

Deep Learning for Distribution Channels' Management

Sabina-Cristiana NECULA

Department of Research, Faculty of Economics and Business Administration

Alexandru Ioan Cuza University of Iasi

sabina.mihalache@gmail.com

This paper presents an experiment of using deep learning models for distribution channel management. We present an approach that combines self-organizing maps with artificial neural network with multiple hidden layers in order to identify the potential sales that might be addressed for channel distribution change/ management. Our study aims to highlight the evolution of techniques from simple features/learners to more complex learners and feature engineering or sampling techniques. This paper will allow researchers to choose best suited techniques and features to prepare their churn prediction models.

Keywords: Artificial Neural Network, Distribution Channel, Self-Organizing Maps, Deep Learning

1 Introduction

Gartner predicts that by 2018, 70% of e-Commerce will move from B2C and B2B models to models that focus on the individual customer experience. The applicability of deep learning in business is a research field that intends to research different models that learn patterns from data in a supervised or unsupervised manner. The field presents high interest both for practitioners, as for researchers. Deep learning relates to artificial neural networks with multiple hidden layers, convolutional neural networks, recurrent neural networks, self-organized maps, Boltzmann machine and auto encoders.

The applicability in the business field is well investigated. The approaches are in the field of customer relationship management, human/ talent resource, financial analysis, fraud, bankruptcy, supplier relationship management. Churn prediction, identifying the potential outliers or recommending products/ ideas/ customers/ suppliers are well suited as implementations.

Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, partially supervised or unsupervised.[1][2][3][4]

Deep learning is a class of machine learning algorithms that:[5](pp199–200)

- use a cascade of multiple layers of

nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.

- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- use some form of gradient descent for training via backpropagation.

This paper presents the current research in the field of deep learning for business, the research method, the experiment results and the main conclusions.

2 Current research

In today's business environment, the most manufacturers do not sell their goods directly to their end-consumers but they cooperate with marketing intermediaries to offer the goods to market.

Distribution channels are different based on their consumer and industrial market and based on the number of involving factors in distribution channel. The choice of distribution channels and channel members will have a huge impact on the company's strategy. A careful attention should be also taken into account at different levels in distribution channel's decision making

process. In this regard, there are three strategies including intensive distribution strategy, exclusive distribution strategy, and selective distribution strategy. Intensive distribution strategy is mainly related to daily usage of goods with high consumption and low-priced.

Artificial neural networks can be used for various tasks such as classification, regression, segmentation, and so on. Machine learning can give an answer by recognizing and sorting out patterns from the data provided and then classifying that data into the possible appropriate pattern (predicting) when it faces unknown data as a question.

The information-based prediction models using machine learning techniques have gained massive popularity during the last few decades. Such models have been applied in a number of domains such as medical diagnosis, crime prediction, movies rating, etc. Due to immense financial cost of customer churn, the companies from all over the world have analyzed various factors (such as call cost, call quality, customer service response time, etc.) using several learners such as decision trees, support vector machines, neural networks, probabilistic models such as Bayes, etc. [6]

E-commerce has provided new opportunities for both businesses and consumers to easily share information, find and buy a product, increasing the ease of movement from one company to another as well as to increase the risk of churn. Studies develop churn prediction model [7] by testing the forecasting capability of the support vector machine (SVM). The predictive performance is benchmarked to logistic regression, neural network and classic support vector machine.

The product review plays an important role in customer's purchase decision making process on the e-commerce websites. Therefore, there are studies that build predictive models using deep neural network and investigates the influences of emotions that contribute to review helpfulness. In addition, the type of product, reviewer, visibility, readability, linguistics and sentiment related characteristics are also used for comparison and helpfulness prediction. [8]

Machine learning can't do feature engineering. Potentially, there are two types of limitations with machine learning: [9]

- An algorithm can only work well on data with the assumption of the training data - with data that has different distribution. In many cases, the learned model does not generalize well.

- Even the well-trained model lacks the ability to make a smart meta-decision. Therefore, in most cases, machine learning can be very successful in a very narrow direction.

Deep learning took the approach of making each layer learn in advance. This is literally known as pretraining. In pretraining, learning starts from the lower-dimension layer in order. Then, the data that is learned in the lower layer is treated as input data for the next layer. The method that prevents this density problem is called the dropout. Networks with the dropout learn by cutting some linkages randomly within the units of networks. The dropout physically makes the network sparse. Machine learning can be broadly classified into supervised learning and unsupervised learning. With supervised learning, a machine uses labeled data, the combination of input data and output data, and mentions which pattern each type of data is to be classified as. With unsupervised learning, a machine uses unlabeled data. In this case, only input data is given. Then, what the machine learns is patterns and rules that the dataset includes and contains. The purpose of unsupervised learning is to grasp the structure of the data. It can include a process called clustering, which classifies a data constellation in each group that has a common character, or the process of extracting the correlation rule. Data dimension reduction is an important step for customer classification modeling, and feature selection has been a research focus of the data dimension reduction field. [10]

Class imbalance brings great challenges to feature selection in customer identification, and most of the current feature selection approaches cannot produce good prediction on the minority class. [11]

Neural networks are a little different to the machine learning algorithms. While other

methods of machine learning take an approach based on probability or statistics, neural networks are algorithms that imitate the structure of a human brain.

