Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 12, No. 3, 2024, 409-421.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

A New Structure for Perceptron in Categorical Data Classification

Fariba Taghinezhad and Mohammad Ghasemzadeh*

Department of Electrical and Computer Engineering, Yazd University, Yazd, Iran.

Article Info

Article History:
Received 26 August 2024
Revised 05 November 2024
Accepted 30 November 2024

DOI:10.22044/jadm.2024.14981.2594

Keywords:
Neural Network, Qualitative Data, Categorical Data, Non-Mumeric Data, Binary Classification.

**Corresponding author: m.ghasemzadeh@yazd.ac.ir (M. Ghasemzadeh).*

Abstract

Artificial neural networks are among the most significant models in machine learning that use numeric inputs. This study presents a new single-layer perceptron model based on categorical inputs. In the proposed model, every quality value in the training dataset receives a trainable weight. Input data is classified by determining the weight vector that corresponds to the categorical values in it. To evaluate the performance of the proposed model, we have used 10 datasets. We have compared the performance of the proposed model to that of other machine learning models, including neural networks, support vector machines, naïve Bayes classifiers, and random forests. According to the results, the proposed model resulted in a 36% reduction in memory usage when compared to baseline models across all datasets. Moreover, it demonstrated a training speed enhancement of 54.5% for datasets that contained more than 1000 samples. The accuracy of the proposed model is also comparable to other machine learning models.

1. Introduction

Data is a collection of observed facts that can be divided into two categories: numeric and categorical. Numeric data, also known as quantitative data, are measurable and computable data and can be classified into two discrete and continuous categories [1]. Categorical data, also called non-numeric or qualitative data, represents characteristics. Such data is visible but cannot be calculated, such as color, style, nationality, gender [2]. In areas such as natural language processing [3], molecular biology, medicine, social sciences, game theory, education, economy, urbanism, classification of protein sequences [4] we are dealing with categorical datasets.

The number of unique categories in a categorical variable is called cardinality. Low-cardinality refers to categorical variables with few unique values, typically less than 100, and do not change over time. Some of the quality variables are high cardinality, such as product IDs, zip codes, or names, as well as words in documents that contain hundreds or even thousands of distinct values [4]. One of the main challenges of using categorical data is to use it as input for machine learning models and neural networks.

Machine learning is a branch of artificial intelligence and computer science that enable computers to automatically learn from data [5]. Neural networks are a class of machine learning models that are the basis of many important and widely used models, such as deep neural networks, convolutional neural networks and recurrent neural networks. These networks are applied to the vector of real numbers [6]. In order to use categorical data as input for neural networks, we must apply techniques to map it to numeric vectors. Such techniques are known as representation, distributed representation, encoding, and embedding [4].

One of the most common approaches to evaluate the performance of the techniques presented is to use them in algorithms and machine learning models for a specific task and then measure the model's end-to-end performance. The techniques presented so far have problems such as generating large and sparse numeric vectors, high memory consumption and high time required for training

embedded vectors, issues with optimizing the values of parameters, lack of attention to semantic relationships between categorical values, and lack of attention to the statistical information contained in the entire data, losing important information during the generation process of numeric vectors. Sometimes, high-cardinality features and variables contain valuable and important information but due to the large dimensions of vectors derived from their encoding, as well as the loss of useful information when categorizing values, the incorporation of such features in machine learning models is rare, and in most cases, they are ignored [7]. In fields such as medicine, categorical datasets may contain important information but unknown. The use of improper techniques could lead to the inability to observe this information and make predictions about patient treatments [8].

This article introduces a novel neural network model that accepts categorical data as input without the need for any mapping procedures. The model incorporates a single categorical neuron and assigns a distinct weight to each categorical value within a feature. Our proposed model eliminates the need for manual feature extraction and feature engineering.

The rest of the article is organized as follows. Section 2 discusses techniques for encoding categorical data and how they are used in neural networks and machine learning models. Section 3 details the principle of the proposed model that is known as proposed perceptron. Section 4 provides a set of experiments that are used to evaluate the model. Section 5 presents the positive aspects of the proposed approach, along with ideas for more research.

2. Discussion

Reily et al. [8] examine the heuristics for pre-processing, selection of the encoding method, and choose the appropriate algorithm for classifying categorical data. Using heuristics like cardinality, ordinal, or nominal quality features, they have developed a flowchart to choose the suitable encoding technique for every feature in the dataset. A table was created for selecting a classification model based on heuristics such as model accuracy, speed, training time, model parameters, dataset features, and size of the dataset.

Alexandridis et al. [9] have developed a method for using categorical data in radial basis function networks called CRBF, and categorical data are used directly in the neural network without mapping. Their proposed model consists of two stages. In the first stage, with the help of unsupervised learning, the number of first-layer centers and the categorical values of centers are estimated. In the second stage, with the help of supervised learning, the weights of the output layer are trained. To calculate the distance between input and quality centers, different distance indicators such as the Hamming distance have been used. The experiment of this model on 22 categorical datasets shows that for 15 cases of datasets the proposed model has yielded better results, and in the rest it has been relatively good performance and equivalent to the rest of the classifiers.

Cardona et al. [10] have proposed a novel way to classify categorical data using machine learning models. First, the Chi-square distance criterion was used to map categorical values to another space, in which data is more separable. Then, they reduced the dimensions of the features by means of the t-SNE algorithm. The features obtained are used to classify data by machine learning models. By testing this method on seven datasets, it can be concluded that the proposed method's execution time for classification is reduced, and is also at the same level of accuracy other methods are.

Perez and Castillo [4] have developed four hypotheses in order to provide a suitable map of protein data to classify protein sequences. These assumptions include translation, permutation, constant, and eigenvalues. Next, they performed a classification of protein sequences based on various models to evaluate their maps. They have shown that there is no significant improvement in the model's average accuracy when mapping with translation, permutation and consistent assumptions and the resulting maps are equivalent to basic mapping. Using the eigenvalue-based mapping, the accuracy of the models will be comparable and the best model has an accuracy of 83.25 percent.

Hancock et al. [5] present a paper that explores the use of categorical data in machine learning models and neural networks. The article categorized categorical data encoding techniques into three groups: determined, algorithmic, and automatic. Determined techniques are the easiest to use, which involve using distinct numeric values to encode categorical values. These values remain consistent throughout the model's train and test phases. Such techniques are suitable for dealing with low-cardinality data because in this case, they have a low mapping memory and time and always use fixed numbers for encoding. These techniques include one one-hot encoding, integer encoding and so on. These techniques can result in high memory consumption for data with a high diversity of values (such as cities or postal codes). Algorithmic techniques are techniques that are

commonly referred to as the preprocessing stages. In these techniques, the transformation of categorical data into vector space requires both time and computational complexity, such as Latent Dirichlet Analysis (LDA) technique.

Automated techniques are techniques that train the numeric representation of categorical data during the training process of neural network. Most of the time, the automatic technique input is a one-hot encoding vector of categorical value, and their output is a vector with a smaller size. As part of the neural network training process, the input weight matrix will be trained and will serve as the mapping table. The weight matrix that is derived can either be applied directly to the primary neural network or used for transfer learning in a different neural network. The time complexity of these techniques is higher and their output will be directly input into classification models. Such techniques in the field of natural language processing are SGNS, CBOW, or GloVe.

