Sonia Singh // 7/11/2023 // PDP HW#3

1. Please briefly explain why it is the case that the weight from the first input unit (i.e., input:0) to the first output unit (i.e., output:0) equals 0.375. You will also want to discuss why the weight from second input unit (i.e., input:1) to the fifth output unit (i.e., output:4) equals -0.25. To answer both parts of this question effectively, it is important to consider the equation for the Hebb rule, the nature of the training patterns, the learning rate, and the fact that, in using the Delta rule, Lens applies an extra factor of 2 to the weight changes (which are used to update the weights in the network.)

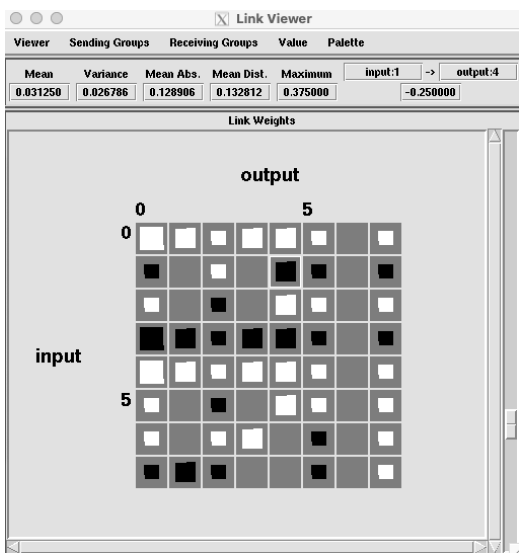
- equation for Hebb rule: $w_{ij} = \epsilon \sum_{p} a_i^{[p]} a_j^{[p]}$

The weight from the first input unit to the first output unit equals 0.375 because as the Hebb rule states, in order to find the weight between a sending and receiving unit, we must look at the activations of those sending and receiving units across all patterns or, in this case, all training examples, "a", "b", and "c". For the first input to the first output, the product of activation energies from training set a is 1; the product of activation energies from training set b is 1; and, the product of activation energies from training set c is 1. 1+1+1=3 which is the sum of activations. Then, all we need is a constant value, epsilon. which we can find by looking at one example. If the weight between units input 0 and output 0 is 0.375, then the equation looks like 0.375 = epsilon (3). Therefore, epsilon is equal to 0.125. (learning rate times factor of 2 -- 0.0625 x 2 = 0.125)

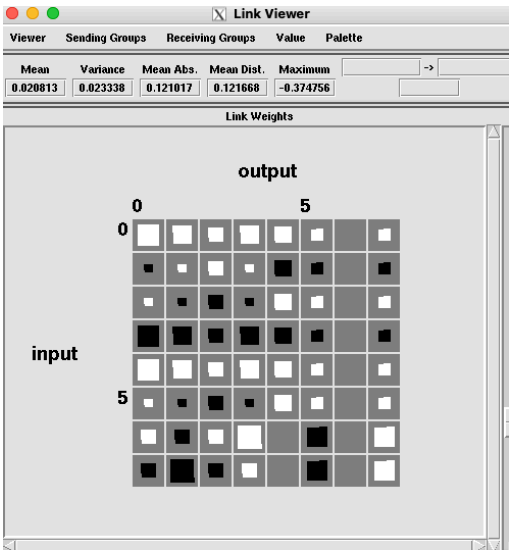
The same logic applies to looking at the connection between the second input unit and the fifth output unit which equals -0.25. The sum of activations comes to (1x-1)+(1x0)+(1x-1)=-2. By multiplying this sum by our epsilon of 0.125, we get -0.25 which is the weight between those units.

2. At this point, if you tried to reapply the Delta rule by re-clicking on “TrainNetwork”, the weights in the network remain unchanged. Why is this the case? Let’s imagine that we trained this network with the Hebb rule. What would have happened if we applied the Hebb rule a second time? (10 points)

The weights in the network remain unchanged because the current weights already satisfy the network in that the target output is the network's output. The difference between the Delta and Hebb rules is that Lens applies an extra factor of 2 to weight changes for the Delta rule; so, if the network was trained using the Hebb rule, it would need to be trained twice and not just once in order to find the correct weights to satisfy the network.



