Activity 17 – Neural Networks

Kenneth V. Domingo
2015–03116
App Physics 186, 1st Semester, A.Y. 2019–20
Corresponding author: kvdomingo@up.edu.ph

Results and Discussion

1.1 Function fitter

For this activity [1], we will first explore the applications of a neural network—or more accurately, a multilayer perceptron (MLP)—as a universal function fitter. Designing an neural network architecture can be quite tedious as there are a lot of hyperparameters to tune in order for you to get desired results. After several attempts of trial-and-error, I ended up with the network shown in Fig. 1 (all network visualizations were done using [2]). For the training data, I generated 1000 random numbers distributed uniformly in the range $[-1, 1]$ as the input, and the corresponding function as the output. Specifically, the functions used were:

- linear: $f(x) = x$
- quadratic: $f(x) = x^2$
- cubic: $f(x) = x^3$
- $\tanh(x)$
- logistic sigmoid: $\frac{1}{1+\exp[-x]}$
- $\sin(2\pi f x)$

The first hidden layer of the network contains 500 nodes, which are all activated by a rectified linear unit (ReLU), defined by

$$g(x) = \max(0, x) \tag{1}$$

$$g'(x) = \begin{cases} 0 & , \quad x < 0 \\ 1 & , \quad x > 0 \end{cases} \tag{2}$$

The second hidden layer contains 10 nodes, also activated by ReLU. The output layer is a single node which maps everything calculated so far to a single output value. Thus, the function arguments (random numbers $\in [-1, 1]$) comprise an entire dataset $\mathcal{M}$. The network objective is to minimize the ℓ_2 loss, defined as

$$\ell_2(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \tag{3}$$

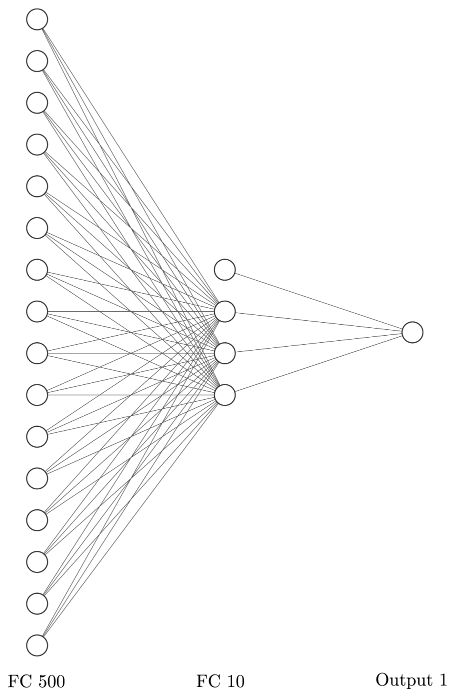


Figure 1: Network architecture for the function fitter.

where $\mathbf{y}, \hat{\mathbf{y}}$ are the ground truth and predicted outputs, respectively, and $\|\mathbf{y}\|_2 := \left(\sum_i y_i^2\right)^{1/2}$ is the ℓ_2 norm operator. The network is trained until the loss drops below a value of 0.01. I opted to use the ℓ_2 loss instead of the sum-of-squares error or mean-squared error because, from experience, the former is more perceptually accurate.

For the test data, I generated 1000 equally-spaced numbers in the range $[-1, 1]$. Figure 2 shows the predictions for each function.

