EEL 4930/ 5934

Introduction to Biomedical Image Analysis

Assignment – 10

Due: 04/16/2024, Noon

Multi-Layer Perceptrons for Image Classification:

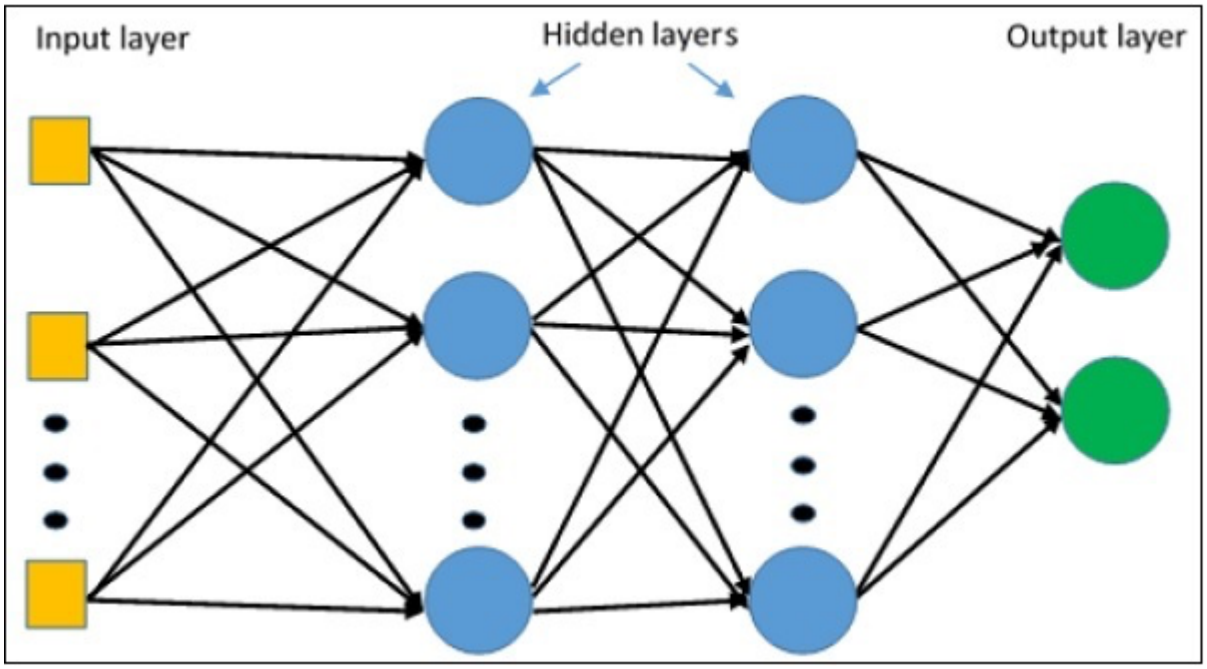


Image taken from: https://www.tutorialspoint.com/tensorflow/tensorflow_multi_layer_perceptron_learning.htm

Multi-layer perceptrons are a way in which we can estimate non-linear relationships between multiple classes within a dataset. In the Input layer, we have features that are extracted from the images with the middle Hidden layers being randomly initialized prior to training. An initial prediction is made by propagating values from the Input layer, through the Hidden Layer(s), and to the Output layer where the final prediction is made. Over many iterations, the weights between Input, Hidden, and Output nodes are updated to correct for erroneous predictions.

Part 1: Dataset Generation:

For the first part of this assignment, we will have to generate training data of images from two different classes that we will attempt to classify.

1. Read in the ‘Circle_bin.png’ and ‘Star_bin.png’ images.
2. Within a for loop from 1 to 100:
    a. Generate a motion blur kernel with a random size between 0 and 100 with a random direction between 0 and 145.
    b. Use *conv2()* to convolve both your star and circle images.
    c. Use *im2bw()* with a random binarization level to binarize the motion blurred images.
    d. Apply Gaussian (size = [20, 20], sigma = 7) and salt & pepper noise (default strength).
        i. Resize the images to [50, 50] and now go on to feature extraction.

This is accomplished in the code submitted with this assignment.

(10 pts.)

Part 2: Feature set Generation:

Now that we have 100 images of each, we’ll need to extract features of the images that we’ll use for classification.

