BACKPROPAGATION

- we initialize weights with some random values or any variable
- It's not mean that whatever weight values we have selected will be correct
- The error value may be huge
- We explain the model to change the parameters (weights), such that error becomes minimum using Backpropagation

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

SCIKIT-LEARN - ML LIBRARY

- Machine Learning Library
- It provides a selection of efficient tools for machine learning and statistical modeling including

- Classification
- Regression
- Clustering
- Dimensionality reduction

Installing Library: pip install scikit-learn

7_DL_Backpropagation.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:05

Comment Share

+ Code + Text

RAM Disk Editing

BackPropagation

Importing Libraries

+ Code + Text

```
[96] import numpy as np #array Operations
     import pandas as pd #Handling Data
     from sklearn.datasets import load_iris #Plant Iris Dataset
     from sklearn.model_selection import train_test_split #Splitting Dataset into Train & Test
     import matplotlib.pyplot as plt #plotting Graph
```

Load Dataset

7_DL_Backpropagation.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

Load Dataset

```
dataset = load_iris()
dataset
```

```
{'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Character
 'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
```

0s completed at 19:24

7_DL_Backpropagation.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

Initializing Weights

```
np.random.seed(10)
firstWeight = np.random.normal(scale=0.5, size=(input_size, hidden_size))
lastWeight = np.random.normal(scale=0.5, size=(hidden_size , output_size))

firstWeight
```

```
array([[ 0.66579325,  0.35763949],
       [-0.77270015, -0.00419192],
       [ 0.31066799, -0.36004278],
       [ 0.13275579,  0.05427426]])
```

Segregating Data into X(Imput) & Y(Output)

0s completed at 19:29

7_DL_Backpropagation.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

Segregating Data into X(Input) & Y(Output)

```
x=dataset.data
y=dataset.target
y
```

+ Code + Text

Convert categorical variable into dummy/indicator variables

```
[101] y = pd.get_dummies(y).values
```

Split dataset into Train & Test

0s completed at 19:35

7_DL_Backpropagation.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

Split dataset into Train & Test

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=20, random_state=4)
```

Initializing Hyper Parameters

```
[123] learningRate = 0.1
      epoch = 5000
      N = y_train.size
      input_size = 4
      hidden_size = 2
      output_size = 3
      results = pd.DataFrame(columns=["MeanSquareError", "accuracy"])
```

0s completed at 19:40

7_DL_Backpropagation.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

```
[123] results = pd.DataFrame(columns=["MeanSquareError", "accuracy"])
```

Activation Function

```
def activationFn(x):
    return 1 / (1 + np.exp(-x))
```

Mean Squared Error

```
[104] def mse(y_pred, y_true):
          return ((y_pred - y_true)**2).sum() / (2*y_pred.size)
```

0s completed at 19:42

Show notifications

Allow Block

ation.ipynb

Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
def mse(y_pred, y_true):
    return ((y_pred - y_true)**2).sum() / (2*y_pred.size)
```

Evaluation Function - Accuracy

```
def accuracy(y_pred, y_true):
    acc = y_pred.argmax(axis=1) == y_true.argmax(axis=1)
    return acc.mean()
```

Training

```
for i in range(epoch):
```

0s completed at 19:45

7_DL_Backpropagation.ipynb - C | NumPy Documentation | numpy.random.normal — NumP | pandas.get_dummies — pandas

colab.research.google.com/drive/1lHZaw_JYoJ_r65QSYGdHSNh0w4RAgAG9#scrollTo=oPtIPJpKU7tn

colab.research.google.com wants to
Show notifications
Allow Block

ation.ipynb

Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
# feedforward propagation on hidden layer
firstNetInput = np.dot(X_train, firstWeight)
firstAFnOp = activationFn(firstNetInput)

# feedforward propagation on output layer
lastNetInput = np.dot(firstAFnOp, lastWeight)
lastAFnOp = activationFn(lastNetInput)

