Namal Institute Mianwali
Computer Science Department
Artificial Intelligence

PROJECT: 02

Document:	Training Multi Level Neurons	Date:	29-Nov-2022
Prepared by:	Attiq Ullah Shah (BSC-2020-21) Abubakar Bhatti (BSCS-2020-41)	Prepared for:	Dr. Junaid Akhter

Contents

Problem Statement: 3
Introduction: 3
Overview of Project: 3
Context: 3
Natural Phenomena: 3
Classification Theory: 3
What is a Neural Network? 4
Computational Model: 5
Characteristics of Neural Network: 6
PROJECT IMPLEMENTATION: 7
LANGUAGE USED: 7
LIBRARIES: 7
SKLEARN.NEURAL NETWORK MLP CLASSIFIER: 7
HIDDEN-LAYERS: 7
ACTIVATION FUNCTION: 7
SOLVER: 7
LEARNING_RATE_INIT: 7
RANDOM_STATE: 7
VERBOSE: 8
MAX_ITER: 8
N_ITER_NO_CHANGE: 8
CODE EXPLANATION: 8
FEATURES: 8
COUNT OF BLACK PIXELS: 8
EDGE-DETECTION: 9
DIAGONAL: 9
CENTROID: 9
HOG: 10
EXPERIMENTATION: 10
HYPOTHESIS 1: 10
EXPERIMENTS: 10
CONCLUSION: 11
HYPOTHESIS 2: 11
EXPERMINET: 11
CONCLUSION: 12
CONCLUSION: 12

Problem Statement:

- Use sklearn library on a hand-drawn roman numerals (I-X), and train a multi-class neural network for maximum training and validation accuracy.

Introduction:

- Do you know what differentiates between humans and other organisms? "**Brain."**
- The most critical, complex, and powerful organ of our body, but have you ever thought about that what makes it so complex and powerful? "**It's Neurons."**
- A neuron is a single unit that helps us to take actions intentionally or unintentionally.
- Our brains consist of billions of neurons that combine and work together to decide and classify what type of action we should take.
- For instance, if we feel pain in any part of our body, the message that the body is feeling pain is delivered to neurons. The neuron decides that there is a pain in some particular body part. This is the same case that the neurons help classify a decision when we see a cat or a dog, and we used to say that this is the cat and the other is the dog.
- But the question that arises here is, can machines do the same? The answer to the question is **"Yes."** This can be done with the help of Artificial Intelligence
- Artificial intelligence (AI) is a branch of computer science that helps make machines able to learn. AI takes these problems as **classification problems** and provides a solution that differentiates between classes.

Overview of Project:

Context:

- The report contains the solution to the problem in which we will train a machine to find similar results with its previous data, provide a final result after a maximum number of iterations, and classify the given scenario to make a better decision. This scenario is related to the classification algorithm of Artificial Intelligence based on Natural Phenomena.

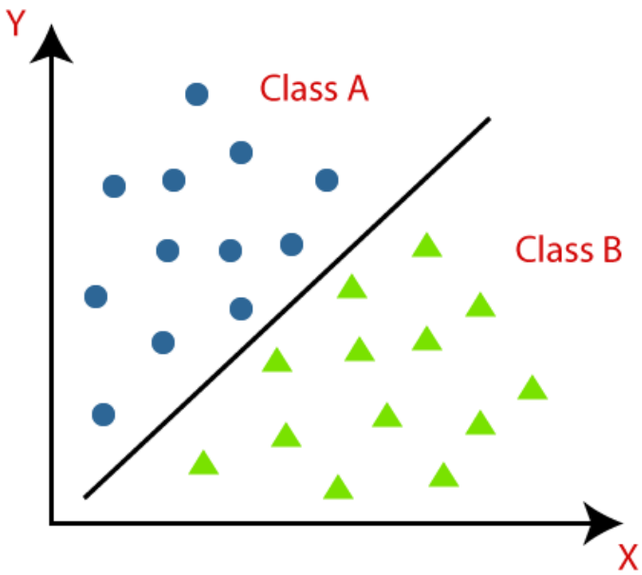
Natural Phenomena:

- Humans are the best example for solving classification problems because we can differentiate things naturally. But it takes extended learning to differentiate everything naturally according to our minds.
- The training of neurons starts from childhood to differentiate between good or bad, cat or dog, and water and fire.
- Now we have trained to an expert level that our sense of recognition is much stronger than in our childhood.
- The same strategy can be adapted to train machines to learn a concept and make decisions for good and bad.

