Volume 57, Number 1, 2016

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

CHURN PREDICTION IN THE TELECOMMUNICATIONS SECTOR USING NEURAL NETWORKS

Ionut B. BRANDUSOIU Gavril TODEREAN
Technical University of Cluj-Napoca
ionut.brandusoiu@gmail.com, gavril.toderean@com.utcluj.ro

Abstract: One of the main concerns of telecommunications companies is the subscriber retention. This sector has an annual churn rate of approximately 30% and is on top of the list. Since it is more costly to acquire a new subscriber than to retain an existing one, it is important for these telecommunications operators to identify subscribers that are at risk of churning by using predictive models. In this paper, we present an advanced methodology that predicts churn in the pre-paid mobile telecommunications industry by applying our own neural network architecture on a dataset that contains call details records. We use the gain measure and the ROC curve to evaluate the model.

Keywords: *call details records, churn prediction, neural networks, telecommunications.*

I. INTRODUCTION

The focal concerns of organizations active in the mobile telecommunications industry are to contract new subscribers and retain the existing ones. Thus, it is extremely important to identify subscribers who are likely to churn. The term churn refers to the change of the service provider, triggered by better rates or services or by the benefits offered at signup. To improve the value of its subscribers, a company must retain them over a long period of time [1].

In the telecommunications industry, the mobile market is the segment that sees the fastest growth. At a global level, approximately 75% of the calls can be made using mobile phones, making this a competitive market in which the attention has been switched from contracting to retention [2]. For companies that are active in industries with low switching costs, the loss of a subscriber represents the main responsibility. With an annual churn rate of approximately 30%, the telecommunications industry is on top of the list of all the industries that present this concern [3].

As a result, a company active in the mobile telecommunications sector must identify the subscribers who are at risk of churn before they are actually going to act. In order do to so, such a company needs to implement a predictive model that precisely identifies churners in the near future, to avoid contacting subscribers that will use the services anyway [4]. To correctly identify only the subscribers who are going to churn, the predictive model has to be very accurate. To achieve this task is not easy and well defined because the pre-paid subscribers do not have a contract. Therefore, to implement an accurate predictive model is a complex task.

The approach taken in this paper for predicting churn is based on a data mining process. To extract knowledge from data, the data mining process makes use of machine learning algorithms, statistics, pattern recognition and visualization techniques. By using these types of techniques, we can implement a prediction model to discover trends and future behaviors that allows telecommunications companies to take smarter decisions based on the knowledge extracted from data.

Several representative papers that predict subscribers who are likely to churn in the mobile telecommunications sector are present in the literature and use the following machine learning algorithms: decision trees [5], support vector machines [6], neural networks [7], Bayesian networks and statistical techniques [8]. All these algorithms, including the neural networks, have a standard architecture that predicts correctly only approximately 85% of the subscribers who are going to churn.

This research paper is organized as follows: the following section presents our own neural network architecture used for predicting churn which yields a much higher accuracy than standard algorithms proposed in the literature. We evaluate the predictive model in Section 3 on a call detail records dataset [9]. Finally, Section 4 concludes the paper.

II. MODELLING THE NEURAL NETWORKS

In this paper, the neural networks are used for classification, i.e. to predict the state of a two class variable based on the independent variables.

The perceptron, referred as McCulloch-Pitts neuron, represents the simplest neural network consisting of a single neuron that performs a weighted sum of its inputs followed by an activation function. For classification problems, the perceptron learning algorithm can operate only in the case of linearly separable data and cannot terminate in the case of linearly inseparable data due to the inability to detect the minimum of the error function. The multilayer perceptron (MLP) can be used to classify linearly inseparable data [10].

The MLP neural networks are feedforward networks with one or more layers of units between the input and output layer. The output units represent a hyperplane in the input space. In this paper, the training process of the MLP neural

Manuscript received January 31, 2016; revised February 18, 2016

networks uses the backpropagation (BP) algorithm based on the generalized delta rule.

A. THE BACKPROPAGATION ALGORITHM

The BP learning algorithm is the most popular learning rule for supervised learning. Due to BP, the MLP can be extended to multiple layers.