After training for 1 epoch (hebb-linindepen.wt)



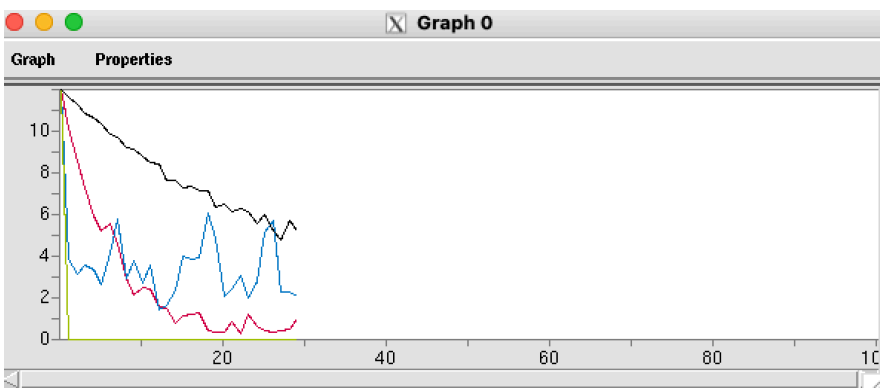
After training for 10 epochs (delta-linindepen.wt)

3. For this question, you will describe (as well as explain) the similarities and differences among the weights produced by the Hebb rule (hebb-linindepen.wt) and those produced by the Delta rule (delta-linindepen.wt) when training on the linearly independent set. (10 points)

The weights produced by the Hebb rule are fairly even in size and come in only 3 values (positive and negative), 0.125, 0.25, and 0.375. In contrast, the weights produced by the Delta rule come in a greater range of numbers, 0.041748, 0.083252, 0.125, 0.166504, 0.208252, and 0.291504. Additionally, many connections between units have a weight of 0 in the Hebb rule diagram, whereas most connections do have some value as a weight in the Delta rule diagram. The connections with a weight of 0 for the Hebb rule generally have a small value for the Delta rule. (patterns a and c reduced differences) (Hebb rule for pattern b) pattern b is orthogonal to a and c; no perfect performance for patterns a and c;

Since the Delta rule seeks to find a set of weights that produce zero error, this rule allows the network to try more specified weights on more of the units. The Hebb rule notably only works perfectly when the set is linearly independent and orthogonal. Since our set is only linearly independent, the network cannot produce zero error while using the Hebb rule. Thus, the weights come in a greater range of values for the Delta rule and there are fewer weights of 0. Also, in the Delta rule, the activation of a receiving unit j cannot equal 0 on subsequent presentations which accounts for why there are fewer connections with a weight of 0 for the Delta rule.

4. You've now trained with high, intermediate, and low learning rates. Why do you think training the network with the learning rate of 0.005 was much more effective than training it with 0.0625 or 0.001. Include the error graph in your response. [A useful hint here is to consider the Delta rule equation.] (10 points).

