Reservoir computing Echo State Network classifier training

Reservoir computing Echo State Network classifier training

V Krylov[1, 2] and S Krylov[1]

[1] Higher School of Economics National Research University, Nizhny Novgorod, Russian Federation
[2] Artezio LLC

vladimir.v.krylov@gmail.com

Abstract. Reservoir Computing (RC) is taking attention of neural networks structures developers because of machine learning algorithms are simple at the high level of generalization of the models. The approaches are numerous. RC can be applied to different architectures including recurrent neural networks with irregular connections that are called Echo State Networks (ESN). However, the existence of successful examples of chaotic sequences predictions does not provide successful method of multiple attribute objects classification. In this paper ESN binary classifiers are researched. We show that the reason of low precision of classification is using of unbalanced training dataset. Then the method to solve the problem is proposed. We propose to use stochastic perforation balancing algorithm on training data set and method of data temporalization. The resulting errors matrixes are pretty good. The proposed method is illustrated by the usage on synthetic data set. The features of ESN classifier are demonstrated in the case of anomaly events detection such as based on transaction attributes fraud detection.

1. Introduction

Usage of machine learning neural structures for complicated tasks demands new neural networks architectures. The feed-forward networks are very popular but there is place for Recurrent Neural Networks (RNN). One of the most sophisticated architectures called Long Short Term Memory (LSTM) is used frequently as sequence classification module and for other artificial intelligence tasks. Neural networks are important building blocks of Reinforcement Learning (RL) systems. The characteristics of the networks influence a lot on the result. The researchers look for effective architectures with high level of generalization and low complexity of learning algorithms. The potential possibilities of RNN are promising. However irregular connections of network nodes make back propagation algorithm and it’s modifications ineffective in the search of network weights because of problem of vanishing gradient. This challenge gave birth to special class of neural structures called Reservoir Computing (RC) [1–3].

The distinct feature of the architectures is the separation of network output coefficients subset – ReadOut – made of a perceptron or of simple neural network, all other coefficients of internal and input connections are selected randomly. This way the architecture consists of not learning complex recurrent network – reservoir – and learning ReadOut. One of widely used RC architectures is Echo State Network (ESN) [4]. We selected this type of architectures for our research.

The network defines sequence of output vectors $y(t)$ with dimension – $dout$ length T by defined input sequence of the same dimension and length – din according to some rule formulated implicitly during network training process. ESN in the training is the following dynamic system with the dimension of state space – $dres$:

$$\mathrm{x}(\mathrm{t}+1)=\mathrm{f}\big(\mathrm{W_{in}u(t)}+\mathrm{W_{res}x(t)}+\mathrm{W_{fb}y(t)}\big). \tag{1}$$

The coefficients of the model are in W_{in}, W_{res}, W_{fb} – input, internal and back connection respectively. They are randomly created and they are not changed during the training. f – nonlinear vector function of sigmoid type, usually tanh (). The vector of inner states contains calculated values of reservoir neurons activation function. Each instance of training set is presented as the sequence of states. Each time t state records into matrix X of size $n{*}dres$. After the training another matrix of outlet coefficients is calculated

$$\mathrm{W_{out}} = \mathrm{X^{+}Y}. \tag{2}$$

X^{+} – pseudoinverse (Moore-Penrose, usually) matrix for X. The matrix Y has size $n * dout$ and it defines which outputs correspond n instances of input sequences. In the process of inference for every input sequence $u_{eval}(t)$ the (1) actions, matrix X is created and the predictions are calculated according to the formula:

$$\mathrm{y_{eval}}=\mathrm{f^{out}(W_{out}X)}.$$

Most of publications view at ESN application for tasks of prediction of chaotic movements in dynamic systems. It demonstrates the feasibility of their training for structured chaos. It means that ESN has high level of generalization and that they can find and show deeply hidden patterns. In Adrian Milleea dissertation [5] the detailed description of ESN is present.