To implement the BP algorithm, the activation function must be continuous nonlinear monotonically increasing and differentiable. In this architecture we use the hyperbolic tangent function (1) as the activation function for the hidden layers and the generalized sigmoid function (2) for the output layer. In [11], the generalized sigmoid function, known as *softmax* or *Potts*, is introduced for the neurons in the output layer of an MLP neural network used for classification, offering a greater flexibility to the model.

The BP algorithm can be improved by applying the momentum method which implies adding a term *α* with a value between (0,1] to adjust the weights. A typical value for parameter *α* is 0.9 [10]. The BP algorithm with gradient and momentum has a similar complexity to the BP algorithm but is superior regarding the convergence speed and the ability to avoid local minima.

The cost function selected is the cross entropy, which is a better alternative for classification problems [12]. The cross entropy is a function of relative errors and it assumes to more precisely estimate the low probabilities [13]. In order to calculate the first partial derivatives of the cost function with respect to the weights we use the BP algorithm with momentum as follows:

- For each *n*, *p*, and *r* we set:

$$\frac{\partial E}{\partial w_{rp}^{(n)}} = 0 \tag{1}$$

- For each $n = 2,...,N$, $p = 1,...,k_N$:

$$\delta_{l,p}^{(N)} = -e_{l,p} \tag{2}$$

- For each $n = 2,...,N$, $p = 1,...,k_N$, and $r = 1,...,k_{n-1}$:

$$\frac{\partial E_l}{\partial w_{rp}^{(n)}} = -\delta_{l,p}^{(n+1)} o_{l,r}^{(n)} \tag{3}$$

where *E* is the error function, $w_{rp}^{(n)}$ is the weight from layer $n-1$ neuron *r* to layer *n* neuron *p*, $\delta_{l,p}^{(n)}$ is the delta function for neuron *p* sample *l* from layer *n*, and $o_{l,r}^{(n)}$ is the output of neuron *r* sample *l* from layer *n*.

- We set:

$$\frac{\partial E}{\partial w_{rp}^{(n)}} = \frac{\partial E}{\partial w_{rp}^{(n)}} + \frac{\partial E_l}{\partial w_{rp}^{(n)}} \tag{4}$$

- If $n \geq 2$ and $r > 1$ then:

$$\delta_{l,r}^{(n+1)} = \phi_r'^{(n+1)}(u_{l,r}^{(n+1)}) \sum_{q=1}^{k_{n+2}} \delta_{l,r}^{(n+2)} w_{pr}^{(n+1)} \tag{5}$$

B. THE GRADIENT DESCENT ALGORITHM

The gradient descent method for online learning with learning rate $\eta_0 = 0.4$, minimum learning rate $\eta_{\min} = 0.001$, momentum $\alpha = 0.9$, decay factor for learning rate $\beta = (1/lp)\ln(\eta_0/\eta_{\min})$, number of training samples *l* and the number of epochs *p* needed to reduce the initial learning rate towards $\eta_{\min}$ consists of the following steps:

1. Let $k = 0$ and initialize the weight vector with w_0 and the learning rate with η_0 and $\Delta w_0 = 0$.
2. All the data are read from a randomly selected subset and calculate the error function $E_k(w_k)$ and its gradient $\nabla E_k(w_k)$.
3. If $\eta_k |\nabla E_k(w_k)| \leq \alpha |\Delta w_k|$ then the parameter $\alpha = 0.9\eta_k |\nabla E_k(w_k)| / |\Delta w_k|$.
4. Let $v = w_k - \eta_k \nabla E_k(w_k) + \alpha \Delta w_k$.
5. If $E_k(v) < E(w_k)$ then $w_{k+1} = v$ and $\Delta w_{k+1} = w_{k+1} - w_k$. Otherwise $w_{k+1} = w_k$ and $\Delta w_{k+1} = \Delta w_k$.
6. $\eta_{k+1} = e^{-\beta}\eta_k$. If $\eta_{k+1} < \eta_{\min}$ then $\eta_{k+1} = \eta_{\min}$.
7. If a convergence condition is met then the network is reported according to the convergence criteria, otherwise we increment *k* with one and go to step 2.