The credit scoring aim is to classify the customer credit as defaulter or non-defaulter. It is concluded that artificial neural network (ANN) is better alternative than decision tree and logistic regression when data availability is high in dataset. [12]

Also, there are studies that develop pricing models. The results revealed that the ANN model's overall prediction capability was 94%; the model's robustness was demonstrated by comparing the results produced using the model with those produced using a multiple regression analysis. The proposed model can be applied as an effective decision-support tool for green building pricing and formulating marketing strategies. [13]

3 Research hypothesis

We identified a research hypothesis:

H1: Is deep learning possible to use for channel distribution?

Objectives:

- Identify the types of deep learning models
- Identify the tools needed
- Identify datasets.

Deep learning models:

Supervised learning:

- Artificial Neural Networks
- Convolutional Neural Networks
- Recurrent Neural Networks

Unsupervised learning:

- Self Organized Maps
- Restricted Boltzmann Machine
- Autoencoders

At this moment, there are some software libraries for deep learning:

- Deeplearning4j—An open-source deep-learning library written for Java/C++ with LSTMs and convolutional networks. It provides parallelization with Spark on CPUs and GPUs, and imports models from Keras, Tensorflow and Theano. DL4J is part of the Eclipse Foundation.
- Gensim—A toolkit for natural language processing implemented in the Python programming language.

- Keras—An open-source deep learning framework for the Python programming language.
- Microsoft CNTK (Computational Network Toolkit)—Microsoft's open-source deep-learning toolkit for Windows and Linux. It provides parallelization with CPUs and GPUs across multiple servers.
- MXNet—An open source deep learning framework that allows you to define, train, and deploy deep neural networks. Backed by AWS.
- OpenNN—An open source C++ library which implements deep neural networks and provides parallelization with CPUs.
- Paddle—An open source C++ /CUDA library with Python API for scalable deep learning platform with CPUs and GPUs, originally developed by Baidu.
- Pytorch - Tensors and Dynamic neural networks in Python with GPUs. The Python version of Torch, associated with Facebook.
- TensorFlow—Google's open source machine learning library in C++ and Python with APIs for both. It provides parallelization with CPUs and GPUs.
- Torch—An open source software library for machine learning based on the Lua programming language and used by Facebook.
- Caffe- Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Focused on image processing.

We used a dataset available on the Machine learning repository website <http://archive.ics.uci.edu/ml/datasets/Wholesale+customers>. [14] The dataset has 440 instances and 8 attributes (Table 1).

Attribute Information:

- 1) Fresh: annual spending (m.u.) on fresh products (Continuous);
- 2) Milk: annual spending (m.u.) on milk products (Continuous);

- 3) Grocery: annual spending (m.u.) on grocery products (Continuous);
- 4) Frozen: annual spending (m.u.) on frozen products (Continuous)
- 5) Detergents_paper: annual spending (m.u.) on detergents and paper products (Continuous)
- 6) Delicatessen: annual spending (m.u.) on and delicatessen products (Continuous);
- 7) Channel: customers Channel - Horeca (Hotel/Restaurant/Cafe) or Retail channel (Nominal)
- 8) Region: customers Region Lisbon, Oporto or Other (Nominal)

Table 1. Descriptive Statistics

| Attribute | Minimum | Maximum | Mean | Std. Deviation |
|----------------|------------------|---------|----------|----------------|
| Fresh | 3 | 112151 | 12000.30 | 12647.329 |
| Milk | 55 | 73498 | 5796.27 | 7380.377 |
| Grocery | 3 | 92780 | 7951.28 | 9503.163 |
| Frozen | 25 | 60869 | 3071.93 | 4854.673 |
| Detergents | 3 | 40827 | 2881.49 | 4767.854 |
| Delicatessen | 3 | 47943 | 1524.87 | 2820.106 |
| Region | Frequency | | | |
| Lisbon | 77 | | | |
| Oporto | 47 | | | |
| Other Region | 316 | | | |
| Channel | Frequency | | | |
| Horeca | 298 | | | |
| Retail | 142 | | | |

As for tools, we used Anaconda/ Spyder, Python language and keras, tensorflow, torch, pandas, numpy libraries. Figure 1 presents the neural network architecture with hidden layers. In the hidden layer, the activation function is the rectifier function. In the output layer, the activation function is the sigmoid function.

