

Project Based Learning Report

On

BACKPROPAGATION ON IRIS DATASET USING PYTHON

Submitted in the partial fulfillment of the requirements

For the Project based learning in (Fuzzy Logic,
Neural Networks & Genetic Algorithms)

In

Electronics & Communication Engineering

By

2014111123 Satyam Suresh

2014111107 Shashank

2014111097 Vivek Nagar

Under the guidance of Course In-charge

Prof. V. P. Kaduskar

Department of Electronics & Communication Engineering

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

Academic Year: 2022-23

**Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 4110436**

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the Project Based Learning report entitled, “Backpropagation on Iris dataset using Python” is work done by

2014111123 Satyam Suresh

2014111107 Shashank

2014111097 Vivek Nagar

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **Fuzzy Logic, Neural Networks & Genetic Algorithms** Course of Bachelor of Technology Semester-V, Electronics & Communication Engineering.

Date:

**Prof. V. P. Kaduskar
Course In-charge**

**Dr. Arundhati A.Shinde
Professor & Head**

Index

Page No.	Contents
1-1	Problem Statement with Solution
2-6	Description about project
7-7	Software Used
8-11	Results with Flow process
11-11	Conclusion & Outcome
12-14	Appendix A
15-15	Appendix B

Problem Statement:-

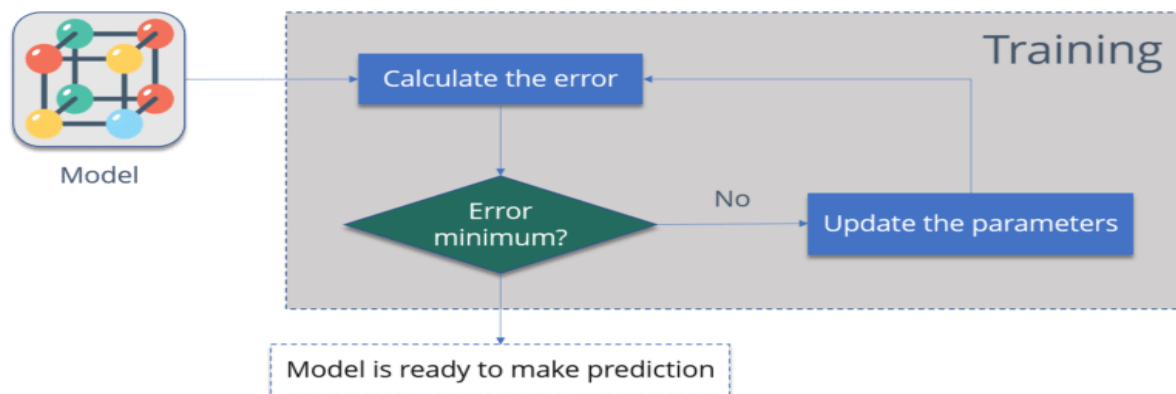
How to implement backpropagation on iris dataset?

Solution:-

The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.

While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact. So, it's not necessary that whatever weight values we have selected will be correct, or it fits our model the best. We have selected some weight values in the beginning, but our model output is way different than our actual output i.e. the error value is huge. Now, how will we reduce the error?

Basically, what we need to do, we need to somehow explain the model to change the parameters (weights), such that error becomes minimum. One way to do it is train our model with Backpropagation algorithm.



Steps needed are:-

1. **Calculate the error** – How far is your model output from the actual output.
2. **Minimum Error** – Check whether the error is minimized or not.
3. **Update the parameters** – If the error is huge then, update the parameters (weights and biases).
After that again check the error. Repeat the process until the error becomes minimum.
4. **Model is ready to make a prediction** – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

Description:-

Iris Dataset:-

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis. It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species. Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".