Arat [11] has come up with a new approach to utilizing high-cardinality quality features in deep neural networks. In this method, the categorical features with high cardinality are first encoded using the mean-target technique, and then a decimal number is obtained for each value. The weights of each decimal value are updated during the training process of a deep neural network. The values obtained from mean-target encoding and the trained weight form a key-value pair for each categorical value that is stored in a mapping table. The key represents a categorical value with a high cardinality, and the value indicates the weight assigned to the key. According to the experiments, this method can learn neural networks and generate mapping vectors without requiring extra memory or time, unlike Van Hat and Minh Target methods. One-hot encoding [12], integer encoding [13], mean-target encoding [14], feature-hashing [15] techniques are commonly used methods for mapping categorical features. These methods generate large and sparse mapping vectors and by ignoring statistical information, they will cause poor results. In some techniques, due to the generation of the same codes for different categorical values, it will cause the loss of useful information in categorical features and the performance of the final model will be reduced.

Various techniques have been presented to classify categorical features with high cardinality and utilize them optimally in machine learning models. These techniques include grouping and clustering values [16,17] to reduce cardinality, as well as techniques based on training embedded vectors [18,19]. However, these techniques face certain challenges. One challenge is the lack of consideration for the statistical information of the entire data when grouping categorical values. Additionally, the need for an appropriate method to measure similarity between categorical values of categorical features and the disregard for semantic relationships among categorical values in the final vectors have been noted.

Moeyersoms and Martens [8] examine various techniques for using high-cardinality quality features into predictive models. These techniques involve dummy encoding, reducing the cardinality of categorical features by semantically grouping values (e.g., grouping postal code values based on province), and subsequently using dummy encoding to encode the obtained groups. Additionally, mapping functions based on the target function are utilized to convert categorical values into numeric values. Noteworthy mapping functions include Weight of Evidence (WOE), Supervised Ratio (SR), and Perlich Ratio (PR) functions. Through practical experimentation using the same dataset, the researchers have demonstrated that incorporating high-cardinality quality features, along with other common features and appropriate encoding, enhance the performance of the predictive model.

The main issues with the majority of the approaches discussed are: the training or mapping of large datasets requires a lot of time and memory, the statistical information of the entire data is not used, and the semantic relationships between words are not taken into consideration.

In the following section, we will introduce a new model for classifying the categorical data. This model is designed to handle both binary classification and regression challenges. However, this paper will restrict its analysis to the binary classification model, while subsequent works will investigate the model's potential for regression and other problems.

3. Proposed Model

Let's assume that the desired dataset *DS* has *N* data samples with *n* categorical features. So, *n* is the number of features in dataset. Moreover, suppose that for any data sample, we represent the *j*th quality feature as QF_j and the value of the QF_j for the *i*th data sample as $QV_j^{(i)}$.Every categorical feature in the data sample has a categorical value from its own collection of categorical values (according to equation (3)). If we denote the *i*th data sample as $S^{(i)}$ and the target output of the $S^{(i)}$ as $Y_t^{(i)}$, then for the *i*th data sample, we will have:

$$S^{(i)} = [QV_1^{(i)}, QV_2^{(i)}, ..., QV_j^{(i)}, ..., QV_n^{(i)}], \quad i \in \{1, 2, ..., N\} \tag{1}$$

$$Y_t^{(i)} \in \{neg(0), pos(1)\}, i \in \{1, 2, ..., N\} \tag{2}$$

$$\begin{cases} QV_1^{(i)} \in \{qv_{1,1}, qv_{1,2}, ..., qv_{1,d_1}\}, Cardinality(QF_1) = d_1 \\ QV_2^{(i)} \in \{qv_{2,1}, qv_{2,2}, ..., qv_{2,d_2}\}, Cardinality(QF_2) = d_2 \\ ... \\ QV_j^{(i)} \in \{qv_{j,1}, qv_{j,2}, ..., qv_{j,d_j}\}, Cardinality(QF_j) = d_j \\ ... \\ QV_n^{(i)} \in \{qv_{n,1}, qv_{n,2}, ..., qv_{n,d_n}\}, Cardinality(QF_n) = d_n \end{cases} \tag{3}$$

In the above equations, QF_j and $QV_j^{(i)}$, respectively represent *j*th categorical feature and *j*th categorical value of *i*th data sample ($S^{(i)}$); and d_j represents the cardinality of QF_j and $qv_{j,k}$ is a categorical value that represents the *k*th categorical value of QF_j. In the proposed model, any categorical value of any categorical feature will have its own weight, which will be updated during the training procedure of the proposed perceptron. If we display the weight of the $QV_j^{(i)}$ as $W_{QV_j^{(i)}}$, then we have:

$$\begin{cases} W_{QV_0^{(i)}} = w_0 \\ W_{QV_1^{(i)}} \in \{w_{qv_{1,1}}, w_{qv_{1,2}}, ..., w_{qv_{1,d_1}}\} \\ W_{QV_2^{(i)}} \in \{w_{qv_{2,1}}, w_{qv_{2,2}}, ..., w_{qv_{2,d_2}}\} \\ ... \\ W_{QV_n^{(i)}} \in \{w_{qv_{n,1}}, w_{qv_{n,2}}, ..., w_{qv_{n,d_n}}\} \end{cases} \tag{4}$$

where, $w_{qv_{j,k}}$ is a decimal number and represents the weight assigned to $qv_{j,k}$. $W_{QV_0^{(i)}}$ represents the bias weight and we display it with w_0.

The architecture of the common perceptron is shown in figure 1-A and the architecture of the proposed perceptron is shown in figure 1-B. This model is a single-layer neural network based on categorical values. It consists of *n* categorical inputs and one bias input. Its purpose is to perform binary classification on input data.

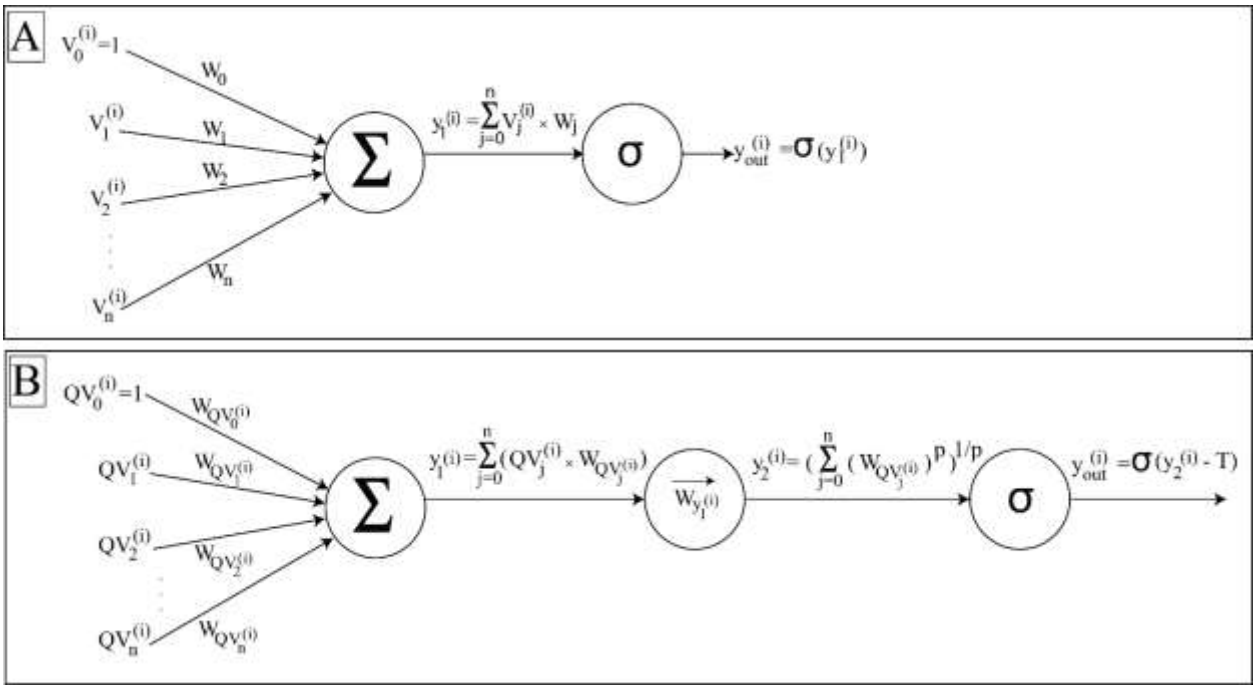

Figure 1. Typical perceptron architecture based on real numbers ($V_j^{(i)}$) and proposed perceptron based on categorical values ($QV_j^{(i)}$), (A) typical perceptron model for binary classification issues ,(B) proposed perceptron for Binary Classification.

