, the system saw almost or all 50 first as not churn hence giving error reading when it find out it is actually a churn data.

The study had run the training using few amount of data and it had concluded that 100 is the best data amount. When the amount of data increase closing to 100, the gradient of line for both graphs become almost more horizontal. At 100 data amounts, the number of error for ERNN starts with 47% and then linearly stays at 50% of error line. Same goes to JRNN but only its error start at 51%. This proved that whenever the data is balanced between churn, and not churn customer, it produced balanced number of error if default values are used.

When then amount of data used is more than 100, it give the same reason as unbalanced of data. Except the curve went higher which mean the number of error increased. In conclusion, the value 100 will be used as default value instead of other value based on the comparisons from the graph.

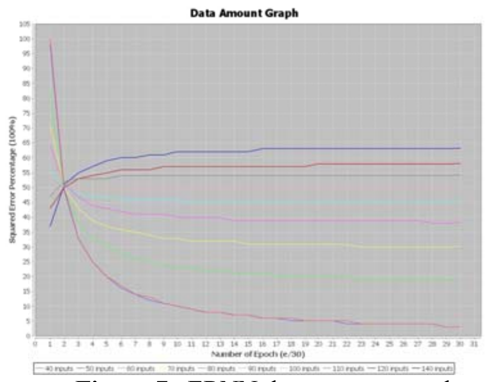

Figure 7. ERNN data amount graph

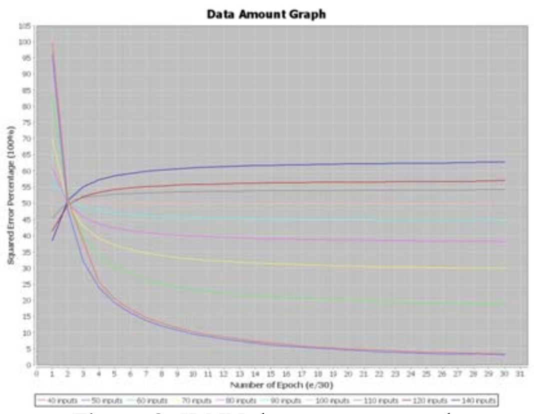

Figure 8. JRNN data amount graph

4.2. Reward

Reward is depending on the Reinforcement Learning Algorithm used. Usually, bigger reward means bigger correction of weight, and vice versa. Following algorithm was also used in research by [10].

$$\text{Correction} = \text{weight} + \text{LR} * (\text{reward} + (\text{weight} * \text{DF}) - \text{weight}) \quad (1)$$

There are several ways of using the algorithm to suit the system itself. For this study, the reward value that used is 0 since this study applied adjusted reinforcement learning algorithm.

4.3. Epoch and threshold value

Epoch basically is the Neural Network number of cycles. For Recurrent Neural Network, there must be at least one epoch to see the feedback input play its roles when it feed back to input layer. On every epoch, wrong output weights were corrected and network value were recalculated if there is errors. The more epoch used in training, the larger possibility that maximize result will be obtained. However, there are also possibilities that small epoch produce better result. There is a consequence of having big number of epoch. Epoch multiply the amount of data and its calculation process by its value. For example when the epoch is 10, the network value calculation for 100 data amount will be done 100 times. This processes consume a lot of resources.

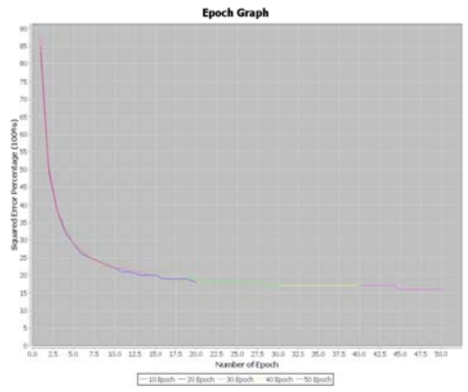

Figure 9. JRNN epoch graph

In this Recurrent Neural Network training, 10 to 50 different epochs value were used. As can be seen in the graph in Figure 9, sum of squared error decrease dramatically in 10 and 20 Epoch. This shows that only in 20 cycles, the error being reduced more than ¾ from its initial value. Since it is a squared value, the graph would not show value reduces exactly to zero. At 30 cycles, the graphs start becomes stable and the changes in error are lesser. Using huge value of epoch only means wasting space and also memory. In order to prevent more resource wasted, using epoch with value 30 is already sufficient to achieve objective.