Classification Theory:

- Classification theory states that objects can be divided into groups according to their similarities and dissimilarities.

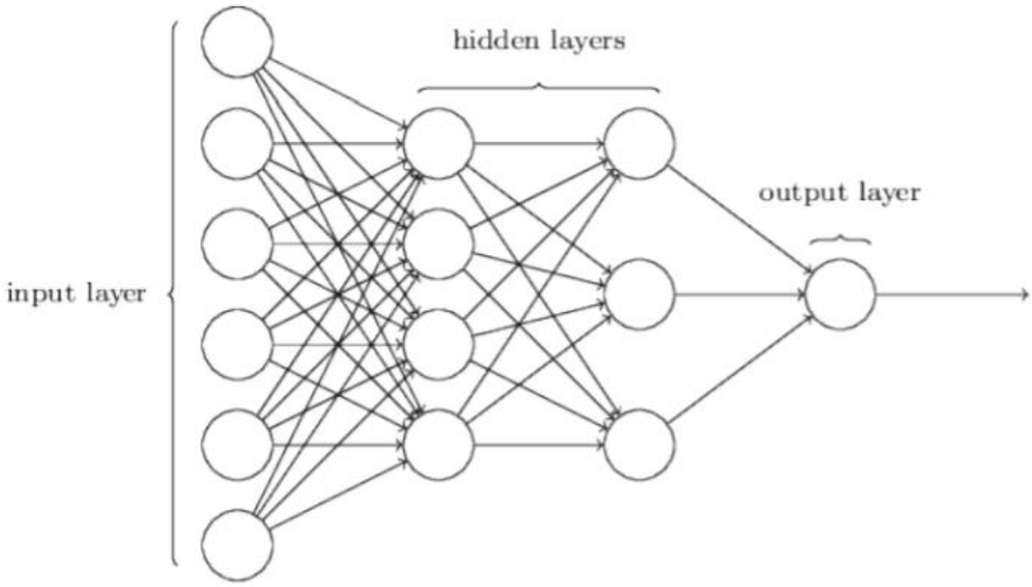
- Classification draws a border line between the object of a similar group that belongs to the same domain. Also, it draws a border line between objects of different groups belonging to the same domain.
- **Learning General Concepts** is a standard used by classification theory to learn from the given data.