2. Model of ESN binary classifier and implementation

In this article ESN model is used for objects classification. The objects are described by the list of attributes that are not related to time sequences usually. To use ESN we have to transform data set to time sequences set (temporalization). The simple method of temporalization is interpretation of feature number as time stamp. In this case, rows of the matrix of instance of training and testing data are calculations of time sequence. For binary classification output matrix Y dimension $n * 2$, using One Hot Coding is:

$$\mathrm{Y}=\begin{matrix}0 & 1\\ 1 & 0\\ 1 & 0\end{matrix}$$

You can see 3 instances. The first instance belongs to class 1, the second and the third – class 0. The research of possibilities of building this type classifier was done for the ESN model implemented with the library of program models EasyESN [6]. Let's review basic characteristics of neural network which was built. In the figure1, the conceptual schema of classifier is shown.

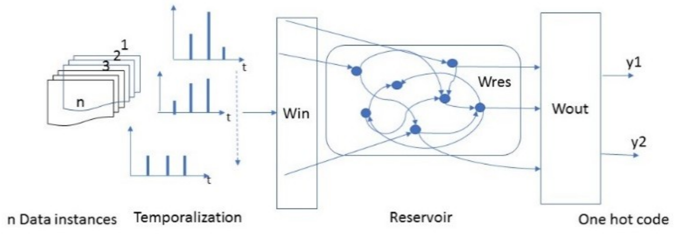

Figure 1. ESN binary classifier.

The features of reservoir are defined by coefficients of matrixes of input and inner connections and by activation functions. Let's use $B * \tanh(x)$ as main nonlinear function. For neurons of reservoir let's have $f_{res}(x) = 0.1 + \frac{0.98x}{1+Bexp(-x)}$. Matrixes of reservoir are generated randomly. However, important condition is the provision of spectral radius of matrix W_{res} value that less than 1. As shown in the work [7], this condition guarantees that reservoir forgets initial state and the trajectory in state space reflects input sequence only. In opposite case the trajectories will be dependent on initial state and the network will not be able to differentiate inputs. The spectral radius of square matrix is the maximum module of matrix's eigenvalues. It can be shown when spectral radius extreme value is close to one in dynamic system defining reservoir the network is functioning at the edge of chaos. That values of spectral radius were supported in the research by normalization of randomly generated coefficients of matrix W_{res}. Other important parameter of reservoir is the density of neuron connections or connectivity. Usually the density is the relation of not null weights to number of possible connections in reservoir. If the number of neurons is $dres$, then the density can be calculated as $\rho = \frac{nonzeroW}{dres^2}$. In the research the density was under control at the level about 0.2. To find output matrix the Moore-Penrose pseudoinverse was used at the final stage of training. In classic case, this algorithm is realized by the function $pinv()$ described in [8].

The library EasyESN makes possible to do training by the couple of lines:
esn = ClassificationESN(din, dres, dout)
esn.fit(inputData, outputData, verbose=1).

For inference one line is enough:
eval = esn.predict(evalData, verbose=1).

3. The results of synthetic data set research

The result of random vectors of defined length with (0,1)-uniform distribution generation was used as synthetic data set. The result of comparison of average value to defined limit was used as the label of the class. The main subject of the research was the influence of on the error of classification:
1) Dimension of features space,
2) Number of instances,
3) Balance of the classes.

The different number of neurons in the reservoir was used. This number was increased till the high level of chaos was reached to classify objects with big number of attributes.

The research was started with toy example using very small dimension reservoir $dres = 3$ and training set of 20 instances ($n = 20$) that have 2 features. The temporalization was executed according to the following schema: values of the features for every instance $k = 0, 1, \dots n-1$ were interpreted as time samples of time sequences $u^k(0), u^k(1)$. The value $din = 1$ was used. For example let's see one of such set of sequences with the matrix defining unitary code of the class label for every instance.

[0.06807841, 0.44933138],	[1., 0.]	class 1
[0.33252574, 0.36319995],	[1., 0.],	class 1
[0.44539369, 0.00985919],	[1., 0.],	class 1
….		…
[0.81826957, 0.8492856]	[0., 1.]	class 0.