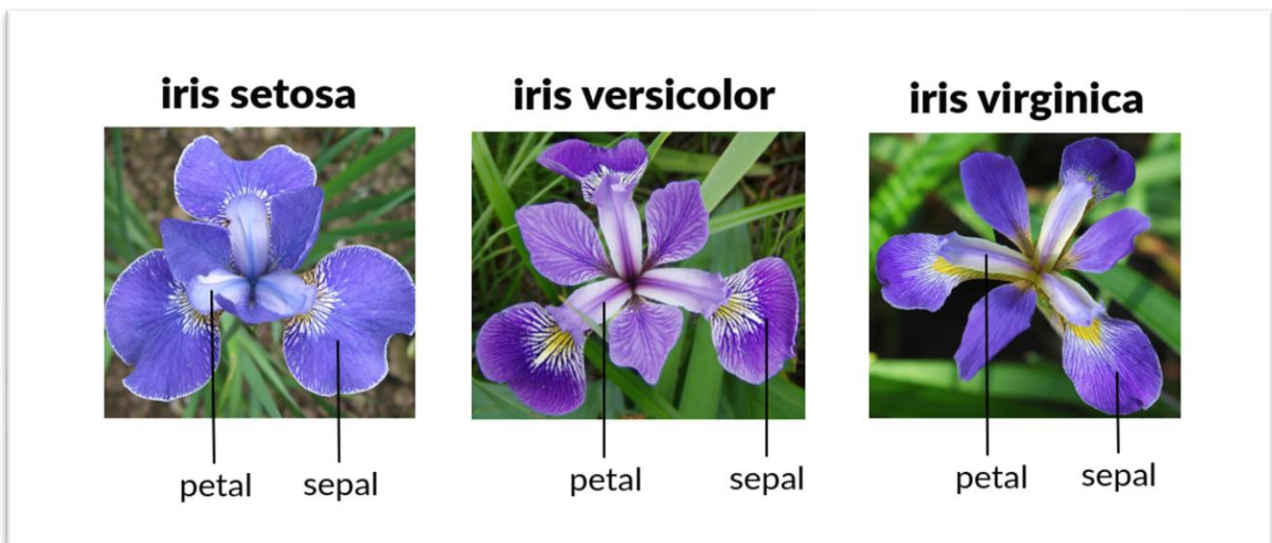


Figure 1

Figure 2

Figure 3

The data set consists of 150 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

Datasets: -

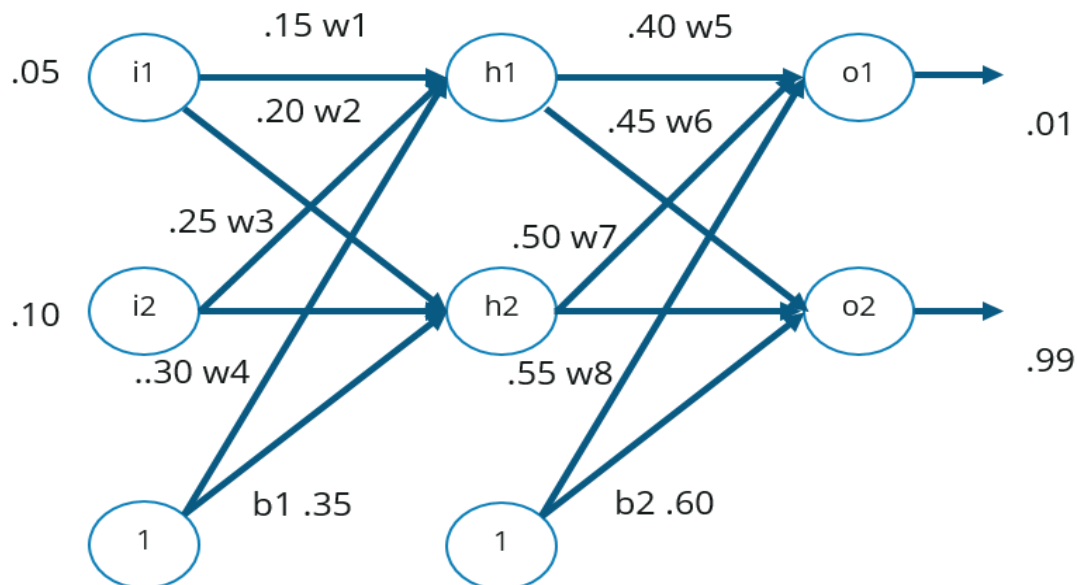
We have loaded the iris dataset from the seaborn library using the “load_dataset” function and performed an analysis in Jupyter Notebook.