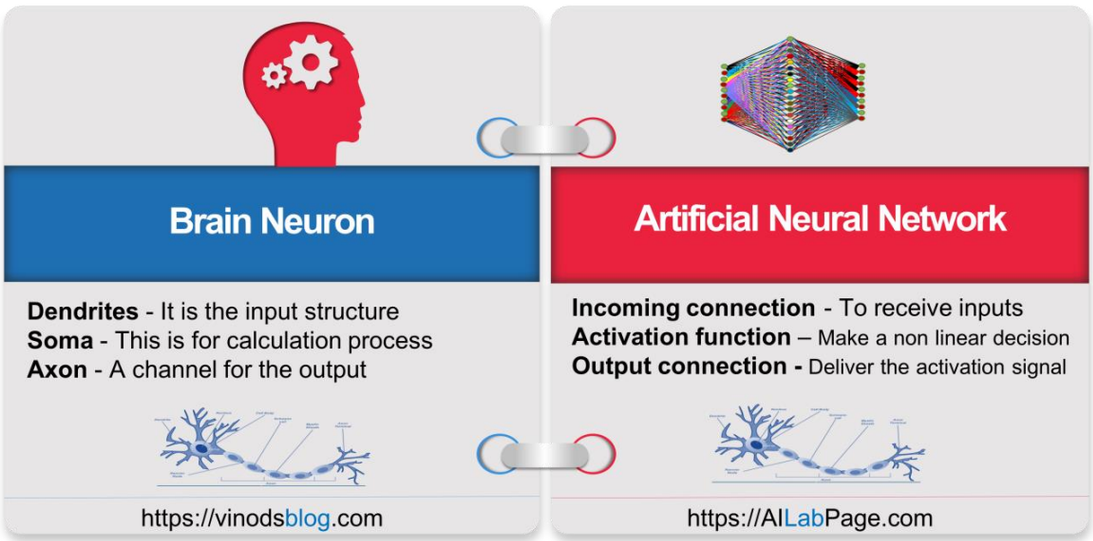
- Source: https://www.javatpoint.com/

What is a Neural Network?

- Artificial neural networks are influenced by animal brains and are based on a collection of connected units called artificial neurons, which model the neurons in a biological brain.
- In artificial neural networks, there is an **input layer, a hidden layer, and an output layer.**
- The input layer depends upon the given data, and the form of data can be an integer or any other.
- A hidden layer is between the input and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.
- There is only one output layer in an artificial neural network, and that is responsible for the final results.
- The following diagram clarifies the concept of the Input, Hidden, and Output layers;

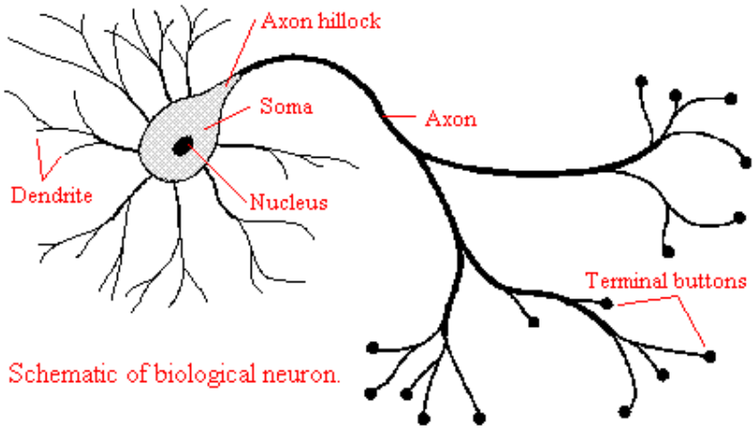


○ Source: https://medium.com/analytics-vidhya/image-classification-with-deep-learning-a-theoretical-introduction-to-machine-learning-and-deep-d118905c6d3a

