Activity 17: Neural Network

Mary Chris Go / 2014 11122

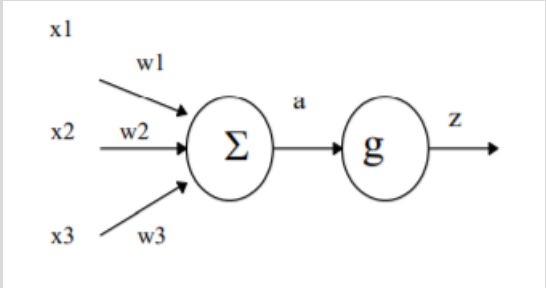
Goal

To be able to build a neural network for classification

Mary Chris Go / 2014 11122

Neural Network

Artificial neurons being interconnected with other neurons, we form a neural network that consists of input, hidden, and output layer.



How it Works

The input feature or the set of signals goes inside the input layer which consists of input neurons. These signals are then forward to the hidden layer. The hidden layer does something to these signals and the output will be passed on to the output layer. We train the network by just adjusting the weights according to the bias we set. This change rule will depend from the optimization of some cost function.

Mary Chris Go / 2014 11122

Theoretical Process of training a multilayer neural network

1. Per feature in the dataset, we evaluate the activations of the hidden unit in:

$$z_j = g(a_j), a_j = \sum_{i=0}^{d} w_{ji} x_i \quad .$$

And activation for the output units using:

$$y_k = \tilde{g}(\tilde{a}_k), \tilde{a}_k = \sum_{j=0}^{c} \tilde{w}_{kj} z_j$$

2. Now, we compute the error of each output through:

$$\tilde{\delta}_k = \tilde{g}'(a_k)\{y_k - t_k\}$$

3. For the hidden errros, we compute their errors by:

$$\delta_j = g'(a_j) \sum_{k=1}^{c} \tilde{w}_{kj} \tilde{\delta}_k$$

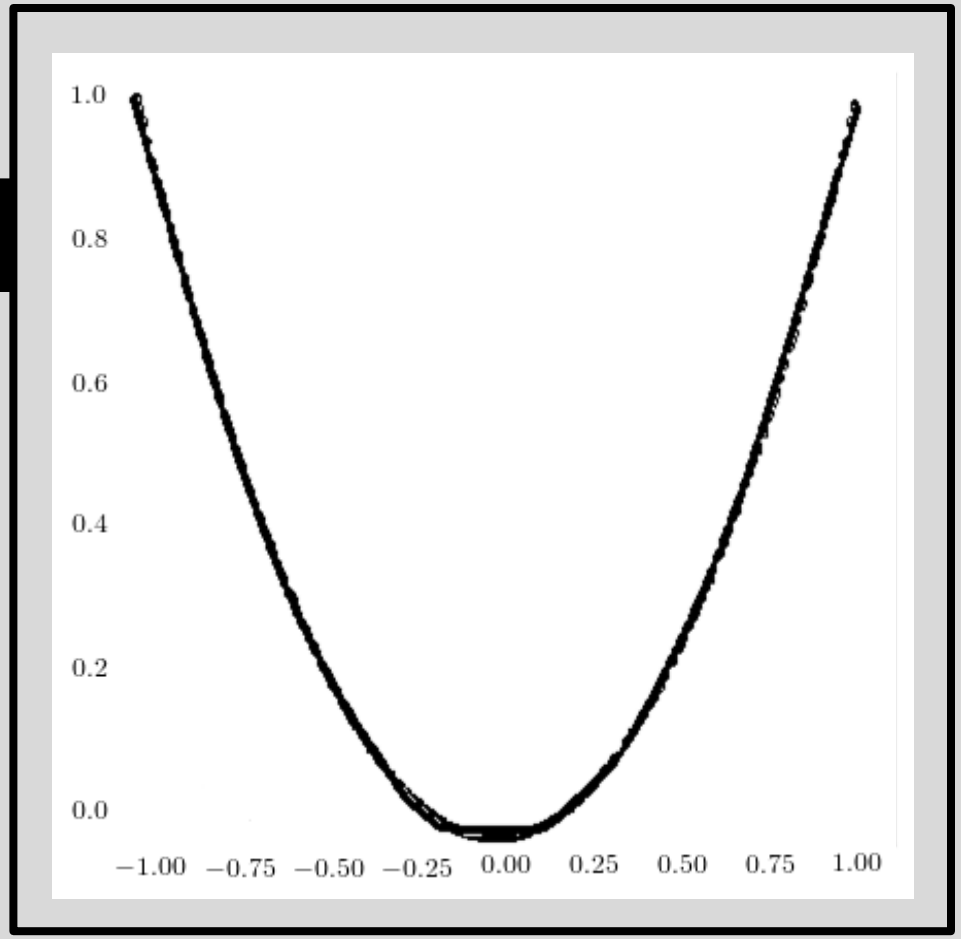
4. Now, we compute the error derivatives using two equations:

$$\frac{\partial E^q}{\partial \tilde{w}_{kj}} = \tilde{\delta}_k z_j \qquad \frac{\partial E^q}{\partial w_{ji}} = \delta_j x_i$$

5. For each pattern, repeat steps 1 to 4.

Mary Chris Go / 2014 11122

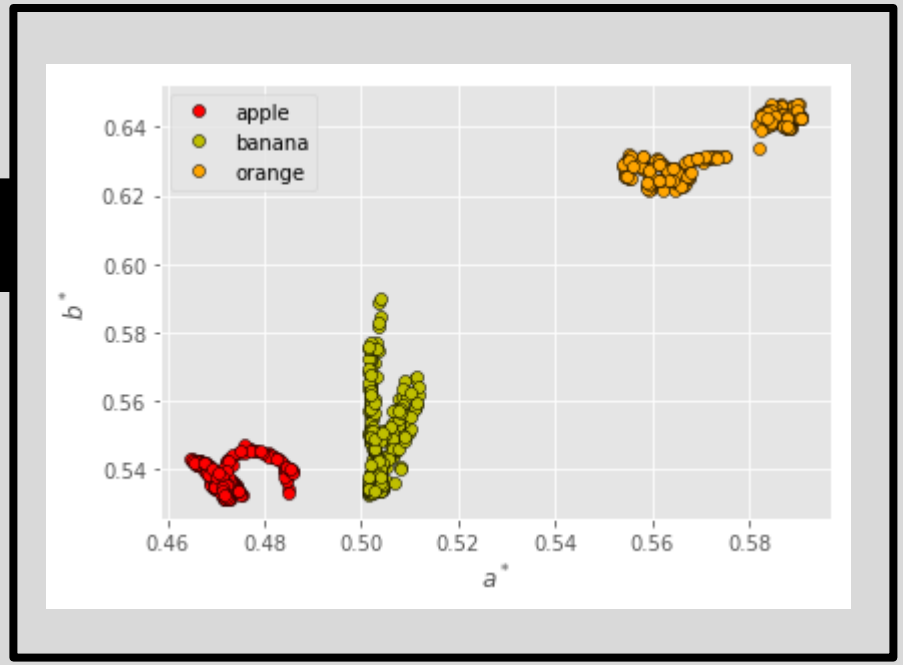
Function Fitter



In understanding neural network, we build it first as a universal function fitter. In here, we see how it tries to fit in a quadratic function. The colors were not friendly to differentiate them though. But all in all, it was able to fit in a specific function.