Libraries used: -

- 1) Numpy library - NumPy is used to perform various mathematical operations on arrays.
- 2) Pandas Library - pandas provide various data structures and operations for manipulating numerical data and time series.
- 3) Matplotlib library:- Matplotlib library has a pyplot module which is used for plotting 2D graphics.
- 4) Seaborn library - Seaborn is a library for making statistical graphics in Python.
- 5) Sklearn library - It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

How Backpropagation works?

Consider the below Neural Network:



The above network contains the following:

- two inputs
- two hidden neurons
- two output neurons
- two biases

Below are the steps involved in Backpropagation:

- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Step – 1: Forward Propagation

We will start by propagating forward.

Net Input For h1:

$$\text{net h1} = w1*i1 + w2*i2 + b1*1$$

$$\text{net h1} = 0.15*0.05 + 0.2*0.1 + 0.35*1 = 0.3775$$

Output Of h1:

$$\text{out h1} = 1/1+e^{-\text{net h1}}$$

$$1/1+e^{-.3775} = 0.593269992$$

Output Of h2:

$$\text{out h2} = 0.596884378$$

We will repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Output For o1:

$$\text{net o1} = w5*\text{out h1} + w6*\text{out h2} + b2*1$$

$$0.4*0.593269992 + 0.45*0.596884378 + 0.6*1 = 1.105905967$$

$$\text{Out o1} = 1/1+e^{-\text{net o1}}$$

$$1/1+e^{-1.105905967} = 0.75136507$$

Output For o2:

$$\text{Out o2} = 0.772928465$$

Now, let's see what is the value of the error:

Error For o1:

$$E_{o1} = \Sigma 1/2(target - output)^2 \rightarrow \frac{1}{2} (0.01 - 0.75136507)^2 = 0.274811083$$

Error For o2:

$$E_{o2} = 0.023560026$$

Total Error:

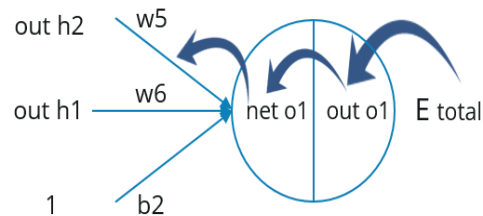
$$E_{total} = E_{o1} + E_{o2} \rightarrow 0.274811083 + 0.023560026 = 0.298371109$$

Step – 2: Backward Propagation

Now, we will propagate backwards. This way we will try to reduce the error by changing the values of weights and biases.

Consider W5, we will calculate the rate of change of error w.r.t change in weight W5.

$$\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta out_{o1}} * \frac{\delta out_{o1}}{\delta net_{o1}} * \frac{\delta net_{o1}}{\delta w_5}$$



Since we are propagating backwards, first thing we need to do is, calculate the change in total errors w.r.t the output O1 and O2.

$$E_{total} = 1/2(target_{o1} - out_{o1})^2 + 1/2(target_{o2} - out_{o2})^2$$

$$\frac{\delta E_{total}}{\delta out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Now, we will propagate further backwards and calculate the change in output O1 w.r.t to its total net input.

$$\text{out } o1 = 1/(1+e^{-neto1})$$

$$\frac{\delta out \ o1}{\delta net \ o1} = \text{out } o1 (1 - \text{out } o1) = 0.75136507 (1 - 0.75136507) = 0.186815602$$

Let's see now how much does the total net input of O1 changes w.r.t W5?

$$\text{net } o1 = w5 * \text{out } h1 + w6 * \text{out } h2 + b2 * 1$$

$$\frac{\delta net \ o1}{\delta w5} = 1 * \text{out } h1 = 0.593269992$$

Step – 3: Putting all the values together and calculating the updated weight value

Now, let's put all the values together:

$$\frac{\delta E_{total}}{\delta w5} = \frac{\delta E_{total}}{\delta out \ o1} * \frac{\delta out \ o1}{\delta net \ o1} * \frac{\delta net \ o1}{\delta w5}$$

