Neurocomputing 175 (2016) 121–131

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A profit-driven Artificial Neural Network (ANN) with applications to fraud detection and direct marketing

Ashkan Zakaryazad, Ekrem Duman*

Department of Industrial Engineering, Özyeğin University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 15 December 2014
Received in revised form
6 July 2015
Accepted 14 October 2015
Communicated by Dorothy Ndedi Monekosso
Available online 23 October 2015

Keywords:
Neural network
Profit-driven neural network
Individual profit and cost
Sum of squared errors (SSE)

ABSTRACT

The rapid growth in data capture and computational power has led to an increasing focus on data-driven research. So far, most of the research is focused on predictive modeling using statistical optimization, while profit maximization has been given less priority. It is exactly this gap that will be addressed in this study by taking a profit-driven approach to develop a profit-driven Artificial Neural Network (ANN) classification technique. In order to do this, we have first introduced an ANN model with a new penalty function which gives variable penalties to the misclassification of instances considering their individual importance (profit of correctly classification and/or cost of misclassification) and then we have considered maximizing the total net profit. In order to generate individual penalties, we have modified the sum of squared errors (SSE) function by changing its values with respect to profit of each instance. We have implemented different versions of ANN of which five of them are new ones contributed in this study and two benchmarks from relevant literature. We appraise the effectiveness of the proposed models on two real-life data sets from fraud detection and a University of California Irvine (UCI) repository data set about bank direct marketing. For the comparison, we have considered both statistical and profit-driven performance metrics. Empirical results revealed that, although in most cases the statistical performance of new models are not better than previous ones, they turn out to be better when profit is the concern.

1. Introduction

Classification algorithms label the observations of a data set using their different attributes. In recent years, Artificial Neural Network (ANN) classifier has become popular because of its broad application areas. In most of these applications there is a focus on cost-sensitive learning as there are different costs for different types of misclassifications [1–6]. Cost of misclassification of an instance is different from one context to another. In most of the cost-sensitive binary classifications such as diagnosis problems, there are two different misclassifications (false negatives and false positives) and each of them has a particular cost. However, in business problems such as credit card fraud detection and direct marketing each misclassified observation can have a different cost and moreover there may be a profit for correctly classified ones (true positives and true negatives). Thus, in such cases, there is a necessity to develop a classification model that considers individual profits and costs.

"Credit card (CC) fraud detection" is one of the well-known classification problems. In this type of classification, a data set which contains various information (attributes) about the credit card transactions of a bank is used [7–10]. For each of the records, there is a dependent attribute which takes the value of one if the transaction is fraudulent and takes the value of zero if it is legitimate. In CC fraud detection if a classifier correctly detects a fraudulent transaction, it will save the usable limit of the corresponding card and if it mis-classifies the transaction, the usable limit of the card will be lost. The other application of data mining and classification used in this study is Bank direct marketing. In order to target a specific segment of customers, banks use data mining algorithms to classify the customers as buyers and non-buyers [11–13]. In this context, if a model correctly detects a potential customer for a campaign, there will be a particular profit of gaining that customer and if a potential buyer is not identified, the profits that could be gained from him/her might be lost. These two applications of classification are discussed in this study, where individual costs and profits of each instance is considered.

Although ANN has become a popular classification algorithm in recent years, there are a few studies about variable individual costs of misclassifications. Excluding this issue and using the simplistic model, ANN will just minimize the number of misclassifications

* Corresponding author.
E-mail address: Ekrem.Duman@ozyegin.edu.tr (E. Duman).

http://dx.doi.org/10.1016/j.neucom.2015.10.042
0925-2312/© 2015 Elsevier B.V. All rights reserved.

which will result in a suboptimal solution in terms of cost minimization or profit maximization [14]. In real life problems, the most important reason which motivates business administrations to invest in data science is the amount of profit they gain from implementing their models. In credit card frauds, a possible fraud transaction with a high amount of available limit is more important for bank to be detected than a transaction with a low usable limit. Equivalently, in the direct marketing context (for a bank), a customer with a high potential balance is more important to be detected and receive an offer than a customer with a lower balance.

For comparing cost-sensitive classifications, we had to choose a performance metric which does not represent the statistical performance of the models, but their actual ability to maximize the profit gained from their implementation. This issue is addressed in this study by introducing the net profit gained from classification where the threshold is the number of positive instances in the test set (i.e. top 10%) instead of the classical threshold of score 0.5. While using this threshold we imply that only the number of positives labeled by a classifier is not important because in this case we may incorrectly choose a model which labels most of the instances as positive to increase the profit of detecting positive cases.

In this study, a new case-based (individual) net profit matrix is considered to be compared with traditional class-based cost matrix. Table 1 illustrates the profits and costs of two individual instances from two different classes. One of them is case (positive) and the other is non-case (negative).

In Table 1 the instance i is a case (positive) and instance j is non-case (negative) and there are variable profits for correctly classification of each instance and also variable costs for incorrectly classification of them. These profits and costs are different from one context to another. For instance, in credit card fraud P^i_{11} is the usable limit of the card used for transaction i minus the fixed action cost, C^i_{10} is the usable limit of the card of transaction i (cost), C^j_{01} is action cost and P^j_{00} is zero. It means that correct classification of non-case instances have no profit or cost.

In their previous study on credit card fraud detection, Sahin and Duman [36] analyzed the performance of variations of Decision tree and SVM models on fraud data set and the result represented the out-performance of decision tree. Accordingly we just brought the best classifier in their previous study to compare with ANN. Another well-known and common used classifier, Naïve Bayes classifier, is used in order to benchmark the proposed models and original ANN. These two algorithms have been studied in lots of applications such as fraud detection and direct marketing [9,15–18]. Also we have used traditional class-based cost-sensitive ANN (mentioned as CNN) to compare its performance with the newly presented models.

We introduced seven versions of ANN classifier where each of them is a modification of the original ANN classifier. We changed the error function and made it sensitive to individual cost and profit of each instance. While five of these versions are originally proposed in this study, one of them is adapted from literature [8] and one is inspired from literature [21] and modified here to fit to our setting. We performed experiments on three bank data sets with the objective of maximizing the net profit gained from implementing the classification model. Two of them are credit card fraud data and the other one is bank direct marketing. The experimental results revealed that there is no champion model for all of the data sets but the different versions of proposed model have statistically significant improvement in the total net profit as compared to standard ANN, Decision tree, Naive Bayesian classifier and class-based cost-sensitive ANN.

Table 1
Individual net profit matrix.

		Predicted class	
		Case (i)	Non-case (j)
Actual class	Case (i)	P^i_{11}	C^i_{10}
	Non-case (j)	C^j_{01}	P^j_{00}

Table 2
A general cost matrix, a two-class case.

		Predicted class	
		Case	Non-case
Actual class	Case	0	C_1
	Non-case	C_2	0

The remainder of the paper is organized as follows: the next section (Section 2) presents a literature survey on cost-benefit wise learning. Then, Section 3 outlines neural network and its original error function based on sum of squared error. After that, modified error functions are introduced which take the individual net profits into consideration. Section 4 introduces the three data sets used in our empirical study and the experimental results obtained. Finally Section 5 briefly compares all of the models in terms of both statistical performance and total profit.

2. Related works

A considerable amount of literature has been published on cost-sensitive learning, however, there has been relatively little literature published on maximization of the total net profit using individual profits and costs. Numerous studies have the tendency to focus on cost-sensitive learning algorithms such as over-sampling, under-sampling, meta-cost, cost-sensitive boosting, and meta heuristics [1,9,14,19–25] while some of them have worked on cost-sensitive ANN [2–4,14,21,23,26–29]. In most of these researches the authors have aimed to address the imbalance between classes and the different types of misclassification. By considering these issues resulting models are cost-sensitive classifiers which work well in imbalanced data and minimize total cost instead of minimizing total number of misclassifications [30,35]. However the central issue in these model is the class-based costs.