The scatterplot of training input sequences is shown on the figure 2a. The histogram of instances distribution in classes on the figure 2b is shown.

As you can see the input dataset can contain different numbers of instances for the classes. The ratio of number of instances of one class to another is called excess of selection exc. In our example $exc = 0.666$. After the starting training the classifier got 20 more instances for the testing and evaluation. The F1 – measure for this toy example is 0.823. The normalized and not normalized confusion matrix are shown in figure 3.

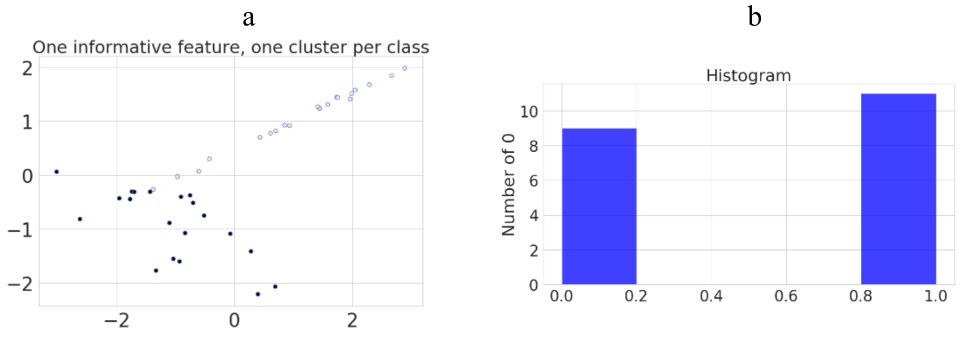

Figure 2. Training data set: a – input points, b – distribution of class labels.

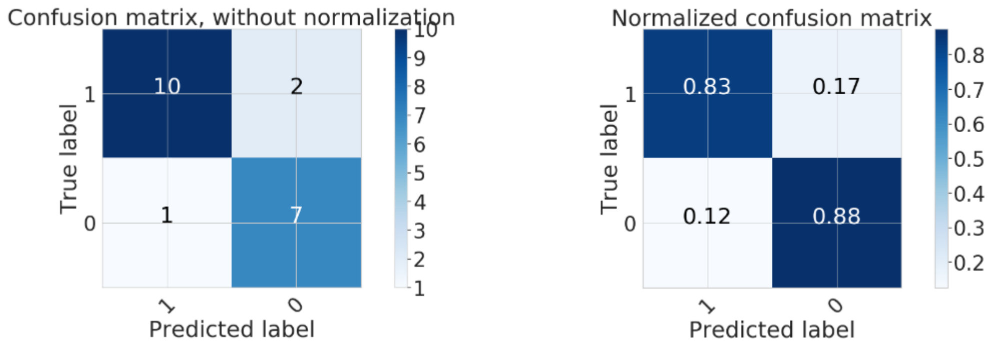

Figure 3. Confusion matrix of classifier.

The classifier wrongly defines label of the class in 15% cases for each class. The next step of research was the increase of number of neurons of reservoir. Also the size of data set and number of features and number of instances were investigated. The following table demonstrates the results.

Table 1. Reservoir size, dataset size, balance metric and testing F-score.

dres	**dataset**	**F1**	***exc***	**Wrong classifications**
3	20×2	0.823	0.666	0.12/0.17
5	20×2	0.705	0.818	0.33/0.18
5	20×2	0.823	0.818	0.22/0.09
10	20×2	0.875	0.818	0.22/0.0
30	20×2	1.0	0.820	0/0
100	20×2	1.0	0.666	0/0
30	20×20	0.0	0.25	1.0/0
500	20×20	1.0	0.176	0/0
500	2000×20	0.367	0.229	0.77/0.01
500	2000×20	0.899	0.797	0.06/0.19
500	2000×2	0.888	0.942	0.11/0.11