1. The first set of features we’ll be extracting will be the vectorized pixel values. Before the image generation for-loop, initialize two matrices of zeros that have a size of [100, 50*50]. Replace each row with the resized image pixel values.
    a. Use *reshape(Image, [], 1)*
2. The second set of features will be based on the binarized noisy images. Initialize two matrices of zeros that have a size of [100, 5].
    a. Binarize each of the images with *imbinarize(Image)*.
    b. Use *regionprops(Image, ‘Area’, ‘Eccentricity’, ‘Solidity’, ‘Perimeter’, ‘Extent’)* to calculate the area, eccentricity, solidity, perimeter, and extent of the image.
        i. You can convert the ‘struct’ to a matrix using *cell2mat(struct2cell())*.
    c. Replace each row in the second set of feature matrices with these features.
3. Generate a label matrix with 1’s for circle images and 2’s for star images.

This is also accomplished in the code submitted with this assignment.

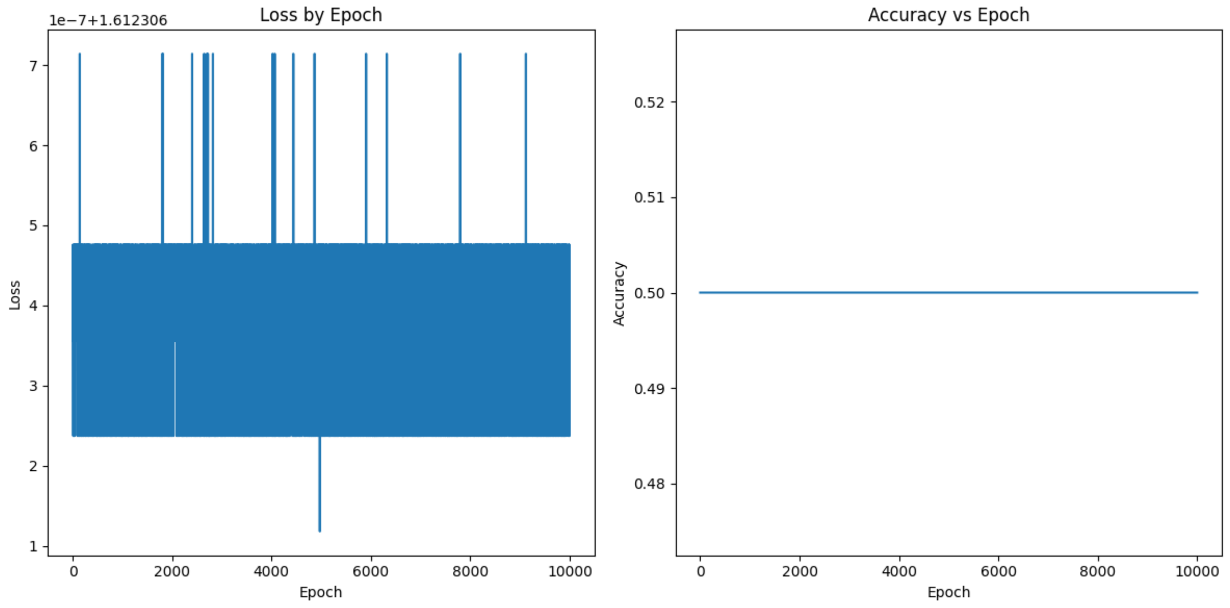
(10 pts.)

Part 3: Training MLP:

Now that we have the two feature sets, let's try to optimize the parameters for the MLP provided in *'backprop_Assignment.m'.*

1. Replace data and label variables with your own variable names for the first feature set (vectorized pixel values).
2. Run '*backprop_Assignment.m'* with the default values for learning rate, number of iterations, and size of hidden layer. **Include the output figure below. What is the classification rate after 10000 iterations? What does this tell you about how well the network is performing?**