One of the activation functions introduced is the rectifier. A unit-applied rectifier is called a Rectified Linear Unit (ReLU).

$$f(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

This activation function is what fires the

neuron's output, based on the sum of all incoming signals. Mathematically it adds nonlinearity to neural network processing, thereby providing the artificial neuron nonlinear behaviors, which will be very useful in emulating the nonlinear nature of natural neurons.

$$\text{sigmoid} = \frac{1}{1 + e^{-ax}}$$

An activation function is usually bounded between two values at the output, therefore being a nonlinear function, but in some special cases, it can be a linear function

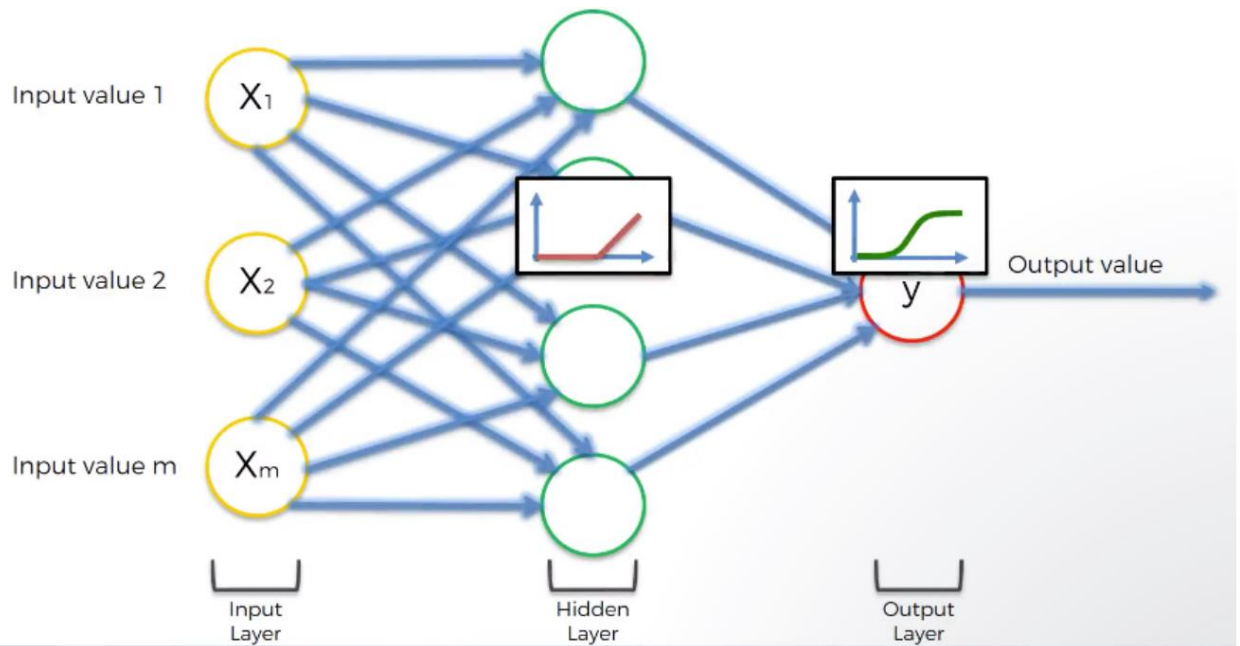


Fig. 1. A neural network representation (the rectifier linear function in the hidden layer and the sigmoid function for the output layer)

A neural network learns by adjusting weights in an intent to minimize the errors. The error is back propagated to the network.

Backpropagation is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data is processed.

Technically it calculates the gradient of the loss function. It is commonly used in the gradient descent optimization algorithm. It is also called backward propagation of errors, because the error is calculated at the output and distributed back through the network layers.

Backpropagation requires a known, desired output for each input value—it is therefore considered to be a supervised learning method (although it is used in some unsupervised networks such as auto encoders).

The goal of any supervised learning algorithm is to find a function that best maps a set of inputs to their correct output. The motivation for backpropagation is to train a multi-layered neural network such that it can learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to

output.

Sometimes referred to as the cost function or error function, the loss function is a function that maps values of one or more variables onto a real number intuitively representing some "cost" associated with those values. For backpropagation, the loss function calculates the difference between the network output and its expected output, after a case propagates through the network.

The standard choice for the loss function is the square of the Euclidean distance between the vectors y and \hat{y} (y predicted) as in Figure 2.

- Perceptron: a single layer feed forward neural network
- Output value (\hat{y}): predicted value by the neural network / Actual value: y
- Cost function: $\frac{1}{2}(y-\hat{y})^2$
- The goal is to minimize the cost function
- After having the cost function, we will feed the information back to the neural network, then the weights get updated to minimize the cost function.
- 1 epoch: is when we go through the whole dataset.