The first observation is that the increase of number of neurons in reservoir influence a lot on classification precision. At some point the precision stays constant and depends on training and testing sets only. The increase of features requires increase of number of instances for training. For objects having few dozens of features the thousands of instances of each class are required. The research has shown that unbalanced classes influence a lot on precision of classification of big data sets even if number of neurons in the reservoir is big. The table demonstrates results of classification for different values of exc. You can see that with the same number of neurons in reservoir – 500 – metric of quality is not good F1=0.367 if *exc* is down to value 0.229. It means the classifier in architecture ESN is very sensitive to unbalance of the classes of training set. To compensate this drawback the stage balancing by of stochastic perforation of instances of set of domination class was included in training process. For every instance the random number with pseudorandom uniform distribution in interval (0,1) was set along with class label. The perforation parameter $0 \leq R \leq 1$ defines the probability of instance exclusion from data set. Using the definition of excess it can be stated that the balancing of the classes must be done by stochastic perforation of domination class with the parameter of perforation $R = 1 - exc$. In the program script this perforation was done by 2 lines:

```
inputData1=inputData[np.logical_or(outputData[:,0]==0,inputData[:,2]>R)]
outputData=outputData[np.logical_or(outputData[:,0]==0,inputData[:,2]>R)]
```

The below table demonstrates results after stochastic perforation stage inclusion. The first row is result without this stage, the second – with it. The parameter of perforation was set equal to 0.797.

Table 2. Stochastic perforation impact.

dres	dataset	F1	*exc*	Wrong classifications
500	2000×20	0.367	0.229	0.77/0.01
500	2000×20	0.899	0.797	0.06/0.19

After synthetic dataset testing the ESN classifier was tested with real data.

4. Application of binary ESN classifier to fraud detection

The detection of anomaly events such as fraud with credit cards was the following step of the research. We used data set from Kaggle called *creditcard.csv*. It contains transactions by credit cards by Eurobonds holders in September of 2013 [9]. This data set includes 492 frauds of 284 807 transactions. The data set is unbalanced. The target class (fraud) is 0,172% of all transactions. The data set contains digital variables only. The values are results of PCA transform. Unfortunately because of confidential problems Kaggle cannot give original attributes and additional information on data. The features V1, V2 ... V28 are main components by PCA. Only features out of PCA transform are *Time* and *Sum*. Feature *Time* contains seconds between every transaction and first transaction in the data set. The feature *Sum* is sum of transaction, it can be used for training as sum of spending. The column *Class* contains label of class. It is 1 for fraud and 0 – otherwise. For research *Sum* was scaled and *Time* was excluded as not useful.

Using standard procedures the division of data set on training and testing sets was done. 20% transactions were separated for testing randomly:

```
x_train, x_test = train_test_split(data, test_size=0.2, random_state=RANDOM_SEED)
```

The training set was unbalanced on classes with excess 0.00173. To balance the stochastic perforation with $R = 0.98$ was done. The number of transaction instances decreased to 797. After the perforation the new excess value was 1.02. The training was executed next. The dimension of reservoir was set to 500. The testing was done on original testing set which included 55556 transactions. 94 were fraud instances. The testing error matrix is shown in figure 4.

You can see the classifier is taught very good and 87% cases of the fraud were detected correctly. The miss detection is at the level 13%, false positives – 2%. Let's compare the results with one of the best methods of classification of unbalanced data – boosting. The boost classifier of XGboost [10] package was used. The following configuration of 50 estimators was set:

```
model1=xgb.XGBClassifier( learning_rate =0.1, n_estimators=50, max_depth=5,
min_child_weight=1, gamma=0, subsample=0.8, colsample_bytree=0.8, objective= 'binary:logitraw',
nthread=4, scale_pos_weight=1, seed=27).
```