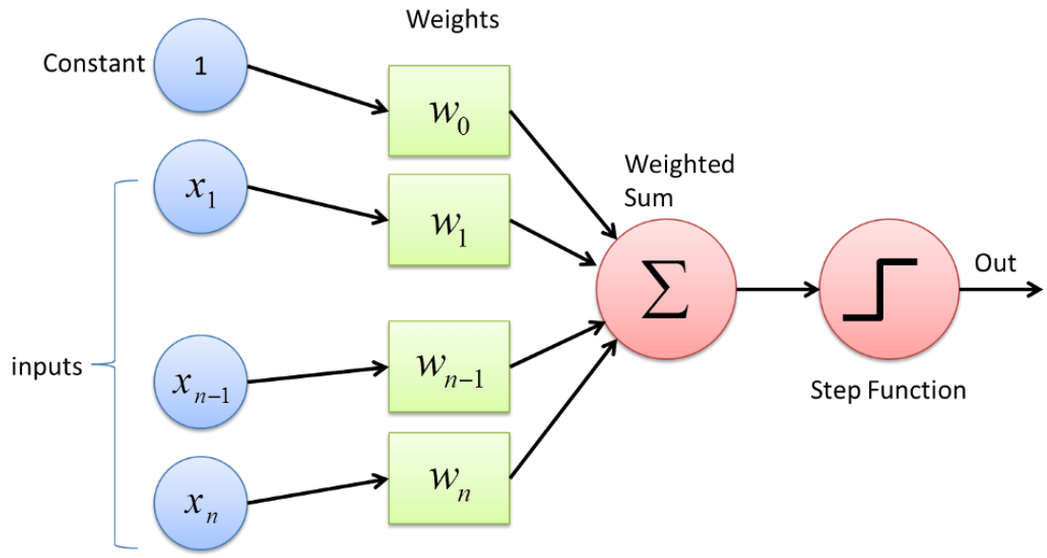


Computational Model:

- The computational model for classification is derived from the biological drawing of a neuron.
- Neurons consist of three parts;
    - Nucleus
    - Dendrites
    - Terminals
- Dendrites are responsible for fetching inputs when the nucleus performs the function on inputs and terminals pass out results.



○ Source: https://web.bvu.edu//



○ Source: https://cacm.acm.org/

- The above diagram shows a representation of a neuron.
- It takes input from the given data, represented by x variables and w weights.
- Learning General Concepts means finding the best possible weights for the training data, which classifies it.
- The neuron space holds a function that performs combined operations on all inputs.
- In this figure, the neuron calculates the dot product of x and w inputs which is then stepped up or stepped down by another active function to shrink the output of the neuron to a single integer.
- Step function also changes according to problem variation.

Characteristics of Neural Network:

- Neurally implemented mathematical model.
- It contains many interconnected processing elements called neurons to do all operations.

- Information stored in the neurons is the weighted linkage of neurons.
- The input signals arrive at the processing elements through connections and connecting weights.
- It can learn, recall and generalize from the given data by suitable assignment and adjustment of weights.
- The collective behavior of the neurons describes its computational power, and no single neuron carries specific information.

PROJECT IMPLEMENTATION:

LANGUAGE USED:

This project is implemented in python language.

LIBRARIES:

Python has many of libraries used for machine learning, data science, data visualization and many other fields as well. We used scikit-learn library in our project which is one of the most popular libraries used for machine learning. It can perform features like classification, regression, and clustering. Since our problem is related to classification, so we used **Multi Layer Perceptron Classifier** (MLP classifier).

Also, libraries like numpy, matplotlib, open-cv are used in our project for dealing matrix of images, plotting, and reading of images.

SKLEARN.NEURAL NETWORK MLP CLASSIFIER:

Multi-layer Perceptron classifier.
This model optimizes the log-loss function using LBFGS or stochastic gradient descent.
There are many parameters of this function but according to our interest we have used the following parameters:

HIDDEN-LAYERS:

hidden_layer_sizes: tuple, length = n_layers -2, default = (100,)

this parameter sets the size of layers as well as neurons. In tuple, first one represents the input layer, the middle represents the hidden layers while the last represent the output layer.

ACTIVATION FUNCTION:

Activation = "logistic"

Activation function is just a function which is used to get the output of node. It also known as transfer function. Here we are using logistic activation function because it exists between 0 and 1. Since our project is based on classification, so logistic activation function will give us the prediction of probability as an output.

SOLVER:

Solver = "sgd"

It is used for weight optimization.

We used "**Stochastic gradient descent**" solver. It is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs.

LEARNING_RATE_INIT:

learning_rate_init = 0.1

It tells the difference to keep in the result of each iteration. We keep it as minimum as possible because large number can lead to high jump which leads in missing some useful data.

RANDOM_STATE:

random_state = 1

It specifies random number generation for weights initialization.

VERBOSE:

Verbose = True

If we set the value of verbose to True, then it will print loss for every iteration. We set the value of verbose to true because we want to check loss figure at every iteration.

MAX_ITER:

max_iter = 800

it specifies the maximum number of iterations to be reached.