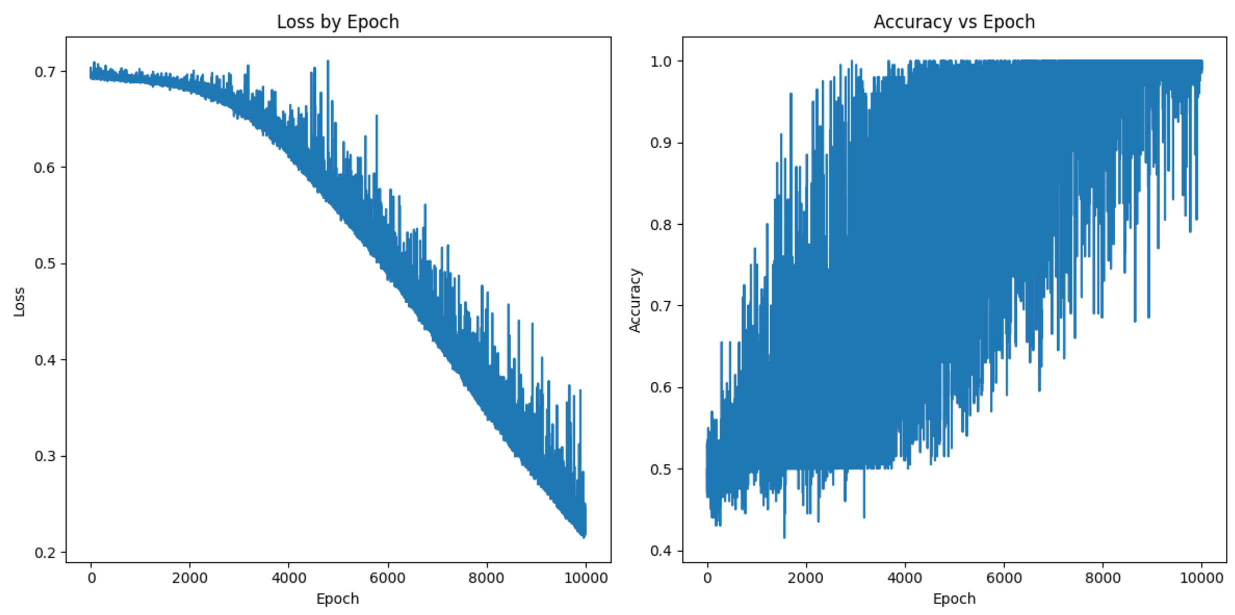
(Note: I read the MATLAB code and I think created the same figures it does, but there may be some differences. If there is something missing just let me know and I can modify the figure output.)



The classification rate is 0.5 (printed in code output) – coupled with the unchanging loss, we likely have the learning rate too low. Our network is performing very poorly (just guessing 50%).

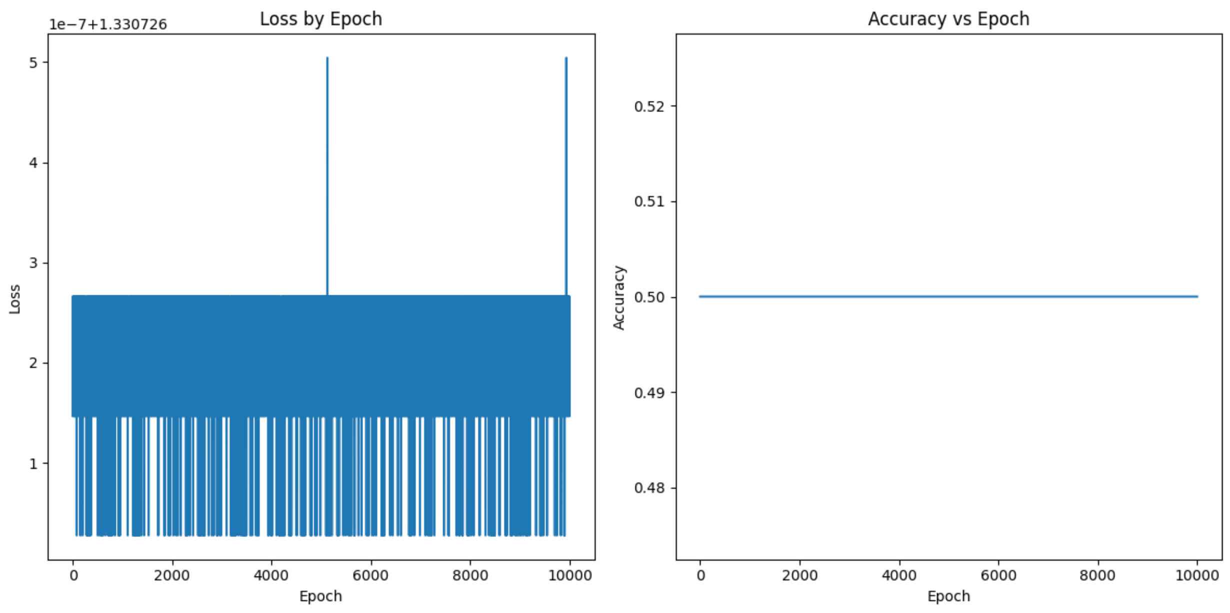
3. **Vary the value for size of hidden layer, learning rate, and number of iterations until you get a model that reaches classification rate > 0.9. Include the output figure below.**