In order to get the best threshold value to be used, a set of threshold experiment has been conducted, and the value 0.2 give the best accuracy rate compared to other threshold values as shown in Table 2.

Table 2. Experiment result

Testing Value	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Threshold Accuracy Rate (%)	54.9	73.4	60.6	50.1	50	50	50	50	50
Discount Factor Accuracy Rate (%)	80.7	70.3	79.9	75.1	77.3	74.5	72.2	70.1	68.1
Learning Accuracy Rate (%)	62.5	74.3	73.1	71.8	68.2	65.2	63.6	62.1	60.1

4.4. Discount factor and Learning rate

Discount factor and learning rate are two attributes that used together in Reinforcement Learning Algorithm. So, they have relation to each other. Small value of learning rate will cause system to learn slower because it took more time to learn. The same goes for discount factor. Smaller correction maybe will take some time, but the advantage is; it produces less error hence more accurate result. Following table 2 is the result obtained from discount factor experiment and learning rate experiment.

4.5. Sigmoid Bound

Sigmoid bound is the value that will decide the size of churn and not churn domain. It is applied after the sigmoid activation function value is obtained with the purpose of rounding the value only to either 1 or 0 which is representing values for churn and not churn.

In Table 3 sigmoid bound with value 0.6, 0.7 and 0.8 shows the highest accuracy rate among all. Since the value of weight generated is random on every run, and the slightest change of network value will change the result, therefore further experiment need was conducted to make sure the value obtained is consistent and accurate. The experiment had shown that sigmoid bound with 0.75 values give the smallest average error compared to others after 10 trainings. The value 0.75 was chosen for ERNN sigmoid bound and 0.72 for JRNN sigmoid bound based on the experiment conducted.

Table 3. Sigmoid bound experiment result

Sigmoid Bound	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Accuracy Rate (%)	50	51.74	58.57	68.67	75	77.27	79.04	78.94	68.07

5. Testing Result

The training phase has gives us the best values to be used for both recurrent neural systems in order to acquire the best prediction for customer churn. To prove it, a testing on the system by using the values obtained must be conducted. Testing phase is almost the same as training phase, except it use only one epoch and the value of weight is based on the best weight values generated from training phase.

5.1 Elman and Jordan RRN

This study had run ERNN and JRNN testing using different set of data and it is found that the amount of error predicted in ERNN is less than value in testing phase. These prove that the value of weight from training phase can produce better result when the data is different in testing phase. However, since it is only a slight different, the accuracy of weight still can be considered. But JRNN testing produces the same result as training with the percentage of error only 3%. This shows that by using the best weight, the same accuracy percentage can be obtained even with different sets of data.

5.2 Analysis on Reinforcement Learning Algorithm

The basic of Q-Learning formula that normally applied for most recurrent neural network researcher is formula (1), where LR stands for learning rate and DF for discount factor. There are several ways of manipulating this formula. The formula is manipulated with the purpose of matching the learning pattern with the system build to get the desired output. The main concept of Q-Learning is to update weights by either rewarding or punishing Neural Network whenever the objective is not achieved or achieved. Using this concept, first, second and third Q-Learning Experiment was conducted. The purpose is to understand better about Q-Learning, hence clarifies the usage of Q-Learning in both Elman and Jordan Recurrent Neural Network.

On the first experiment, 3 different reward values were used; 0.1, 0.5 and 0.9. The reward (positive value) is only given when there is no error occurs; when the actual and desired output is equal. Following figure 10 is the learning graph obtained based on the experiment conducted.

