Gordon Petry
Yuriy Deyneka
Amulya Badineni
Vihan Patel

NOTE: Our group used a learning rate of 0.1.

T5.1

We noticed that learning problems 2 and 3 require one epoch of learning. However, the minimum number of errors made on the training data for those problems is greater than 0. It means that learning problems 2 and 3 are not linearly separable. A problem is linearly separable when a perceptron can classify training examples with 100% accuracy. Learning problem 1 is linearly separable. A perceptron can use the output of -0.44*x_4 + -1.04*x_3 + 0.82*x_2 + 0.26*x_1 + 0.2 to decide whether something is an iris setosa or not.

T5.2

The initial weight vector for a perceptron affects your ability to decide whether a learning problem is linearly separable or not. For task 2, our group could see that learning problem 1 is linearly separable. However, we could not do the same for task 3.1. A perceptron's initial weight vector also affects the amount of training that is needed. For learning problems 2 and 3, the perceptron required at least fifteen epochs of learning for task 3.1 and only one for task 2.

The initial weights of a perceptron were random numbers for task 3.2. Compared to task 3.1, task 3.2 needed a lower number of epochs for learning problems 2 and 3. If the initial weight vector for a perceptron contains random numbers, a perceptron might avoid an unnecessary amount of training.

For task 3.3, our group observed that learning problem 1 is linearly separable. For learning problem 1, the perceptron can use the output of 0.10293*x_4 - 0.51883*x_3 + 0.29447*x_2 + 0.06544*x_1. For task 3.3, w_4 = 0.34293, w_3 = 0.14117, w_2 = 0.23447, and w_1 = 0.44544 for the initial weight vector of the perceptron. Using random numbers for the initial weight vector of a perceptron increases your chances of deciding whether a learning problem is linearly separable or not.

T5.3

We found that learning problem 1 is linearly separable. For learning problem 1, a perceptron can use the output of 0.2 + 0.36*x_1 + 1.9*x_2 - 2.38*x_3 - 1.06*x_4 for task 4.1 and the output of 0.2 + 0.04*x_1+ 0.84*x_2 - 1.22*x_3 - 0.72*x_4 for task 4.2. Less than 4 epochs of learning were needed for learning problems 2 and 3. For perceptron learning, shuffling training data randomly appears to be more effective than using random initial weights.