EECE 6036 Intelligent Systems
Homework 2
Benjamin Middleton

Problem 1

Problem Statement

We would like to train a binary threshold neuron to classify images of handwritten zeroes and ones.

System Specification

For this system I decided on 20 epochs with a learning rate η of 0.01. Higher learning rates tested were inconsistent, while lower ones required more epochs (computation cost) with equal results. Less epochs resulted in poorer or less consistent performance, and further epochs tested served only to overtrain with little impact on test data results.

Results

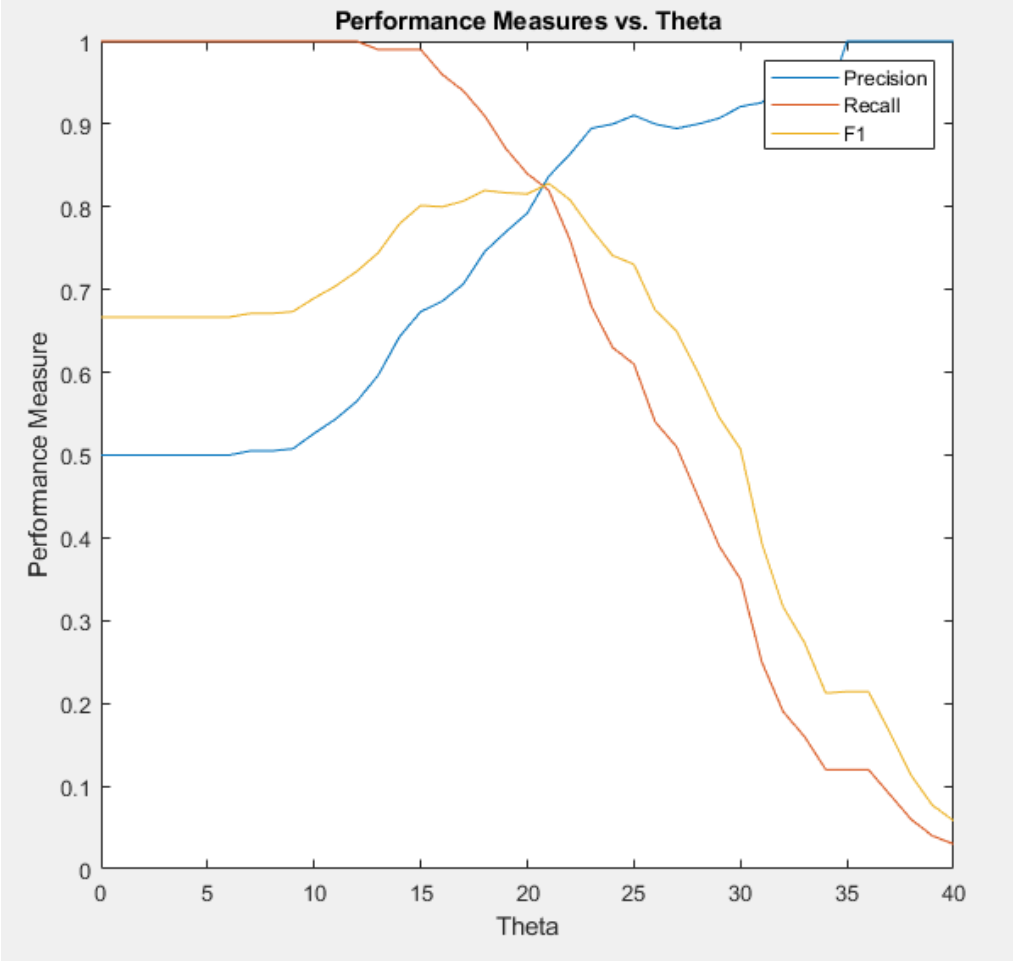

Figure 1.1: *Plots for performance measures precision, recall, and F1-score with varying theta (threshold value). Since there is an equal "cost" for a false positive and a false negative, max(F1-score) was chosen as the optimal theta (in this case theta = 21 and F1 = 0.8283).*