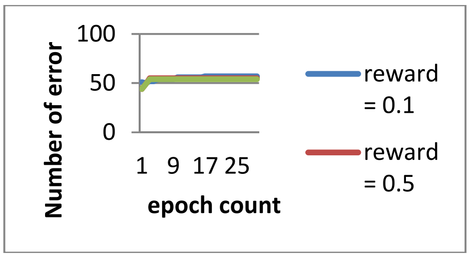

Figure 10. Only give reward to weight when no error

Referring to figure 10, it shows that when using with ERNN, the learning algorithm fail to produce better error reduction when the cycle of epoch increase. This is mainly because the system was taught only to give reward for any non error occurrence and ignore error. The system did not correct the errors, thus producing only more error.

Second experiment is based on concept “Reward (positive value) when no error and punish (negative) when error occur.” It means that the system will reward any weight that produce correct answer and in the same times, punish or correct the error. In this experiment, punish value is simply substitute zero value in Q-Learning formula which will treat the formula as punishment to the weight. Following figure 11 is the graph created from this experiment result.

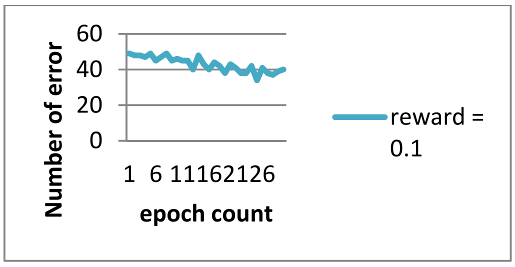

Figure 11. Reward and punishment to weight

Base on Figure 11 above, it can be seen that the value of error is inconsistently decreased as the epoch cycle grows. Eventually at bigger or longer epoch, it will give the smallest error reads, but it will take a lot of time. Rewarding is good in term of broadening the system view/sensor in searching or exploring surrounding for better options.

In this last experiment, punishment (negative) was imposed when error occurs for error correction purposes. Giving punishment can be in the sense of giving negative value, or simply using zero value in Q-Learning formula which treated as a punishment to the weight. Hence, the formula can be simplified as:

$$Correction = weight + LR * - ((weight * DF) - weight) \qquad (2)$$

Figure 12 shows that the algorithm corrects error in almost every epoch, hence reducing the number of error to the smallest.

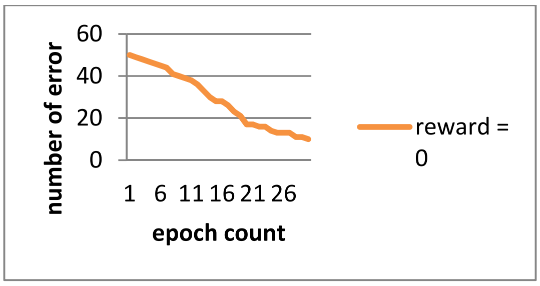

Figure 12. Only give punishment when errors occurs

5.3. Comparisons

A comparative study was performed to see the difference between Elman and Jordan Recurrent Neural Network. The comparison was made based on the result of error produced either for ERNN and JRNN Q-Learning. The percentage of error for ERNN training in Table 4 is 25.97% and for JRNN training is 21%. Meanwhile for ERNN testing, the error percentage is only 6% and 3% for JRNN as shown in table 5. The percentage value in training phase is obtained base on sum of squared error on each epoch.

Table 4. Training result for ERNN and JRNN

No. of Epoch	**Number of error on Epoch**	
	Elman RNN	**Jordan RNN**
First Epoch	43	42
Last Epoch	7	3
Accuracy Percentage (after 30 Epoch cycles)	74.03%	79%

In this comparative study, it is proved that JRNN have better accuracy percentage compared to ERNN with 79% accuracy rate during the training and 97% during the testing phase.

