Introduction to IoT and Its Industrial Applications (CS698T)
Indian Institute of Technology Kanpur
Assignment 2

QUESTION

1

Member Names: Kajal Sethi, Utkarsh Srivastava
Member Emails: kajals21@iitk.ac.in, utkarshs21@iitk.ac.in
Member Roll Numbers: 21111033, 21111063
Date: November 3, 2021

---

1 DESCRIPTION

The goal is to create an intelligent irrigation system that uses servo motors to manage water flow based on a Machine Learning algorithm's forecast. To anticipate the water flow, this algorithm uses the humidity and temperature data produced by the sensor associated with each individual servo motor sensor. In addition, the water flow is only controlled during the day. During night time, system is switched off.

2 COMPONENTS

This project is simulated on Arduino Mega board for processing. There is a single Multi Layer Perceptron (MLP) Model deployed, which has been trained on a dataset a priori to the simulation[6]. For sensing the temperature and humidity of the surrounding, DHT22 sensor is used[4]. Water flow to the field is controlled by Servo motors. The Photoresistor(LDR) sensor detects time of day as either Day or Night[1] . And the LCD 16x2 (I2C) module is used to display the percentage of water being supplied by each unit.

3 SENSORS USED

We have used four sensors in this assignment, that are stated below:

3.1 DHT22 Sensor

The DHT22 sensor is a basic temperature and humidity sensor with a digital display. It measures the ambient air with a capacitive humidity sensor and a thermistor and outputs a digital signal on the data pin[4]. It has a VCC pin (5V positive), an SDA pin (signal line to a digital pin), an NC pin (not connected), and a GND pin (ground negative line)[3] .

3.2 Servo Motor

A servomotor is a rotary or linear actuator that can control angular or linear position, velocity, and acceleration with precision. It is made comprised of an appropriate motor and a position feedback sensor. There are three pins: a ground pin, a VCC pin, and a PWM pin (signal pin). We can turn its arm 180 degrees in either a looping or a single motion[2] .

3.3 Photoresistor(LDR) Sensor

A photoresistor or a light-dependent resistor is a passive component that reduces resistance when brightness (light) is received on its sensitive surface[1]. The resistance of a photoresistor lowers as the incident light intensity increases; in other words, it is photoconductive. VCC, GND, DO, and AO are the four pins of the sensor. Depending on which pin among DO and AO is utilised for recording the results, it can produce both digital and analog values.

3.4 LCD 16x2 (I2C) display

We have used LCD 16x2 (I2C) display to show the amount of water that needs to be supplied by each unit of the irrigation system. This LCD display screen has a 16x2 resolution and an I2C interface. It can show 16x2 characters, or 16 characters on two lines (including white characters). The display has 4 pins namely VCC(5V positive), GND(ground), SDA(serial data pin), and SCL(serial clock pin)[1].

4 WORKING OF THE IRRIGATION SYSTEM

4.1 Expected

We are supposed to build four independent units of irrigation system such that each unit senses temperature and humidity values. Thereafter, send these readings to ML model deployed on the Arduino Mega board for water flow prediction. This prediction will be given as input to the servo motor. And display the % of water flow that has to be fed as input to servo motors on the LCD display.

4.2 Input

The sensors provide input to the system. The DHT22 sensor provides the temperature of the surrounding in degrees Celsius unit and provides the humidity readings in the range of 0 to 100. There are four DHT22 sensors deployed in the assignment.

4.3 Process

The Arduino Mega board processes[5] the data sent by photoresistor sensor(mega:AO) and anticipates Day/Night by the help of LDR Sensor. If it is 'DAY'(i.e. value of luminous is above 50 units on LDR sensor shows "DAY" time as stated in the question) time , then the DHT22 sensor(mega:4,5,6,7) values are sent to the Arduino mega board for processing. Firstly, the temperature and humidity values received are scaled by Min Max Scaling to account for different scales of reading of temperature and humidity sensors.
For example: t = 50, min = 20, max = 100. Then $new\_t = \frac{(t-min)}{(max-min)}$
This data is now combined with the weights that were learned by our ML model and the prediction for the quantity of water that is needed to be supplied is made. This process is performed for each of the four units. If the photoresistor sensor(mega:AO) data senses the time to be 'NIGHT' time (i.e. value below 50 units on LDR sensor shows time as "NIGHT" as stated in the question), then no operation is performed and the system is switched off. The predicted water quantity(in percentage) is then converted into degree using the following formula:

$$Servo\ motor\ angle = \frac{water\ percentage * 180}{100}$$

Now this angle is provided to the respective servo motor(mega:8,9,10,11).