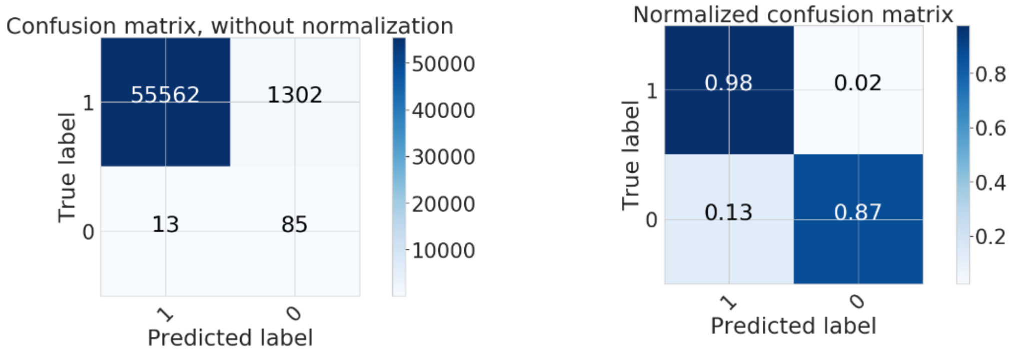

Figure 4. Not normalized and normalized error matrixes for real data.

The similar to stochastic profiler balancing method quality was achieved. However the results of detection are not so good (see confusion matrix on the figure 5).

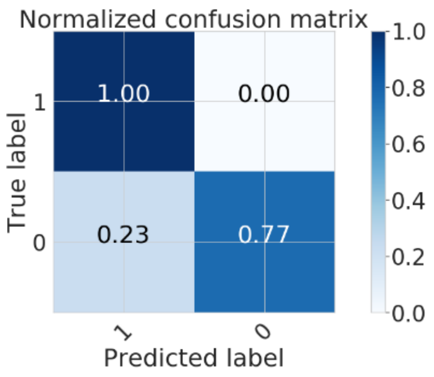

Figure 5. The results of testing by XGboost classifier.

The boosting provided worse miss fraud number in about twice. As expected the decision trees were well tuned to domination class. But the detection of anomaly events was weak.

5. Conclusion

The research of binary classifier based on reservoir computing architecture Echo State Network lead to statement that it can be used to solve traditional prediction tasks such as detection of anomaly events. The successful application to fraud detection using 3 dozens of attributes of transactions is proved. For the classifier the additional procedure is developed – stochastic perforation. It improves the results a lot. The profit of RC architecture is training time reduction. The research was done on simple ESN implementation. The future improvement of the method to reduce time may require more complicated architectures of RC. Also the interesting discoveries can be done for classifications with greater number of attributes. The temporalization for such tasks enlarges length of sequences. Probably it will lead to creation of new effective tools of real time classification.

References

[1] Melandri L 2014 Introduction to reservoir computing methods *Alma Mater Studiorum Universita di Bologna* 51
[2] Verstraeten D, Schrauwen B, D'Haene M and Stroobandt D 2007 An experimental unification of reservoir computing methods *Neural Networks* 20
[3] Schrauwen B, Verstraeten D and Campenhout J V 2007 An overview of reservoir computing: theory, applications and implementations *In ESANN'2007 proceedings*

[4] Lukosevicius M 2012 A practical guide to applying Echo State Networks *Neural networks: Tricks of the trade* 659–86
[5] Millea A 2014 *Explorations in Echo State Networks* (University of Groningen, Department of Artificial Intelligence, Nijenborgh 9 9747 AG, Groningen, The Netherlands, June)
[6] *GitHub repository* https://github.com/kalekiu/easyesn
[7] Jaeger H 2001 *The "echo state" approach to analysing and training recurrent neural networks* (Technical report, German National Research Center for Information Technology)
[8] FreeMat v4.0 – Online Documentation, PINV Moore-Penrose Pseudoinverse http://freemat.sourceforge.net/help/array_pinv.html
[9] *Credit Card Fraud Detection* https://www.kaggle.com/samkirkiles/credit-card-fraud/data
[10] Sharma N 2018 *XGBoost. The Extreme Gradient Boosting for Mining Applications* (GRIN Verlag)