C. CONVERGENCE CRITERIA

The training process runs at least during an epoch and it stops based on the following criteria which are verified in the stated order:

- When the model is updated, the training total error is computed at the end of each iteration. From the step K_1 , if the training error does not decrease below the current minimum error E_1 during the next $n = 1$ steps, then it stops and reports the weights obtained at step K_1 .
- If the change in the training error is relatively small (0.0001), the training process stops and it reports the weights obtained at step K_1 .
- If the ratio between the current training error and the initial error is small (0.001), it reports the weights obtained at step K_1 .

D. THE WEIGHT INITIALIZATION

In this paper, the weights are initialized using the simulated annealing algorithm and the training process alternatively [14]. This procedure is applied over a random subset to derive the initial weights 4 times. The simulated annealing algorithm is used to escape the local minima obtained during the training process by perturbing this local minima 4 times. If the local minima is successfully avoided, for the next training cycle the simulated annealing algorithm initializes the weights with proper values. To find the global minimum this procedure is repeated 3 times.

For large datasets, this method is computationally expensive and therefore for weight initialization we use only a random subset:

1. Randomly generate 4 weight vectors between

$[-0.5, 0.5]$ and calculate the training error for each weight vector and select as initial weights the ones with the minimal training error.

2. Initialize a loop $k = 0$.
3. Train the network using the initial weights and obtain the trained weights $\mathbf{W}$.
4. If the training error is smaller or equal to 0.5 the loop stops and the vector $\mathbf{W}$ is the result, otherwise we increment the loop with one.
5. If $k < 4$, the old weights are perturbed to obtain 4 new weight vectors $\mathbf{W}' = \mathbf{W} + \mathbf{W}_n$ by adding random noise $\mathbf{W}_n$ between $[-0.5^{k+1}, 0.5^{k+1}]$. If $E(\mathbf{W}_{\min}) < E(\mathbf{W})$, then the initial weights are set to $\mathbf{W}_{\min}$ and go to step 3, where $\mathbf{W}_{\min}$ is the perturbed weights vector that produced the minimal training error.
   Else, the loop stops and the vector $\mathbf{W}$ is considered the result.

If the weights produce a training error greater than 0.1, then this algorithm is repeated until the training error becomes smaller or equal to 0.1 or is repeated 3 times and the weights that produce the minimal testing error in the *k* loops are selected.

E. OPTIMIZING THE NETWORK STRUCTURE

Because the generalization is better in the small networks with few parameters, we apply the pruning strategy based on sensitivity to remove insignificant units during the learning process. The pruning method starts from a neural network with a large number of units and then it removes the unuseful neurons form the input and hidden layers. In this paper, we use the pruning method proposed in [15] and it consists of two steps: first it removes the hidden neurons and then the neurons from the input layer.

The pruning process of the hidden neurons stops: if the global convergence conditions are met, if the best network has an error three times smaller than the error produced by the current network, or if the persistence limit from the hidden layer is exceeded. If no convergence criteria are met, then a sensitivity analysis over the hidden neurons is performed to identify the insignificant neurons.

During the pruning step of input neurons, the neural networks is trained on the entire training set and if any convergence criteria are met, then the global convergence criteria are verified and if necessary the two-step process is repeated. The pruning process of input neurons stops: if the global convergence criteria are met, if the best network has an error three times smaller than the error produced by the current network, or if the persistence limit from the input layer is exceeded. If no convergence criteria are met, then a sensitivity analysis over the input neurons is performed to identify the insignificant neurons.

III. MODEL EVALUATION

In this section we apply the neural networks algorithm described in the previous section to a dataset that contains the call detail records of 3333 subscribers (one row per subscriber) [9]. We must remember that the BP learning algorithm is a method based on the gradient descent and has a slow convergence speed. The preprocessing of the training set avoids the *curse of dimensionality* and improves the generalization capacity of a neural network.

This dataset is detailed in [16] where in order to reduce the dimensionality and remove the collinearity between the independent variables, Brându□oiu and Toderean applied the PCA algorithm. The dataset used in this paper consists of the 11 principal components resulted in [16], the binary variables *International Plan* and *Voice Mail Plan*, and the dependent variable *Churn*.