As shown in figure 1 and to maintain formula simplicity, the weight of bias is denoted as $W_{QV_0}^{(i)}$, and the value of bias is represented as $QV_0^{(i)} = 1$. The following subsections presents a comprehensive description of the feedforward procedure, the loss function calculation, and the backpropagation procedure as proposed in the model.

3.1. Feedforward on the proposed perceptron for binary classification

Suppose that $y_{out}^{(i)}$ is the estimated output by the proposed perceptron for the $S^{(i)}$. As the figure 1-B

illustrates, in a binary classification problem, the feedforward procedure of the proposed perceptron will be as follows:

$$\begin{cases} y_1^{(i)} = \sum_{j=0}^{n} QV_j^{(i)} \times W_{QV_j^{(i)}} \\ \overrightarrow{W_{y_1^{(i)}}} = (W_{QV_0^{(i)}}, W_{QV_1^{(i)}}, W_{QV_2^{(i)}}, ..., W_{QV_n^{(i)}}) \\ y_2^{(i)} = (w_0{}^p + \sum_{QV_j^{(i)} \in S^{(i)}} (W_{QV_j^{(i)}})^p)^{\frac{1}{p}} \end{cases} \quad (5)$$

In equation (5), $y_2^{(i)}$ is the middle output of the proposed perceptron. The $y_1^{(i)}$ is an algebraic expression. The $\overrightarrow{W_{y_1^{(i)}}}$ is the coefficient vector of $y_1^{(i)}$ which can be called as the coefficient vector of $S^{(i)}$. The p is a natural number. The output $y_2^{(i)}$ depends on the p value, with the difference that in equation (5).

In mathematics, a norm is a function from a real or complex vector space to the non-negative real numbers that behaves in certain ways like the distance from the origin. In the norm of a vector, we use the absolute value of each element. So, the norm of each vector will always be a positive number. This study introduces a new formula that is similar to the norm function for calculating output values ($y_2^{(i)}$), where the actual values of the elements in $\overrightarrow{W_{y_1^{(i)}}}$ are used rather than their absolute values. When p is an odd number, this strategy leads to a distribution of outputs in both positive and negative spaces, which simplifies the classification of the input samples. We can call this new formula as semi-norm function.

It is essential to clarify that in the practical implementation of the proposed model, the algebraic expression $y_1^{(i)}$ and the vector $\overrightarrow{W_{y_1^{(i)}}}$ are not actually generated. These expressions have been utilized to enhance the comprehension of the proposed model. These expressions have been used to illustrate the relationship between the input and the variable $y_2^{(i)}$.

As observed in equation (5), for odd values of p, $y_2^{(i)}$ will always be a real number in the range $(-\infty, +\infty)$ and for even values of p, $y_2^{(i)}$ will always be a real number in the range $[0, +\infty)$. So, for even values of p, we have $0.5 \le \sigma(y_2{}^{(i)}) \le 1$ and the predicted class will always be *'positive'* for all data samples, similar to common neural networks. This means that if we only use the '*bias'* as an input to the neural network then, for even values of p, we will face a problem and all samples will be classified as '*positive'* class.

To avoid this problem, there are two primary approaches: 1) using a *threshold* that is greater than 0.5 to determine the final class, 2) shifting the activation function T units to the right of the y_2-*axis*. In this study, we used the second approach and proposed a new activation function for even values of p. Therefore, $y_{out}^{(i)}$ is obtained from equation (6):

$$y_{out}{}^{(i)} = \sigma(y_2{}^{(i)} - T) = \frac{1}{1 + e^{-(y_2{}^{(i)} - T)}}, T \ge 0 \quad (6)$$

Where, T is the transfer parameter and depends on the p value. If p be an odd number then $T = 0$, and if p be an even number so $T > 0$. The $y_{out}^{(i)}$ is a real number in the range of *(0, 1)*. It depends on the type of problem and other parameters. In a binary classification problem, $y_{out}^{(i)}$ is calculated according to equation (6) and depends on the T value. Now the final class of the data sample $S^{(i)}$ is obtained as follows:

$$\begin{cases} if\ y_{out}{}^{(i)} \ge 0.5 \rightarrow class^{(i)} = pos(1) \\ otherwise \rightarrow class^{(i)} = neg(0) \end{cases} \quad (7)$$

According to the equation (7), $class^{(i)}$ is the final predicted class for the data sample $S^{(i)}$. In a binary classification problem, a data sample can be classified as either *positive (pos)* or *negative (neg)*. Consider equations (5)-(7), it can be observed that:

$$\begin{cases} \forall\ p = 2k + 1, k \ge 0, T = 0: \\ y_{out}{}^{(i)} = \sigma(y_2{}^{(i)} - T) = \sigma(y_2{}^{(i)}) \in (0,1) \\ \forall\ p = 2k, k \ge 1, T > 0: \\ y_{out}{}^{(i)} = \sigma(y_2{}^{(i)} - T) \in [\frac{1}{1+e^T}, 1) \end{cases} \quad (8)$$

According to the equations (7) and (8), it can be proven that:

$$\begin{cases} \forall\, p = 2k+1, k \geq 0, T = 0: \\ \quad \begin{cases} y_{out}{}^{(i)} \geq 0.5 \rightarrow class^{(i)} = pos\,(1) \\ y_{out}{}^{(i)} < 0.5 \rightarrow class^{(i)} = neg\,(0) \end{cases} \\ \forall\, p = 2k, k \geq 1, T > 0: \\ \begin{cases} y_2{}^{(i)} \geq T \rightarrow class^{(i)} = pos\,(1) \\ 0 \leq y_2{}^{(i)} < T \rightarrow class^{(i)} = neg\,(0) \end{cases} \end{cases} \quad (9)$$

Figure 2 displays the proposed activation function of the proposed perceptron.