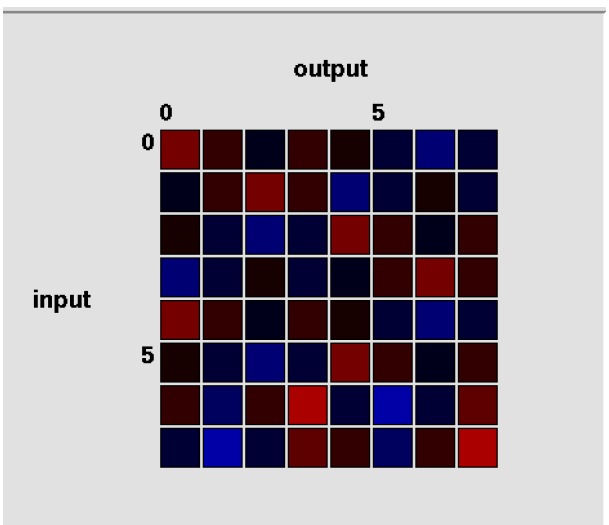


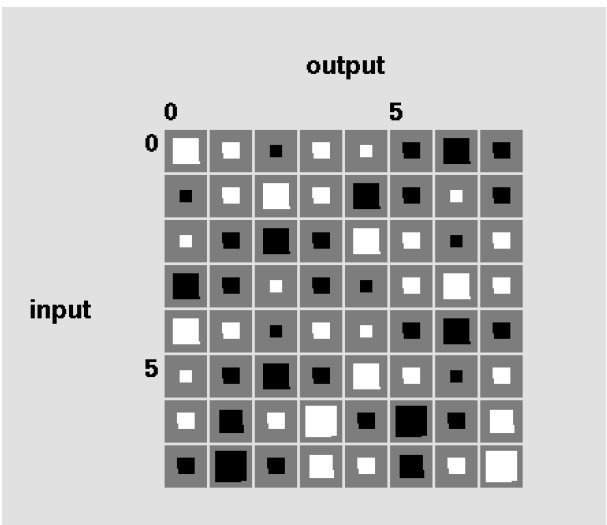
Training the network with the learning rate of 0.005 (red) was much more effective than 0.0625 or 0.001. Since the Delta rule is error correcting, a learning rate of 0.0625 holds greater potential for the weights to jump around, going from high to low in order to converge on the set of weights with the lowest error. In contrast a learning rate of 0.001 jumps around less but goes far more slowly than the other learning rates. Training with the learning rate of 0.005 is more effective since it still jumps

around just enough to converge on the best set of weights, and it is quicker in doing so.

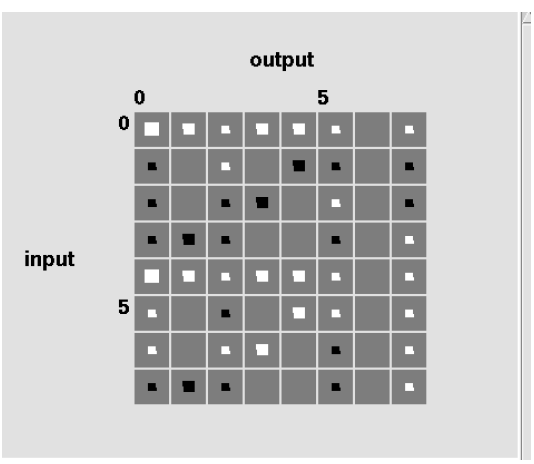
5. Notice that learning is considerably slower when using sigmoidal output units than when using linear output units. Why is that the case? In your response, it is important to refer to the similarities and differences in the resulting set of weights. (10 points)

Sigmoidal Output Units (40 epochs)





Linear Output Units (40 epochs)



In the resulting set of weights for the sigmoidal output units, each connection between an input and output has a weight of some value; whereas, in the set of weights for the linear output units, some connections have a weight of 0. The latter network has already found which connections need weights and which do not. The weights in the linear network are also smaller than in the sigmoidal network. The Delta equations differ between the sigmoidal and linear units.

This is the sigmoidal equation. $$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}} = -\epsilon (t_k^{(p)} - o_k^{(p)}) a_j^p (1 - a_j^p) a_i^p$$

This is the linear equation. $$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}} = -\epsilon (t_k^{(p)} - o_k^{(p)}) a_i^p$$

The sigmoidal equation has an extra factor, the "a(1-a)" (which is the activation of receiving unit j at pattern p multiplied by 1 minus the activation of receiving unit j at pattern p), which causes the network to learn slower compared to the linear equation. The weight change is smaller because this extra term causes the entire equation to become smaller (multiplying by a small fraction). So, the weights changes are slower and smaller.