Salchenberger et al. [31] developed a neural network model with variable thresholds considering statistical type I and type II errors and they used this model in thrift failure prediction. They showed the high performance of neural network in discriminating between healthy and failed institutions in comparison to other traditional models. Berardi and Zhang [30] studied the effect of different misclassification costs on neural network and their results represented that this scenario can be used to reach the optimal decision making based on cost of misclassification. Also they implied that most of the statistical performance metrics are not suitable to show the better performance of cost-sensitive classifiers.

In the literature, some of the cost-sensitive models have been developed using different costs in cost matrix. Here, a cost matrix has been considered which contains different costs for different types of misclassifications. Table 2 illustrates a general cost matrix for these types of cost-sensitive models where there is a specific cost for misclassifying a case (positive) as non-case (negative) which is shown as C_1 and another cost for misclassification of a non-case as case (C_2).

In threshold varying cost-sensitive models, different algorithms have been used to find the optimal threshold, such as bisection method [3] or evaluating number of correct classifications using variable thresholds and finding the threshold which maximizes it [2]. Also it can be done by minimizing the cost function with respect to threshold point and finding the optimum value for it [27]. In sampling-based methods for developing cost-sensitive ANN, different costs have been conveyed by appearance of instances. Instances with high costs are over-sampled and/or instances with low costs are under-sampled and the problem of imbalanced data set is solved. Then the model becomes sensitive to the instances with high frequencies and minimizes the total misclassification cost [23].

One of the approaches is to develop a model or modify the original ANN to minimize the root mean square error (RMS) [32]. Adapting the learning rate of the error function with respect to different types of misclassification is a continued study which started by Breiman's altered priors approach [33]. It has been used to make the neural network more sensitive to variable amounts of cost and it increases the learning rate for more costly instances by giving them more weight. This approach has been used in neural network learning in some researches such as Kukar's [21]. Besides, Kukar and Kononenko used a new approach called minimization of misclassification cost which was correction of the error function by multiplying it a cost factor. By this change in error function they make it a cost function using different class-based costs (not uniform) and in back propagation of neural network, in fact, instead of squared errors, the cost of misclassification is minimized and their results showed the superior performance of the network using this approach.

Zheng [27] made use of three cost-sensitive boosting algorithms to classify defect-prone modules and not-defect-prone ones in software defect prediction context where most of the problems have different costs of misclassification, i.e. misclassification of defect-prone are much more expensive than not-defect-prone ones. The first boosting algorithm is based on threshold-moving and the two others are weight-updating based algorithms. This study represents that threshold moving in most of the software defect prediction problems can be the best choice to develop a cost sensitive ANN.

Ma et al. [28] developed some scenarios for multiple-cost extension of back-propagation with different class-based costs and compare them in terms of both accuracy-based and cost-based performance measures. Their findings imply that there is a significant relation between accuracy and misclassification cost. Also they represent that, to reach a high accuracy and consequently a lower misclassification cost, in the case of non-constant cost matrix, the best way is to maintain separate matrices and not to merge them on an overall cost matrix.

The most relevant approach to this study is proposed by Kukar and Kononenko [21] which is based on an earlier study of Breiman et al. [33]. They studied a modification on the learning rate of cost function in neural networks which assigns variable penalties to different misclassifications of instances with respect to their type of misclassification as follows:

$$\eta(p) = \frac{\eta \cdot CostVector[class(p)]}{\max_i CostVector(i)} \tag{1}$$

Here, p is the training example and as demonstrated above, the learning rate for each instance is modified considering its type of misclassification. We used this approach as a benchmark to our instance-based weights which are discussed in the next section.

3. The original and profit-based Artificial Neural Networks

3.1. Original ANN

Artificial Neural Network is the mathematical representation of biological neural network in human's body [34]. Neural networks have two types: single-layer perceptron and multi-layer perceptron (MLP). Multi-layer perceptron which has been used in this paper has three kinds of layers where information is transferred between them with weighted connections: input layer, hidden layers and output layer. Input layer has some neurons which get the inputs from data set multiplied by their weights and transfer them to the hidden layer with some other set of weights. Hidden layers also have neurons which get the information from input layer and map them between zero and one by using sigmoid function in this study. Then the hidden layers transfer their outputs to the output layer with weighted connections. In each layer, there is some bias for each neuron to make the model easier to predict the exact target. If the output neurons have linear function, the target will be a real number and it is an appropriate function for regression. Nonetheless, as the problem proposed in this paper is the prediction of two classes, cases (positives) and non-cases (negatives), the model has to be a classification model and here the sigmoid function which is represented below, is used instead of simple linear function:

$$y = \frac{1}{1+e^{-z}} \tag{2}$$

Here y is between zero and one and z comes from the following formula:

$$z = b + \sum_i x_i w_i \tag{3}$$

Where b is bias, x_i is the input vector for instance i and w_i is its corresponding weight.

Typically sigmoid function is used in neural network because it has nice derivatives which simplify learning procedure. The following figure (Fig. 1) shows the behavior of sigmoid function as a transferring and learning function:

In neural network representations, the bias is often given the value of 1 to be able to write it in vector representation as follows:

$$z = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x} \tag{4}$$

Here $\mathbf{x}$ is the input vector and $\mathbf{w}$ represents the weight vector which the network is going to learn and predict the target as the error of the prediction is in its minimum amount.

The network uses the chain rule to get the derivatives needed for learning the weights of a logistic unit. To learn the weights which minimize the error function, we need the derivative of the output with respect to each weight.

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - y_n)^2 \tag{5}$$

$$\frac{\partial y}{\partial w_i} = \frac{\partial z}{\partial w_i} \frac{\partial y}{\partial z} = x_i y(1-y) \tag{6}$$

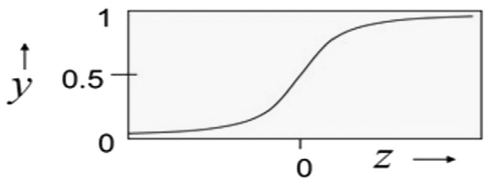

Fig. 1. Sigmoid function.

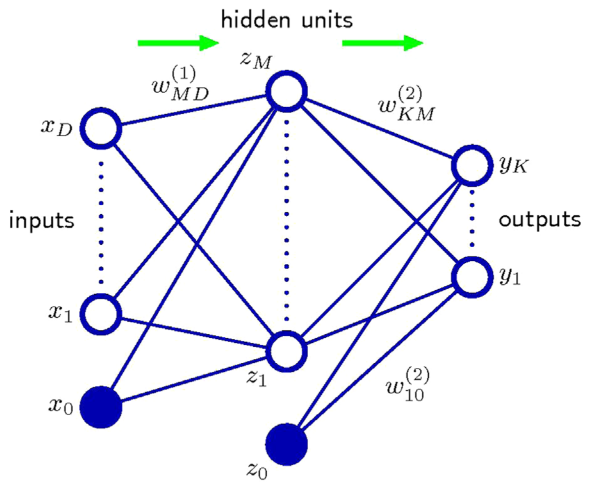

Fig. 2. ANN with one hidden layer [34].

$$\frac{\partial E}{\partial w_i} = \sum_n \frac{\partial y^n}{\partial w_i}\frac{\partial E}{\partial y^n} = -\sum_n x_i^n y^n(1-y^n)(t^n-y^n) \tag{7}$$

x^n,y^n,t^n are respectively the nth input, its corresponding output and its target value.