0.082167041

Let's calculate the updated value of W5:

$$w5^+ = w5 - \eta \frac{\delta E_{total}}{\delta w5}$$

$$w5^+ = 0.4 - 0.5 * 0.082167041$$

Updated w5

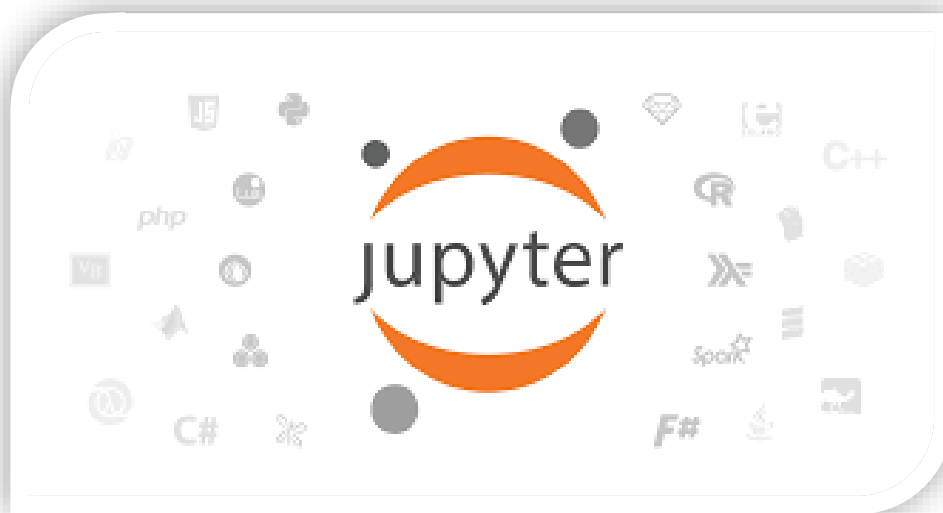
0.35891648

- Similarly, we can calculate the other weight values as well.
- After that we will again propagate forward and calculate the output. Again, we will calculate the error.
- If the error is minimum we will stop right there, else we will again propagate backwards and update the weight values.
- This process will keep on repeating until error becomes minimum.

Software Used:-

Jupyter Notebook: -

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating notebook documents. A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media. Underneath the interface, a notebook is a JSON document, following a versioned schema, usually ending with the “. ipynb” extension.



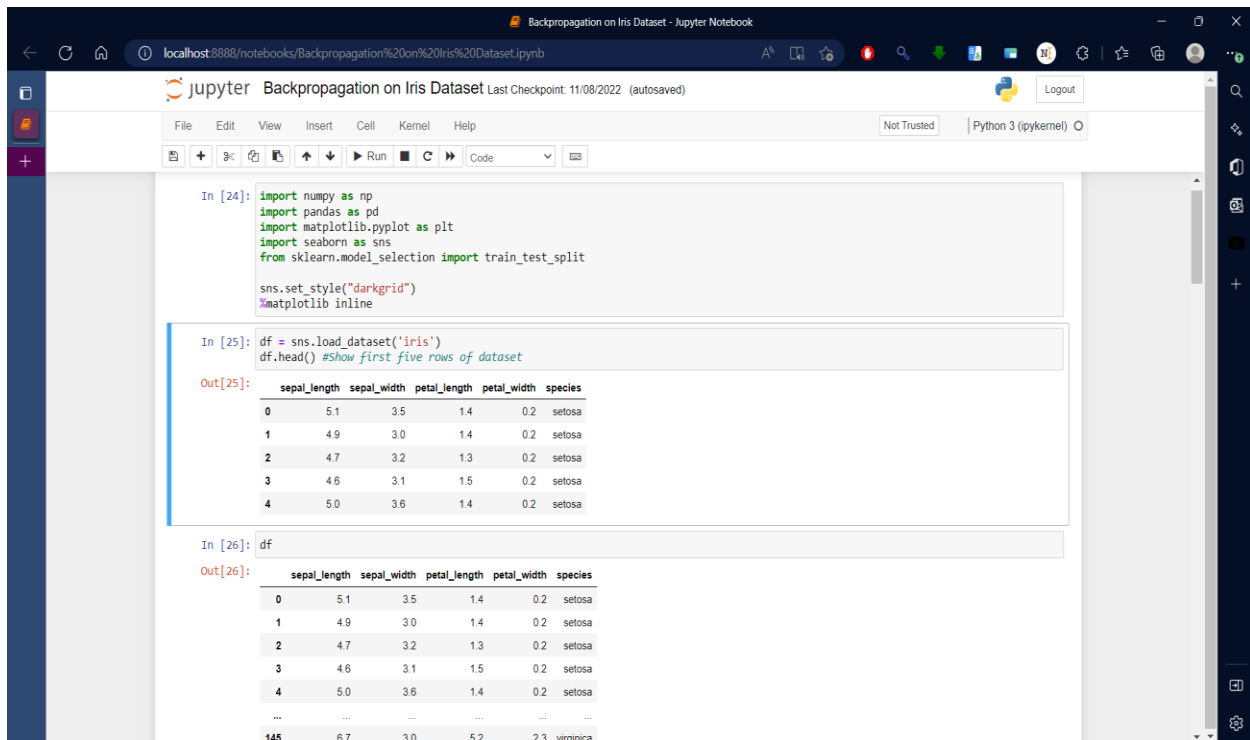
Jupyter notebooks are built upon a number of popular open-source libraries. Jupyter Notebook can connect to many *kernels* to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