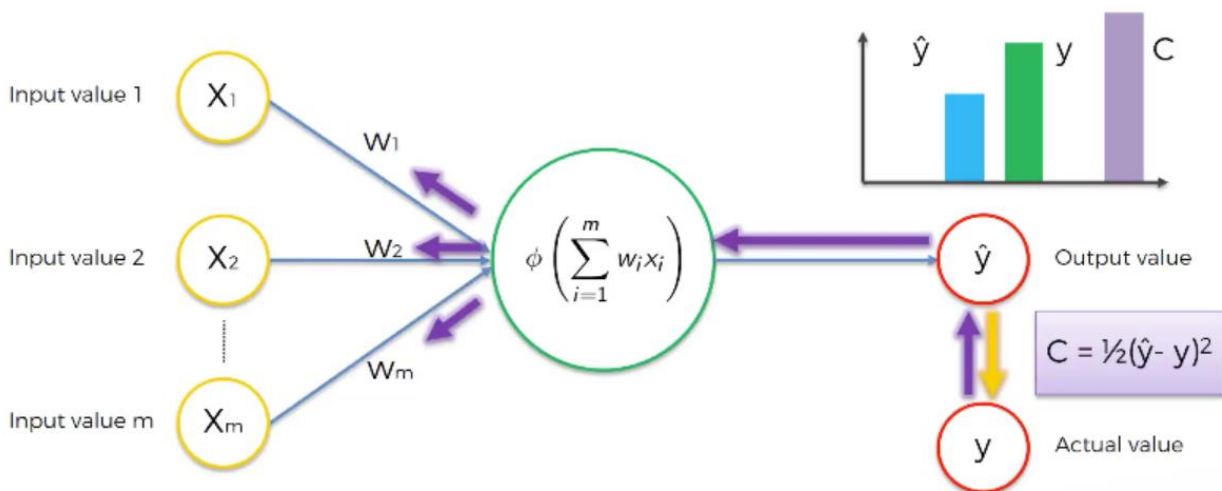


Fig. 2. The backpropagation algorithm

Stochastic gradient descent is a popular algorithm for training a wide range of models in machine learning, including (linear) support vector machines, logistic regression and graphical models. When combined with the backpropagation algorithm, it is the de facto standard algorithm for training artificial neural networks.

Many improvements on the basic stochastic gradient descent algorithm have been proposed and used. In particular, in machine learning, the need to set a learning rate (step size) has been recognized as problematic. Setting this parameter too high can cause the algorithm to diverge; setting it too low makes it slow to converge.

RMSProp (for Root Mean Square Propagation) is a method in which the learning rate is adapted for each of the parameters. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight.

Adam (short for Adaptive Moment Estimation) is an update to the RMSProp optimizer. In this optimization algorithm, running averages of both the gradients and the second moments of the gradients are used.

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space.

The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron whose weight vector is most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance (within the lattice) from the BMU. The architecture of a self-organizing map is presented in Figure 3.