N_ITER_NO_CHANGE:

n_iter_no_change = 200

Our classifier will stop when a given number of iterations is reached.

CODE EXPLANATION:

First, we must access the paths of our train data and validation data. Then we must read all the images in the train directory and val directory one by one.

We have made a function **Read_IMG_Path** which will just take a path of train and validation data directory, then it will store the path of all images in a list and total number of images in a folder of each class will be returned by this function.

```
#   Reading Multiple Images From File
#   Using os.walk module, to give us roots , directories and files in our selected path.

def Read_IMG_Path(Path):
    Total_Images = []
    No_Of_Imgs = []
    for roots, dir, files in os.walk(Path):
        No_Of_Imgs.append(len(files))
        for file in files:
            p = os.path.join(roots, file)
            Total_Images.append(p)

    return Total_Images, No_Of_Imgs
```

FEATURES:

One of the main and major steps of project is the feature extraction from image which we will pass to our classifier for training and validation. There are many features which can be extracted from an image for classification, but we have used some of them in our project.

Below are the features used:

COUNT OF BLACK PIXELS:

This feature will return the count of black pixels in our image.

```
#   Black Pixels Count
def Black_Pixels(Data):
    FeatureExtract = []
    for image in Data:
        imageRead = cv.imread(image, 0)
        imageRead = np.round(imageRead)
        count = np.count_nonzero(imageRead == 0)
        FeatureExtract.append(count)
    return FeatureExtract
```

EDGE-DETECTION:

This feature will get the edges of an image. We have used canny module of skimage which will give us edges of our picture.

```
#   Edge-Detection Feature
def Edge_Detection(Data):
    Feature_Extract = []
    for image in Data:
        imageRead = cv.imread(image, 0)
        # imageRead = maImg.imread(image)
        # print(imageRead)
        count = 0
        img = canny(imageRead)
        count = np.count_nonzero(imageRead == 0)
        Feature_Extract.append([count])
    return Feature_Extract
```

DIAGONAL:

This feature will get the diagonal of an image. Diagonal is obtained by using numpy np.diag function.

```
#   Diagonal Feature
def Diagonal(Data):
    FeatureExtract = []
    for image in Data:
        imageRead = cv.imread(image, 0)
        imageRead = np.resize(imageRead, (300, 300))
        diag = np.diag(imageRead)
        FeatureExtract.append(diag)
    return FeatureExtract
```

CENTROID:

This feature will give us the central part of our image.

```
#   Centroid Feature
def Centroid(Data):
    Feature_Extract = []
    for image in Data:
        imageRead = cv.imread(image, 0)
        mid_pixel = len(imageRead)//2
        new_img = imageRead[mid_pixel-5:mid_pixel+5]
        train_img = []
        for i in new_img:
            train_img.append(i[(len(i)//2)-50:(len(i)//2)+50])
        train_img = np.array(train_img)
        train_img = np.reshape(train_img, (1, 1000))
        # print(train_img)
        Feature_Extract.append(train_img)
    return Feature_Extract
```

HOG:

The Histogram of Oriented Gradients (HOG) is a feature descriptor that is used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of Gradient orientation in localized portions of an image. This method is like that of Edge Orientation Histograms, Scale-Invariant Feature Transform descriptors, and shape contexts.

```
def Hog(Data):
    FeatureExtract = []
    for image in Data:
        imageRead = cv.imread(image, 0)

        # cv.imshow("img",img)
        # cv.waitKey(0)

        #creating hog features
        fd, hog_image = hog(imageRead, orientations=9, pixels_per_cell=(8, 8),
                            cells_per_block=(2, 2), visualize=True, channel_axis=None)
        # hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range = (0,5))
        hog_image = asarray(hog_image)
        FeatureExtract.append(hog_image)
    return FeatureExtract
```

After extracting all features from images, we will simply pass our feature to our MLP classifier that will get train on our data.

```
# ----------********************---------------- Multi Layer Perceptron CLASSIFIER ----------------**************************------------
classification = MLPClassifier(hidden_layer_sizes=(100,50,30), activation="logistic", solver="sgd",
                learning_rate_init=0.1, random_state=1, verbose=True, max_iter=800, n_iter_no_change=200).fit(train_x, train_y)
```

EXPERIMENTATION:

HYPOTHESIS 1:

Changing Feature Extraction technique will affect our score values.

