Heaven's Light is Our Guide

RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

Computer Science & Engineering

Course No: CSE 4204

Sessional based on CSE 4203

Name of the Experiment

Design and implementation of single layer perceptron learning algorithm

Submitted by

Tushar Das

Roll: 1803108

Dept. of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Submitted to

Rizoan Toufiq

Assistant Professor

Dept. of Computer Science & Engineering

Rajshahi University of Engineering & Technology

1 The Perceptron Learning Algorithm

The single-layer is the first proposed neural model. The contents of the neuron's local memory consist of a vector of weights. The calculation of the single-layer, is done by multiplying the sum of the input vectors of each value by the corresponding elements of the weight vector. The value displayed in the output is the input of the activation function.A single layer perceptron (SLP) is a feed-forward network based on a threshold transfer function. SLP is the simplest type of artificial neural networks and can only classify linearly separable cases with a binary target (1, 0).

1. Initialise weights and threshold
Define $w_i(t), (0 \leq i \leq n)$, to be the weight from input i at time t, and θ to be the threshold value in the output node. Set w_0 to be $-\theta$, the bias, and x_0 to be always 1.
Set $w_i(0)$ to small random values, thus initialising all the weights and the threshold.
2. Present input and desired output
Present input $x_0, x_1, x_2, \ldots, x_n$ and desired output $d(t)$
3. Calculate actual output

$$y(t) = f_h\left[\sum_{i=0}^{n} w_i(t)x_i(t)\right]$$

4. Adapt weights—Widrow-Hoff delta rule

$$\begin{aligned} \Delta &= d(t) - y(t) \\ w_i(t+1) &= w_i(t) + \eta \Delta x_i(t) \\ d(t) &= \begin{cases} +1, & \text{if input from class A} \\ 0, & \text{if input from class B} \end{cases} \end{aligned}$$

where $0 \leq \eta \leq 1$, a positive gain function that controls the adaption rate

Figure 1: SLP Algorithm

2 The Abalone Dataset

The Abalone dataset [3] is a widely used dataset in machine learning and statistics. It contains measurements and other attributes of abalone, a type of marine mollusk. The dataset has the following characteristics:

- **Features:** The dataset includes 8 physical and geometrical measurements such as Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, and Shell Weight. Additionally, there is a Sex attribute representing the gender of the abalones.
- **Target Variable:** The target variable is 'Rings,' which is a numerical value indicating the age of the abalones. This variable is crucial for age prediction.

- **Size:** The dataset consists of a total of 4,177 instances.
- **Data Types:** The dataset contains a mixture of numerical and categorical data types, making it suitable for various data preprocessing techniques.

2.1 Exploratory Data Analysis & Preprocessing

- **Class Distribution & Reduction:** The 'Rings' attribute includes a wide range of classes, with ages from 1 to 29. The class distribution reveals that some age classes have significantly more instances than others. Classes [9, 10, 8, 11, 7, 12] are the most abundant, while the other classes have fewer examples. To simplify the classification task and facilitate the implementation of a binary perceptron model, the decision was made to reduce the classes to two. The two classes with the highest frequency were identified and retained, while the remaining classes were grouped into the 'Other' category.
- **Label Encoding:** The 'Sex' column in the dataset contained categorical values ('M', 'F', 'I') that needed to be converted into a numerical format for the perceptron model. Label encoding was applied to map these categorical values to numerical equivalents. Additionally, the values were scaled between 0 and 1 to normalize the data.

2.2 Training and Test Dataset Ratio

For model development and evaluation, it is common to split the dataset into training and test sets. The recommended ratio is typically 80/20 or 70/30. In this analysis, we have divided the dataset into a training set and a test set with an 80/20 split. This means that 80% of the data is used for training the SLP classifier, while the remaining 20% is reserved for testing and evaluating the classifier's performance.

3 Perceptron Model

The perceptron model was implemented with a tanh activation function and trained on the preprocessed dataset.