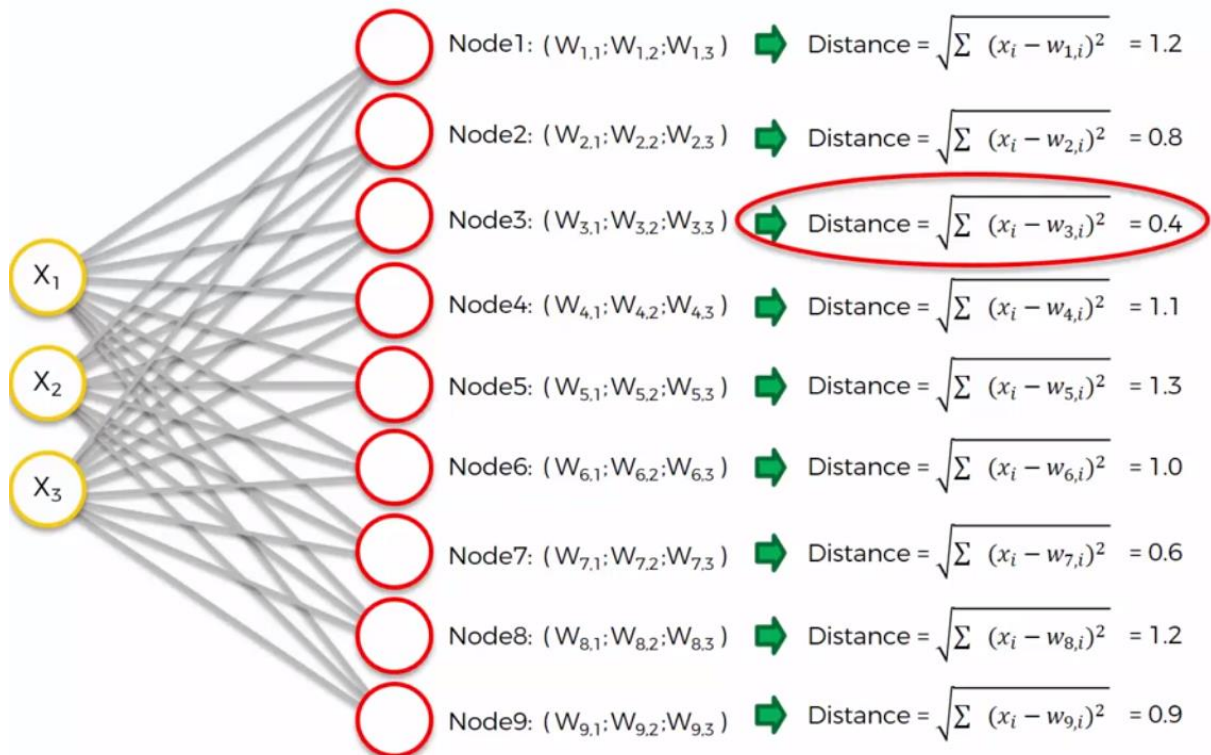


Fig. 3. Self Organized Maps Architecture

One of the most interesting aspects of SOMs is that they learn to classify data without supervision. With the artificial neural network approach an input vector is presented to the network (typically a multilayer feedforward network) and the output is compared with the target vector. If they differ, the weights of the network are altered slightly to reduce the error in the output. This is repeated many times and with many sets of vector pairs until the network gives the desired output. Training a SOM however, requires no target vector. A SOM learns to classify the training data without any external supervision whatsoever. Training occurs in several steps and over many iterations:

- Each node's weights are initialized.
- A vector is chosen at random from the set of training data and presented to the lattice.
- Every node is examined to calculate which

one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).

- The radius of the neighborhood of the BMU is now calculated. This is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighborhood.
- Each neighboring node's (the nodes found in step 4) weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.
- Repeat step 2 for N iterations.

4 Experimental results analysis

We splitted the dataset into the training set and the test set. We normalized data using StandardScaler and LabelEncoder. (Figure 4)

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Wholesale.csv')
X = dataset.iloc[:, 1:8].values
y = dataset.iloc[:, 0].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, tes

from sklearn.preprocessing import StandardScaler, LabelEncode
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
labelencoder_X = LabelEncoder()
y_train=labelencoder_X.fit_transform(y_train)
y_test=labelencoder_X.fit_transform(y_test)

```

Fig. 4. Preprocessing the data

Therefore, we manage to have the data prepared in vectors. (Figure 5)

| | | | |
|-----------------|---------|----------|--|
| X | int64 | (440, 7) | array([[3, 12669, 9656, ..., 214, 2674 [3, ... |
| X_test | float64 | (88, 7) | array([[-1.95240475, -0.8380839 , -0.2783966 , ..., -0.26996686, ... |
| X_train | float64 | (352, 7) | array([[0.60241485, -0.54591673, 2.10173577, ..., -0.29812628, ... |
| best_accuracy | float64 | 1 | 0.9176136363636365 |
| best_parameters | dict | 3 | {'batch_size':32, 'epochs':500, 'optimizer': 'rmsprop'} |

Fig. 5. The results of preprocessing the data

There is a common standard to tune the artificial neural network. Therefore, we build a classifier that receives as parameter the optimizer (it could be adam or rmsprop). It can be noticed that the artificial neural network was set to 2 hidden layers, the batch size is another parameter that is tuned. The

number of epochs is also tuned. As it can be observed in the previous picture, it seems that the optimum number of observations from a batch size is 32, the number of epochs is 500 and the best algorithm optimizer is rmsprop (Figure 6).