Network without hidden units are very limited in the input–output mappings they can model hence, in complex data sets, adding a hidden layer makes them much more powerful. A neural network with just one hidden layer has been visualized in Fig. 2 which depicts some neurons in each of three layers. The hidden layer and output units use sigmoid functions so the output of the network will be a value between 0 and 1 (the probability of taking one for the output).

The idea behind the back-propagation is using error derivatives with respect to hidden activities instead of using only desired activities to train the hidden units. Each hidden activity can affect many output units and can therefore have many separate effects on the error. These effects must be combined. Error derivatives can be computed for all of the hidden units efficiently at the same time. Once we have the error derivatives for the hidden activities, it is easy to achieve the error derivatives for the weights going into a hidden unit.

The following equation represents the general sum of squared errors (SSE) and its derivative which has been used in many research to back-propagate in the network using gradient descent. Each weight is updated using the following equations:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - y_n)^2 \tag{8}$$

$$\frac{\partial E}{\partial y_i} = -(t_i - y_i) \tag{9}$$

$$\mathbf{w}^{T+1} = \mathbf{w}^T - ?\nabla E(\mathbf{w}^T) \tag{10}$$

Where $\mathbf{w}^T$ and $\nabla E(\mathbf{w}^T)$ indicate the vector of weights and vector of derivatives of the weights in Tth iteration respectively and η represents the learning rate.

This paper proposes a new error function which modifies the original SSE function to increase the total net profit. In this study, we defined different versions of modifying the error function.

3.2. *Direct weights (DW)*

In this version, error function has been directly multiplied by instances' individual weights. In other words, each instance's error is multiplied by its profit. Accordingly, as profitability of instances increase, the penalties of their misclassification increase proportionally. The main drawback for this model is its zero error for non-profitable instances. If the profit for instance is zero and it has been incorrectly classified, the model will not try to correct this misclassification. The following expression is the error function for this version of modification:

$$E(\mathbf{w}) = (t_i - \mathbf{y}(x_i, \theta))^2 * A_j \tag{11}$$

Where A_j is the amount of individual profit for instance j.

3.3. *Profit-based ANN (PNN)*

Our main goal is to correctly classify the profitable instances as much as possible with minimum decrease in the accuracy of detecting other instances (i.e. less profitable ones). For this reason, an indicator has been used in the error function to make the algorithm more sensitive to high profitable instances. Accordingly, we used a multiplier to intensify the individual penalty of profitable false negatives (in CC Fraud, fraudulent transactions whose usable limit is more than the average).

Indicator should indicate the profitable (important) instances which is the Usable Limit in the context of credit card fraud and the customer revenue (balance) in direct marketing. Thus, indicator has been defined as

$$I(A_j) = \begin{cases} 1 & \text{if } A_j \geq \overline{A} \\ 0 & \text{otherwise} \end{cases}$$

Here A_j is the individual profit of instance j and $\overline{A}$ is the total average of instances' individual profits. In this version, multiplier in the error function (penalty) can be defined as profitability of ith instance which is sensitive to its individual profit. Consequently, the error function can be defined as

$$E(\mathbf{w}) = \sum_{j \in train} (\mathbf{t}_j - \mathbf{y}(x_j, \theta))^2 * \left(\frac{A_j}{\overline{A}}\right)^{I(A_j)} \tag{12}$$

As low profitable instances are not going to be multiplied by a value, we can assume that they will be multiplied by one. We aim to make a connection between instances and their individual penalties to ensure that profitable ones will be classified correctly as much as possible.

We can consider this modification from another point of view. A learning rate is a user-defined value to determine how much the weights of examples can be modified at each iteration. We can assume that the learning rate has been modified to assign an appropriate individual penalty for each example and penalize the misclassified important examples considering their individual importance. Moreover, a cost matrix (net profit matrix) can be represented to show the individual costs and profits (Table 3). Where A_i and A'_j are profits of true positive and true negative and C_i and C'_j are costs of false negative and false positive, respectively.

3.4. *PNN using logarithm (LOG-PNN)*

Neural network is very sensitive to any multiplier in error function and if it is a large value the error function will be unstable

Table 3
Individual cost matrix (net profit matrix).

		Predicted class	
		Case	Non-case
Actual class	Case	A_i	C_i
	Non-case	C'_j	A'_j

and if it is a small value, the rate of learning will be decreased considerably.

As the ratio $A_j/\overline{A}$ in the previous version can give out large values it may cause instability in the model, so for the sake of limiting the values it can take, we can use logarithm function in an alternative version of error function. Hence, the penalty for each instance can be defined as

$$P_j = \log\left(\frac{A_j}{\overline{A}}+1\right) \tag{13}$$

The value of one inside the logarithm guarantees that the output will always be positive. The penalty function and weight updating equations can be expressed as

$$E(\mathbf{w}) = \sum_{j \in train} (\mathbf{t}_j - \mathbf{y}(x_j,\theta))^2 * \left(\log\left(\frac{A_j}{\overline{A}}\right)+1\right)^{I(A_j)} \tag{14}$$

$$w_j^{T+1} = w_j^T - \eta \nabla E_n(w_j^T) * \left(\log\left(\frac{A_j}{\overline{A}}\right)+1\right)^{I(A_j)} \tag{15}$$

This version of error function can make the network more stable when the profitability ratio has a large spread, for example, in the first data set used in this study, the usable limit attribute is considered as profit based metric. The range of this attribute is [0, 99 ,714] with average of 1816. Accordingly the profit ratio $A_j/\overline{A}$ has a range which contains very large values as a multiplier to the error function. If we use logarithm function, the penalty will have the range $[0, \log(54.9)] = [0, 1.7474]$.

3.5. LOG-PNN without using indicator (LPWI)

In this version of error function we use the modified version of LOG-PNN without the indicator. Therefore, not only misclassified profitable instances but also all misclassified ones are penalized proportional to their profitability. The error function is as follows:

$$E(\mathbf{w}) = \sum_{j \in train} (\mathbf{t}_j - \mathbf{y}(x_j,\theta))^2 * \left(\log\left(\frac{A_j}{\overline{A}}\right)+1\right) \tag{16}$$

3.6. LOG-PNN without average (LPWA)

This version of ANN uses the logarithm of each instance's profit as its multiplier in the error function:

$$E(\mathbf{w}) = \sum_{j \in train} (\mathbf{t}_j - \mathbf{y}(x_j,\theta))^2 * (\log A_j)^{I(A_j)} \tag{17}$$

3.7. Weights of modified Fisher (MF) [8]

This version of ANN has been benchmarked from a recent profit-based study in fraud detection context which was the best choice to generate weights for Fisher Discriminant classifier. In this model, there is no indicator for profitable instances where all of the instances are given a weight related to their potential profit. We use this approach as a benchmark to compare its performance with other weight generation methods. The error function for this approach is as follows:

$$E(\mathbf{w}) = \sum_{j \in train} (\mathbf{t}_j - \mathbf{y}(x_j,\theta))^2 * \left(\left(\frac{A_j}{\overline{A}}\right)+1\right)^{1/2} \tag{18}$$

3.8. MAX-PNN

This version of error function uses the Kukar's way of weight generation [21] and it gives different weights for different instances considering their profit of correct classification (originally cost of misclassification). The only difference between this approach and Kukar's approach is that here all of the instances have individual cost of misclassification where in the original paper they studied class-based costs:

$$E(\mathbf{w}) = \sum_{j \in train} (\mathbf{t}_j - \mathbf{y}(x_j,\theta))^2 * \left(1+\left(\frac{A_j}{\max_j(A_j)}\right)\right) \tag{19}$$

3.9. Class-based cost-sensitive ANN (CNN)

In this case we take into account the imbalance of data and importance of classes and assign different costs of misclassification to each of the classes. The ratio of misclassification cost of each class is inversely proportional to the number of instances from that class in the test set. For example if the ratio of positive instances to negative ones is 1/10, then the cost of misclassified positive (FN) is 10 times of cost of a misclassified negative (FP).