1.2 Fruit classification

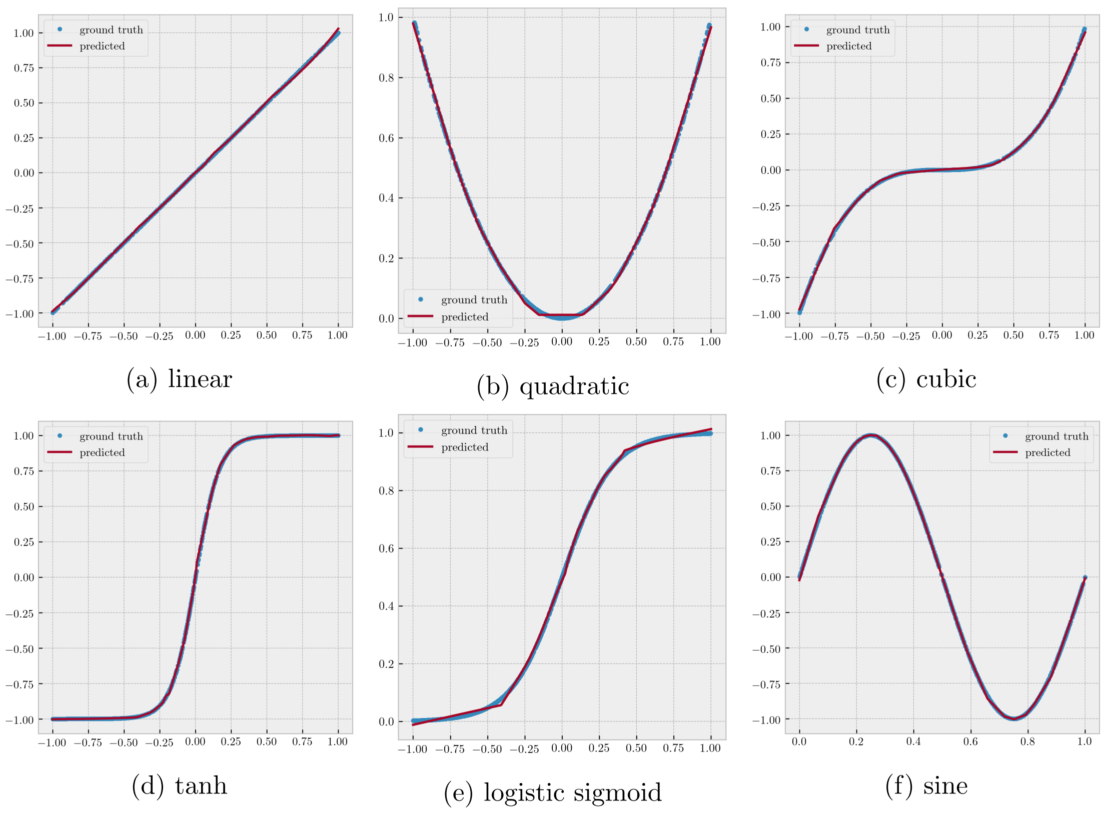
For this section, I used my extracted $a^* - b^*$ color features of bananas, apples, and oranges from previous activities. The architecture I came up with is shown in Fig. 3. The dataset consists of 50 feature vectors for each fruit (total of 150 feature vectors), which were split 80% for training and 20% for testing.

The network was designed such that the first hidden layer contains as many nodes as data points. Layers are added which contain $\frac{1}{10}$ many nodes as the previous layer, until the number of nodes in the current layer is on the order of magnitude of 10^1. The output layer contains as many nodes as the classes. The hidden layers are activated by a ReLU except for the last, which is activated by a logistic sigmoid, defined as

$$g(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

$$g'(x) = g(x)(1 - g(x)) \tag{5}$$

and the output layer is activated by a softmax, defined by



(a) linear (b) quadratic (c) cubic (d) tanh (e) logistic sigmoid (f) sine

Figure 2: Neural network as a universal function fitter.

$$g(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{6}$$

$$g'(x_i) = g(x_i)(\delta_{ik} - g(x_k)) \tag{7}$$

which outputs probabilities such that the sum of the output nodes should equal unity. Thus, for this dataset, the network topology is 120-12-3 (two hidden layers, 3 nodes in output layer).

This time, the network objective is to minimize the mean-squared error (MSE), and the network is trained until it drops below a value of 0.01. Feeding in the test data shows a test accuracy of 100%, and the decision boundaries are shown in Fig. 4. This is not surprising since the fruit feature vectors form distinct clusters and do not overlap.

Now, let's consider adding more samples to the training/testing sets. Using the fruits dataset from [3], I added more varieties of apples, bananas, and oranges, as well as an additional class of cherries. Figure 5 shows the distribution of their feature vectors in $a^* - b^*$ space. Notice now that the clusters occupy more area and are almost touching each other.

With a total of 4752 feature vectors, I split the data 50-50 among the train-test sets. Therefore, the network topology now is 2376-237-23-4 (3 hidden layers, 4 output nodes). Plugging in the test data afterwards yields a test accuracy of 99.96%, and the decision boundaries are shown in Fig. 6.

