Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Final Year B. Tech (EE)

Trimester: I **Subject: Artificial Intelligence and Machine Learning**

Name: Shreerang Mhatre **Class: TY**

Roll No: 52 **Batch: A3**

Experiment No: 04

Name of the Experiment: Classify species of Iris flower using MLP

Performed on:14/09/2023

Submitted on:14/09/2023

Marks	Teacher's Signature with date

Aim: To create a multilayer neural network and classify species of iris flower using Python.

Prerequisite: Knowledge of MLP, iris flower data and its classes.

Objective:

To create a multi-layer neural network and classify iris flower data using Python Programming.

Components and Equipment required:

SkLearn Python module, Python software, NumPy and Panda Libraries

Theory

A hobby botanist is interested in distinguishing the species of some iris flowers that she has found. She has collected some measurements associated with each iris, which are:

- the length and width of the petals
- the length and width of the sepals, all measured in centimetres.

She also has the measurements of some irises that have been previously identified by an expert botanist as belonging to the species setosa, versicolor, or virginica. For these measurements, she can be certain of which species each iris belongs to. We will consider that these are the only species our botanist will encounter.
The goal is to create a machine learning model that can learn from the measurements of these irises whose species are already known, so that we can predict the species for the new irises that she has found.

Building our model

As we have measurements for which we know the correct species of iris, this is a supervised learning problem. We want to predict one of several options (the species of iris), making it an example of a classification problem. The possible outputs (different species of irises) are called classes. Every iris in the dataset belongs to one of three classes considered in the model, so this problem is a three-class classification problem. The desired output for a single data point (an iris) is the species of the flower considering it's features. For a particular data point, the class / species it belongs to is called its label.

Procedure:

- SkLearn is a pack of Python modules built for data science applications

- load_iris: The classic dataset for the iris classification problem. (NumPy array)
- train_test_split: method for splitting our dataset.
- KNeighborsClassifier: method for classifying using the K-Nearest Neighbor approach.
- NumPy is a Python library that makes it easier to work with N-dimensional arrays and has a large collection of mathematical functions at its disposal. It’s’ base data type is the “numpy.ndarray”.

Output

The target array contains the species of each of the flowers that were measured. This array is composed of numbers from 0 to 2.

The meaning of those numbers are directly related to our target names (classes):

- setosa (0)
- versicolor (1)
- virginica(2)

Conclusion:

Post Lab Questions:

1. What are the limitations of a perceptron?
2. Explain Generalization.
3. How many training data patterns should be used to train a back propagation network.
4. How to determine the number of Hidden Layer Nodes.

Untitled2 Last Checkpoint: 36 minutes ago (unsaved changes)

In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
```

In [4]:
```python
iris = sklearn.datasets.load_iris()
iris_data = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_data['class'] = iris.target
iris_data.head()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [5]:
```python
iris_data.describe()
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000

Untitled2 Last Checkpoint: 37 minutes ago (unsaved changes)

In [5]: iris_data.describe()

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [6]:
```python
iris_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   class              150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [7]:
```python
iris_data.isnull().sum()
```

Out[7]:
```
sepal length (cm)    0
```

```
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [7]:
```python
iris_data.isnull().sum()
```

Out[7]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
class                0
dtype: int64
```

In [8]:
```python
iris_data.corr()
```

Out[8]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
sepal length (cm)	1.000000	-0.117570	0.871754	0.817941	0.782561
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126	-0.426658
petal length (cm)	0.871754	-0.428440	1.000000	0.962865	0.949035
petal width (cm)	0.817941	-0.366126	0.962865	1.000000	0.956547
class	0.782561	-0.426658	0.949035	0.956547	1.000000

In [9]:
```python
X = iris_data.drop('class',axis = 1)
Y = iris_data['class']
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size=0.30)
```

In [10]:
```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [11]:
```python
model.fit(X_train, Y_train) #model training
```


	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126	-0.426658
petal length (cm)	0.871754	-0.428440	1.000000	0.962865	0.949035
petal width (cm)	0.817941	-0.366126	0.962865	1.000000	0.956547
class	0.782561	-0.426658	0.949035	0.956547	1.000000

In [9]:
```python
X = iris_data.drop('class',axis = 1)
Y = iris_data['class']
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size=0.30)
```

In [10]:
```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [11]:
```python
model.fit(X_train, Y_train) #model training
```


Out[11]:
```
LogisticRegression()
```

In [19]:
```python
#print metric to get performance
print('Acuracy:', model.score(X_test, Y_test)*100)
```

```
Acuracy: 91.11111111111111
```

In [ ]:

Exp 4 Classify species of Iris flower using MLP

PAGE No.
DATE 14/9/23

* Post lab Questions-

Q1) What are the limitations of perceptron?

-> Limitations of perceptron -
1. Linear seperability
2. Lack of Hidden Layers
3. Binary outputs
4. sensitivity to Input scale
5. Inability to Learn Complex functions
6. Lack of Representation Learning
7. Unsuitable for Image & Speech Recognition.
8. Training Challenges.

Q2) Explain Generalization

-> Generalization, in the context of machine learning, refers to a model's ability to perform well on new, unseen data that it has not been specifically trained on. It means that the model has learned underlying patterns and relationships in the training data without simply memorizing it. A good generalization implies that the model can make accurate predictions or classifications for a wide range of inputs beyond the training set, indicating its ability to capture.

Q3) How many training data patterns should be used to train a back propagation network

-> The number of training data patterns needed for a back propagation neural network varies based on factors like problem complexity and network design. Generally, thousands to tens of thousands of diverse data points are often reauired, but it depends on the specific problem. overfitting should be avoided, so having enough data is crucial for good generalization. Experementation & cross-validation help determine the right data set size

Q4) How to determine the number of Hidden Layer Nodes

-> The no. of hidden layer Nodes are -

1. start simple
2. Gradually increase
3. Avoid over fitting
4. use of Rule of Thumb
5. Consider Architectural Variations
6. Regularization Techniques
7. Cross - validation
8. Domain Knowledge