Note that as expressed previously, accuracy and other performance metrics based on accuracy are not suitable for cost-sensitive or profit-based classification models. We have used four different measures to compare the models, where two of them are accuracy-based measures which are: Accuracy and True Positive (TP) rate. Next two measures are cost/profit based measures. “Saving” measures the amount of profit in each model with threshold 0.5. The “Net profit in top 10%” (10% is the proportion of actual positives in the test set) evaluates net profit when the cutoff point is the score of top 10%th instance. This measure has an advantage that does not care about the number of total positives in

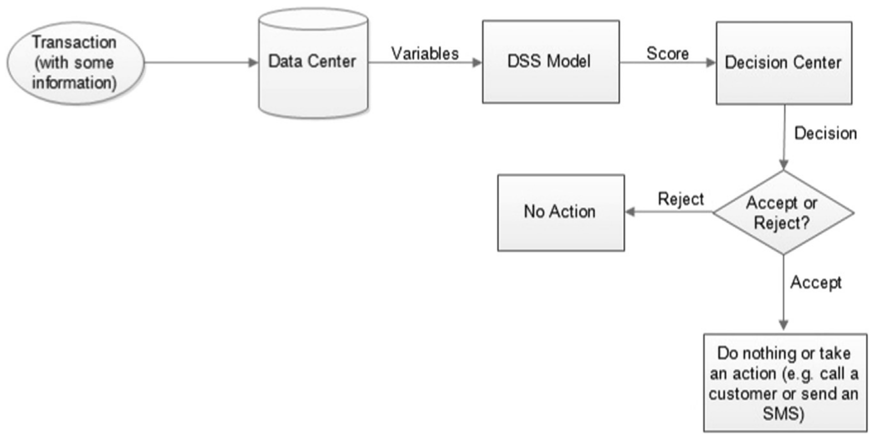
Transaction (with some information)
Data Center
Variables
DSS Model
Score
Decision Center
Decision
Accept or Reject?
Reject
No Action
Accept
Do nothing or take an action (e.g. call a customer or send an SMS)

Fig. 3. System model for the proposed decision support system (DSS).

each classifier, but it gives more importance to the actual number of positives detected in the first top positives in each model and sums their net profits.

The proposed decision support system is given in Fig. 3.

4. Research design and experimental results

In this paper, three data sets have been used to investigate how the proposed models work. All of our ANN versions have benefited from the Levenberg–Marquardt algorithm to train the neural networks. Note that we used one hidden layer ANN with three hidden units and there is just one output unit in the output layer. Also we used default value of 30 for the number of epochs. Two of data sets are new real-life bank credit card data sets which have been acquired from two well-known Turkish banks and the third one is a bank direct marketing data set obtained from UCI repository [11]. The description of each data set is as follows:

The first data set includes 9388 transactions' information made by customers of a bank with a balance of 9 to 1 (legitimate transactions to fraudulent transactions); 939 of them are fraudulent cases and the rest are legitimate ones. The number of attributes is 102. The second data set has 5960 examples which include 1019 fraud cases and 4941 legitimate ones and it has 46 attributes. The third data set is related to direct marketing campaigns of a Portuguese banking institution which contains 3463 customers' information, 460 of them are subscribed (positive) and 3003 are not subscribed (negative) examples and it has 16 attributes. In the empirical study of each data, the data set has been divided in a way that 2/3 proportion is used to train the model and 1/3 is used to test the trained model. In all of the data sets, we have a profit-based attribute which individualize the penalties in each case where they are usable limit for credit card data sets and balance of customers in direct marketing data set. The ANN model automatically extracts a validation data set (15%) from training data set in order to avoid over-fitting. Table 4 represents all of the information about data sets and ANNs used.

Empirical results of testing the seven presented models, class-based cost-sensitive ANN and also the original neural network has been analyzed thoroughly. In all of the models the train sets and test sets are the same, however, as the initial weights are generated randomly in neural network, each of the models has been run ten times and the average of runs plus their standard deviations are considered as classifiers' final performance. Moreover, note that we applied decision tree (C4.5) first and then trained ANN models with those variables used in DT. As such, 27, 20 and 14 variables are used for the training of data set one, two and three, respectively. The expression below demonstrates how to calculate the amount of net profit for each model:

$$NP = \sum_{i=1}^{N_{TP}} (A_j - c) + \sum_{k=1}^{N_{FP}} (-c) \tag{20}$$

Where c is the fixed cost of action (cost of contacting the customer) and N_{TP} and N_{FP} indicate the number of true positives and false positives, respectively. As mentioned above, A_i is the amount of profit gained when the instance i is classified correctly. In context of credit card fraud prediction it is the usable limit of the corresponding card used to make the transaction i. Furthermore, there is a fixed cost of applying the decision support system, but since it is fixed and has to be added for all models, this is not taken into consideration in the comparison procedure. The threshold has been changed from 0.5 to the number of cases (positives) in test set to show that in the top most probable instances (top 10%), which of the classifiers is successful. For the threshold of 0.5 case, we presented the net profit under the column titled "saving" to make difference between these two concepts.

For the first data set the results are presented in Table 5. In the test set of this data set we have overall 3130 instances, where 313 of them are fraudulent transactions and 2817 are legitimate. As the number of positive instances is 313 in the test set, the threshold here has been chosen the 313th instance's output to analyze classifiers' performance. In the context of credit card fraud, the most important profit-based attribute is the usable limit of each card. If we correctly detect fraudulent cases, we save their usable limit subject to a cost of contact. Let us consider the base scenario as the case where all transactions are supposed to be legitimate. It is a common approach for evaluating the profit of applying data mining algorithms. Then, the total saving that can be obtained from the implementation of a model will be

$$NP = \sum_{i=1}^{N_{TP}} (UL_i - c) + \sum_{k=1}^{N_{FP}} (-c) \tag{21}$$

Where, UL_i indicates the usable limit of the card of the i^{th} transaction and c is the contact cost which is fixed for all cases.

Table 5 illustrates the performance of seven models on the first data set. There are two types of measures here. Six of them (Threshold = 0.5) are accuracy based and show the number or percentage of correct classifications . Original ANN has the greatest accuracy as it tries to classify instances as correct as possible where instances' profitability is not important. Naïve Bayes classifier has the best performance in saving when threshold is 0.5. But as the amount of saving depends on the number of the frauds detected, if a classifier like Naïve Bayes labels most of the instances as positive (fraudulent) the saving amount will be very high accompanied by a decrease in accuracy which is what happened in this case. True positive rate and false positive rate have their best amount in original ANN and this demonstrates that Original ANN has the best performance in statistical measures. Amount of net profit when threshold is top 10%, has its highest amount in LPWA (a version of PNN which makes use of logarithms of usable limit as weights), however its accuracy based measures are not very less than original ANN. As shown in Fig. 4, most of the usable limits are distributed between 0 and 5000 with some large limits as well. In Fig. 5 the distribution of weights generated by the LPWA has been shown which have been distributed between 3 and 5 and most of them are around three. These weights are more efficient than other models' weights as they have maximized the net profit amount in this data set. Note that in Fig. 4, there are considerable

Table 4
Data sets and ANNs used in this study.