Mary Chris Go / 2014 11122

Feature Space




```
for j in range(3):
    filenames = os.listdir(dirs[j])
    for i,f in enumerate(filenames):
        if i == 50:
            break
        #eccentricity
        img = cv.imread(dirs[j] + f)
        img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        thres, out = cv.threshold(img_gray, 127, 255, cv.THRESH_OTSU)
        out = (img_gray < thres).astype(float)
        img_label = meas.label(out)
        props = meas.regionprops(img_label)
        ecc = props[0]['eccentricity']

        # a* b*
        img_Lab = cv.cvtColor(img, cv.COLOR_BGR2Lab).astype(float)
        img_Lab /= img_Lab[:,:,0].max()
        img_L, img_a, img_b = cv.split(img_Lab)

        ass[j].append(img_a.mean())
        bss[j].append(img_b.mean())
        ecs[j].append(ecc)
```

This snippet of code was already used several times in clustering three kinds of fruit: orange, banana, and mango. These values were saved as csv files.

Mary Chris Go / 2014 11122

Setting up the model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden1 (Dense)              (None, 729)               2187
_________________________________________________________________
hidden2 (Dense)              (None, 72)                52560
_________________________________________________________________
hidden3 (Dense)              (None, 7)                 511
_________________________________________________________________
output (Dense)               (None, 3)                 24
=================================================================
Total params: 55,282
Trainable params: 55,282
Non-trainable params: 0