To be able to empirically estimate the generalization error of the neural networks algorithm, we randomly partitioned the dataset into a training dataset (80%) and a testing dataset (20%). In the training dataset, the distribution of the dependent variable *Churn* is unbalanced, i.e. 388 (14%) subscribers for the *Yes* class and 2279 (86%) subscribers for the *No* class. To optimally train the neural networks we balanced the distribution of the dependent variable by oversampling. In this way we randomly cloned the subscribers that correspond to the *Yes* class until the number of subscribers almost reaches the one for the *No* class. At the end of this process we obtained an equilibrated training dataset with 2294 (50%) subscribers for the *Yes* class and 2279 (50%) subscribers for the *No* class, having 4573 subscribers in total. Regarding the testing dataset, the *Churn* variable has 95 (15%) subscribers for the *Yes* class and 571 (85%) subscribers for the *No* class.

We can evaluate the performance of this predictive model by using the confusion matrix in Table 1. In this table, the correct predictions are indicated by the cells on the diagonal of the cross-classification of cases, whilst the incorrect predictions by the cells off the diagonal.

In the training dataset, the predictive model correctly classified all the 2294 subscribers that churned, having a true positives rate (sensitivity) of 100.00%, and correctly classified all the 2279 subscribers that did not churn, having a specificity of 100.00%. Overall, the predictive model correctly classified 100.00% of the subscribers from the training dataset.

In the testing dataset, the predictive model correctly classified 94 subscribers that churned out of the total of 95, having a sensitivity of 98.95%, and correctly classified 569 subscribers that did not churn out of the total of 571, having a specificity of 99.65%. Overall, the predictive model correctly classified 99.55% of the subscribers from the testing dataset. By comparing the percentages of the incorrect predictions in the training and testing dataset one can observe that we have almost equal percentages, meaning that the predictive model does not overtrain.

Set	*Observed*	*Predicted*		
		No	Yes	% correct
	No	2279	0	100.00%
Training	Yes	0	2294	100.00%
	Overall %	49.84%	50.16%	100.00%
	No	569	2	99.65%
Testing	Yes	1	94	98.95%
	Overall %	85.59%	14.41%	99.55%

Table 1. Confusion matrix.

Another way to evaluate the performance of this predictive model is the gain measure. On the vertical axis of the gain chart we have the percentage of the correct answers while on the horizontal axis we have the percentage of the contacted subscribers. The proportion of the respondents present in each percentile with respect to the total number of respondents represents the gain measure. The cumulative gain chart illustrates the prediction rate compared to the random expectation (red line). Figure 1 shows the curve for the neural network model for the *Yes* class of the dependent variable *Churn*. We can observe that in the 16th percentile, the predictive model reaches its highest performance of approximately 99%.

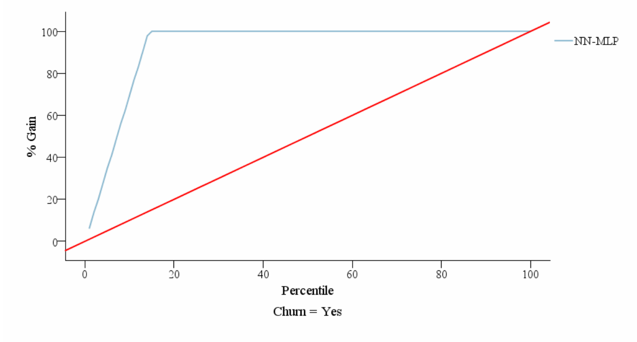

Fig. 1 Cumulative gain chart.

To evaluate the predictive model we can also use the ROC curve which is derived from the confusion matrix and uses only the sensitivity and the false positives rate (equal to 1 minus specificity). The ROC curve that corresponds to the neural network model is illustrated in Figure 2. In this figure, the coordinate point (0, 1) from top left corner implies a perfect prediction and we can observe that the ROC curve approaches this point. For the neural network model, the sensitivity is approximately 99% and the specificity is well over 99%.