Table 5. Testing result for ERNN and JRNN

	Elman RNN	**Jordan RNN**
Number of error	6	3
Accuracy Percentage	94%	97%

6. Conclusion

In today's fierce competitive business market, many industries has to find a trusted method in order to predict the customers churning and use the knowledge to strategize the marketing plan. The study has proved that RNN with Reinforcement Learning could assist the business in the prediction. The study have two phases which are training and testing. In the training section, several experiments were conducted in order to choose the best variable to be used in the system. Next, the testing section was conducted to see whether the result generated in training phase is consistent with testing phase or not. This is an important part where the best weight from training phase was used inside testing phase.

The analysis on Reinforcement Learning Algorithm section clarifies the usage of Q-Learning algorithm in this research. Lastly is the comparative study where ERNN and JRNN being differentiate from each other. Research on two different Recurrent Neural Networks show that Jordan RNN able to predict better accuracy percentage than Elman RNN.

References

[1] Yossi Richter, Elad Yom and Tovy Noam Slonim, "Predicting customer churn in mobile networks through analysis of social groups" in Proceeding SIAM International Conference on DataMining, pp732-741, 2010.
[2] Jorge B. Ferreira, Marley Vellasco, Marco Aurélio Pacheco and arlos Hall Barbosa "Data Mining Techniques on the Evaluation of Wireless Churn", European Symposium on Artificial Neural Networks Bruges (Belgium), 2004
[3] Bingquan Huang, Mohand T K, Buckley B ,"Customer Churn Prediction in Telecommunications", Expert Systems with Application Vol 39, pp1414-1425,2012.
[4] Ohn Hadden, Ashutosh Tiwari, Rajkumar Roy and Dymitr Ruta, "Computer Assisted Customer Churn Management- State-Of-The-Art and Future Trends", Journal of Computers & Operations Research v34(10), pp2902-2917, 2007.
[5] Triin Kadak ," Soft Computing Method for Customer Churn Management", Helsinki, 2007
[6] Zhang kai-hui, Li Lei, Li Peng, " Customer Churn Prediction Based on Cluster Stratified Sampling Logistic Reggression" International Journal of Digital Content Technology and Its Applications, v5(10), 2011.
[7] Wu H, Zhang WeiWei, "A Customer Churn Analysis Model in E-Business Environment" International Journal of Digital Content Technology and Its Applications, v6(9), 2012.
[8] Qiu Yihui, Mi Hong, "Application of Feature Extraction method in Customer Churn Prediction Based on Random Forest and Transformation" Journal of Convergence Technology V5(3), 2010.
[9] Yaya Xie, Xiu Li, Ngai E.W.T, Weiyun Ying, " Customer Churn Prediction using Improved Balanced Random Forests", Expert System with Application v36, pp3445-3449, 2009.
[10] Yuan-Chu Cheng, Wei-Min Qi, Wei-You Ca, "Dynamic Properties Of Elman And Modified Elman Neural Network " in proceeding of Internation Conference of Machine Learining and Cybernatic, 2002.
[11] Mikael Bod´en, "A guide to recurrent neural networks and back propagation", School of Information Science, Computer and Electrical Engineering Halmstad University, November 13, 2001
[12] Zhiqiang Zhang, Zheng Tang, Shangce Gao and Gang Yang, "Training Elman Neural Network For Dynamic System Identification Using An Adaptive Local Search Algorithm", International Journal of Innovative. Computing, Information and Control, ISSN 1349-4198. Volume 6, Number 5, May 2010

[13] Djarfour N., T. Aifa, K. Baddari, A. Mihoubi, J. Ferahtia, “Application of Feedback Connection Artificial Neural Network to Seismic data Filtering”, Comptes Rendus Geoscience, V340(6),pp 335-344, 2008.

[14] Magdy M. Abdelhameed and Farid A. Tolbah, A, “Recurrent neural network-based sequential controller for manufacturing automated systems", International Journal of Mechatronics vol.12, pp617-633, 2002

[15] Richard S. Sutton , “ Reinforcement Learning”, 1999.