3.1 Perceptron Training Function

- The **perceptron_train** function takes input data, labels, and a learning rate 'alpha'.
- It initializes the weights randomly and stores the original weights.
- In the **perceptron_train** function, a threshold is implicitly used as part of the hyperbolic tangent (tanh) activation function.

  a = tanh (weighted sum of inputs)

- The threshold in this case is 0.9. If the activation is greater than or equal to 0.9, the predicted label (dt) is set to 1. Otherwise, if the activation is less than 0.9, the predicted label is set to 0. This threshold determines the point at which the perceptron "fires" and makes a positive prediction.

- The weight update rule in the Widrow Delta Rule is mathematically represented as follows:

  $weights = weights + \alpha \cdot (dt - Y[i]) \cdot X[i]$

- The difference between the final and original weights is printed, and the trained weights are returned.

3.2 Perceptron Testing Function

The **perceptron_test** function takes input data for testing, the shape of target labels, and the trained weights. It iterates through the testing set, calculates the activation using tanh, and predicts the label based on the activation. The predicted labels are returned.

3.3 Model Evaluation

The **score** function takes predicted labels and actual labels as input. It calculates the difference between them, counts the number of correct predictions, and computes the accuracy score. The accuracy score is then printed.

3.4 Training and Testing the Perceptron Model

Choosing an optimal learning rate is crucial for the success of the perceptron model. A learning rate that is too high may lead to overshooting, causing the model to fail to converge, while a learning rate that is too low may result in slow convergence or getting stuck in local minima.

Figure 2: Learning Rate vs Accuracy

After experimenting with different learning rates, 0.00009 was found to strike a balance, providing a reasonable convergence rate without sacrificing stability. This value was determined through iterative testing and observing the model's performance on the testing set.

I used a learning rate of 0.00009 to train and test the perceptron model. The accuracy, or how well the model predicted the outcomes, was around 60% on the test data. This means the model got 60% of the predictions right.

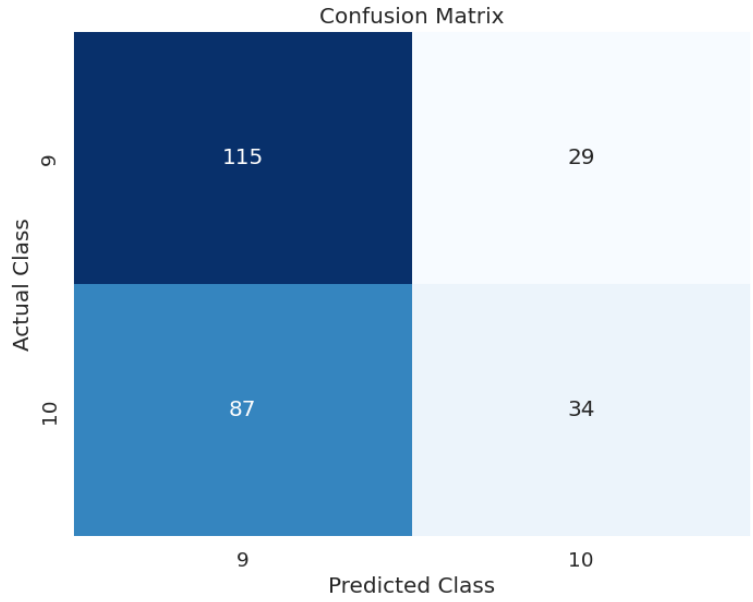

Figure 3: Confusion Matrix

4 Classifier Limitations

- It might not perform well when there are intricate relationships between the features, and the model might not generalize effectively to new, unseen data.
- The perceptron model used in the analysis is a relatively simple algorithm. Its single-layer structure might not capture the intricate relationships within the data, especially if the age prediction is influenced by non-linear interactions among features.
- Choosing the value of random weights, and learning rate was a complex task and the accuracy is highly dependent on these values.
- There might be a loss of valuable information because some classes were removed. The dataset is not suitable for this classifier model and the model is a simple model of one-layer perception. So, the accuracy of around 60% is quite a good one.

References

[1] Complete Guide to Single Layer Perceptron with Implementation "https://shorturl.at/iDKPS"

[2] Source Code "https://shorturl.at/aOUX0"

[3] Abalone Dataset "https://rb.gy/2ov6cr"

[4] Neural Computing: An Introduction - R Beale and T Jackson