Programming Test: Learning Activations in Neural Networks

Dataset Used: Bank Note Authentication

Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.

Data Set Information:

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tools were used to extract features from images.

Attribute Information:

1. variance of Wavelet Transformed image (continuous)

2. skewness of Wavelet Transformed image (continuous)

3. curtosis of Wavelet Transformed image (continuous)

4. entropy of image (continuous)

5. class (integer)

Algorithms used: Stochastic Gradient Descent

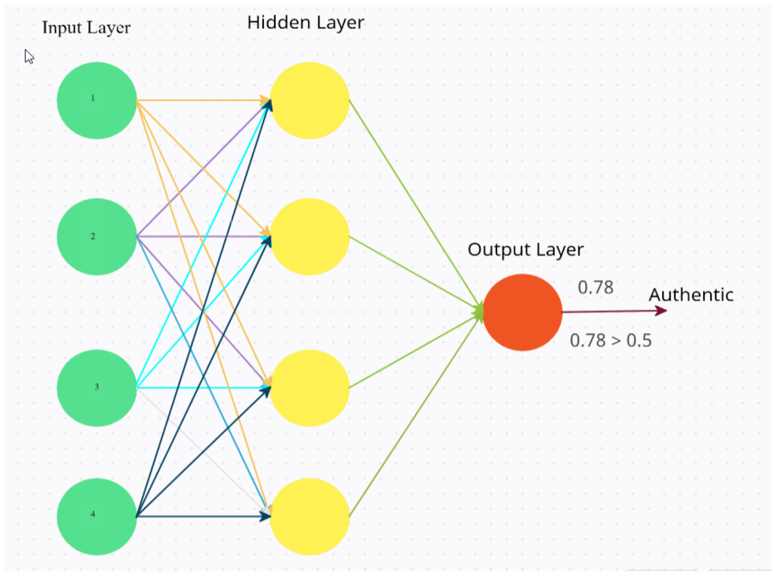
I. **Artificial Neural Network**

Steps:

1. Define neural network structure
2. Initialize model's parameters
3. Forward Propagation
4. Computing loss
5. Backward Propagation to get gradients
6. Update parameters(gradient descent)
7. Neural network model
8. Prediction
9. Metrics
10. Visualizing model

- We will import some basic python libraries like numpy, matplotlib, sklearn, etc., that we need.
- We need to define the number of input units, the number of hidden units, and the output layer. The input units are equal to the number of features in the dataset 4 ,

hidden layer is set to 4, and the problem is the binary classification we will use for a single layer output.



- We need to initialize the weight matrices and bias vectors. Weight is initialized randomly while bias is set to zeros.
- For forward propagation, given the set of input features (X), we need to compute the activation function for each layer. For the hidden layer, we are using the **tanh** activation function. Similarly, for the output layer, we are using **sigmoid** activation function.
- Now, we need to calculate the gradient w.r.to different parameters.
- After calculating gradients for different parameters we need to update the parameters using the gradient descent rule.
- Finally, putting together all the functions we can build a neural network model with a single hidden layer.
- Using the learned parameters, we can predict the class for each example by using forward propagation.
- If the activation is greater than 0.5, then the prediction is 1 otherwise it’s 0.

Parameters

```
 Weights and bias for the activation of hidden layer

 <-----W1----->
 [[ 0.06521107  0.06642471 -0.00685537  0.01800601]
  [-0.56387281 -0.30989762 -0.34719833 -0.09989109]
  [-0.45605825 -0.23006511 -0.24961856 -0.07175504]
  [-0.47052236 -0.24868128 -0.26897584 -0.08918061]]

 <-----b1----->
 [[-0.00852729]
  [ 0.29754159]
  [ 0.1833203 ]
  [ 0.20013005]]
```

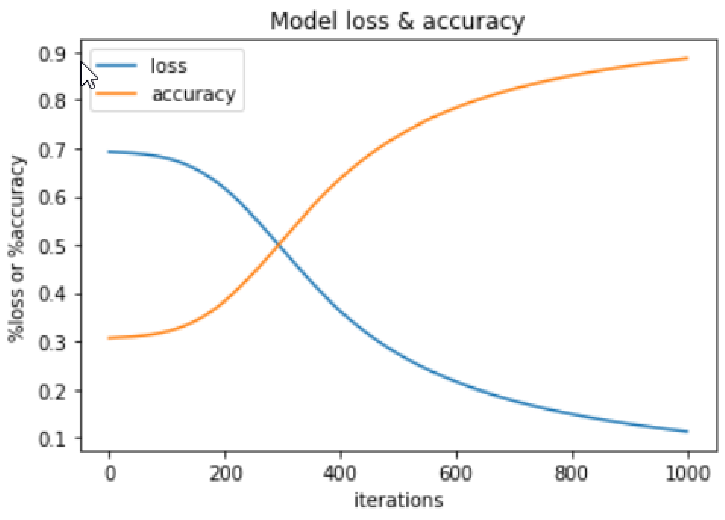
```
Weights and bias for the activation of output layer

<-----W2----->
[[-0.10448563  1.25174565  0.86289524  0.92569912]]

<-----b2----->
[[0.0798439]]
```

Classification Report

```
----------Test classification report----------

              precision    recall  f1-score   support

         0.0       0.99      0.95      0.97       154
         1.0       0.94      0.98      0.96       121

    accuracy                           0.96       275
   macro avg       0.96      0.97      0.96       275
weighted avg       0.96      0.96      0.96       275
```

Test Accuracy: 96.36363636
Train Accuracy: 97.9033728

Model loss & accuracy Vs Iterations


Model loss & accuracy
loss
accuracy
%loss or %accuracy
iterations