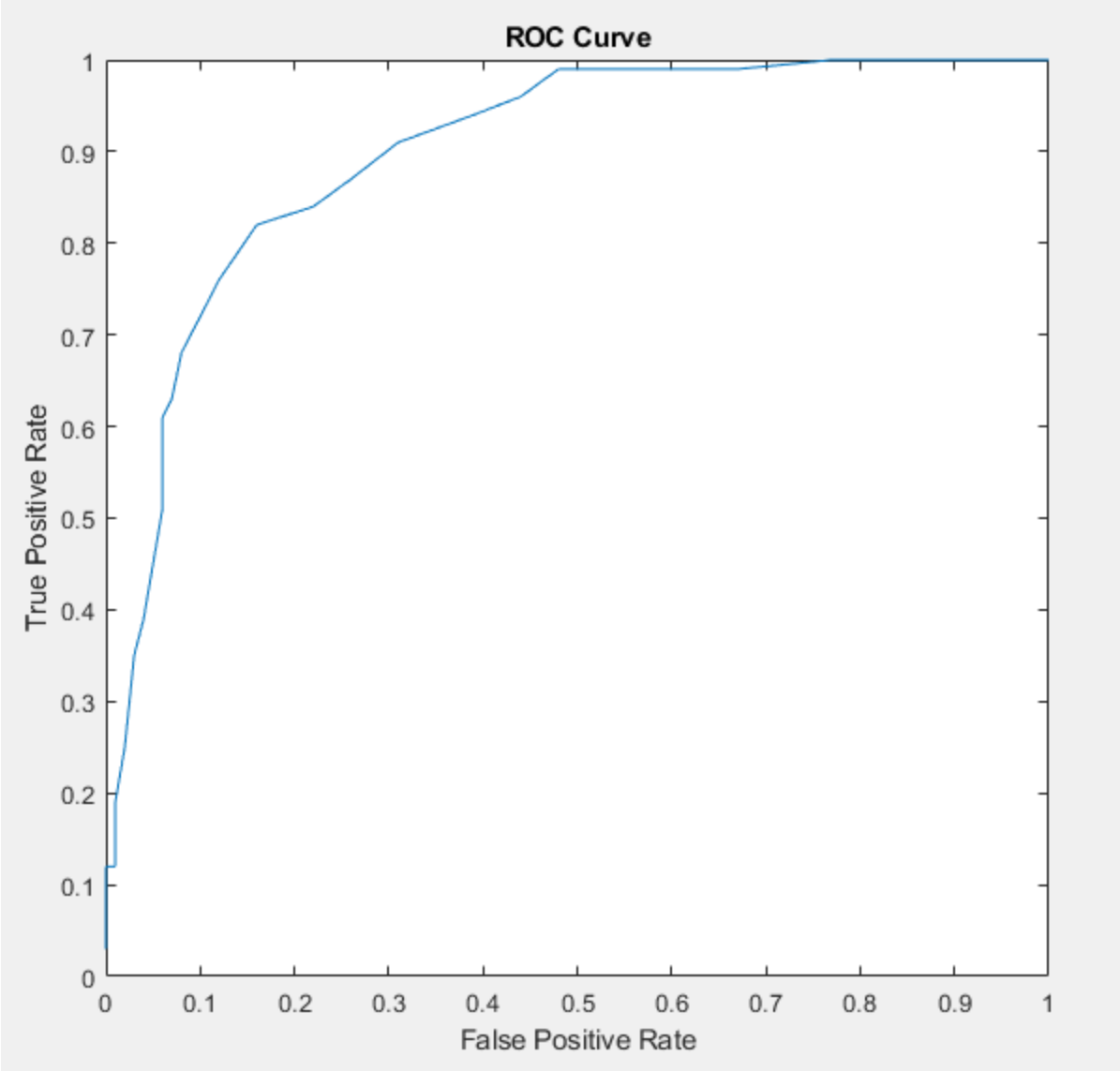

Figure 1.2: *A plot of the ROC curve (true positive rate versus false positive rate) for the various thetas. The significant upwards curve shows that the model has learned well.*