Name	No. of features	Size of samples	Training ratio	Testing ratio	Validation ratio	No. of hidden layers	No. of hidden units	No. of Epochs	Training method
Credit card fraud 1	27	9388	2/3	1/3	15% of training set	1	3	30	Levenberg–Marquardt
Credit card fraud 2	20	5960	2/3	1/3	15% of training set	1	3	30	Levenberg–Marquardt
Direct marketing	14	3463	2/3	1/3	15% of training set	1	3	30	Levenberg–Marquardt

amount of usable limits around zero which have not been considered in Fig. 5 because the model LPWA just assign weights for profitable instances, i.e. the instances which have usable limits more than average. The superiority of LPWA to original ANN is statistically significant based on a t-test at $\alpha = 0.1$ level.

Table 5 illustrates performance of proposed methods for the second credit card fraud data set. For this data set, test set includes 1986 instances of 339 frauds and 1647 legitimates. Therefore, the threshold here is the 339th instances score (top 17%).

As Table 6 demonstrates, proposed PNN has the best performance in terms of net profit when the threshold is top 17%. However, considering statistical measures, original ANN and Naïve Bayes outperform others. As can be seen in the table, in terms of saving when the cutoff point is 0.5, Naïve Bayes has the best performance, however its accuracy is low and this shows that most of the instances are classified as positive (Fraud) and there is a high false positive rate. As shown in Fig. 6, most of the usable limits of instances are near to each other and around 3500 and there are some large usable limits near 100,000. These large limits make the average a bit skewed. Fig. 7 shows the weights generated by PNN for profitable instances. We confirmed that the superiority of PNN over ANN is statistically significant based on a t-test with $\alpha = 0.1$.

For the third data set which is about the prediction of the success of telemarketing calls for selling bank long-term deposits, the results are presented in Table 7. In the test set we have 1154 instances overall, which contains 153 cases (customers who accept to subscribe) and 1001 non-cases (who reject to subscribe). Threshold has been chosen the 153rd instance's score as the number of positives is 153 (top 10%). Here the base scenario which can be used to find the profit and cost of implementing the model is making no contacts with customers. Hence, the expression to calculate the saving amount is shown as follows:

$$NP = \sum_{i=1}^{N_{TP}} (B_i - c) + \sum_{k=1}^{N_{FP}} (-c) \tag{22}$$

Where B_i demonstrates the amount of balance gained from subscription of customer i, c is the fixed cost of contacting to each customer. This expression means that if the model is implemented, there will be such benefits and costs gained from all instances compared to the case of not implementing the model.

In Table 7 again original ANN has the largest accuracy and Naïve Bayes classifier reaches the largest saving when the threshold is 0.5; however, it has the worst accuracy due to large number of false positives. A version of proposed model which uses the benchmark weights [8] has the greatest net profit when the threshold is top 10%. These results represent that, although the proposed model using benchmark weights, has less number of true positives in the test set, it has detected the most profitable cases and maximizes the total net profit. Fig. 8 shows the distribution of balance in the customer's account and it shows that most of them are between zero and 5000 and there are some

Table 5
The results of all models for the first fraud data set.

Version of ANN	Threshold = 0.5						Threshold = 10%		
	Accuracy (%)	Saving (%)	TN (#)	FN (#)	TP (#)	FP (#)	TP rate	TN rate	Net profit (%)
ANN	**94.73 ± 0.43**	81.69 ± 9.46	2476.57 ± 816.74	99.70 ± 18.35	213.30 ± 18.35	65.30 ± 27.75	0.73 ± 0.01	**0.97 ± 0.00**	84.54 ± 7.17
DW	92.51 ± 4.55	76.56 ± 14.19	2718.40 ± 111.63	136.20 ± 36.23	159.40 ± 58.94	97.60 ± 111.63	0.64 ± 0.17	**0.97 ± 0.00**	78.73 ± 18.02
PNN	94.63 ± 0.34	77.52 ± 10.38	2758.00 ± 12.63	110.10 ± 14.02	202.90 ± 14.02	58.00 ± 12.63	**0.74 ± 0.01**	0.90 ± 0.01	81.33 ± 9.81
LOG-PNN	94.56 ± 0.43	80.98 ± 8.19	2748.20 ± 14.55	102.50 ± 9.77	210.50 ± 9.77	67.80 ± 14.55	0.72 ± 0.03	**0.97 ± 0.00**	82.71 ± 8.45
LPWI	45.76 ± 34.49	47.55 ± 35.22	1311.40 ± 1175.64	192.60 ± 111.50	84.00 ± 60.92	1504.60 ± 1175.64	0.09 ± 0.05	**0.97 ± 0.00**	20.27 ± 10.34
LPWA	94.68 ± 0.37	84.56 ± 5.12	**2759.10 ± 11.38**	109.10 ± 16.48	203.90 ± 16.48	**56.90 ± 11.38**	0.72 ± 0.02	**0.97 ± 0.01**	**87.74 ± 4.67**
MF	94.31 ± 1.11	78.38 ± 17.33	2754.70 ± 16.88	107.80 ± 53.10	196.80 ± 42.37	61.30 ± 16.88	0.71 ± 0.07	**0.97 ± 0.01**	83.37 ± 12.31
MAX-PNN	94.55 ± 1.01	81.31 ± 8.48	2750.80 ± 11.65	105.40 ± 33.09	207.60 ± 33.09	65.20 ± 11.65	0.71 ± 0.10	0.96 ± 0.02	83.40 ± 8.83
CNN	92.90 ± 2.70	75.99 ± 16.76	2685.80 ± 90.65	92.30 ± 46.71	220.70 ± 46.71	130.20 ± 90.65	0.67 ± 0.11	0.96 ± 0.01	79.66 ± 9.97
DT	94.25	84.11	2728	92	221	88	0.7	**0.97**	84.4
NB	82.01	**90**	2290	**37**	**276**	526	0.67	0.96	84.55

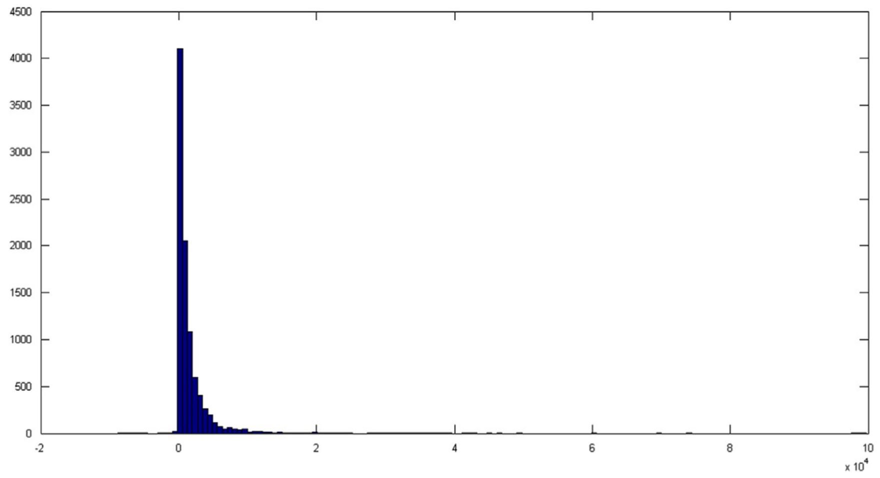

Fig. 4. Distribution of usable limits in the first data set.