_________________________________________________________________
```

This part of the code is where we set up the model in sequence. 3 hidden layers were implemented. About 55 thousand parameters were trainable and 0 non-trainable parameters. Given that we have small dataset and we are just trying to build a network, numerous layers are not recommended. As a newbie in ML, this is a good start.

Mary Chris Go / 2014 11122

Setting up the model

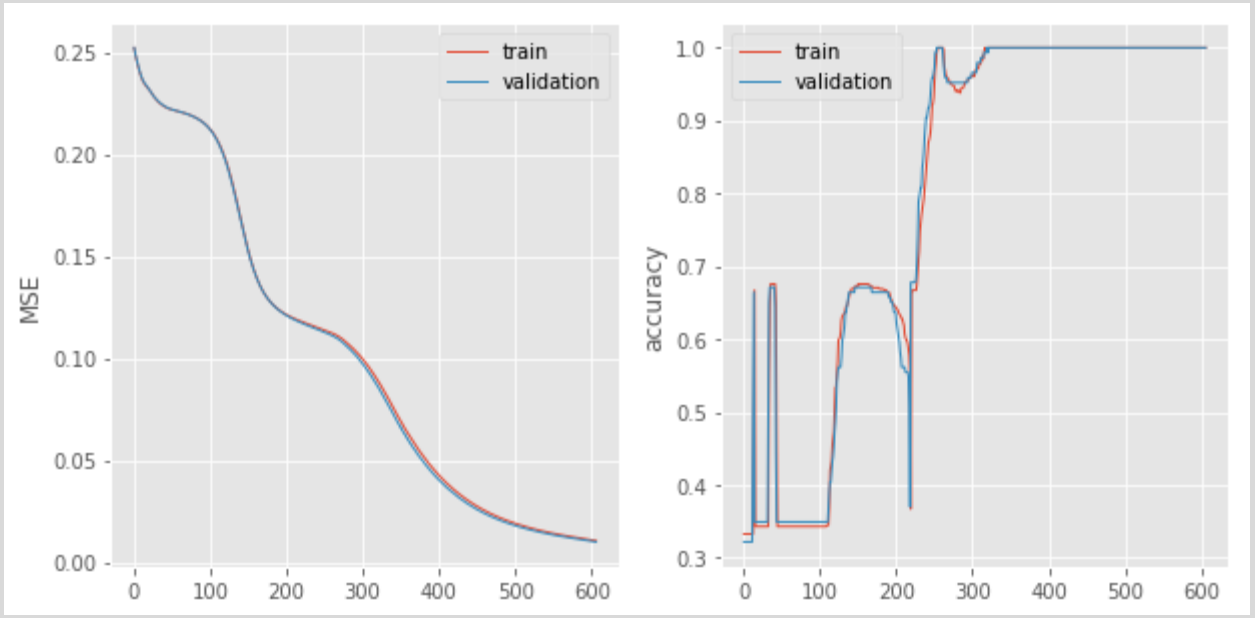
```
history = model.fit(X_train, y_train,
                    batch_size=len(X_train),
                    validation_split=0.2,
                    epochs=int(1e6),
                    verbose=0,
                    callbacks=[stopval, checkpoint])
model.save('fruits.model')
```

executed in 30.1s, finished 12:18:48 2019-11-30

Now, we try to train the model with EarlyStopping and ModelCheckpoint. It was more of a trial and error to find the right values for patience, verbose, and monitor setup. We compile the model using adam optimizer which will help in the future. Now we save the model and we will use it later. We set up our own batch_size and epochs. This would depends also with the capability of the machine you are using. As I am only using my laptop, I use the least amount to maximize it.

Mary Chris Go / 2014 11122

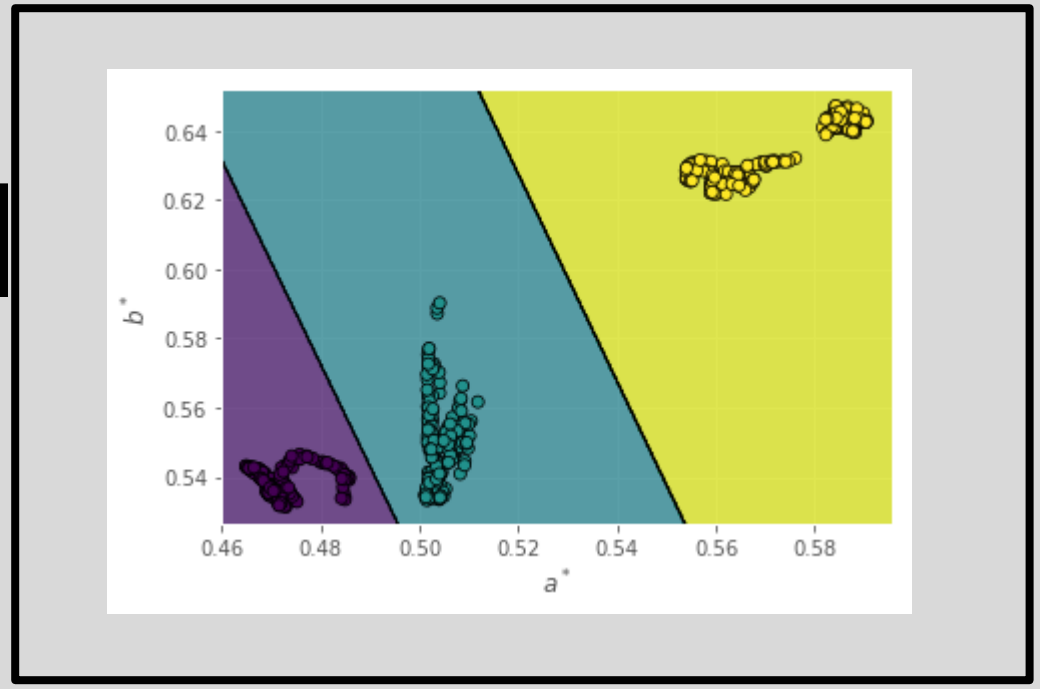
Setting up the model



We plot the epoch vs the history of loss and accuracy for both train and validation. As you can see, the two lines are almost distinguishable. We expect a good value for accuracy in this portion.

Mary Chris Go / 2014 11122

Setting up the model