4.4 Output

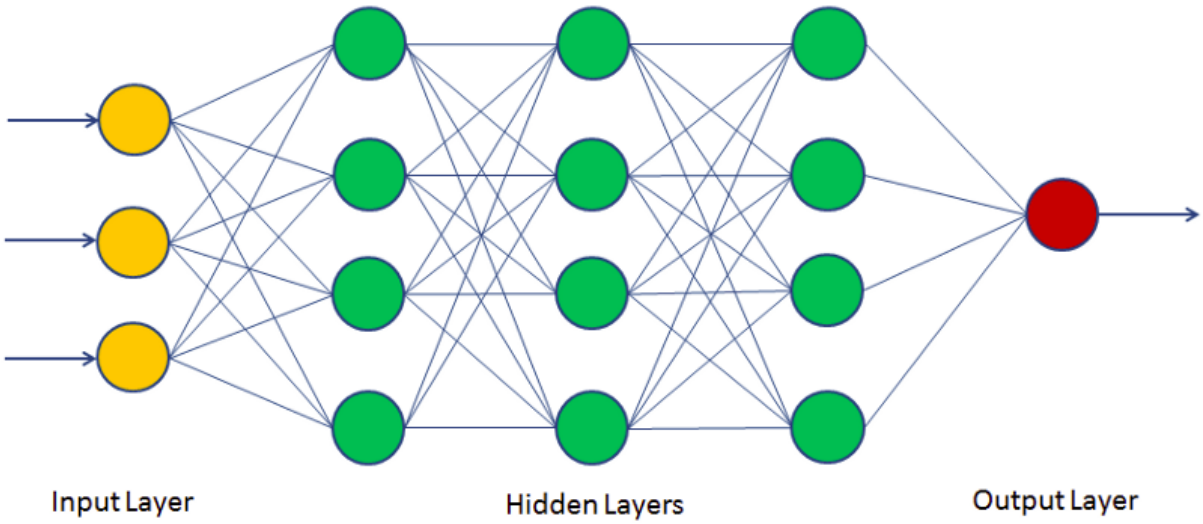
Depending on the angle calculated, the write command is given to the servo motor(mega:8,9,10,11), thus opening the water flow in the farm. This is done for all four units and the percentage of water supplied is displayed on the LCD display(mega:SDA-20,SCL-21).

5 THE ML MODEL

5.1 Multi Layer Perceptron: Architecture

The simplest sort of artificial neural network is the multi-layer perceptron (MLP). It's a hybrid of several perceptron models. Neurons in perceptron are inspired by the human brain and attempt to address problems by simulating its functionality. These neurons in MLP are extensively linked and parallel. This parallelization aids in the speeding up of computations. Frank Rosenblatt introduced Perceptron in 1950. It, like the human brain, is capable of learning complex things. Sensory Unit (Input Unit), Associator Unit (Hidden Unit), and Response Unit (Output Unit) make up a perceptron network as shown in Figure 1[6].

Figure 1: Layers in MLP



The Perceptron is made up of two fully connected layers: an input layer and an output layer. MLPs have the identical input and output layers, but as shown in Figure 2[6], they can have numerous hidden layers in between them.

Figure 2: Gradient Decent in MLP

If(y is not equal to t)
$W_i(new)=w_i(old) + \alpha t x_i$;
b(new)=b(old) + αt;
else
$w_i(new)=w_i(old)$;
b(new)=b(old);

Multi-Layer Perceptron trains model in an iterative manner as shown in Figure 2. In each iteration, partial derivatives of the loss function are used to update the parameters. We can also use regularization of the loss function to prevent overfitting in the model.

5.2 Parameters of ML Model

- **Epochs:** We have used 8000 epochs in the form of argument max_iter = 8000 as it gives optimal accuracy.
- **Layers:** We have used 1 input layer + 2 hidden layers + 1 output layer as mentioned in the question.
- **Hidden Layer Sizes:** The hidden layer sizes denotes the number of neurons present in each hidden layer. Here the sizes that we have used are 16,8 in the form of function argument hidden_layer_sizes =(16,8).
- **Random State:** random state is provided so that the output of an experiment can be reproduced anytime for validation purposes. Providing the same random number will yield the same output. It is provided in the form of argument random_state = 56.
- **Activation Function:** The activation function links the weighted sums of units in a layer to the values of units in the succeeding layer. It is provided in the form of argument activation='relu'.

  Mathematically, f(x) = max (0, x).

6 PSEUDO CODE

6.1 Libraries Required

The following libraries are needed to run the Python code.