# Calculating error
mseValue = mse(lastAFnOp, y_train)
acc = accuracy(lastAFnOp, y_train)
results=results.append({"MeanSquareError":mseValue, "accuracy":acc},ignore_index=True)

# backpropagation
E1 = lastAFnOp - y_train
dfirstWeight = E1 * lastAFnOp * (1 - lastAFnOp)
```

7_DL_Backpropagation.ipynb - C | NumPy Documentation | numpy.random.normal — NumP | pandas.get_dummies — pandas

colab.research.google.com/drive/1IHZaw_JYoJ_r65QSYGdHSNh0w4RAgAG9#scrollTo=oPtIPJpKU7tn

colab.research.google.com wants to
Show notifications
Allow Block

ation.ipynb

Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
# backpropagation
E1 = lastAFnOp - y_train
dfirstWeight = E1 * lastAFnOp * (1 - lastAFnOp)

E2 = np.dot(dfirstWeight, lastWeight.T)
dlastWeight = E2 * firstAFnOp * (1 - firstAFnOp)

# weight updates
lastWeight_update = np.dot(firstAFnOp.T, dfirstWeight) / N
firstWeight_update = np.dot(X_train.T, dlastWeight) / N

lastWeight = lastWeight - learningRate * lastWeight_update
firstWeight = firstWeight - learningRate * firstWeight_update
```

colab.research.google.com wants to
Show notifications
Allow Block

ation.ipynb

Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

```
results.MeanSquareError.plot(title="Mean Squared Error")
```

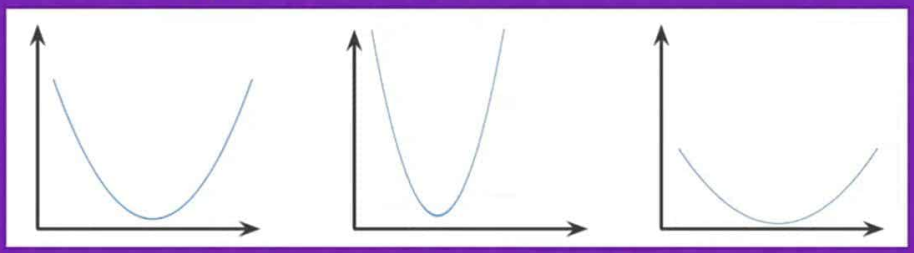
```
[112] results.accuracy.plot(title="Accuracy")
```

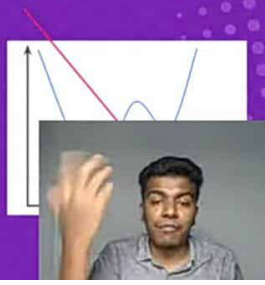
```
[114] # feedforward
      firstNetInput = np.dot(X_test, firstWeight)
      firstAFnOp = activationFn(firstNetInput)

      lastNetInput = np.dot(firstAFnOp, lastWeight)
      lastAFnOp = activationFn(lastNetInput)

      acc = accuracy(lastAFnOp, y_test)
      print("Accuracy: {}".format(acc))
```

Deep Learning Terminology - 1

convex function

- A function in which the region above the graph of the function is a convex set
- It is shaped something like the letter U
- has exactly one local minimum point, which is also the global minimum point.

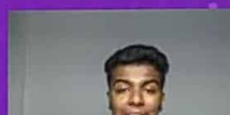




Deep Learning Terminology - 2

Multilayer Perceptron

- An entry point towards complex neural nets where input data travels through various layers of artificial neurons
- Every single node is connected to all neurons in the next layer which makes it a fully connected neural network
- Input and output layers are present having multiple hidden Layers
- It has a bi-directional propagation i.e. forward propagation and backward propagation
- Used for deep learning [due to the presence of dense fully connected layers and back propagation]



Deep Learning Terminology - 3

Artificial Neural Network

- It is a group of multiple perceptron or neurons at each layer
- ANN is also known as a Feed-Forward Neural network because inputs are processed only in the forward direction
- The network may or may not have hidden node layers, making their functioning more interpretable