```

from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import Dense
def build_classifier(optimizer):
    classifier = Sequential()
    classifier.add(Dense(units = 4, kernel_initializer = 'uniform', activation = 'relu', input_dim = 7))
    classifier.add(Dense(units = 4, kernel_initializer = 'uniform', activation = 'relu'))
    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
    classifier.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])
    return classifier
classifier = KerasClassifier(build_fn = build_classifier)
parameters = {'batch_size': [25, 32],
              'epochs': [100, 500],
              'optimizer': ['adam', 'rmsprop']}
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 10)
grid_search = grid_search.fit(X_train, y_train)
best_parameters = grid_search.best_params_
best_accuracy = grid_search.best_score_

```


Fig. 6. Obtaining the best parameters to train the artificial neural network

We defined the confusion matrix. In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error

matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. The confusion matrix is presented in Figure 7.

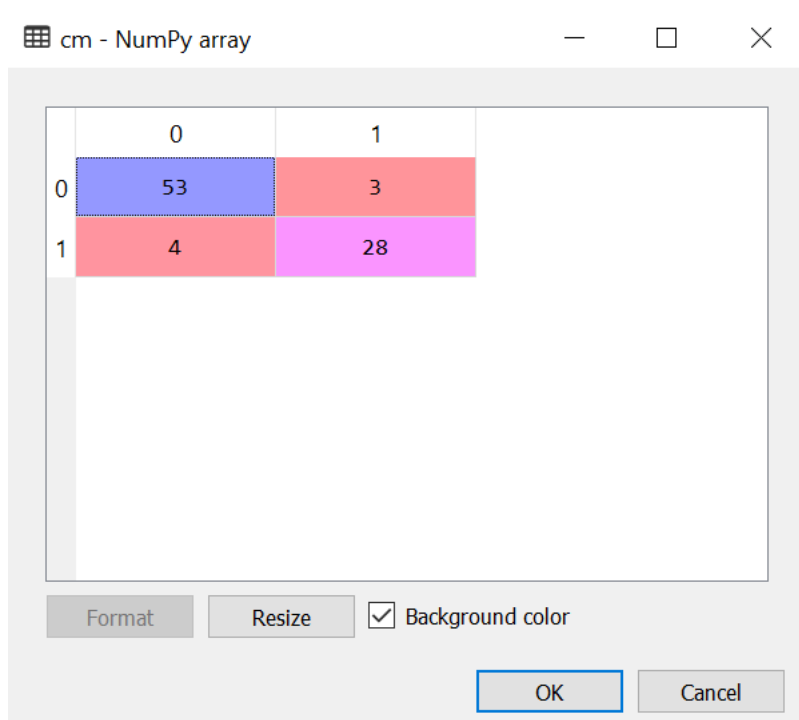


Fig. 7. The confusion matrix

The actual accuracy is given by dividing $(53+28)/88=93\%$ as in the Figure 8.

```
352/352 [=====] - 0s - loss: 0.1840 - acc: 0.9347
Epoch 497/500
352/352 [=====] - 0s - loss: 0.1838 - acc: 0.9347
Epoch 498/500
352/352 [=====] - 0s - loss: 0.1838 - acc: 0.9347
Epoch 499/500
352/352 [=====] - 0s - loss: 0.1838 - acc: 0.9347
Epoch 500/500
352/352 [=====] - 0s - loss: 0.1840 - acc: 0.9347
```

Fig. 8. The accuracy of the artificial neural network

The self-organizing maps are a type of unsupervised deep learning model that clusters data and identifies the outliers. A usual way to use this model is to identify the outliers by using self-organizing maps and, after that, predicting a model of churn by

using an artificial neural network with multiple hidden layers. We build a self-organizing map. In the same manner, we scaled the data as it can be observed in Figure 9.

```
# Importing the dataset
dataset = pd.read_csv('Wholesale.csv')

X=dataset.iloc[:,1:]
y=dataset.iloc[:,0]

# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
X = sc.fit_transform(X)
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Training the SOM
from minisom import MiniSom
som = MiniSom(10, 10, 1, sigma=0.1, learning_rate=0.1)
som.random_weights()
som.train_random(X, 1000)

