***Vishwakarma Institute of Technology, Pune Artificial Neural Networks Lab***

***Name: Tanishka SinghPRN- 12311695 Rollno. 44Class- TY CSAI- B***

# **Experiment Number: 02**

**Title:** Write a program to use a Perceptron neural network to recognize even and odd numbers represented in ASCII form, illustrating the basic principles of a single-layer neural network.

|  |  |  |  |
| --- | --- | --- | --- |
| **Title of Experimentation** | **CO**  **Mapping** | **CO-Statements** | **PO**  **Mapping** |
| Write a program to use a Perceptron neural network to recognize even and odd numbers represented in ASCII form, illustrating the basic principles of a single-layer neural network | CO1  CO2 | Implement a single-layer perceptron to solve a simple classification task and processed in binary/ASCII form | PO2  PO3  PO5 |

## **Objective**

To implement a perceptron neural network that classifies numbers (0–9) into **even** and **odd** classes using their ASCII representations. This demonstrates the working of a single-layer neural network.

## **Theory**

### **Perceptron Model**

* Proposed by **Frank Rosenblatt (1958)**.
* A single-layer perceptron consists of:
  1. **Input Layer** – receives input features.
  2. **Weights** – assigned to inputs.
  3. **Summation Function** – computes weighted sum.
  4. **Activation Function** – step function for binary classification.

Mathematically:

y=f(∑i=1nwixi+b)y = f\left(\sum\_{i=1}^{n} w\_i x\_i + b \right)y=f(i=1∑n​wi​xi​+b)

Where,

* xix\_ixi​: Input features
* wiw\_iwi​: Weights
* bbb: Bias
* fff: Activation function (step function here)

### **ASCII Representation**

* Each digit (‘0’ to ‘9’) has an ASCII code.
  + Example: '0' → 48, '1' → 49, ... '9' → 57
* Input to perceptron = ASCII value of digit.
* Output = **0 (even)** or **1 (odd)**.

### **Activation Function (Step)**

f(x)={1if x≥00if x<0f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}f(x)={10​if x≥0if x<0​

## **Algorithm**

1. Take input digits (0–9) in ASCII form.
2. Define training dataset with labels:
   * Even → 0
   * Odd → 1
3. Initialize perceptron weights and bias randomly.
4. For each training sample:
   * Compute weighted sum.
   * Apply step activation function.
   * Update weights using learning rule:

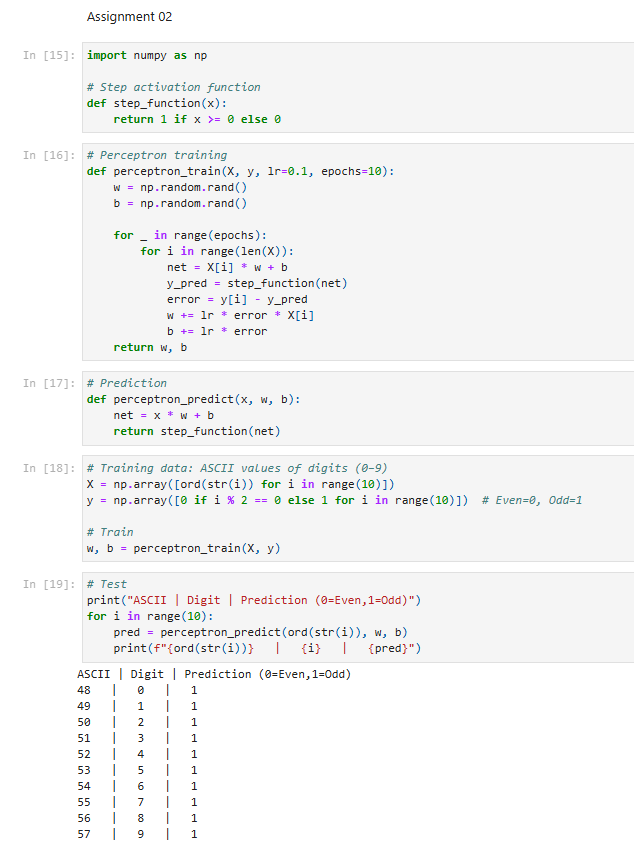
wnew=wold+η(d−y)xw\_{new} = w\_{old} + \eta (d - y) xwnew​=wold​+η(d−y)x bnew=bold+η(d−y)b\_{new} = b\_{old} + \eta (d - y)bnew​=bold​+η(d−y)

Where,

* η\etaη = learning rate
* ddd = desired output
* yyy = predicted output

1. Train until error becomes 0 (convergence).
2. Test with unseen digits and classify as Even/Odd.

## **Program (Python Implementation)**



## **Conclusion**

* A single-layer perceptron successfully learned to classify even and odd numbers based on their ASCII codes.
* This experiment demonstrates the **basic principles of perceptron learning**: weighted sum, activation, and weight updates.
* Although perceptron is limited to linearly separable problems, this simple case illustrates its working effectively.