EXPERIMENTS:

Different feature extraction responds to different result accuracy because the classifier learns differently. If we train the MLP classifier based on black count in image arrays, it may give some accurate result because somewhere in the data the darker part of train images may be the same as the darker part of validation images. We have performed various experiments to prove our hypothesis.

CENTROID FEATURE EXTRACTION:

First, we check score for the centroid Feature by keeping same hidden layers and other features as well.

RESULT:

Training Score: 14.000000000000002

Validation Score: 10.0

```
Iteration 360, loss = 2.28802429
Iteration 361, loss = 2.28790899
Training loss did not improve more than tol=0.000100 for 200 consecutive epochs. Stopping.
Training Score:  14.000000000000002
Validation Score:  10.0
```

EDGE DETECTION FEATURE:

Then we use edge detection feature to check the effect on our score.

RESULT:

Training Score: 15.0

Validation Score: 10.0

```
Iteration 585, loss = 2.27450345
Iteration 586, loss = 2.27488526
Iteration 587, loss = 2.27420289
Training loss did not improve more than tol=0.000100 for 200 consecutive epochs. Stopping.
Training Score:  15.0
Validation Score:  10.0
PS C:\Users\it computer world>
```

DIAGONAL FEATURE:

Checking the change in score if we set our feature extraction to diagonal Feature Extraction.

RESULT:

Training Score: 14.000000000000002

Validation Score: 10

```
Iteration 487, loss = 2.28784302
Iteration 488, loss = 2.28843654
Training loss did not improve more than tol=0.000100 for 200 consecutive epochs. Stopping.
Training Score:  14.000000000000002
Validation Score:  10.0
PS C:\Users\it computer world>
```

BLACK PIXELS COUNT FEATURE:

Then we switch to Black Pixels Count feature to check the effect on our score.

RESULT:

Training Score: 15.0

Validation Score: 10.0

```
Iteration 585, loss = 2.27450345
Iteration 586, loss = 2.27488526
Iteration 587, loss = 2.27420289
Training loss did not improve more than tol=0.000100 for 200 consecutive epochs. Stopping.
Training Score:  15.0
Validation Score:  10.0
PS C:\Users\it computer world>
```

CONCLUSION:

There occurs a minor change when we change our feature extraction technique.

HYPOTHESIS 2:

Score value also changes when we change hidden layers in our classifier.

EXPERMINET:

For this experiment, we will set our feature to Black Pixels Count. Number of Hidden Layers will be changed in different experiments.

1. **Hidden_layer_size = (100, 50, 60)**
   RESULT:
   Training Score: 15.0
   Validation Score: 10.0

2. **Hidden_layer_size = (50, 100, 50)**
   RESULT:
   Training Score: 14.0
   Validation Score: 10.0

3. **Hidden_layer_size = (10,10,10)**
   RESULT:
   Training Score: 12.0
   Validation Score: 10.0

   CONCLUSION:
   Change in hidden layers affects our Score Values.

CONCLUSION:

Human brain has natural capability of understanding things or objects but implementing this to computer, force us to think on the working of human brain. Artificial Neural Network is one of the most popular models in the world of Artificial Intelligence which we have implemented in this project. This project gives hand on practice of python famous library Scikit-Learn, its features like canny, hog, classifier like MLP Classifier etc. and how they work. Enjoyed in making a machine learn.

NOTE:

It will not be a good practice to say that our implemented code is best because we are achieving score up to 15 to 20 which is very low. A lot of improvements can be done by combining multiple features together and more.