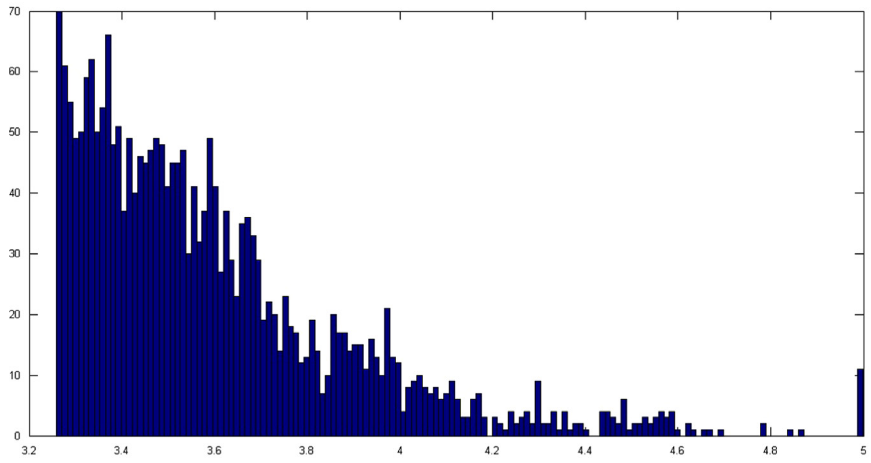

Fig. 5. Distribution of weights generated by LOG-PNN in the first data set.

Table 6
The results of all models for the second fraud data set.

Version of ANN	Threshold = 0.5						Threshold = 17%		
	Accuracy (%)	Saving (%)	TN (#)	FN (#)	TP (#)	FP (#)	TP rate	TN rate	Net profit (%)
ANN	**80.91** ± **1.03**	50.29 ± 4.26	1430.60 ± 33.86	182.80 ± 15.50	156.20 ± 15.50	215.40 ± 33.86	0.46 ± 0.03	**0.91** ± **0.01**	46.49 ± 3.22
DW	80.26 ± 0.93	44.26 ± 3.45	1444.60 ± 25.68	190.80 ± 11.48	148.20 ± 11.48	201.40 ± 25.68	0.43 ± 0.02	0.90 ± 0.01	43.15 ± 2.34
PNN	82.30 ± 0.27	49.95 ± 9.91	1460.80 ± 3.92	156.40 ± 13.95	182.60 ± 13.95	185.20 ± 3.92	**0.48** ± **0.04**	0.89 ± 0.01	**49.26** ± **4.70**
LOG-PNN	80.34 ± 1.39	48.12 ± 10.00	1440.80 ± 29.74	166.00 ± 30.40	173.00 ± 30.40	205.20 ± 29.74	0.44 ± 0.04	**0.91** ± **0.01**	44.46 ± 4.36
LPWI	78.70 ± 1.09	38.98 ± 9.20	1430.60 ± 30.08	207.20 ± 32.34	131.80 ± 32.34	215.40 ± 30.08	0.36 ± 0.08	0.89 ± 0.01	35.76 ± 7.19
LPWA	76.18 ± 5.47	34.79 ± 7.15	1394.80 ± 120.79	221.80 ± 24.47	117.20 ± 24.47	251.20 ± 120.79	0.33 ± 0.07	0.90 ± 0.01	33.17 ± 7.89
MF	81.46 ± 1.14	41.03 ± 16.26	**1479.20** ± **47.27**	201.40 ± 53.27	137.60 ± 53.27	**166.80** ± **47.27**	0.43 ± 0.06	0.90 ± 0.01	43.99 ± 6.74
MAX-PNN	80.02 ± 1.51	47.97 ± 12.88	1427.00 ± 64.71	177.80 ± 43.08	161.20 ± 43.08	219.00 ± 64.71	0.44 ± 0.04	0.90 ± 0.00	44.25 ± 4.33
CNN	70.42 ± 9.44	46.57 ± 14.54	1233.83 ± 220.64	174.67 ± 50.04	164.33 ± 50.04	227.17 ± 90.72	0.34 ± 0.06	0.86 ± 0.01	35.18 ± 6.00
DT	78.34	50.33	1383	170.00	169.00	263.00	0.43	0.90	43.88
NB	39.95	**71.22**	548	**94.00**	**245.00**	1098.00	0.32	0.88	32.45

negative balances in the data set. Fig. 9 shows the weights generated by the proposed model using benchmark weights. Most of the generated weights are between 1 and 2 so it may result in a stable neural network. We confirmed that the superiority of MF over ANN is statistically significant based on a t-test with $\alpha = 0.1$.

Note that in all of the data sets, Naïve Bayes classifier has the best amount of saving while the threshold is 0.5. However, in all of the data sets it has low “accuracy” and “true positive rate in top 10%” (i.e. highest false positive rate) and this shows that in all of the cases it has labeled more instances as positives. When the threshold is the top 10%, all of the classifiers have the same opportunity to detect true positives among these instances. Other important issue is the low performance of the Classifier “LOG-PNN without indicator” which comes up from the fact that in this model the generated weights are less than one for the less profitable instances and this makes the error less than its regular value which consequently has bad effects on ANN learning process.

There is no single champion model for all of the data sets in these classification problems and the best model in terms of saving is different from one case to another. The reason is that different ways of weight generations have different distribution of weights for each of the data sets. The best model is the one which generates the most appropriate penalties for instances with different profits in each data set. Thus, we can recommend that one should try all alternative error functions to determine which one will perform best for his/her data set. Also in the traditional class-based cost-sensitive ANN (CNN), The model is not reasonably consistent in terms of saving (profit) and the proposed models significantly outperform it in terms of total profit.

5. Summary and conclusion

In this study, a novel profit-based neural network has been proposed which makes the classification considering all individual costs and profits of each of the instances and consequently maximizes the total net profit captured from applying the classification model. For this purpose, we modified the neural network error function which is sensitive to each instance's misclassification considering its profitability. Different models have been proposed to generate weights (penalties) for modification of error function. All of the models, class-based cost-sensitive ANN (CNN) and two well-known classifiers, Decision tree and Naïve Bayes, have been tested on two real-life fraud data sets and a UCI direct marketing data set. In order to evaluate the classifiers, both accuracy-based and profit-based performance metrics have been used.

Results represent that, original ANN has the best performance in terms of statistical measures (accuracy and true positive rate). However, in terms of net profit, different versions of the proposed model outperform others and the way of generating weights are different from a data set to another. Moreover, Naïve Bayes classifier has the highest performance in saving when threshold is 0.5 but its true positive rate is lower than others and when threshold is changed from 0.5 to top n^{th} instance's score (n is the

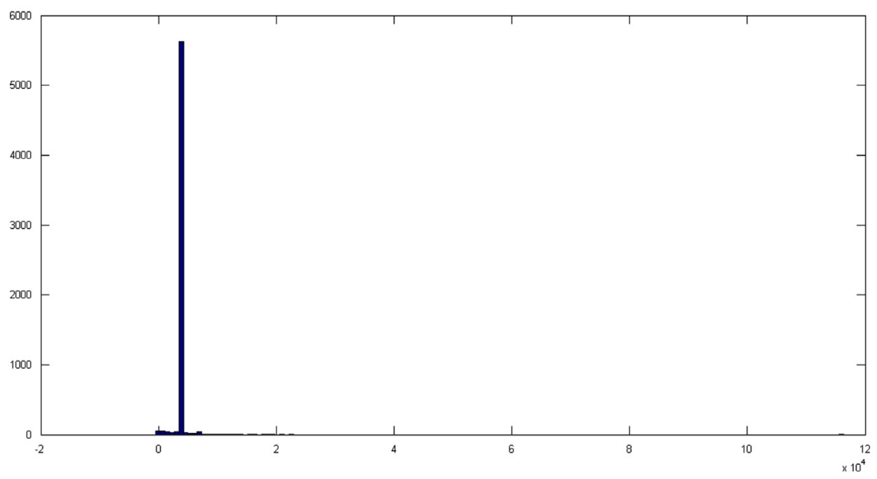

Fig. 6. Distribution of usable limits in the second data set.