Parameters selected (that aren't default): hidden layer size of 15, learning rate of 1e-4. Achieved a final accuracy of 0.990



4. Repeat process for second feature set (morphological features). **Include output figures below for initial values (learning_rate = 5e-15, n_iter = 1e4, and M = 10) and model with classification rate > 0.9.**
    a. Be sure to clear your workspace in between training runs.

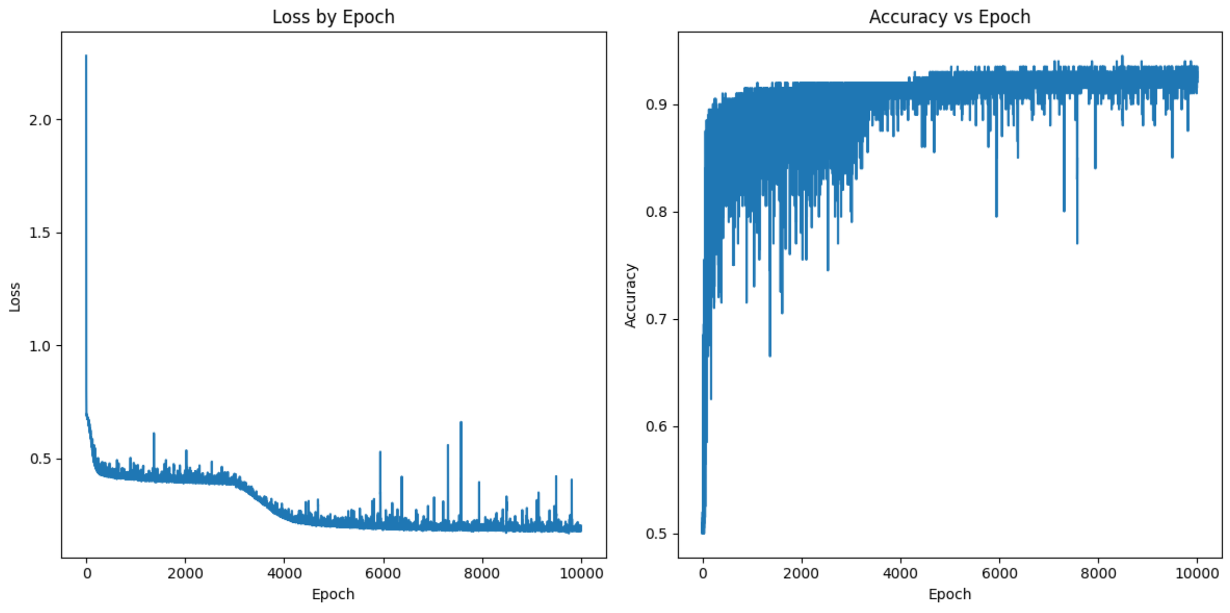
<u>Default parameters:</u>



Similarly, we have basically unchanging high loss, and 50% accuracy. Our learning rate is too low.

<u>Modified:</u>

Parameters: Hidden layer size 20, learning rate 5e-3.



Finally

5. **Which combination of features and training parameters (learning rate, number of iterations, and hidden layer size) converged fastest? Why do you think some combinations resulted in constant classification rate over many iterations?**

The one with the highest learning rate converged fastest. This makes sense – it means that the model can learn the shapes of the features faster. The combinations resulting in constant classification rate over many epochs is caused by the model weights not being changed enough to re-classify any specific image separately.

(20 pts.)