Result with Flow process:-

Flow process: -

1. First, we have imported five libraries – numpy as np, pandas as pd, matplotlib.pyplot as plt and seaborn library as sns and sklearn.
2. Secondly, we have loaded our dataset – iris using load_dataset function of seaborn library and used head() function for displaying first five rows of the dataset.
3. Then, we set the first four columns of dataset as input and last column “Species” as output of dataset.
4. Then, we divide 80% of data for training the model and 20% of data for testing the model using “train_test_split” function of sklearn library.
5. We first initialized some random values to the weights between different layers using “random.normal” function of numpy library.
6. After that, we feedforward the model with the weights to predict the output.
7. Then, we noticed that there is some error between the actual output and the predicted output. To reduce that error, we propagated backwards and update the weights.
8. We again feedforward the new weights and calculate the mean squared error in the output.
9. We do the above step for 15000 epochs to get a minimum mean squared error and maximum accuracy.

Screenshots of code: -



```
In [24]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

sns.set_style("darkgrid")
%matplotlib inline

In [25]: df = sns.load_dataset('iris')
df.head() #show first five rows of dataset

Out[25]:
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
In [26]: df

Out[26]:
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...
145	6.7	3.0	5.2	2.3	virginica

Backpropagation on Iris Dataset - Jupyter Notebook

localhost:8888/notebooks/Backpropagation%20on%20Iris%20Dataset.ipynb

jupyter Backpropagation on Iris Dataset Last Checkpoint: 11/08/2022 (autosaved)

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

```
146 6.3 2.5 5.0 1.9 virginica
147 6.5 3.0 5.2 2.0 virginica
148 6.2 3.4 5.4 2.3 virginica
149 5.9 3.0 5.1 1.8 virginica
```

150 rows x 5 columns

```
In [27]: a = df.species.unique()
a
Out[27]: array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
In [28]: y = pd.get_dummies(df.species).values
x = df.drop("species", axis=1).values
print(y)
print()
print(x)
```

```
[[1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [1 0 0]]
```

Backpropagation on Iris Dataset - Jupyter Notebook

localhost:8888/notebooks/Backpropagation%20on%20Iris%20Dataset.ipynb

jupyter Backpropagation on Iris Dataset Last Checkpoint: 11/08/2022 (autosaved)

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

```
In [29]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=20, random_state=4)
```