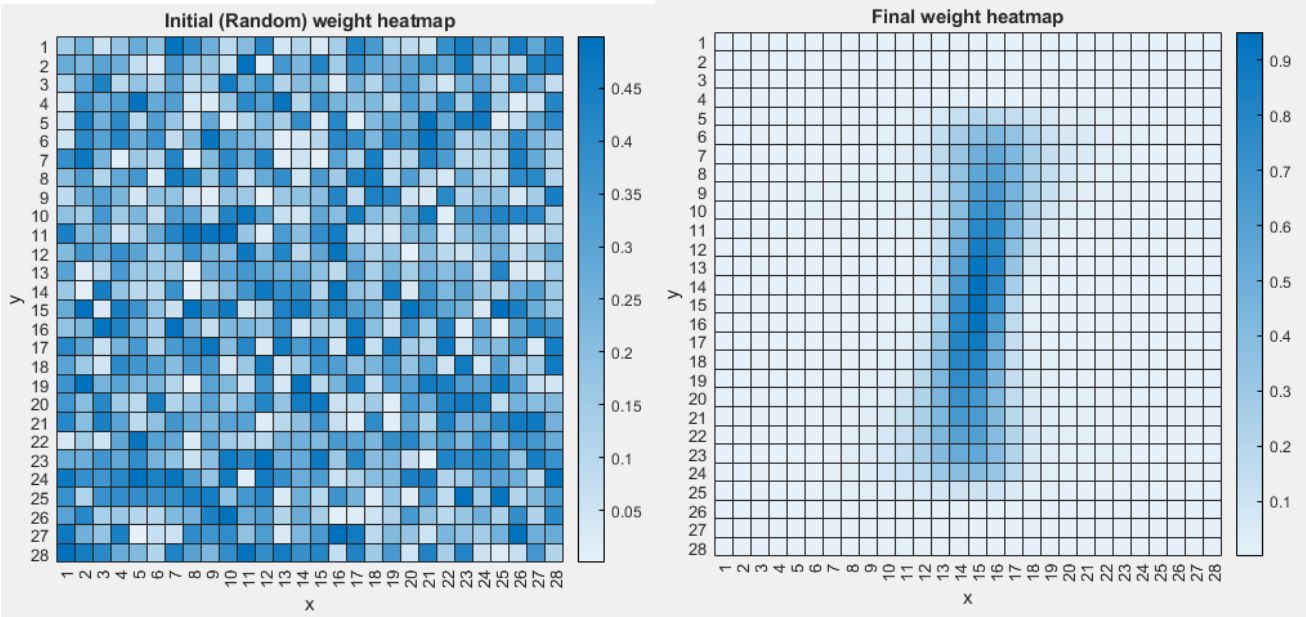

Figure 1.3: *Heatmaps of randomly initialized initial pixel weights (left) and the final pixel weights (right). Note that the left figure scales up to 0.5 while the right figure scales up to 1.0. The right figure shows which pixels the neuron has learned are most commonly darkened in a "1" character.*

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	42	35	83	64	47	68	16	67
1	58	65	17	36	53	32	84	33

Figure 1.4: *Results of having the neuron classify the challenge set (numbers 2-9) as either 0s or 1s.*

Analysis of Results

With the chosen epoch amount and eta value, the neuron consistently learns with an F1-score over 0.8, which shows its consistency in successfully learning. With regards to the challenge results, they make intuitive sense. Since the neuron has only positive weights (which push the neuron to classify the image as a "1"), digits that are mostly classified as ones (3 and 8) have many pixels in the same place as the "1" pixels. It is important to note that the *extra* pixels that are *not* matching with a typical "1" don't detract from the "1" classification. Digits that are mostly classified as zeroes (4 especially, and to some extent 5, 7, and 9) do *not* typically have many markings that match up with the vertical bar "1". Finally, digits that are roughly 50/50 (2 and 6) have only *some* pixels that pass through the vertical middle of the square.

Problem 2

Problem Statement

We would like to train a perceptron to classify the same images of handwritten zeroes and ones.

