Simulation Exercise 2

Hebb and Delta Rules

Be concise: Try to stay under a total of about 1200 words in answering all the questions. Note that some questions have multiple parts; be sure to address all of them.

This simulation exercise requires you to demonstrate an understanding of both the Hebb and Delta learning rules in feedforward pattern associator networks, and how these procedures extract regularities from (possibly noisy) examples.

You will need to download the file 8x8.zip and unzip it into your Lens `Examples` subdirectory (or wherever you'd like to run the simulations from). You can find this zip file in the `simulation_exercises` subdirectory within the main course folder. This should give you the following six files: `8x8-imposs.ex, 8x8-li.ex, 8x8-orth.ex, 8x8.tcl, 8x8lin.in,` and `8x8sig.in`. These files define two versions of a pattern associator with 8 inputs and 8 outputs: one with linear units (`8x8lin.in`) and one with sigmoid units (`8x8sig.in`). Three example files are loaded by each of the networks:

- `8x8-orth.ex` (loaded as example set "orth") in which the input patterns are orthogonal
- `8x8-li.ex` (loaded as set "li") in which the input patterns are non-orthogonal but linearly independent
- `8x8-imposs.ex` (loaded as set "imposs") that is impossible to learn.

First, open Lens, click on "Run Script", and select `8x8lin.in` to load the linear version of the pattern associator. You can also navigate to the directory that contains this file, open up the command line or terminal and type `lens 8x8lin.in`. The Link Viewer and a graph that plots the network error will open automatically. Note that the training set is initially the orthogonal examples ("orth") and that the weights in the network are all initialized to 0.0. Thus, if you open the Unit Viewer and click on each of the three examples, you'll see that the output of each unit for each example is 0.0. This is because, for linear units, the activation of a unit is equal to its net input, which is 0.0 if all of its weights are 0.0. Note that, if you move your mouse over the units, the input activations range from -1 to 1 whereas the targets of the output units are 0 or 1. You can reset the weights back to all zeros at any time by clicking on "Reset Network" on the main panel.

Now click on "Train Network". Because "Weight Updates" is set to 1, this will train the network on one presentation of each of the three training examples (called an "epoch"). For the purposes of this simulation exercise, you can consider this specific situation to be equivalent to applying the Hebb rule using each example. [In actuality, the network is training with the Delta rule but, as we have discussed in class, the result is identical to that for the Hebb rule when training for one presentation of orthogonal patterns starting with zero weights.] Note that, after training, the weights now have a range of values.

Q1. Explain why the weight from `input:0` to `output:0` is equal to 0.375, and why the weight from `input:1` to `output:4` is equal to -0.25. To do this, you will have to consider the Hebb rule equation, the training patterns, the learning rate, and the fact that Lens applies an extra factor of 2.0 to the weight changes (from the derivative of the error function).

Save these weights by clicking on "Save Weights" and **replacing** the Selection field with `hebborth.wt`. The file will be saved into the current directory (shown in the top-left panel). [Note: Recall that Lens under Windows doesn't handle paths or filenames with spaces well. If you get an error when attempting to save a weight file, it may be because you didn't remove the full path from the Selection field and it contains one or more spaces (e.g., `C:\Documents and Settings\Name\Lens\hebborth.wt`).] You will also need to save the Link Viewer display to a file to include in your submission [either take a screen shot or print to a ".ps" file, as in Simulation Exercise 1.]

Q2. If, at this point, you apply the Delta rule by clicking on "Train Network", the weights remain unchanged. Why? What would have happened if the Hebb rule had been applied a second time instead?

Now click on "Reset Network" to reset the weights to zero, and switch to training on the linearly independent patterns by clicking on "Training Set" at the top of the main panel and selecting "li". Then train for 1 epoch, save the weights as `hebb-li.wt`, and save a display of them (using the Hinton Diagram palette). [Note that, even though the patterns are not all orthogonal, the weights after 1 epoch are still equivalent to Hebbian learning in this case because Lens is not updating the weights after each example, but only after all three examples are run.] Now continue training for 9 more epochs (total of 10), save the weights as `delta-li.wt`, and save a display of them.

Q3. Describe and explain the similarities and differences among the weights produced by the Hebb rule (`hebb-li.wt`) and those produced by the Delta rule (`delta-li.wt`) when training on the linearly independent set.

Reset the network and set "Weight Updates" to 30. [Remember to hit "Enter" after changing any value in the Lens interface, so the field turns from pink back to gray---otherwise the value won't actually be changed.] Also reset the error graph ("Graph 0") by selecting "Clear" under the "Graph" menu of the graph. Now click "Train Network" to train the network for 30 epochs and note that the network learns perfectly with this learning rate.

Reset the network again, but now run

```
lens> noiseOn
```

from the command line. This **adds noise to both the inputs and targets** when each example is presented. [You can run `noiseOff` to turn the noise off.] Then train again for 30 epochs. You'll see (in the error graph) that the error jumps around wildly over the course of training. Now reset the network, set the learning rate to 0.005, and retrain the network for 30 epochs. Note that learning is now much more effective. Finally, reset the network one more time, set the learning rate to 0.001, and retrain for 30 epochs.

Q4. Why was training with the intermediate learning rate more effective than either the higher or lower rate? Include the error graph in your response. [Hint: consider the Delta rule equation.]

Now load the version of the pattern associator with sigmoid units by clicking on "Run Script" and selecting `8x8sig.in`. This creates a new network with zero weights and "li" as the training set. Train this network for 40 epochs. Save the resulting weights as `delta-li-sig.wt` and save a display of them.

Q5. Why is learning so much slower using sigmoid units than when using linear units? Describe and explain the similarities and differences in the resulting sets of weights.

Finally, reset the network and switch to training on the "imposs" set. Train for 40 epochs, print a display of the weights, and save them as `delta-imposs-sig.wt`.

Q6. Explain why learning fails here, even with the Delta rule. To answer this, you will have to examine the patterns carefully, noting that some input values are "redundant" with each other—that is, they provide no additional information—and also the relationship of these input values to the targets.

In addition to your written answers to the questions above, please **include** an image of the Link Viewer diagrams (using the "Hinton Diagram" palette) for `hebb-orth.wt, hebb-li.wt, delta-li.wt, delta-li-sig.wt`, and `delta-imposs-sig.wt`, as well as the error graph from Q4.