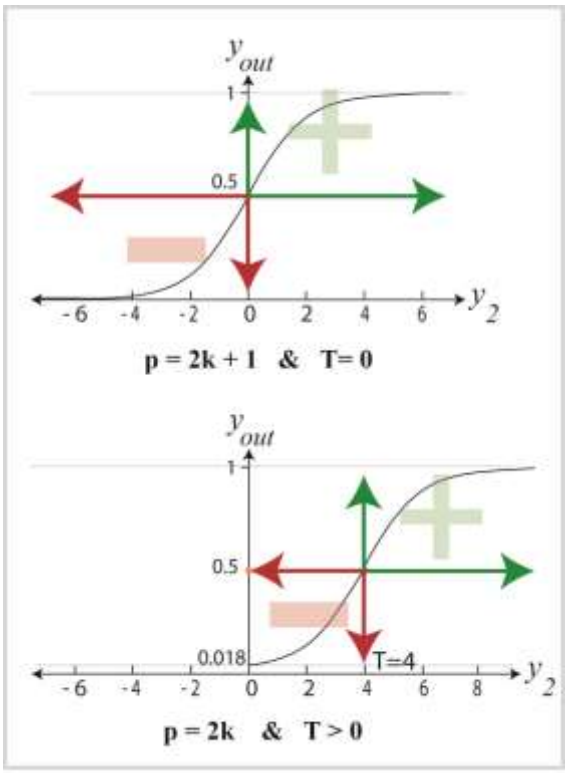

Figure 2. The proposed activation function and binary classification of data samples in the proposed perceptron.

As shown in figure 2, the activation function for $p = 2k+1$ is just like the one in the common perceptron, but for $p = 2k$ if $T = 4 > 0$, then the value of y_{out} will be 0.018 at least. Therefore, the area corresponding to each class will be similar to that shown in the figure 2-B.

As we know, in all neural networks with sigmoid as an activation function in the output layer, there are two types of bias: one bias in input and one fixed threshold (=0.5) in activation function. In this paper, we introduce the shifted activation function or the trainable threshold (T).

3.2. Loss function on the proposed perceptron for binary classification

Until now, various loss functions have been provided for neural networks, and there are also different optimizers to optimize these loss functions. The two primary and common optimizer types are Gradient Descent (GD) and Stochastic Gradient Descent (SGD). In this article, binary cross entropy was used as the loss function and the SGD optimizer. Now, to measure the value of loss function for S_i:

$$L_{SGD}{}^{(i)} = -Y_t{}^{(i)} \times \log(y_{out}{}^{(i)}) - (1 - Y_t{}^{(i)}) \times \log(1 - y_{out}{}^{(i)}) \quad (10)$$

Where, $L_{SGD}{}^{(i)}$ is representing the value of the loss function based on the SGD. $Y_t^{(i)}$ is the expected output for $S^{(i)}$ (equation (2)), which can be either *zero* or *one* and $y_{out}^{(i)}$ is also the output given by the proposed perceptron for $S^{(i)}$ (equation (6))), which is a decimal number within the range *(0, +1)*.

3.3. Backpropagation on the proposed perceptron for binary classification

Considering the optimizer SGD for binary cross entropy (equation (10)), the backpropagation procedure in the proposed perceptron for binary classification problems and weight update will be as follows:

$$W_{QV_j{}^{(i)}}(t+1) = W_{QV_j{}^{(i)}}(t) - \eta \times (y_{out}{}^{(i)} - Y_t{}^{(i)}) \times \left(\frac{W_{QV_j{}^{(i)}}(t)}{y_2{}^{(i)}}\right)^{p-1} \quad (11)$$

If p is odd, T will always be a fixed value of zero. The value of T will need to be trained and optimized during the model training for even values of p. To update the T value, the formula is as follows:

$$T(t+1) = T(t) - \eta \times (Y_t{}^{(i)} - y_{out}{}^{(i)}) \quad (12)$$

3.4. An example of applying the proposed model to a data sample

To learn more about how the proposed model works, we use an example. Consider the dataset in figure 3, which is related to the prediction of playing based on weather features. According to the "weather" dataset, we have $N = 14$ data samples, $n = 4$ categorical features and the problem is a binary classification with '*positive'* class for '*Play'* and '*negative'* class for '*Don't*

Play'. The weight and cardinality parameters in the proposed model will be as follows:

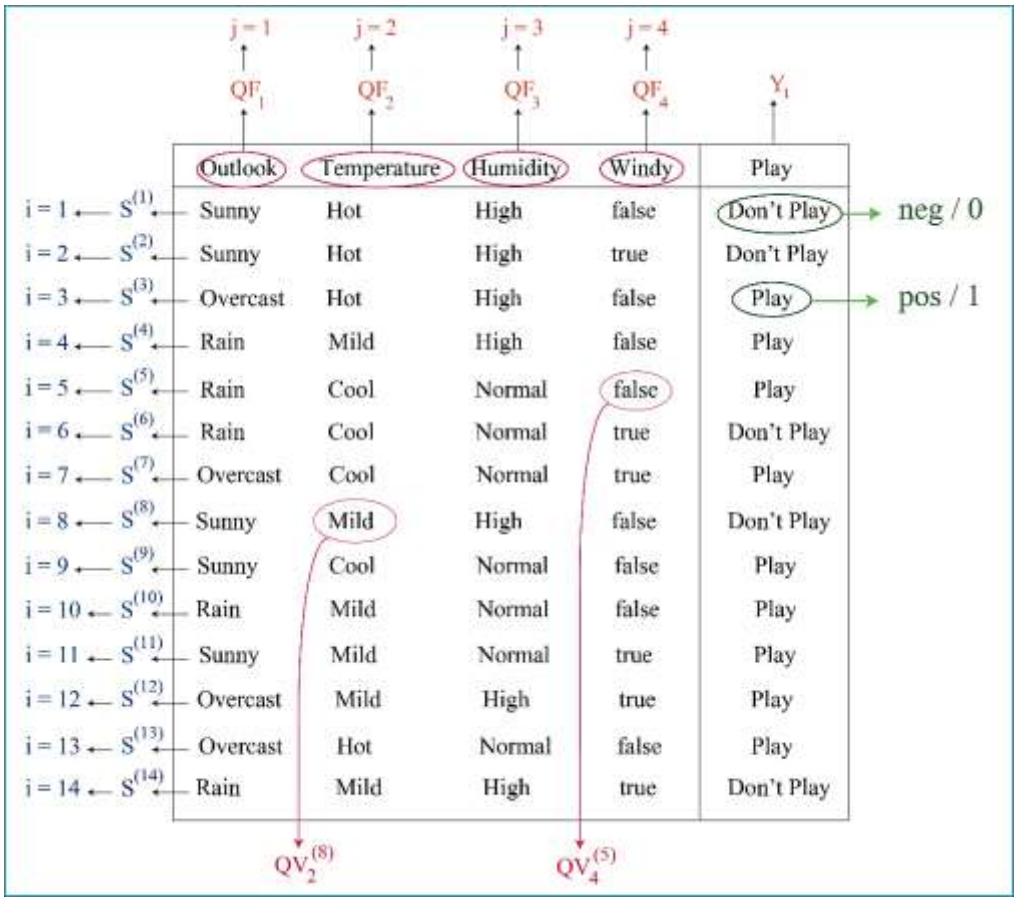

Figure 3. The weather dataset and the symbols of the proposed model.

$$
\begin{cases}
W_{QV_0^{(i)}} = w_0 \\
W_{QV_1^{(i)}} \in \{w_{qv_{1,1}}, w_{qv_{1,2}}, \ldots, w_{qv_{1,d_1}}\} = \{w_{sunny}, w_{overcast}, w_{rainy}\} \; and \; d_1 = 3 \\
W_{QV_2^{(i)}} \in \{w_{qv_{2,1}}, w_{qv_{2,2}}, \ldots, w_{qv_{2,d_2}}\} = \{w_{hot}, w_{mild}, w_{cold}\} \; and \; d_2 = 3 \\
W_{QV_3^{(i)}} \in \{w_{qv_{3,1}}, w_{qv_{3,2}}, \ldots, w_{qv_{3,d_3}}\} = \{w_{high}, w_{normal}\} \; and \; d_3 = 2 \\
W_{QV_4^{(i)}} \in \{w_{qv_{4,1}}, w_{qv_{4,2}}, \ldots, w_{qv_{4,d_4}}\} = \{w_{true}, w_{false}\} \; and \; d_4 = 2
\end{cases}
$$