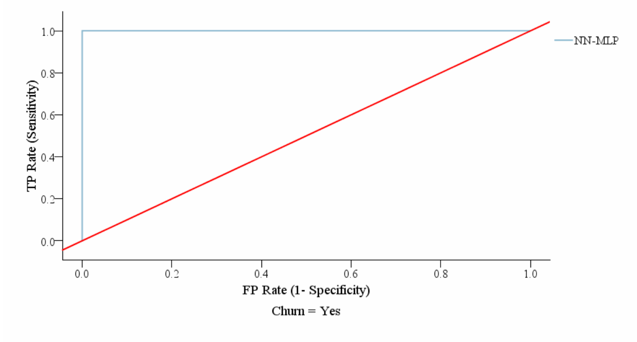

Fig. 2 ROC curve.

IV. CONCLUSIONS

Neural networks are a popular machine learning algorithm inspired from biological neural networks that model the human brain. A neural network is characterized by the network architecture, the nodes characteristics and by the learning rules. The learning process of the MLP neural network uses the BP algorithm with gradient and momentum. To accelerate the convergence speed and to avoid local minima, the weights are initialized using the simulated annealing algorithm. Finally, to improve the generalization ability, we apply the pruning strategy based on sensitivity.

In this paper we implemented a model using our own neural network architecture that predicts subscribers in the pre-paid mobile telecommunications industry that are likely to churn. From the technical point of view, the overall performance of this model to predict churners and non-churners is 99.55%.

From the practical point of view, the overall performance of this model to predict churners in a pre-paid mobile telecommunications company is approximately 99%. A telecommunications company can use this churn prediction model in other subscriber response models such as: cross-sell, up-sell, or subscriber acquisition.

REFERENCES

[1] M. Freeman, “The 2 customer lifecycles”, *Intelligent Enterprise*, Vol. 2, Nr. 16, 1999.

[2] H. Kim and C. Yoon, “Determinants of subscriber churn and customer loyalty in the Korean mobile telephony market”, *Telecommunications Policy*, Vol. 28, 2004.

[3] P. Kotler and L. Keller, *Marketing management 12th Edition*, Pearson Prentice Hall, 2006.

[4] K. Coussement and D. Van den Poel, “Integrating the voice of customers through call center emails into a decision support system for churn prediction”, *Information and Management*, Vol. 45, 2008.

[5] I.B. Brânduşoiu and G. Toderean, “Churn prediction modeling in mobile telecommunications industry using decision trees”, *University of Oradea Journal of Computer Science and Control Systems*, Vol. 6, Nr. 1, 2013.

[6] I.B. Brânduşoiu and G. Toderean, “Churn prediction in the telecommunications sector using support vector machines”, *Annals of the Oradea University Fascicle of Management and Technological Engineering*, Vol. 22, Nr. 1, 2013.

[7] I.B. Brânduşoiu and G. Toderean, “A neural networks approach for churn prediction modeling in mobile telecommunications industry” *University of Pitesti Scientific Bulletin Series: Electronics and Computers Science*, Vol. 13, Nr. 1, 2013.

[8] I.B. Brânduşoiu and G. Toderean, G. “Predicting churn in mobile telecommunications industry”, *ACTA Technica Napocensis Electronics and Telecommunications*, Vol. 54, Nr. 3, 2013.

[9] C.L. Blake and C.J. Merz, “Churn data set”, *UCI repository of machine learning Databases*, University of California, Department of Information and Computer Science, 1998.

[10] G. Toderean, M. Coşteiu, and M. Giurgiu, *Retele neuronale Ediția a doua*, Microinformatica, 1995.

[11] D.E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, “Backpropagation: the basic theory”, *Backpropagation: Theory, Architecture, and Applications*, 1995.

[12] K. Matsuoka and J. Yi, “Backpropagation based on the logarithmic error function and elimination of local minima”, *In Proceedings of the International Joint Conference on Neural Networks*, 1991.

[13] G.E. Hinton, “Connectionist learning procedure”, *Artificial Intelligence*, Vol. 40, 1989.

[14] T. Masters, *Practical neural network recipes in C##*, Academic Press, 1993.

[15] H.J. Xing and B. G. Hu, “Two-phase construction of multilayer perceptrons using information theory”, *IEEE Transactions on Neural Networks*, Vol. 20, Nr. 4, 2009.

[16] I.B. Brânduşoiu and G. Toderean, “Applying principal component analysis on call detail records”, *ACTA Technica Napocensis Electronics and Telecommunications*, Vol. 55, Nr. 4, 2014.