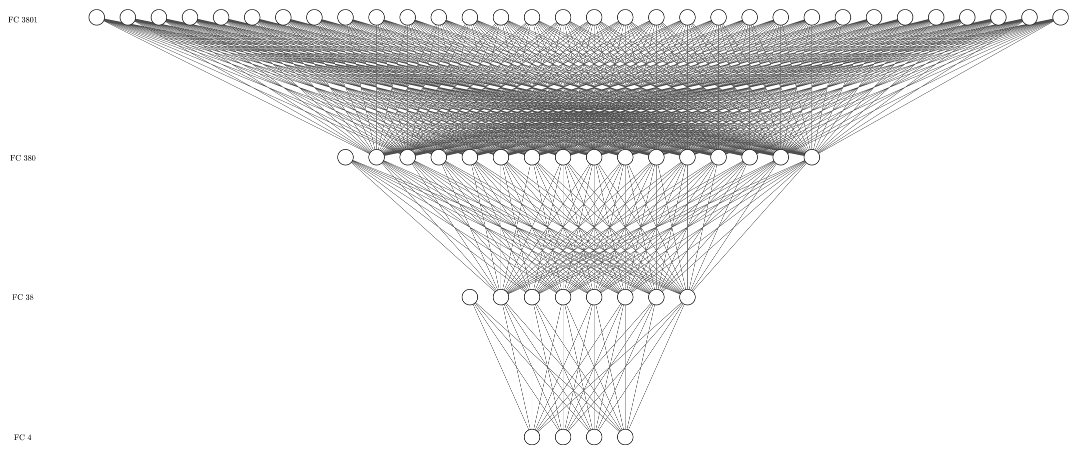


Figure 3: Network architecture for the fruit classifier.

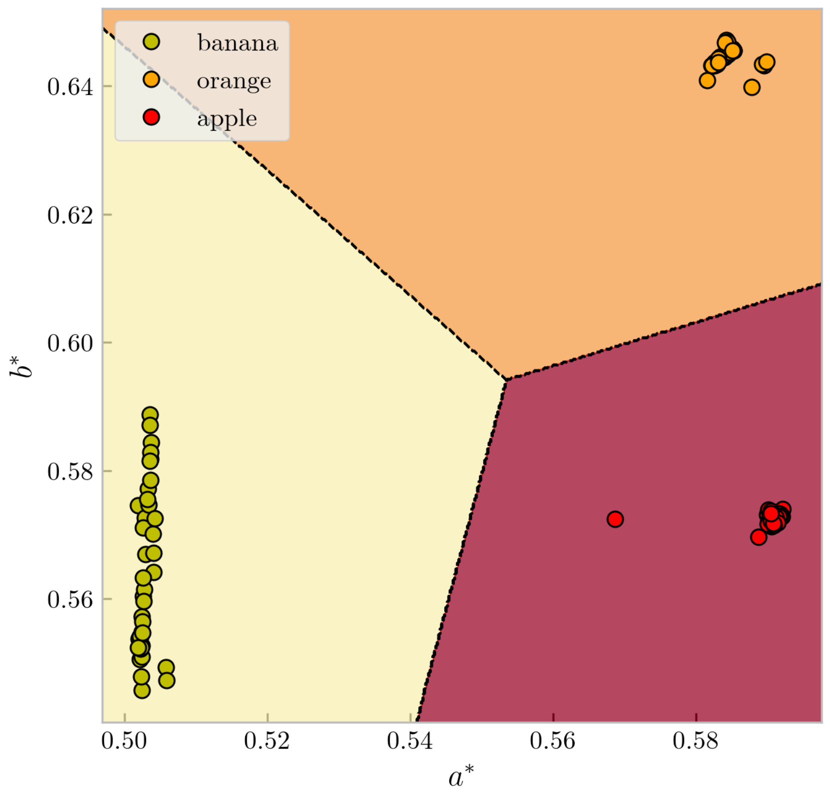


Figure 4: Decision boundaries for bananas, apples, and oranges in $a^* - b^*$ feature space.

