ARDUINO 101

Session 5

www.arduino101.ml

www.github.com/Hardware-Focus-Group

Machine Learning

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

In general, any machine learning problem can be assigned to one of two broad classifications into Supervised learning and Unsupervised learning.

Supervised Learning

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorised into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

(a) **Regression** - Given a picture of a person, we have to predict their age on the basis of the given picture

(b) **Classification** - Given a patient with a tumour, we have to predict whether the tumour is malignant or benign.

Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

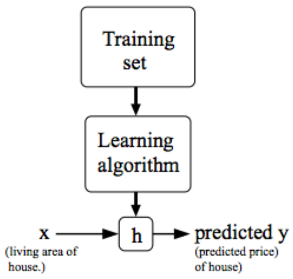
Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment.

Model Representation

To establish notation for future use, we'll use x(i)x(i) to denote the "input" variables (living area in this example), also called input features, and y(i)y(i) to denote the "output" or target variable that we are trying to predict (price). A pair (x(i),y(i))(x(i),y(i)) is called a training example, and the dataset that we'll be using to learn—a list of m training examples (x(i),y(i));i=1,...,m—is called a training set. Note that the superscript "(i)" in the notation is simply an index into the training set, and has nothing to do with exponentiation. We will also use X to denote the space of input values, and Y to denote the space of output values. In this example, X = Y = ℝ.

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a "good" predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like this:



When the target variable that we're trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

Project - Weather Station

STEP 1 - Components

- Arduino Uno
- LDR
- DHT11 sensor
- Breadboard
- Resistor (1K)
- Jumpers

STEP 2 - Environment Setup

- Arduino IDE
  https://www.arduino.cc/en/Main/Software

- Python 3.7.x
  https://www.python.org/downloads/

- PyFirmata Library
  https://pypi.org/project/pyFirmata/

- Numpy Library
  https://pypi.org/project/numpy/

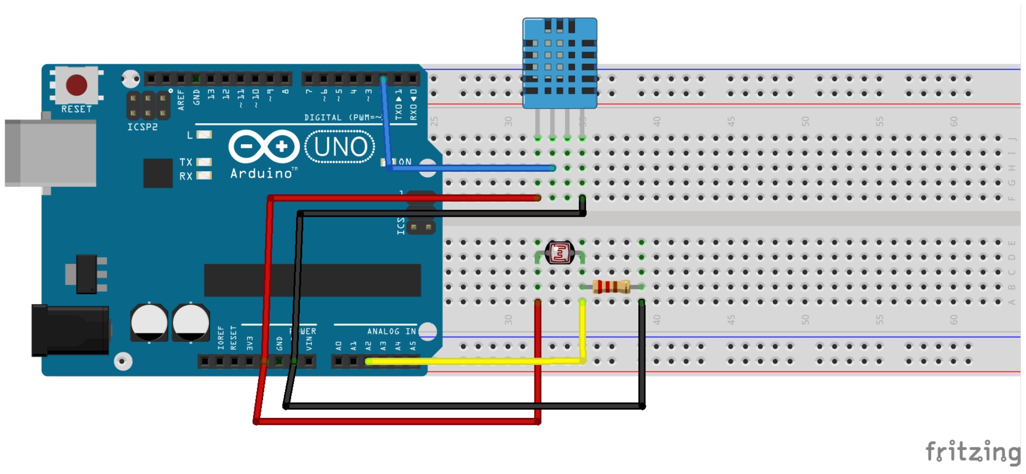
- Sklearn Library
  https://pypi.org/project/sklearn/

- Pandas Library
  https://pypi.org/project/pandas2/

- DHT Library
  https://github.com/adafruit/DHT-sensor-library

STEP 3 - Circuit

- Analog Pin **A2** -> **LDR**
- Digital Pin **2** -> **DHT11**



STEP 4 - Sketch #1 Using ANN

This is an artificial neural network developed for an Arduino Uno microcontroller board. The network described here is a feed-forward back-propagation network, which is perhaps the most common type. It is considered a good, general purpose network for either supervised or unsupervised learning. The code for the project is provided as an Arduino sketch. It is plug and play - you can upload it to an Uno and run it, and there is a section of configuration information that can be used to quickly build and train a customised network.

To modify the network to a new training set you must enter the appropriate truth table values in the Input and Target arrays, and you must also adjust the corresponding parameters in the configuration section to match the new truth table:

- PatternCount: The number of training items or rows in the truth table.
- InputNodes: The number of input neurones.
- OutputNodes: The number of output neurones.

There are several additional items that can be configured in this section for experimentation.

- HiddenNodes: The number of hidden neurones.
- LearningRate: Adjusts how much of the error is actually back-propagated.
- Momentum: Adjusts how much the results of the previous iteration affect the current iteration.
- InitialWeightMax: Sets the maximum starting value for weights.
- Success: The threshold for error at which the network will be said to have solved the training set.

Although the code is not at the absolute beginner level, if you have familiarity with the concepts of arrays and looping, you should hopefully be able to read through the sketch which accompanies this article and follow the logic. Here is a high level breakdown:

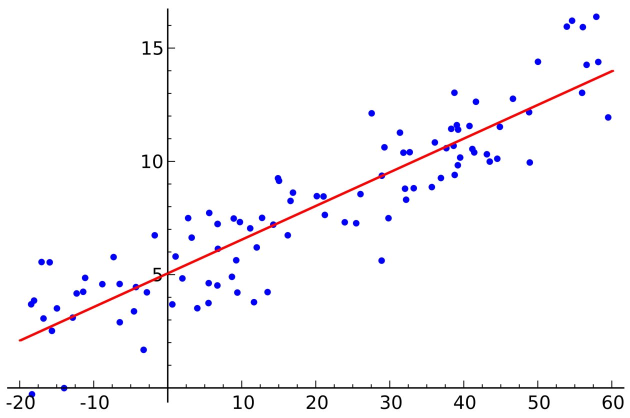
- Initialise the arrays. The weights are set to random numbers and two additional arrays that hold change values used in back-propagation are set to zeros.
- Begin a large loop that runs the system through a complete set of the training data.
- Randomise the order in which the training sets run on each iteration to reduce oscillation or convergence on local minimums.
- Calculate the hidden layer activations, output layer activations and errors.
- Back-propagate the errors to the hidden layer.
- Update the weights.
- If the system error is greater than the success threshold then run another iteration of the training data.
- If the system error is less than the success threshold than break, declare success, and send data to the serial terminal.
- Every 1000 cycles send the results of a test run off the training set to the serial terminal.

Download sketch by clicking here **ANN.ino**

STEP 5 - Sketch #2 Using LR

Firmata is a protocol for communicating with microcontrollers from software on a computer (or smartphone/tablet, etc). The protocol can be implemented in firmware on any microcontroller architecture as well as software on any computer software package. For up-to-date information on the Firmata protocol and various host implementations please see the following website **http://firmata.org** .

Linear regression is implemented using Scikit-Learn, which is one of the most popular machine learning libraries for Python. The term “linearity” in algebra refers to a linear relationship between two or more variables. If we draw this relationship in a two-dimensional space (between two variables), we get a straight line. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression. If we plot the independent variable (x) on the x-axis and dependent variable (y) on the y-axis, linear regression gives us a straight line that best fits the data points, as shown in the figure below.



Download microcontrollers sketch by clicking here **LR.ino**

Download host/computer sketch by clicking here **LR.py**

- Upload the LR.ino sketch on Arduino.
- Make circuit connections as shown in circuit diagram.
- Run LR.py sketch on host machine.