```
In [30]: def sigmoid(x):
         return 1 / (1 + np.exp(-x))

         def mean_squared_error(predictions, labels):
             N = labels.size
             mse = ((predictions - labels)**2).sum() / (2*N)

             return mse

         def accuracy(predictions, labels):
             predictions_correct = predictions.argmax(axis=1) == labels.argmax(axis=1)
             accuracy = predictions_correct.mean()

             return accuracy
```

```
In [31]: learning_rate = 0.1
         epochs = 15000
         N = y_train.size

         n_input = 4
         n_hidden = 8
         n_output = 3
```

```
In [32]: np.random.seed(10)
         weights_1 = np.random.normal(scale=0.5, size=(n_input, n_hidden)) # (4, 2)
         weights_2 = np.random.normal(scale=0.5, size=(n_hidden, n_output)) # (2, 3)

         monitoring = {"mean_squared_error": [], "accuracy": []}
```

```
In [33]: weights_1
Out[33]: array([[ 0.66579325,  0.35763949, -0.77270015, -0.00419192,  0.31066799,
                 -0.36004278,  0.13275579,  0.05427426],
```

```

Backpropagation on Iris Dataset - Jupyter Notebook
localhost:8888/notebooks/Backpropagation%20on%20Iris%20Dataset.ipynb

jupyter Backpropagation on Iris Dataset Last Checkpoint: 11/08/2022 (autosaved)
Python 3 (ipykernel)

[ 0.00214572, -0.08730011, 0.21651309, 0.60151869, -0.48253284,
 0.51413704, 0.11431507, 0.22256881],
[-0.56830111, 0.06756844, 0.7422685, -0.53990244, -0.98886414,
-0.87168615, 0.13303508, 1.19248367],
[ 0.56184563, 0.83631111, 0.04957461, 0.69899819, -0.13562399,
 0.30660209, -0.13365859, -0.27465451]]

In [34]: for epoch in range(epochs):

# feedforward
hidden_layer_inputs = np.dot(x_train, weights_1)
hidden_layer_outputs = sigmoid(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = sigmoid(output_layer_inputs)

# monitor training process
mse = mean_squared_error(output_layer_outputs, y_train)
acc = accuracy(output_layer_outputs, y_train)

monitoring["mean_squared_error"].append(mse)
monitoring["accuracy"].append(acc)

# backpropagation
output_layer_error = output_layer_outputs - y_train
output_layer_delta = output_layer_error * output_layer_outputs * (1 - output_layer_outputs)

hidden_layer_error = np.dot(output_layer_delta, weights_2.T)
hidden_layer_delta = hidden_layer_error * hidden_layer_outputs * (1 - hidden_layer_outputs)

# weight updates
weights_2_update = np.dot(hidden_layer_outputs.T, output_layer_delta) / N
weights_1_update = np.dot(x_train.T, hidden_layer_delta) / N

```

```

Backpropagation on Iris Dataset - Jupyter Notebook
localhost:8888/notebooks/Backpropagation%20on%20Iris%20Dataset.ipynb

jupyter Backpropagation on Iris Dataset Last Checkpoint: 11/08/2022 (autosaved)
Python 3 (ipykernel)

# weight updates
weights_2_update = np.dot(hidden_layer_outputs.T, output_layer_delta) / N
weights_1_update = np.dot(x_train.T, hidden_layer_delta) / N

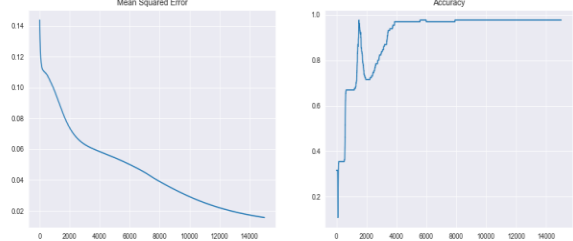
weights_2 = weights_2 - learning_rate * weights_2_update
weights_1 = weights_1 - learning_rate * weights_1_update