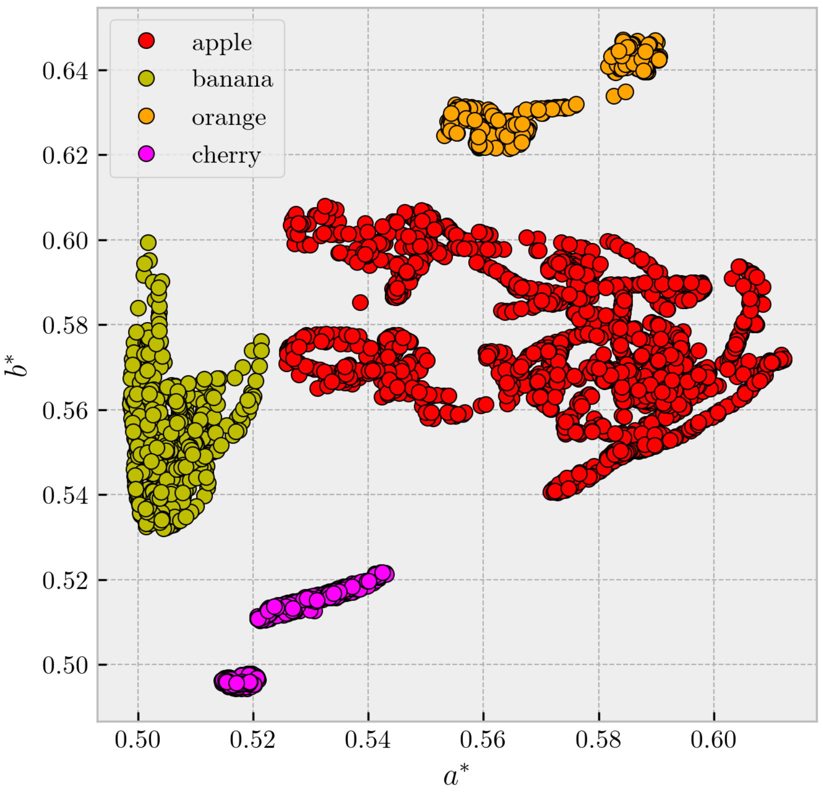


Figure 5: $a^* - b^*$ feature space of apples, bananas, oranges, and cherries.

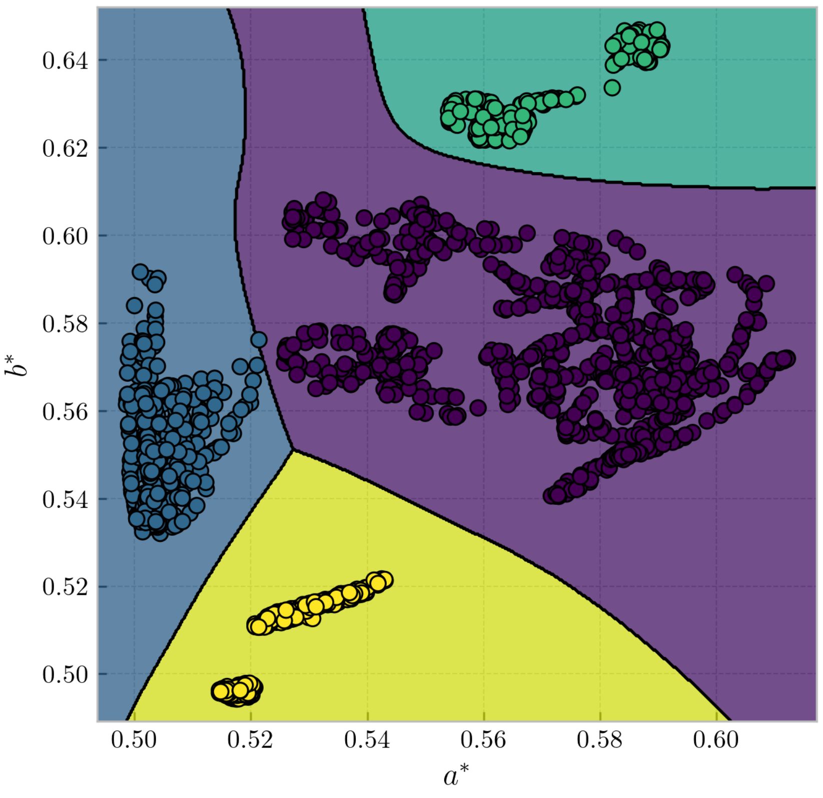


Figure 6: Decision boundaries for bananas, apples, oranges, and cherries in $a^* - b^*$ feature space.

Table 1: Self-evaluation.

Technical correctness	5
Quality of presentation	5
Initiative	2
TOTAL	**2**

References

[1] M. N. Soriano, *A17 – Neural Networks* (2019).

[2] A. LeNail, Publication-ready neural network architecture schematics, *Journal of Open Source Software* **4**, 747 (2019).

[3] H. Muresan and M. Oltean, Fruit recognition from images using deep learning, *Acta Univ. Sapientiae, Informatica* **10**, 26 (2018).