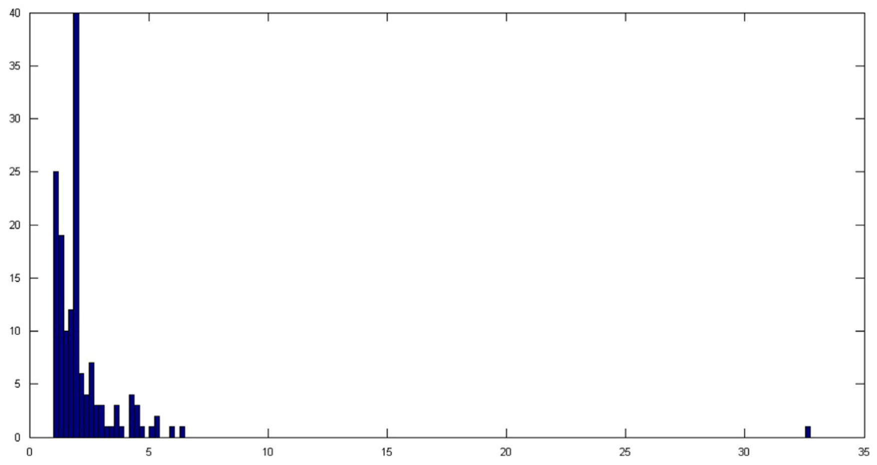

Fig. 7. Distribution of weights generated by PNN in the second data set.

Table 7
The results of all models for the direct marketing data set.

Version of ANN	Threshold = 0.5						Threshold = 10%		
	Accuracy (%)	Saving (%)	TN (#)	FN (#)	TP (#)	FP (#)	TP rate	TN rate	Net profit (%)
ANN	**88.30 ± 0.55**	27.24 ± 8.36	982.00 ± 8.29	104.80 ± 43.66	42.60 ± 33.06	23.80 ± 14.99	**0.55 ± 0.01**	**0.93 ± 0.00**	61.15 ± 3.19
DW	87.12 ± 0.70	8.50 ± 10.29	**993.20 ± 7.63**	141.60 ± 15.24	11.40 ± 15.24	**6.80 ± 7.63**	0.44 ± 0.17	0.91 ± 0.03	49.80 ± 19.23
PNN	87.90 ± 0.46	21.73 ± 5.68	983.60 ± 7.58	123.00 ± 7.46	30.00 ± 7.46	16.40 ± 7.58	0.53 ± 0.02	**0.93 ± 0.00**	59.29 ± 3.21
LOG-PNN	88.38 ± 0.81	28.97 ± 10.92	982.20 ± 5.84	112.80 ± 11.05	40.20 ± 11.05	17.80 ± 5.84	0.53 ± 0.02	**0.93 ± 0.00**	58.37 ± 4.26
LPWI	87.82 ± 0.91	17.26 ± 13.07	988.83 ± 9.42	129.17 ± 18.87	23.83 ± 18.87	11.17 ± 9.42	0.40 ± 0.11	0.91 ± 0.02	44.82 ± 12.22
LPWA	87.80 ± 1.04	24.41 ± 10.50	978.80 ± 4.92	119.20 ± 15.47	33.80 ± 15.47	21.20 ± 4.92	0.50 ± 0.07	0.92 ± 0.01	57.06 ± 9.78
MF	87.76 ± 0.42	20.80 ± 8.04	987.20 ± 9.28	125.40 ± 11.48	27.60 ± 11.48	12.80 ± 9.28	0.54 ± 0.03	**0.93 ± 0.00**	**63.13 ± 4.64**
MAX-PNN	87.73 ± 0.86	20.73 ± 13.80	981.67 ± 16.18	123.00 ± 19.71	30.00 ± 19.71	18.33 ± 16.18	0.46 ± 0.17	0.83 ± 0.03	51.56 ± 20.50
CNN	82.62 ± 6.35	38.87 ± 19.04	900.40 ± 97.81	101.00 ± 27.88	52.00 ± 27.88	99.60 ± 97.81	0.38 ± 0.09	0.90 ± 0.01	42.78 ± 9.71
DT	85.86	9.81	975	141	12	22	0.24	0.88	24.4
NB	85.34	**41.35**	925	**94**	**59**	75	0.42	0.91	43.79

number of actual positives in the test set) to ignore the total number of positives labeled by classifiers, Naïve Bayes has lower performance in terms of both accuracy and net profit when compared to proposed models.

As for the future research, we are working on a profit-based ANN which does not work based on minimizing an error function but maximizing a net profit function and finds the appropriate weights based on it.

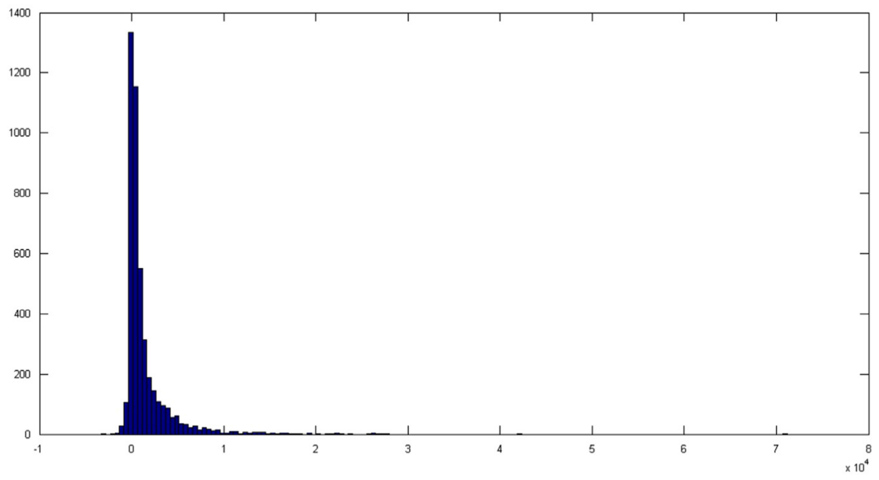

Fig. 8. Distribution of weights in the third data set.

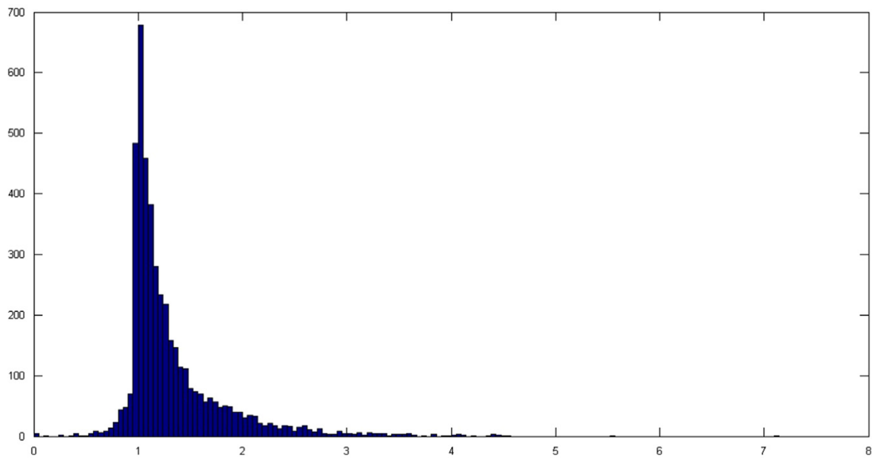

Fig. 9. Distribution of weights generated by MF in the third data set.

Acknowledgments

With a deep sense of gratitude the authors would like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project no. 113M063.