- Python 3.8
- Numpy
- Pandas
- Sklearn
- Matplotlib
- Warnings

The following libraries are needed to run the Arduino code.

- DHT
- Servo
- LiquidCrystal_I2C
- stdio

6.2 Variables

The following variables are of concern in this assignment.

- DHT dht[ ] = {{pin_number, DHT22}, . . . , }; . . . . creating array of object of DHT22 sensor
- float humidity[4];
- float temperature[4];
- int water_percent[4];
- Servo servo1, servo2, servo3, servo4; . . . . create 4 objects of servo motor
- double layer1_bias[ ];
- double layer2_bias[ ];
- double output_layer_bias[ ];
- double layer1_weights[LAYER1][DIMENSION];
- layer2_weights[LAYER2][LAYER1];
- output_layer_weights[OUT_LAYER];
- double weights_output_layer[OUT_LAYER];
- double out_layer_1[LAYER1];
- double out_layer_2[LAYER2];

6.3 Extra Functions

We have defined our code in a modular fashion to ease readability. Here we give a brief idea about different functions present in our Simulation Code.

- **read_temp_humid():** Reads in sensor data
- **activation_function():** return max(0,x) as 'ReLu' activation function
- **predict():** takes in sensor values and returns the water-flow required as predicted by ML model
- **water_level_in_all_motors():** takes in sensors values as arrays and computes the water level required in all servo motors
- **convert_water_percentage_to_angle():** takes in water percentages and computes required angle of opening in servo motor for corresponding sensor values.

6.4 Setup

This function is the initialization function which is used to setup the Arduino board and sensors and define the required connections and their locations respectively. The tasks performed during setup function are listed below:

- Initialize lcd display
- Start temperature and humidity sensors
- Attach servo motors

6.5 Loop

read values of LDR sensor
if(day)
read_temp_humid();
predict_water(temperature, humidity);
lcd.print(percentage for all four units);

else
servo_object_names.write(0);
lcd.print(0 for all four units);

7 RESULTS

We have recorded the accuracy of our machine learning model using two different methods explained below. The accuracy of the model was recorded for different values of random_state, and selected 56 to maintain consistency and reproducibility of our observations.

- R^2 **goodness-of-fit** : We have used R^2 score to determine the accuracy of the predictions of the model. The best possible score that can be obtained is 1.0 and it can be negative as well (when the model is arbitrarily worse). And in case of a constant model which always predicts the same value of y, disregarding the input features, would get a R^2 score of 0.0[6].

  The R^2 score is defined as,

  $$R^2 = 1 - \frac{u}{v}$$

  where u is the residual sum of squares and v is the total sum of squares given by,

  $$u = \sum ((y\_true - y\_predicted)^2)$$

  $$v = \sum ((y\_true - y\_true.mean())^2)$$

  Command used was:-

  $$r2\_score(y\_pred, y\_test)$$

- **RMS Error** : The mean_squared_error function computes mean square error, a risk metric corresponding to the expected value of the squared (quadratic) error or loss.

  RMSE is defined as,

  $$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (y\_true - y\_predicted)^2}{N}}$$

  Command used was:-

  $$mean\_squared\_error(y\_test, y\_pred, squared = False)$$

Result Table

Method used	Recorded score
R^2	0.8355453119132263
$RMSE$	12.777447968938661

Table 1: Recorded Result

8 How To Run

To run this project, please go through the following steps:

1. Download and unzip 21111033_21111063.zip in your local machine.
2. Open Jupyter Notebook and open code.ipynb in it.
3. Goto Menu Bar and click on Cell and then Run All
4. Go to link https://www.wokwi.com and register your account.
5. Click on Start A New Project: Arduino Mega
6. Extract sketch.zip in your local system and paste the sktech.ino and diagram.json file contents in the simulator window in sktech.ino and diagram.json section of simulator respectively.
7. Now the simulator is ready which you can see on your screen.
8. Click on Start the simulation (Play) button.
9. Click on any DHT sensor or LDR sensor to change values and record observations as and when needed.
10. Thank You.

9 Link to the Project Simulation

https://wokwi.com/arduino/projects/314230949315347008

References

[1] https://docs.wokwi.com/parts/wokwi-photoresistor-sensor.

[2] https://www.youtube.com/watch?v=X83d8RRx9yo.

[3] https://www.youtube.com/watch?v=OzK7z8IkEnw.

[4] https://www.youtube.com/watch?v=PVvCt3AT2Jk.

[5] https://docs.wokwi.com/?utm_source=wokwi.

[6] https://machinelearninggeek.com/multi-layer-perceptron-neural-network-using-python/.