from pylab import figure, axes, scatter, show, colorbar, plot
figure()
ax = axes()
scatter(X[:,0], X[:,1], s=100, c=y)
colorbar()
markers = ['o']
colors = ['r']
for i, x in enumerate(X):
    w = som.weights[0][0]
    plot(w[0]+0.05, w[1]+0.05, markers[i], colors[i])
    marker
```

| | | |
|---------|-----------|----------|
| X | DataFrame | (440, 7) |
| dataset | DataFrame | (440, 8) |
| y | Series | (440,) |

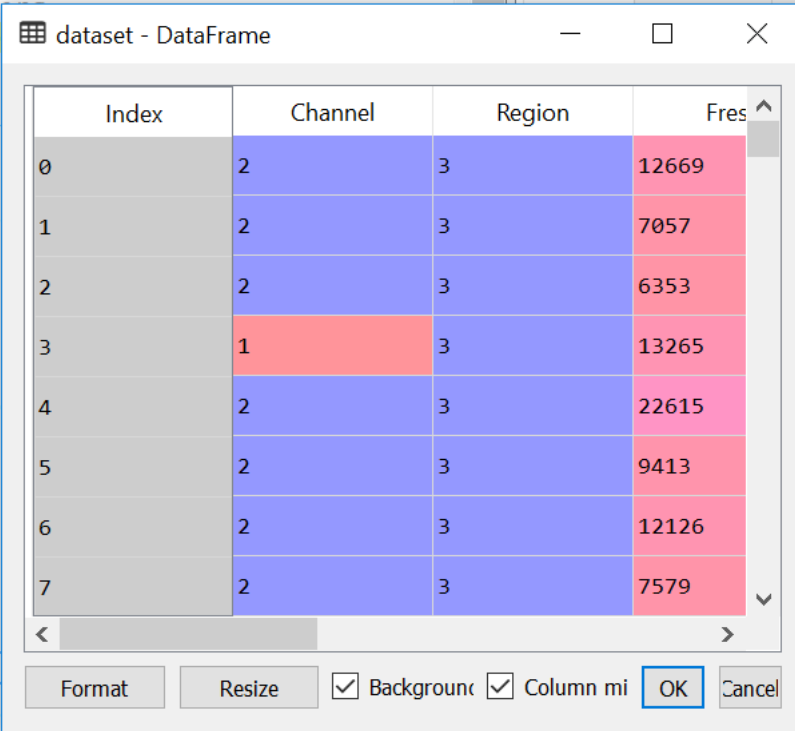


Fig. 9. The self-organizing map data preprocessing

Therefore, if in the beginning the dataset had scaled data as in Figure 10 continuous values, now the vectors contain

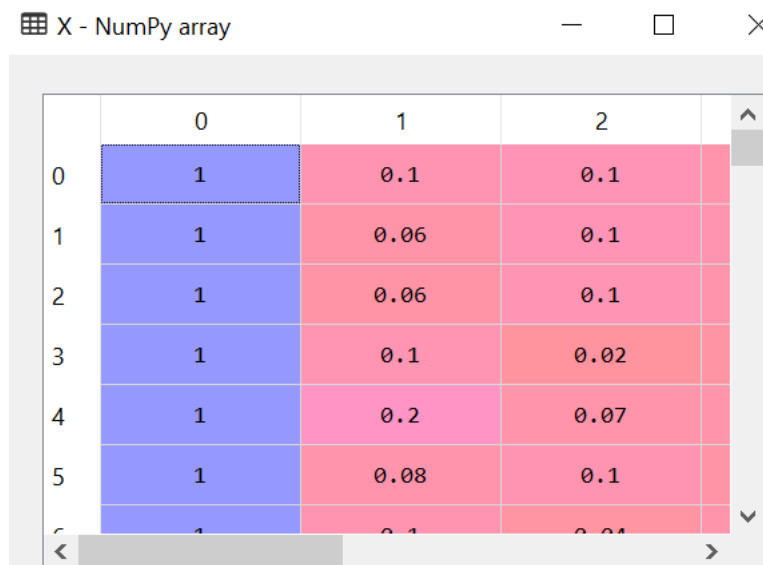


Fig. 10. Scaled data

We identified the outliers by designing a chart (Figure 11).

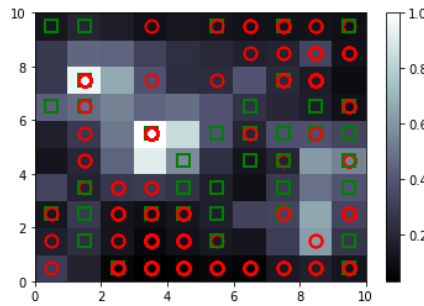


Fig. 11. The outliers from the SOM

The sales that are the outliers identified by SOM are targeted as potential sales to investigate in order to change the channel of

distribution. These sales are presented in Figure 12.

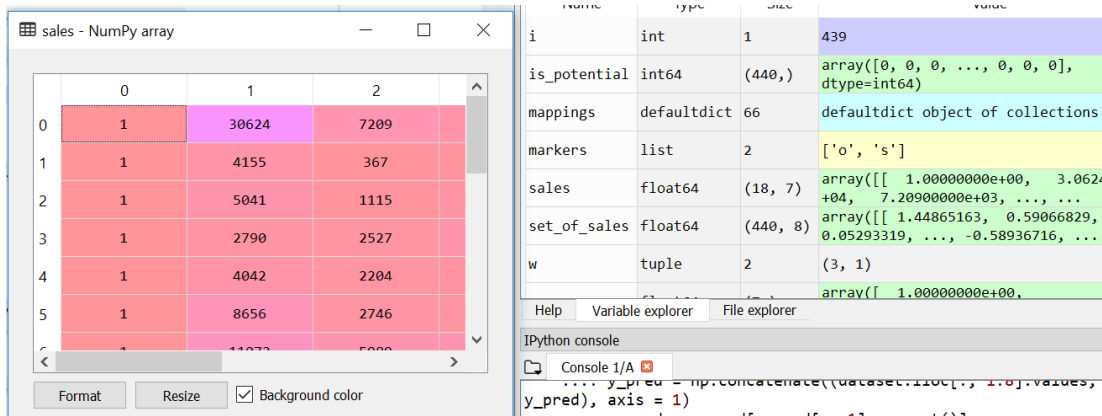


Fig. 12. The vector that contains the potential sales to investigate

Predicting a model of churn consisted in architecting an artificial neural network that uses on the input layer the ReLu activation

function and the sigmoid in the output layer. The training, the loss and the accuracy are presented in Figure 13.