References

[1] N. Chen, B. Ribeiro, A.S. Vieira, J. Duarte, J.C. Neves, A genetic algorithm-based approach to cost-sensitive bankruptcy prediction, Expert Syst. Appl. 38 (September (10)) (2011) 12939–12945.
[2] P.C. Pendharkar, A threshold-varying artificial neural network approach for classification and its application to bankruptcy prediction problem, Comput. Oper. Res. 32 (October (10)) (2005) 2561–2582.
[3] P. Pendharkar, A threshold varying bisection method for cost sensitive learning in neural networks, Expert Syst. Appl. 34 (February (22)) (2008) 1456–1464.
[4] A. Ghazikhani, R. Monsefi, H. Sadoghi Yazdi, Ensemble of online neural networks for non-stationary and imbalanced data streams, Neurocomputing 122 (December) (2013) 535–544.
[5] H.K. Lam, U. Ekong, H. Liu, B. Xiao, H. Araujo, S.H. Ling, K.Y. Chan, A study of neural-network-based classifiers for material classification, Neurocomputing 144 (November) (2014) 367–377.
[6] G. Sateesh Babu, S. Suresh, Meta-cognitive neural network for classification problems in a sequential learning framework, Neurocomputing 81 (April) (2012) 86–96.
[7] E. Duman, M.H. Ozcelik, Detecting credit card fraud by genetic algorithm and scatter search, Expert Syst. Appl. 38 (September (10)) (2011) 13057–13063.
[8] N. Mahmoudi, E. Duman, Detecting credit card fraud by modified Fisher discriminant analysis, Expert Syst. Appl. 42 (April (5)) (2015) 2510–2516.
[9] Y. Sahin, S. Bulkan, E. Duman, A cost-sensitive decision tree approach for fraud detection, Expert Syst. Appl. 40 (November (15)) (2013) 5916–5923.
[10] R.J. Bolton, D.J. Hand, F. Provost, L. Breiman, Statistical fraud detection: a review, Stat. Sci. 17 (3) (2002) 235–255.
[11] S. Moro, P. Cortez, P. Rita, A data-driven approach to predict the success of bank telemarketing, Decis. Support Syst. 62 (Jun.) (2014) 22–31.
[12] Kevin Lane Keller Philip Kotler, Framework for Marketing Management, 5th edition, Prentice Hall, Upper Saddle River, New Jersy, 2012.
[13] F. Talla Nobibon, R. Leus, F.C.R. Spieksma, Optimization models for targeted offers in direct marketing: exact and heuristic algorithms, European Journal of Operational Research 210 (May (3)) (2011) 670–683.
[14] J. Lan, M.Y. Hu, E. Patuwo, G.P. Zhang, An investigation of neural network classifiers with unequal misclassification costs and group sizes, Decis. Support Syst. 48 (March (4)) (2010) 582–591.
[15] R. Bhowmik, Data mining techniques in fraud detection, in: Proceedings of the Conference on Digital Forensics, Security and Law, vol. 3, no. 2, April 2008, pp. 57–72.
[16] G.M. Di Nunzio, A new decision to take for cost-sensitive Naïve Bayes classifiers, Inf. Process. Manag. 50 (September (5)) (2014) 653–674.
[17] C.X. Ling, Test-cost sensitive Naïve Bayes classification, in: Fourth IEEE International Conference on Data Mining (ICDM04), 2004, pp. 51–58.
[18] Q. Deng, Detection of fraudulent financial statements based on Naïve Bayes classifier, in: 2010 5th International Conference on Computer Science & Education, 2010, pp. 1032–1035.
[19] J. Kim, K. Choi, G. Kim, Y. Suh, Classification cost: an empirical comparison among traditional classifier, Cost-Sensitive Classifier, and MetaCost, Expert Syst. Appl. 39 (March (4)) (2012) 4013–4019.

[20] C. Elkan, The foundations of cost-sensitive learning, in: International Joint Conference on Artificial Intelligence, 2001.
[21] M. Kukar, I. Kononenko, Cost-sensitive learning with neural networks, in: European Conference on Artificial Intelligence (ECAI), 1998.
[22] E. Zheng, C. Zou, J. Sun, L. Chen, P. Li, SVM-based cost-sensitive classification algorithm with error cost and class-dependent reject cost, in: 2010 Second International Conference on Machine Learning and Computing, 2010, pp. 233–236.
[23] Zhi-Hua Zhou, Xu-Ying Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Trans. Knowl. Data Eng. 18 (January (1)) (2006) 63–77.
[24] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, Philip K. Chan. AdaCost: misclassification cost-sensitive boosting, in: International Conference on Machine Learning (ICML), 1999, pp. 97–105.
[25] V. Lpez, A. Fernndez, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (June (7)) (2012) 6585–6608.
[26] C.-H. Tsai, L.-C. Chang, H.-C. Chiang, Forecasting of ozone episode days by cost-sensitive neural network methods, Sci. Total Environ. 407 (March (6)) (2009) 2124–2135.
[27] J. Zheng, Cost-sensitive boosting neural networks for software defect prediction, Expert Syst. Appl. 37 (June (6)) (2010) 4537–4543.
[28] G.-Z. Ma, E. Song, C.-C. Hung, L. Su, D.-S. Huang, Multiple costs based decision making with back-propagation neural networks, Decis. Support Syst. 52 (February (3)) (2012) 657–663.
[29] P.C. Pendharkar, A computational study on the performance of artificial neural networks under changing structural design and data distribution, Eur. J. Oper. Res. 138 (April (1)) (2002) 155–177.
[30] V.L. Berardi, G.P. Zhang, The effect of misclassification costs on neural network classifiers, Decis. Sci. 30 (June (3)) (1999) 659–682.
[31] L.M. Salchenberger, E.M. Cinar, N.A. Lash, Neural Networks: A New Tool for Predicting Thrift Failures, Decis. Sci. 23 (July (4)) (1992) 899–916.
[32] Emad W. Saad, Danil V. Prokhorov, Donald C. Wunsch, Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks, IEEE Trans. Neural Netw. 9 (6) (1998) 1456–1470.
[33] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, Taylor & Francis, 1984.
[34] C. Bishop, Pattern Recognition and Machine Learning, 2006.
[35] E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (September (9)) (2009) 1263–1284.
[36] Y. Sahin, E. Duman, Detecting credit card fraud by decision trees and support vector machines, in: International MultiConference of Engineers and Computer Scientists, vol. I, 2011.

Ashkan Zakaryazad received the BSc degree in industrial engineering from Iran University of Science and Technology (IUST), Tehran, Iran in 2012. Since 2013, he has been working toward the MSc degree at the department of industrial engineering at the Ozyegin University, Istanbul, Turkey. Being a member of the Scientific and Technological Research Council of Turkey (TUBTAK), his main research focuses on data mining in business settings, such as credit card fraud detection, churn prediction, direct marketing and credit scoring.

Ekrem Duman is an associate professor at Ozyegin University, Istanbul, Turkey. He obtained his BSc degree from Bogazici University Electrical-Electronics Engineering Department in 1990. Then, he got the MSc and PhD degrees from the Industrial Engineering Department of the same university in years 1994 and 1998, respectively. After a couple of years of industrial experience he worked as a professor in the Industrial Engineering Department of Dogus University between 2001 and 2011 and after that he joined to Ozyegin University. He has done extensive research on predictive analytics, data mining, fraud detection and credit risk management and his findings have been published in well-known international journals (e.g., Computers & Operations Research, System Dynamics Review, International Journal of Production Research, Information Sciences and Expert Systems with Applications). Besides theoretical studies, he has been involved in many industrial (mostly banking) projects as a coordinator or consultant.