As you can see, each categorical value in each feature will be assigned a unique weight. Updating the weights for the first sample from the above dataset will be done as follows:

$$\forall i = 1 : S^{(1)} = [\text{'sunny'}, \text{'hot'}, \text{'high'}, \text{'false'}]$$

$$Y_t^{(1)} = 0 \rightarrow$$

$$QV_1^{(1)} = sunny, QV_2^{(1)} = hot,$$

$$QV_3^{(1)} = high, QV_4^{(1)} = false.$$

$$y_1^{(1)} = w_0 + W_{sunny} \times sunny + W_{hot} \times hot + W_{high} \times high + W_{false} \times false$$

$$y_2^{(1)} = \sqrt[p]{(W_{sunny})^p + (W_{hot})^p + (W_{high})^p + (W_{false})^p + (w_0)^p}$$

$$y_{out}^{(1)} = \sigma(y_2^{(1)} - T), T \geq 0 \rightarrow$$

$$if \; y_{out}^{(1)} \geq 0.5 \rightarrow class^{(1)} = 1, else \; class^{(1)} = 0$$

$$L_{SGD}^{(1)} = -Y_t^{(1)} \times \log(y_{out}^{(1)}) - (1 - Y_t^{(1)}) \times \log(1 - y_{out}^{(1)})$$

$$\begin{cases} W_{sunny}(t+1) = W_{sunny}(t) - \eta \times (y_{out}^{(1)} - Y_t^{(1)}) \times (\frac{W_{sunny}(t)}{y_2^{(1)}})^{p-1} \\ W_{hot}(t+1) = W_{hot}(t) - \eta \times (y_{out}^{(1)} - Y_t^{(1)}) \times (\frac{W_{hot}(t)}{y_2^{(1)}})^{p-1} \\ W_{high}(t+1) = W_{high}(t) - \eta \times (y_{out}^{(1)} - Y_t^{(1)}) \times (\frac{W_{high}(t)}{y_2^{(1)}})^{p-1} \\ W_{false}(t+1) = W_{false}(t) - \eta \times (y_{out}^{(1)} - Y_t^{(1)}) \times (\frac{W_{false}(t)}{y_2^{(1)}})^{p-1} \\ W_0(t+1) = W_0(t) - \eta \times (y_{out}^{(1)} - Y_t^{(1)}) \times (\frac{W_0(t)}{y_2^{(1)}})^{p-1} \end{cases}$$

$$T(t+1) = T(t) - \eta \times (Y_t^{(1)} - y_{out}^{(1)})$$

As seen, for each data sample, only the weights of the categorical values in the sample were updated. And the above calculations are done for all samples in one epoch. The above example shows how the proposed model works.

4. Experiments

In this section, we will detail the experimental results obtained from the proposed model.

4.1. Experimental setup

The proposed perceptron was implemented in Spyder environment and carried out on a PC with a 2.5 GHz Intel Core i5 processor with 8 GB of memory. To evaluate the performance of the proposed perceptron, 10 categorical datasets with binary output were used. The UCI machine learning repository ([20]) has online availability of the datasets and their corresponding descriptions.

In datasets that contain multiple classes, considering the number of samples in each class and maintaining a balanced distribution of classes, the entire class has been reduced to two classes and every problem has been turned into a binary classification problem. When a dataset contains unknown values for some features, we consider a new categorical value for that feature, which is known as MV or 'Missing Value'. All numeric features in the datasets were considered categorical features. Table 1 provides an overview of the pre-processed datasets obtained after applying the above settings to all datasets.

4.2. Parameter optimization of proposed perceptron

The available data was randomly distributed in three subsets, with 55% being used for training, 15% for validation, and 30% for testing. The optimization process was executed using only training and validation. The parameters required to optimize the proposed perceptron are: 1) ±weight_range (WR) related to the initial weight of the connections, 2) the number of epochs, 3) learning_rate (LR) and 4) the initial value of the transmission parameter (T). The mean and standard deviation (SD) were calculated for all datasets. Table 2 displays the results.

Table 1. Benchmark datasets overview.

Datasets	Abbreviation	# of samples	# of inputs	# of Classes
Car Evaluation	CE	1728	6	*2BC
Breast Cancer	BCR	286	9	2BC
Chess (KR vs. KP)	CRP	3196	36	2BC
****Clave Direction**	CD	10800	16	2BC
****Connect-4**	C4	67557	42	2BC
Congressional Voting Recording	CVR	435	16	2BC
Dermatology	DRM	366	33	2BC
Hayes Roth	HR	160	33	2BC
HIV	HIV	6590	33	2BC
Lymphography	LYM	148	18	2BC

***BC: 'Binary Classification' problem**
****These datasets are not involved in the optimization procedure.**

Table 2. Mean and standard deviations obtained for the proposed perceptron.

Parameters		Optimal Values (Mean ± SD)
Learning-Rate (LR)	p = 1	0.09 ± ~~0.16~~ (0.01)
	p = 2	0.1 ± ~~0.16~~ (0.01)
	p = 3	0.02 ± ~~0.03~~ (0.01)
	p = 4	0.18 ± ~~0.24~~ (0.01)
Epochs	p = 1	167 ± ~~122~~ (0)
	p = 2	200 ± ~~100~~ (0)
	p = 3	256 ± ~~113~~ (0)
	p = 4	189 ± ~~89~~ (0)
Weight-Range (WR)	p = 1	0.6 ± ~~1.7~~ (0)
	p = 2	6.1 ± 2.2
	p = 3	5 ± 3.5
	p = 4	6.1 ± 2.2
T	p = 1	0
	p = 2	36.7 ± 33.5
	p = 3	0
	p = 4	31.1 ± 30.6

Considering that in reality, some parameters are unable to have a negative or decimal value, so to prevent negative numbers and decimals, the standard deviation values were replaced by numbers in parentheses during the implementation of the proposed model.
In the final training of the proposed perceptron, the parameters will be initialized based on the normal distribution using the optimal values obtained for them.

4.3. Complexity analysis of proposed perceptron

Our next task is to evaluate the memory required to keep up the mapping table and the proposed perceptron structure, as well as the time required to train the proposed model. To assess the complexity of time and memory in the proposed model more effectively, we have compared the complexity of it with that found in the single perceptron neural network that is based on one-hot encoding and the single perceptron neural network is built on integer encoding. Two metrics were used to measure the complexity of these three models: 1) The total amount of the lookup table's size (LUT) and the size of the network (SON), 2) The approximate time that model training will take per epoch.
During this phase, the Train and Validation sets, which make up 70% of the total data, carried out the model training process, and the testing of the models was done by the Test set. Table 3 displays the total amount of training and testing data, both individually and per class. The first and third columns are related to *'positive'* data, while the second and fourth columns are related to *'negative'* data. It is evident that the data for each class is distributed equally in both Train and Test sets.
The total amount of memory required for the *DS* dataset with *N* samples and *n* categorical features can be calculated by using the following formulas.

$$Space(\mathrm{mod}el) \approx Space_{\mathrm{mod}el}(LUT) + Space_{\mathrm{mod}el}(SON) \quad (13)$$