```
Test accuracy: 100.0
```

The accuracy is 100 percent which is a good thing. But as physicists, we must understand as to why it has a perfect accuracy. The model was able to classify the images perfectly. To be honest, I doubt my own model. Maybe this is because of the dataset I used. I was able to divide them but the images really look alike. The colors and shape were perfectly different from each other and maybe that's why.

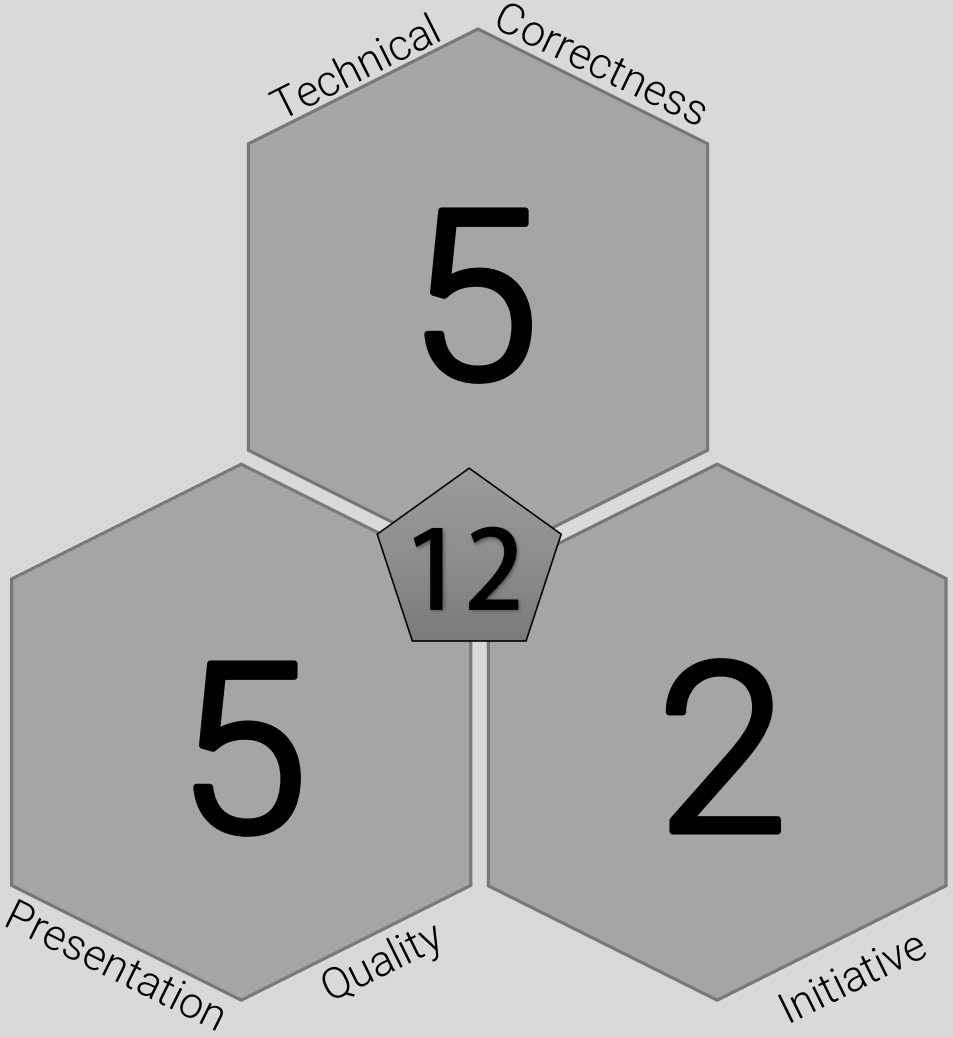
Summary

What's crucial about neural network classifier is the dataset. With my limited and not-so-broad dataset, the neural network's accuracy cant be trusted fully. The image of the fruits were not really different from each other. Maybe next time, we can put images which put the specific fruit in a certain lighting that will train your model to be robust when classifying.

Setting up of your own neural network using hard code was a bit tough. But it was fun knowing that you get good results in the end. Although there are many available packages already, it is good to try to see what is the flow by coding it. But also, to be more efficient, packages are still better.☺ Again, this wouldn't be possible without the help of my classmates.

Mary Chris Go / 2014 11122

Self-Evaluation



References

- Soriano, M., “Neural Networks”. 2019