System Specification

For the perceptron I settled on 10 epochs at learning rate η of 0.01. Many times fewer than 10 epochs were needed, but 10 guaranteed that the perceptron learned the data all the way. Lower learning rates tended to cause overfitting, so this combination allowed the most consistent learning without overfitting. It could be argued that any more than *one* epoch in this dataset is overtraining since the error fraction is so low after only one epoch, but for this small example it does not harm the test error to train more.

Results

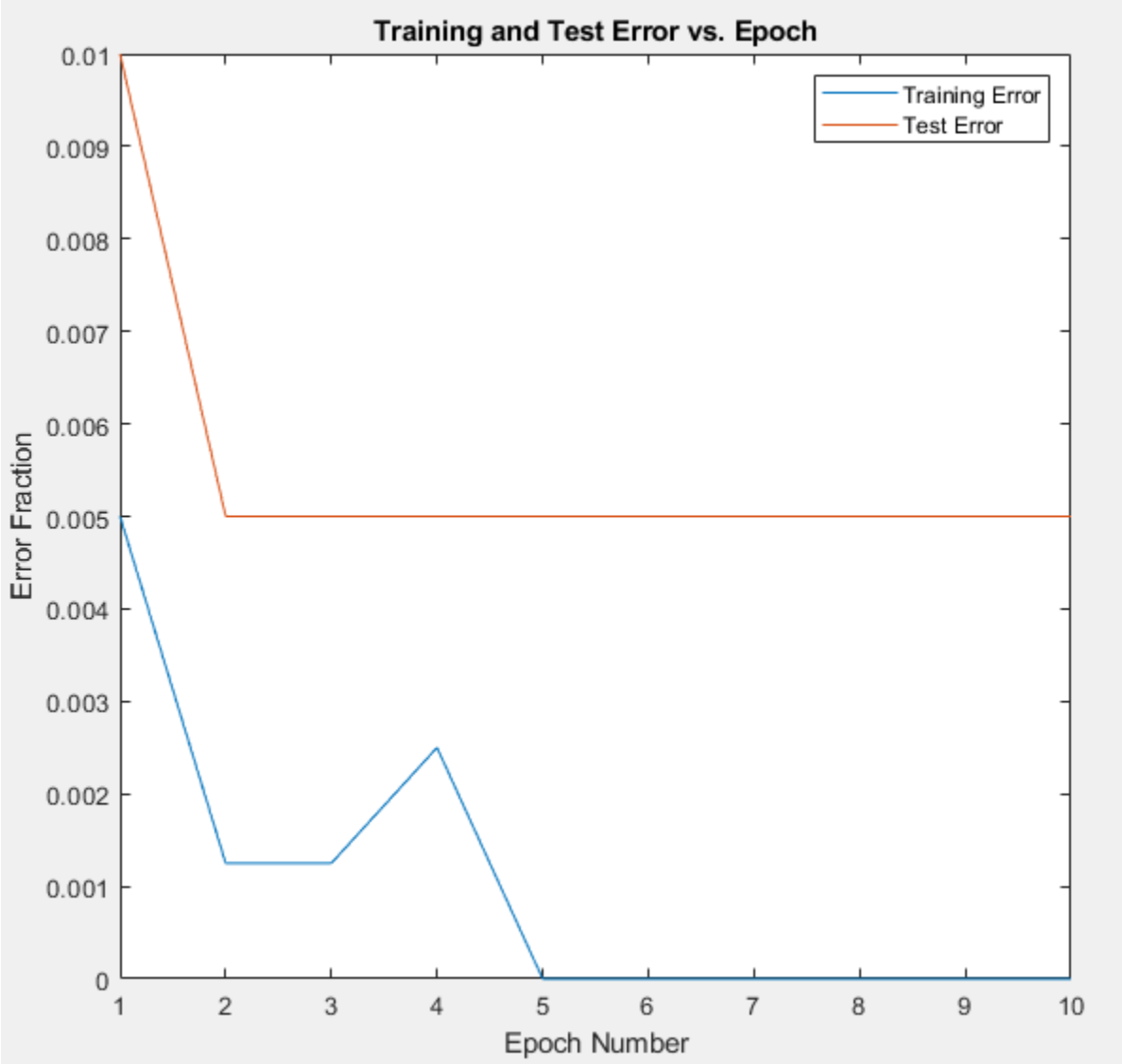

Figure 2.1: *Training and test error at each epoch. The perceptron performed excellently after only one epoch, but did improve slightly after additional epochs.*