So, it can be concluded:

$$Space(OHP) \approx (1+\sum_{j=1}^{n} d_j) + \sum_{j=1}^{n} d_j \quad (14)$$

$$Space(INP) \approx (1+n) + \sum_{j=1}^{n} d_j \quad (15)$$

$$Space(\mathrm{Pr}oposed) \approx 1 + \sum_{j=1}^{n} d_j \quad (16)$$

Where, the Space is the approximate total memory required for various models, INP refers to integer encoding based perceptron, OHP refers to one-hot encoding based perceptron and Proposed refers to proposed perceptron in this paper. According to equals (14-16), we can draw a conclusion:

$$Space(\mathrm{Pr}oposed) \le Space(INP) \le Space(OHP) \quad (17)$$

The Space column in the table 3 displays practical evidence of equal 17. It is evident that across all datasets, the proposed perceptron demonstrates a lower memory requirement compared to the traditional OHP and INP models.
To facilitate a more comprehensive analysis of these findings, the percentage change in memory usage of the proposed model relative to the OHP and INP models is computed using the following formula, with the results illustrated in chart 1.

$$Percentage - Of - Space - \mathrm{Re}duction(A, \mathrm{Pr}oposed) = \frac{Space_A - Space_{\mathrm{Pr}oposed}}{Space_A} \times 100 \quad (18)$$

The findings indicate that the proposed model reduces memory consumption by an average of 49% compared to the OHP model and 23% compared to the INP model. This reduction allows for the application of the proposed model in environments with limited memory.
Furthermore, the correlation coefficient between the 'space' column of the proposed model and the number of samples is 0.2, while the correlation coefficient between the 'space' column of the proposed model and the number of features (inputs) is 0.8. This indicates that memory consumption tends to increase with the number of features, regardless of their cardinality. In contrast, it appears that memory consumption is largely independent of the number of samples present in the dataset. Consequently, the proposed model is particularly suitable for datasets containing a large number of data.
To determine the approximate time needed for training the model, we calculated in practical the model training time for each epoch. In the 'Training Time' column of the table 3, you can see the results of this metric in centisecond ($\times 10^{-2}\,\mathrm{sec}$). You can observe that the proposed model is more effective in managing a dataset with numerous samples. For datasets with a low number of samples, the INP models perform better than others.
In order to gain a clearer insight into the results achieved, the percentage change in training duration of the proposed model, in comparison to the OHP and INP models, is calculated using the following formula, with the results illustrated in chart 2.

Table 3. Table of data distribution in the Train and Test datasets, comparing the memory requirements of each model and the time required to train each model per epoch.

Datasets	Class Distribution in Train Data (70%)		Class Distribution in Test Data (30%)		Space (Memory Unit)			Training Time per Epoch ($*10^{-2}$ Sec)						
										Proposed Perceptron				
	Class: 0	Class: 1	Class: 0	Class: 1	OHP	INP	Proposed perceptron	OHP	INP	p=1	p=2	p=3	p=4	Avg
CE	856 (70.8 %)	353 (29.2 %)	354 (68.2 %)	165 (31.8 %)	42	28	**22**	6.58	**3.88**	4.17	4.56	4.69	4.8	4.56
BCR	137 (68.5%)	63 (31.5 %)	64 (74.4 %)	22 (25.6 %)	87	53	**44**	0.54	**0.31**	0.89	0.98	0.97	1.03	0.97
CRP	1061 (47.4 %)	1176 (52.6 %)	466 (48.5 %)	493 (51.5 %)	145	109	**73**	122	**24.2**	34.3	39.2	37.1	40.6	37.8
CD	4261 (56.3 %)	3299 (43.7 %)	1831 (56.5 %)	1409 (43.5 %)	65	49	**33**	683	147	**72.5**	**78.1**	**77.2**	**81.1**	**77.2**
C4	16114 (34 %)	31175 (66 %)	6970 (34.4 %)	13268 (65.6 %)	253	169	**127**	11.3×10^4	4.9×10^4	**1.6×10^4**	**1.6×10^4**	**1.7×10^4**	**1.6×10^4**	**1.6×10^4**
CVR	188 (61.8 %)	116 (38.2 %)	79 (60.3 %)	52 (39.7 %)	97	65	**49**	1.01	**0.60**	2.33	2.59	2.53	2.73	2.55
DRM	130 (50.7 %)	126 (49.3 %)	52 (47.2 %)	58 (52.8 %)	379	224	**190**	1.58	**0.7**	3.76	4.33	4.04	4.39	4.13
HR	36 (39.1 %)	56 (60.9 %)	15 (37.5 %)	25 (62.5 %)	295	153	**148**	0.28	**0.11**	0.32	0.36	0.35	0.37	0.35
HIV	3685 (79.8 %)	928 (20.2 %)	1545 (78.1 %)	432 (21.9 %)	321	169	**161**	1.3×10^3	34.0	**6.42**	**6.4**	**7.16**	**7**	**6.75**
LYM	44 (44.4 %)	55 (55.6 %)	17 (39.5 %)	26 (60.5 %)	119	78	**60**	0.23	**0.13**	0.75	0.84	0.81	0.87	0.82

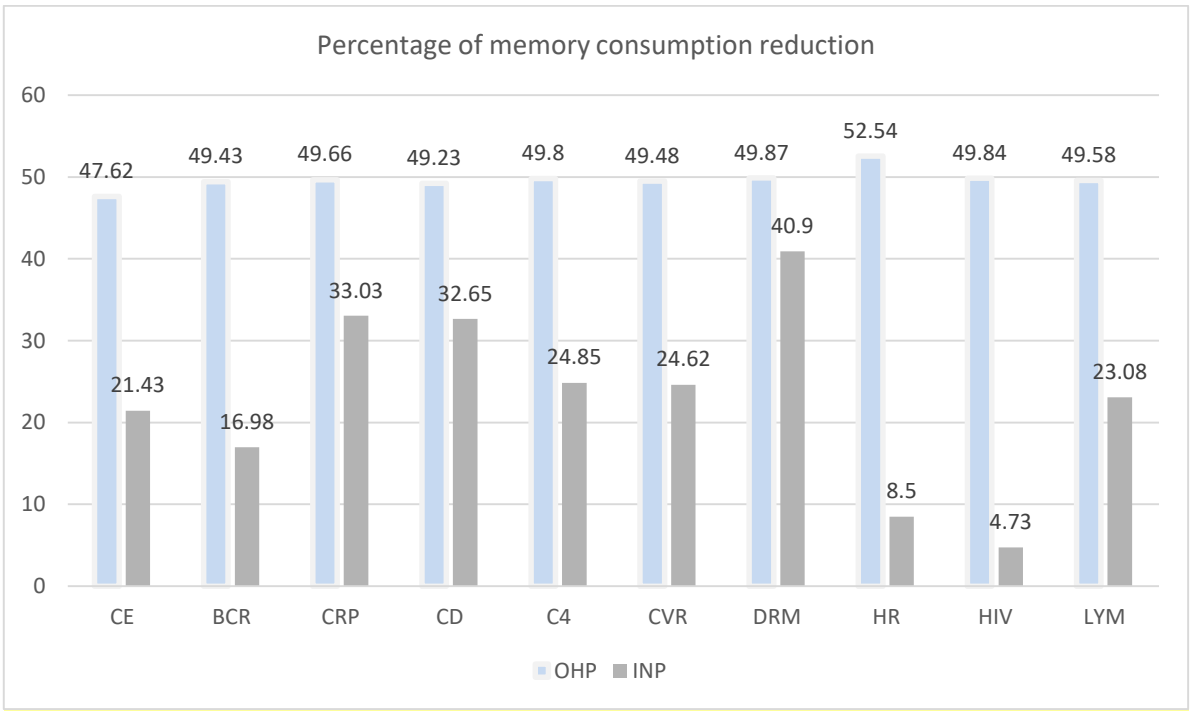

Chart 1. Percentage of memory consumption reduction in the proposed model compared to OHP model and INP model.