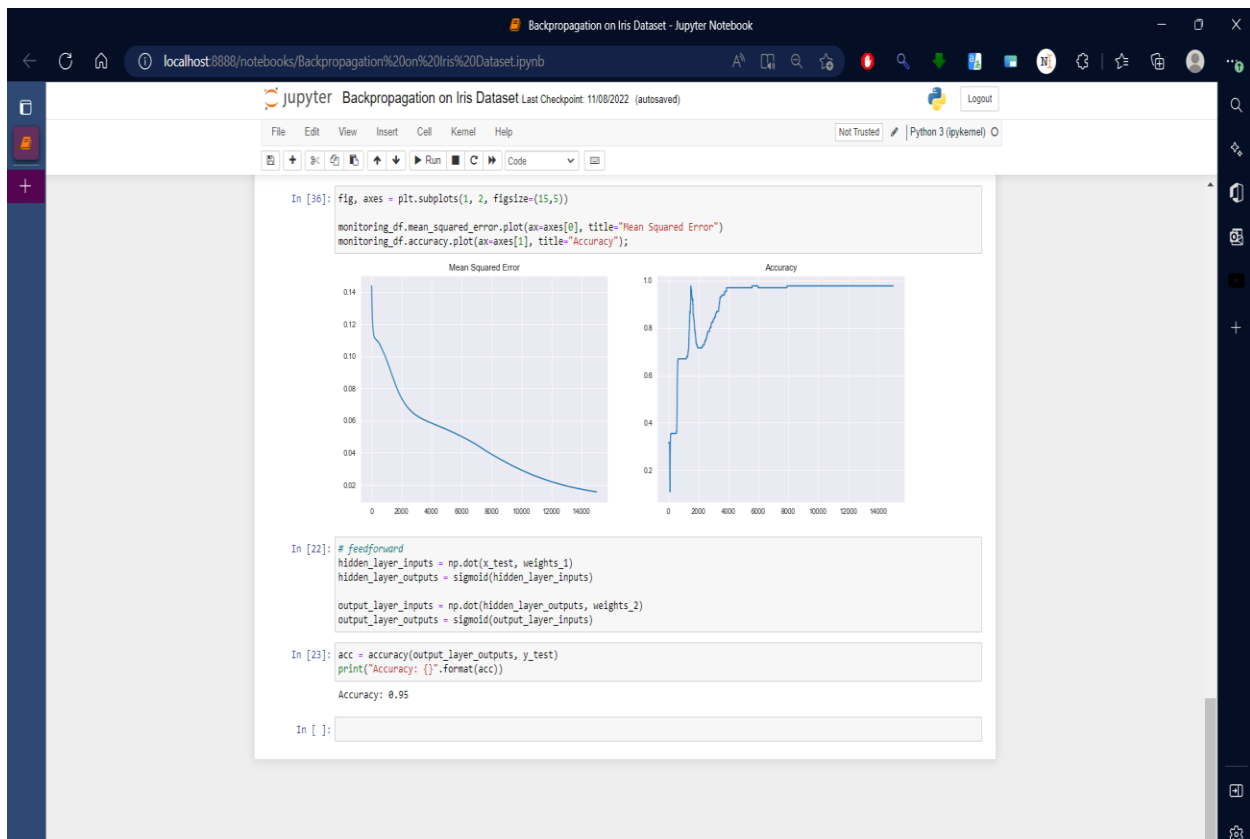
monitoring_df = pd.DataFrame(monitoring)

In [35]: monitoring_df.head()
Out[35]:
   mean_squared_error  accuracy
0         0.143733    0.315385
1         0.143113    0.315385
2         0.142503    0.315385
3         0.141904    0.315385
4         0.141314    0.315385

In [36]: fig, axes = plt.subplots(1, 2, figsize=(15,5))
monitoring_df.mean_squared_error.plot(ax=axes[0], title="Mean Squared Error")
monitoring_df.accuracy.plot(ax=axes[1], title="Accuracy");

```





Project Outcome: -

From this project, we learn to implement backpropagation on iris dataset using python. We also learnt about the use of sigmoid function and mean squared error function in minimizing the error in model of neural networks.

CO4: Develop neural network systems to solve real-world problems.

Project Conclusion: -

From this project, we concluded that every time when we update weights, there is a decrease in mean squared error and a increase in accuracy of model. After a lot of iterations, both of the error and accuracy becomes constant. We also gained the knowledge of software – Jupyter Notebook. We also learnt to analyse the datasets and afterwards, visualizing them.

APPENDIX A

BACKPROPAGATION CODE