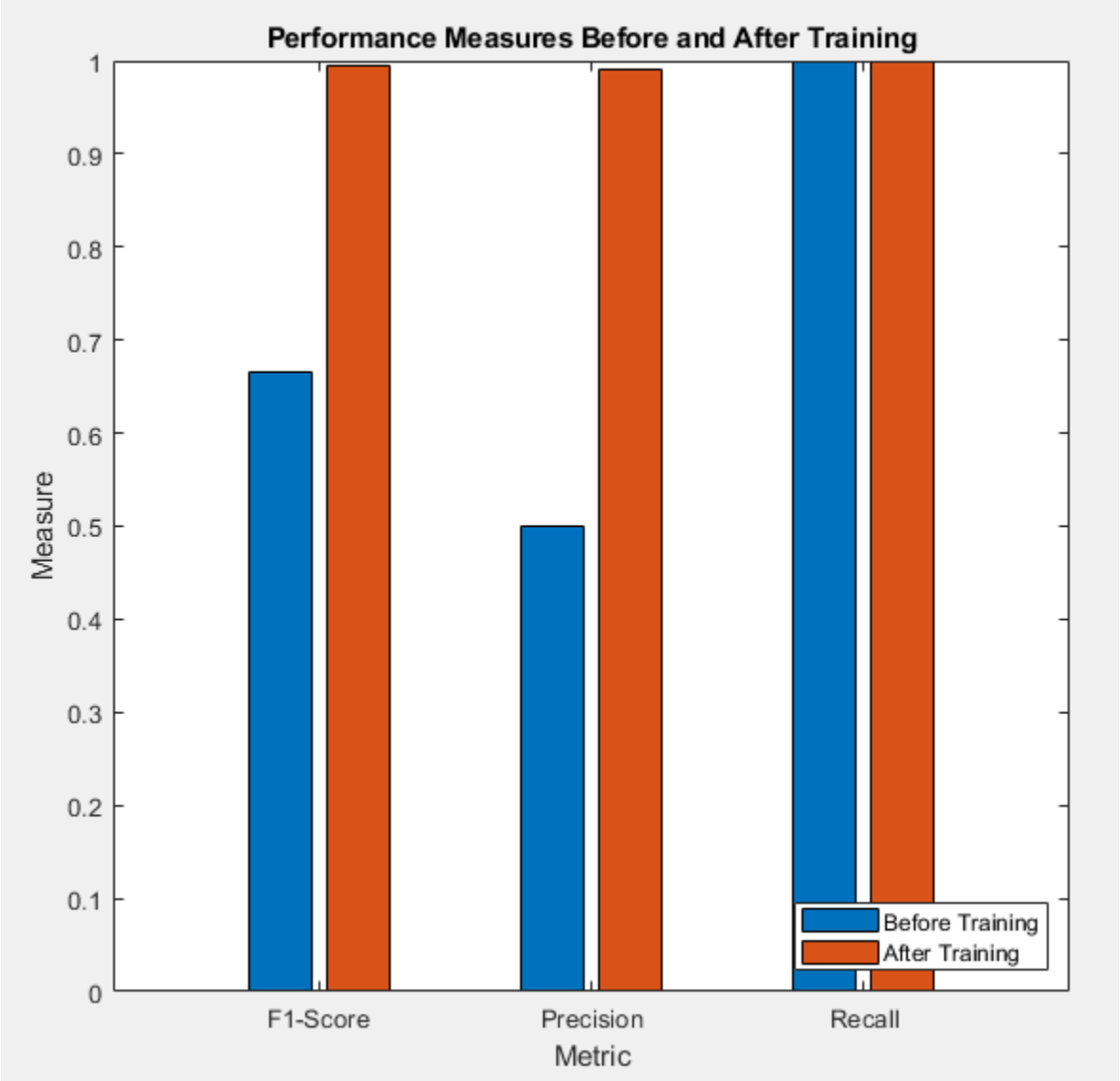

Figure 2.2: *Performance measures of the perceptron before and after training.*

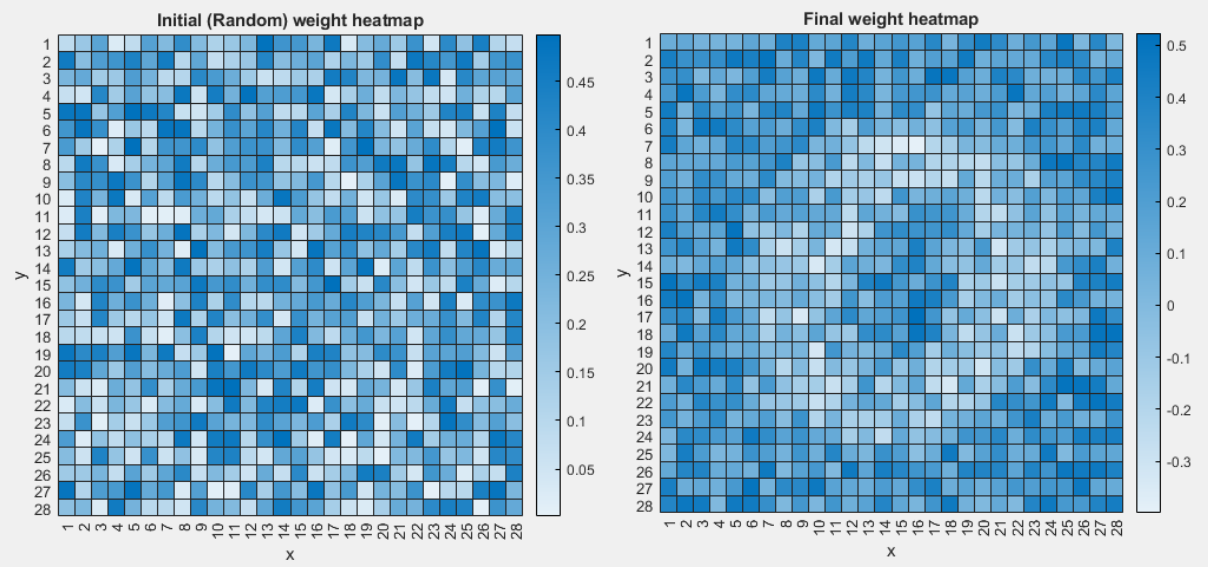

Figure 1.3: *Heatmaps of randomly initialized initial pixel weights (left) and the final pixel weights (right). Note that the left figure scales from 0.0 to 0.5 while the right figure scales from about -0.5 to 0.5. The right figure shows that the perceptron has visibly learned the "0" character due to its ability to assign negative weights.*

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	7	18	5	42	28	7	11	18
1	93	82	95	58	72	93	89	82

Figure 1.4: *Results of having the perceptron classify the challenge set (numbers 2-9) as either 0s or 1s.*

Analysis of Results

The perceptron performed better across the board at this 2-class classification problem, obtaining an F1-score of 0.995 compared to the neuron's score of ~0.82. Additionally, it appeared more certain about many digits in the challenge (which could be looked at as a positive or a negative–perhaps it is good to be unsure of new classes!). However, it is interesting to note that it ended up learning the "0" character more visibly than the "1". This means that the binary threshold's weights neuron would be better for extending this problem to classify every digit, for example. This behavior emerged because of the perceptron's ability to have *negative weights*, and therefore recognize not only the positive class, but also the negative class. From these results, it is clear to me that I would prefer a perceptron for 2-class classification problem, but would prefer several binary threshold neurons for an n-class classification problem.