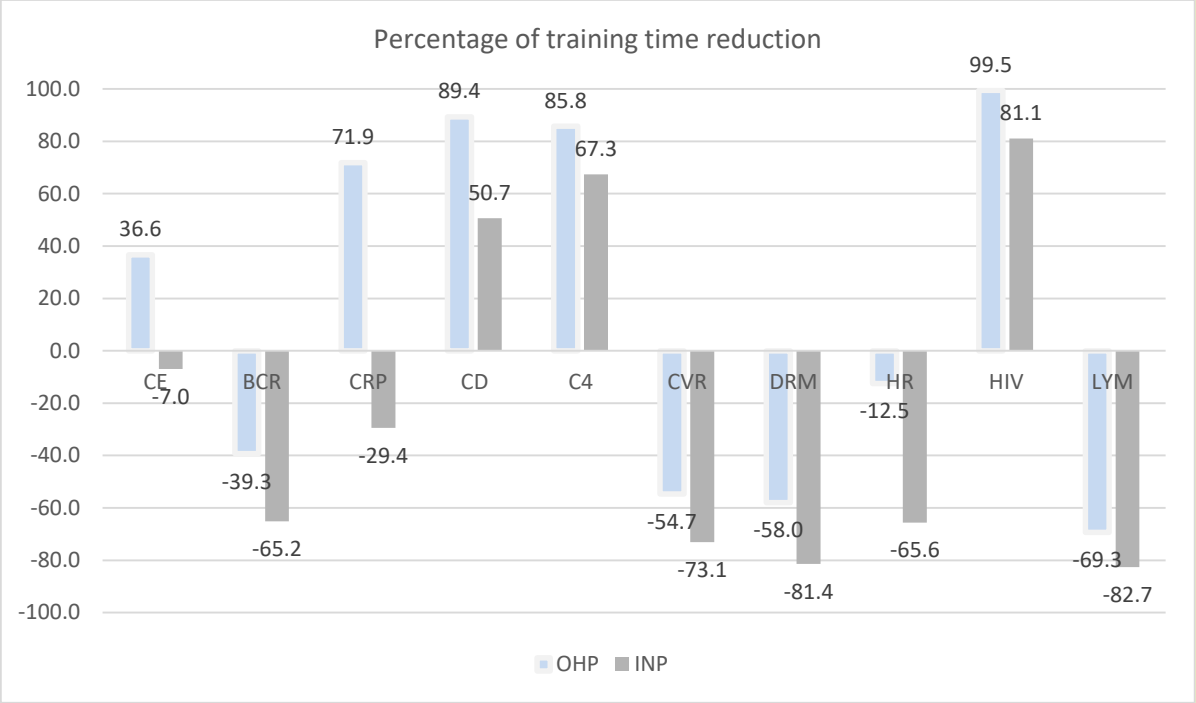

Chart 2. Percentage of training time reduction in the proposed model compare to models OHP and INP.

$$Percentage - Of - Time - \text{Re}duction(A, \text{Pr}oposed) = \frac{Time_{A} - Time_{\text{Pr}oposed}}{\max(Time_{A}, Time_{\text{Pr}oposed})} \times 100 \quad (19)$$

A negative percentage of change in relation to the OHP or INP model indicates that the OHP or INP model has successfully shortened the training time relative to the proposed model.

Chart 2 clearly demonstrates that for datasets with more than 1000 samples, the proposed model exhibits an average improvement in training time of 76.6% compared to the OHP model and 32.5% compared to the INP model. In addition, chart 2 illustrates that the training duration of the proposed model was significantly less than that of both comparison models when applied to the CD, C4, and HIV datasets, which contain the largest number of samples. Consequently, it can be inferred that the proposed model is more efficient in terms of training time for large datasets.

Moreover, the correlation coefficient for the 'Training Time' column of the proposed model in relation to the number of samples is 1.0, while the correlation coefficient with respect to the number of features (inputs) is measured at 0.5. This observation indicates a relatively strong direct relationship between training time and the number of features, as well as a perfect positive correlation with the number of samples.

In general, it can be said that the proposed model appears to be more effective than the other two models when dealing with big data due to its smaller space and faster training time.

4.4. Performance evaluation of proposed model

The proposed perceptron was compared with other methods including multilayer perceptron (MLPs) using one-hot encoding for all the categorical [21], naive Bayes (NB) classifiers [22], support vector machines (SVMs) with Gaussian kernel function [23], random forest trees (RFTs) [24], radial basis function network (RBF) [25] and categorical radial basis function network (CRBF) [9]. We employ both the accuracy (ACC) and the Matthews correlation coefficient (MCC) [26] in the analysis that follows. Table 4 display the comparison based on ACC.

Table 4 shows that the CRBF model performs the best, followed by the OHP and the proposed perceptron ($p = 2$) models, respectively. CRBF and the proposed perceptron, with p values of 1 and 2, perform best in terms of the number of the maximum accuracy in all dataset.

Considering the cosine similarity between accuracy vector of each model and the vector of goal. the CRBF and OHP models have been the first to have a similarity value of 0.996.

Table 4. Performance comparison using the accuracy (ACC) between the proposed model and the other models.

	Accuracy (%)												
	Model	Proposed perceptron				OHP	INP	CRBF [9]	MLP [21]	SVM [23]	NB [22]	RFTs [24]	ERBF [25]	Goal
		p = 1	p = 2	p = 3	p = 4									
Datasets	CE	94.99	94.41	95.37	90.75	93.44	71.86	**97**	96	95	87	96	91	100
	BCR	77.90	75.58	76.74	**80.23**	79	76.74	75	70	76	72	70	71	100
	CRP	96.24	95.3	93.43	70.69	95.41	94.68	**99**	98	98	86	98	81	100
	CD	86.60	**87.82**	66.57	81.85	87.65	87.5	82	84	83	78	81	81	100
	C4	76.46	77.9	68.69	74.54	79.2	66.2	**83**	80	76	73	80	69	100
	CVR	97.7	99.23	98.47	98.47	97.7	96.18	**100**	99	98	97	99	100	100
	DRM	**100**	97.27	98.18	94.54	95.45	**100**	97	95	95	95	96	96	100
	HR	75	80	80	65	75	55	**86**	80	79	82	82	76	100
	HIV	79.26	79.26	79.26	78.14	**93.01**	78.14	93	93	82	91	90	83	100
	LYM	**95.34**	93.02	86.04	83.72	86.04	86.04	82	80	78	78	77	68	100
Mean (±SD)		87.9 (±10)	88.0 (±9)	84.3 (±12)	83.6 (±10)	88.2 (±8)	81.2 (±14)	**89.4 (±9)**	87.5 (±10)	86 (±9)	83.9 (±9)	86.9 (±10)	81.6 (±11)	100
No. of Max		2	1	0	1	1	1	**5**	0	0	0	0	0	10
Cosine to Goal		0.994	0.995	0.991	0.994	**0.996**	0.986	**0.996**	0.994	0.995	0.995	0.994	0.992	+1

The proposed perceptron, with a value of 0.995, and others have the second performance rating.
In general, it can be concluded that models based on categorical data (CRBF, the proposed perceptron) perform better than models based on numeric data in terms of accuracy.