```
1.import numpy as np
   import pandas as pd
   import matplotlib.pyplot as plt
   import seaborn as sns
   from sklearn.model_selection import train_test_split
   sns.set_style("darkgrid")
   %matplotlib inline

2.df = sns.load_dataset('iris')
   df.head() #Show first five rows of dataset
   df

3.a = df.species.unique()
   a

4. y = pd.get_dummies(df.species).values
   x = df.drop("species", axis=1).values
   print(y)
   print()
   print(x)

5.x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=20, random_state=4)

6. def sigmoid(x):
   return 1 / (1 + np.exp(-x))

   def mean_squared_error(predictions, labels):
   N = labels.size
   mse = ((predictions - labels)**2).sum() / (2*N)
```

```
return mse
```

```
def accuracy(predictions, labels):  
    predictions_correct = predictions.argmax(axis=1) == labels.argmax(axis=1)  
    accuracy = predictions_correct.mean()  
    return accuracy
```

7. learning_rate = 0.1

```
epochs = 15000
```

```
N = y_train.size
```

```
n_input = 4
```

```
n_hidden = 8
```

```
n_output = 3
```

8. np.random.seed(10)

```
weights_1 = np.random.normal(scale=0.5, size=(n_input, n_hidden)) # (4, 2)
```

```
weights_2 = np.random.normal(scale=0.5, size=(n_hidden, n_output)) # (2, 3)
```

```
monitoring = {"mean_squared_error": [], "accuracy": []}
```

9. for epoch in range(epochs):

```
    # feedforward
```

```
    hidden_layer_inputs = np.dot(x_train, weights_1)
```

```
    hidden_layer_outputs = sigmoid(hidden_layer_inputs)
```

```
    output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
```

```
    output_layer_outputs = sigmoid(output_layer_inputs)
```

```
    # monitor training process
```

```
    mse = mean_squared_error(output_layer_outputs, y_train)
```

```
    acc = accuracy(output_layer_outputs, y_train)
```

```
    monitoring["mean_squared_error"].append(mse)
```

```
    monitoring["accuracy"].append(acc)
```



```

# backpropagation
output_layer_error = output_layer_outputs - y_train

output_layer_delta = output_layer_error * output_layer_outputs * (1 -
output_layer_outputs)
hidden_layer_error = np.dot(output_layer_delta, weights_2.T)
hidden_layer_delta = hidden_layer_error * hidden_layer_outputs * (1 -
hidden_layer_outputs)

# weight updates
weights_2_update = np.dot(hidden_layer_outputs.T, output_layer_delta) / N
weights_1_update = np.dot(x_train.T, hidden_layer_delta) / N
weights_2 = weights_2 - learning_rate * weights_2_update
weights_1 = weights_1 - learning_rate * weights_1_update
monitoring_df = pd.DataFrame(monitoring)
monitoring_df.head()

10. fig, axes = plt.subplots(1, 2, figsize=(15,5))
    monitoring_df.mean_squared_error.plot(ax=axes[0], title="Mean Squared Error")
    monitoring_df.accuracy.plot(ax=axes[1], title="Accuracy");

11. # feedforward
    hidden_layer_inputs = np.dot(x_test, weights_1)
    hidden_layer_outputs = sigmoid(hidden_layer_inputs)
    output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
    output_layer_outputs = sigmoid(output_layer_inputs)

12. acc = accuracy(output_layer_outputs, y_test)
    print("Accuracy: {}".format(acc))

```

APPENDIX B

GITHUB LINK

[Shashankkrj/Backpropagation-on-Iris_Dataset \(github.com\)](#)

Folders present:-

1. Code Folder name:- **Backpropagation on Iris Dataset.ipynb**
2. Dataset Folder name:- **iris.data.csv**
3. Presentation Folder name :- **FLNNGA_PBL_PRESENTATION.pdf**