```
Epoch 1/2
440/440 [=====] - 0s - loss: 0.5532 - acc: 0.9977
Epoch 2/2
440/440 [=====] - 0s - loss: 0.1838 - acc: 1.0000
```

Fig. 13. Training the network

5 Conclusions

The main conclusion is that the deep learning models are well suited for distribution channel management. The findings of this exploratory study could be used for further testing of hypotheses on the relationship between customer relationship management and predictive modeling with deep neural networks. The study points out the phases of building deep neural models for distribution channel.

The findings from this paper help managerial recommendations for optimizing the decision-making process in a distribution channel churn prediction context. In fact, using neural networks, companies can accurately identify channels who are truly at risk to churn, focus their efforts on these channels of distribution. In this paper, we test the forecasting capability of a churn prediction model, on a dataset of 440 observations. The results show that neural networks have a high prediction accuracy rates.

Distribution channel churn prevention through data mining techniques is an emerging trend that is often employed in order to retain the customers, and thus save millions of dollars for companies. In the literature, many prediction models have been presented that employ various types of features, data mining techniques applied on various private and public datasets.

We tried to implement an artificial neural network model that combines self-organized maps with the model of artificial neural network that has multiple hidden layers in an attempt to improve the accuracy of churn prediction models. It seems that this architecture ensures a high accuracy.

References

- [1] Y., Bengio, A., Courville, A., P., Vincent, (2013). Representation Learning: A Review and New Perspectives, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8): 1798–1828. arXiv:1206.5538 Freely accessible. doi:10.1109/tpami.2013.50.
- [2] J., Schmidhuber (2015), Deep Learning in Neural Networks: An Overview, *Neural Networks*, 61: 85–117. PMID 25462637. arXiv:1404.7828 Freely accessible., doi:10.1016/j.neunet.2014.09.003.
- [3] Y., Bengio, Y., LeCun, G., Hinton, Geoffrey (2015), Deep Learning, *Nature*, 521: 436–444. PMID 26017442. doi:10.1038/nature14539.
- [4] S., Jürgen (2015), Deep Learning, *Scholarpedia*, 10(11):32832. Online
- [5] L., Deng, D., Yu (2014), Deep Learning: Methods and Applications, (PDF), *Foundations and Trends in Signal Processing*, 7 (3–4): 1–199. doi:10.1561/20000000039
- [6] M., Ahmed, H., Afzal, A., Majeed, et al., A Survey of Evolution in Predictive Models and Impacting Factors in Customer Churn, *Advances in Data Science and Adaptive Analysis*, Volume: 9, Issue: 3, 2017
- [7] N., Gordini, V., Veglio, Customers churn prediction and marketing retention strategies. An application of support vector machines based on the AUC parameter-selection technique in B2B e-commerce industry, *Industrial Marketing Management*, Volume: 62, pp: 100-107, 2017
- [8] M. S. I, Malik, A.; Hussain, Helpfulness of product reviews as a function of discrete positive and negative emotions, *Computers in human behaviour*, Volume: 73, pp: 290-302, 2017
- [9] S., Yusuke, et al., *Deep Learning: Practical Neural Networks with Java*, Packt Publishing House, 2017
- [10] J., Xiao, H., Cao, Hanwen, X., Jiang, et al., GMDH-based semi-supervised feature selection for customer classification, *Knowledge Based Systems*, Volume: 132, pp. 236-248, 2017
- [11] B., Zhu, Y., Niu, J., Xiao, et al., A new transferred feature selection algorithm for customer identification, *Neural Computing & Applications*, Volume: 28, Issue: 9, pp: 2593-2603, 2017
- [12] S., Imtiaz, A.J., Brimicombe, A Better Comparison Summary of Credit Scoring Classification, *International Journal of Advanced Computer Science and Applications*, Volume: 8, Issue: 7, pp: 1-4, 2017
- [13] Y-K., Juan, Y-H, Hsu, X., Xie, Identifying customer behavioral factors and price premiums of green building purchasing, *Industrial Marketing Management*, Volume: 64, pp: 36-43, 2017
- [14] The dataset used in this paper, Wholesale Customers, *Machine Learning Repository*, <http://archive.ics.uci.edu/ml/datasets/Wholesale+customers>.



Sabina-Cristiana NECULA (b. July 27, 1979) received her PhD in Accounting (Business Information Systems) (2007). She is currently a Scientific Researcher at Alexandru Ioan Cuza University of Iasi, Faculty of

Economics and Business Administration. Her current research interests include Artificial Intelligence for Business, Machine Learning, Semantic Web standards and technologies, Decision Support Systems, Business Information Systems.