5. Conclusion

This article introduces a single-layer perceptron model that is based on categorical inputs. Instead of mapping categorical inputs to numeric space, the model directly uses them. This model implicitly encodes the input in the weight matrix.
According to the experiments conducted, it can be observed that the proposed perceptron performs better than the other models, in terms of memory consumption. In addition, the training time of a dataset with a large number of samples will be faster. Furthermore, the proposed perceptron model outperformed previous models in terms of accuracy.
The results present that in a specific case, the proposed perceptron performs similarly to the one-hot encoding perceptron, but with a lower memory usage and less training time. However, for datasets that contain low cardinality features, it is more effective to use mapping-based models such as OHP and INP.
Due to the observed low accuracy of the proposed model across certain datasets, it is recommended that the initial weighting process be conducted with intention, and the weights of the proposed model be adjusted using other optimization functions, such as Adam.
Considering the benefits of the proposed model, we suggest that neural network-based model such as MLP, RNN and LSTM be considered and developed in the future based on the proposed model. Additionally, the proposed model can be applied to other datasets that include qualitative features in a range of discipline, such as healthcare, architecture, industrial sectors, and educational contexts.

References

[1] P. Dongare, S. Kannan, R. Garg, and S. S. Harsoor, "Describing and displaying numerical and categorical data," *Airway*, vol. 2, no. 2, p. 64, 2019.

[2] A. Agresti, An introduction to categorical data analysis. Hoboken, New Jersey: John Wiley & Sons, Inc., 2018.

[3] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics and speech recognition" (draft), 2023.

[4] G. Alfonso Perez and R. Castillo, "Categorical variable mapping considerations in classification problems: Protein application," *Mathematics*, vol. 11, no. 2, p. 279, 2023.

[5] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *J. Big Data*, vol. 7, no. 1, 2020.

[6] A. R. Shabrandi, A. Rajabzadeh Ghatari, N. Tavakoli, M. Dehghan Nayeri, & S. Mirzaei. "Fast

COVID-19 Infection Prediction with In-House Data Using Machine Learning Classification Algorithms: A Case Study of Iran," *Journal of AI and Data Mining*, vol. 11, no. 4, pp. 573-585, 2023.

[7] J. Moeyersoms and D. Martens, "Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector," *Decis. Support Syst.*, vol. 72, pp. 72–81, 2015.

[8] D. Reilly, M. Taylor, P. Fergus, C. Chalmers, and S. Thompson, "The categorical data conundrum: Heuristics for classification problems—A case study on domestic fire injuries," *IEEE Access*, vol. 10, pp. 70113–70125, 2022.

[9] A. Alexandridis, E. Chondrodima, N. Giannopoulos, and H. Sarimveis, "A fast and efficient method for training categorical radial basis function networks," *IEEE Trans. Neural Netw*. Learn. Syst., vol. 28, no. 11, pp. 2831–2836, 2016.

[10] L. A. S. Cardona, H. D. Vargas-Cardona, P. Navarro González, D. A. Cardenas Peña, and Á. Á. Orozco Gutiérrez, "Classification of categorical data based on the Chi-square dissimilarity and t-SNE," *Computation (Basel)*, vol. 8, no. 4, p. 104, 2020.

[11] M. M. Arat, "Learning from high-cardinality categorical features in deep neural networks," *Journal of Advanced Research in Natural and Applied Sciences*, 2022.

[12] D. B. Suits, "Use of dummy variables in regression equations," J. Am. Stat. Assoc., vol. 52, no. 280, pp. 548–551, 1957.

[13] K. Potdar, T. S., and C. D., "A comparative study of categorical variable encoding techniques for neural network classifiers," *Int. J. Comput.* Appl., vol. 175, no. 4, pp. 7–9, 2017.

[14] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," *SIGKDD Explor*., vol. 3, no. 1, pp. 27–32, 2001.

[15] C. Seger, "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," *DEGREE PROJECT TECHNOLOGY*. Published online 2018.

[16] W. H. Riska, D. Permana, A. A. Putra, and Zilrahmi, "Categorical data clustering with K-Modes method on fire cases in DKI Jakarta Province," *ujsds*, vol. 2, no. 1, pp. 56–63, 2024.

[17] H. Cho and Y. Chung, "Clustering high-cardinality categorical data using category embedding methods," *J. Korean Data Inf. Sci. Soc*., vol. 31, no. 1, pp. 209–220, 2020.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," arXiv [cs.CL], 2013.

[19] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," *in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[20] A. Asuncion and D. Newman, UCI machine learning repository, 2007.

[21] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochim. Biophys. Acta*, vol. 405, no. 2, pp. 442–451, 1975.

[22] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw*., vol. 5, no. 6, pp. 989–993, 1994.

[23] R. Fernandes de Mello and M. Antonelli Ponti, "Statistical learning theory," in *Machine Learning, Cham: Springer International Publishing*, 2018, pp. 75–128.

[24] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," arXiv [cs.LG], 2013.

[25] H. Zhang, B. Quost, and M.-H. Masson, "Cautious weighted random forests," *Expert Syst*. Appl., vol. 213, no. 118883, p. 118883, 2023.

[26] K.-L. Du and M. N. S. Swamy, "Radial Basis Function Networks," in *Neural Networks and Statistical Learning, London: Springer London*, pp. 299–335, 2014.

ساختاری جدید برای پرسپترون جهت طبقه‌بندی داده‌های دسته‌ای

فریبا تقی‌نژاد و محمد قاسم‌زاده*

گروه کامپیوتر، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.

ارسال ۲۰۲۴/۰۸/۲۶؛ بازنگری ۲۰۲۴/۱۱/۰۵؛ پذیرش ۲۰۲۴/۱۱/۳۰

چکیده:

شبکه‌های عصبی مصنوعی یکی از مدل‌های مهم در یادگیری ماشین است که از مقادیر عددی به عنوان ورودی استفاده می‌کند. این مقاله یک مدل پرسپترون تک‌لایه جدید و مبتنی بر ورودی‌های دسته‌ای را ارائه می‌دهد. در مدل پیشنهادی، به هر مقدار دسته‌ای موجود در مجموعه داده آموزشی یک وزن قابل آموزش اختصاص می‌یابد. داده ورودی با استفاده از بردار وزن مربوط به مقادیر دسته‌ای موجود در آن طبقه‌بندی می شود. برای ارزیابی عملکرد مدل پیشنهادی از ۱۰ مجموعه داده استفاده کرده‌ایم و عملکرد آن را با دیگر مدل‌های یادگیری ماشین شامل شبکه‌های عصبی، ماشین بردار پشتیبان، طبقه‌بند بیز ساده و جنگل تصادفی مقایسه کرده‌ایم. با توجه به نتایج به دست آمده به ازای تمام مجموعه داده‌ها، میزان حافظه مصرفی در مدل پیشنهادی ۳۶٪ کمتر از مدل‌های مبنا بوده است. بعلاوه، سرعت آموزش برای مجموعه داده‌هایی دارای تعداد نمونه‌های بزرگتر از ۱۰۰۰ نمونه، ۵۴٫۵٪ بهبود داشته است. دقت مدل پیشنهادی نیز قابل مقایسه با دیگر مدل‌های یادگیری اشین است.

کلمات کلیدی: شبکه عصبی، داده کیفی، داده دسته‌ای، داده غیرعددی طبقه‌بندی دودویی.