6.034 Artificial Intelligence, Fall 2006
Prof. Patrick H. Winston

Problem Set 4b

This problem set is due Tuesday, November 14th at 11:59 PM. If you have questions about it, ask the TA email list. Your response will probably come from a TA.

To work on this problem set, you will need to get the code, much like you did for earlier problem sets.

The code can be downloaded from the assignments section.

Your answers for the problem set belong in the main file `ps4b.scm`.

Things to remember

- Avoid using DrScheme's graphical comment boxes. If you do, take them out or use *Save definitions as text...* so that you submit a Scheme file in plain text.
- If you are going to submit your pset late, see the problem set grading policy.

1. Neural Nets

In this problem set, you will be experimenting with neural nets. Although you'll only have to do the most superficial of coding, take your time and make sure you really understand what is happening when you run the nets.

1.1. Learning XOR

The first function the neural net will be learning is XOR. XOR is a binary function that returns a 1 when exactly one of its two inputs is 1. The XOR table looks like this:

```
A B XOR(A, B)
1 1    0
1 0    1
0 1    1
0 0    0
```

To do this, we are using a net with 2 input neurons, 2 hidden (internal) neurons, and 1 output neuron. Take a look near the bottom of nnet-train.scm to find the procedures you will use in training the XOR function. In order to set up the neural net for XOR, call `(initialize-xor learning-rate)` , which readies the neural net to learn XOR, at the specified learning rate `learning-rate`.

Now that the neural net is set up, it can be trained. Training the neural net is achieved by calling `(train-xor epochs target-error)` . This trains the neural net on the XOR data for `epochs` epochs (an epoch is a complete cycle through the training data), or until the average error is reduced to less than `target-error`.

1.1.1. Training 1

Initialize the XOR net with a learning rate of 0.3. Then train it in increments of 1000 epochs. Stop training when you have completed 10000 epochs or the average error stops decreasing (changed by less than .01 in the last 1000 epochs), or if the average error is less than 0.1. Do this a few times (remember to re-initialize to reset the weights) to make sure your results are consistent and report the average error at the end of training, and the average number of epochs you trained for.

1.1.2. Training 2

Initialize the XOR net with a learning rate of 0.6. Then train it in increments of 1000 epochs. Stop training when you have completed 10000 epochs or the average error stops decreasing (changed by less than .01 in the last 1000 epochs), or if the average error is less than 0.1. Do this a few times (remember to re-initialize to reset the weights) to make sure your results are consistent and report the average error at the end of training, and the average number of epochs you trained for.

1.1.3. Training 3

Initialize the XOR net with a learning rate of 1.0. Then train it in increments of 1000 epochs. Stop training when you have completed 10000 epochs or the average error stops decreasing (changed by less than .01 in the last 1000 epochs), or if the average error is less than 0.1. Do this a few times (remember to re-initialize to reset the weights) to make sure your results are consistent and report the average error at the end of training, and the average number of epochs you trained for.

1.2. A new data set

Now we'll try using a neural net for classification, using the dataset in nnet.data. To do this, we are using a net with 2 input neurons, some hidden (internal) neurons, and 1 output neuron. You will be experimenting with changing the number of hidden neurons and its effect on the training process. Take a look near the bottom of nnet-train.scm to find the procedures you will use in training the classifier function. In order to set up the neural net, call `(initialize-classifier-net n-hidden learning-rate)` , which readies the neural net for training, at the specified learning rate `learning-rate`, with `n-hidden` hidden neurons.

Now that the neural net is set up, it can be trained. Training the neural net is achieved by calling `(train-classifier-net epochs target-error)`. This trains the neural net on the data for `epochs` epochs (an epoch is a complete cycle through the training data), or until the average error is reduced to less than `target-error`.

1.2.1. Two hidden nodes

Initialize the classifier net with a learning rate of 0.5 and 2 hidden neurons. Then train it in increments of 1000 epochs. Stop training when you have completed 10000 epochs or the average error stops decreasing (changed by less than .01 in the last 1000 epochs), or if the average error is less than 0.1. Do this a few times (remember to re-initialize to reset the weights) to make sure your results are consistent and report the average error at the end of training, and the average number of epochs you trained for.

When you're done training, call `(validate-classifier-net)` to test your classifier on the dataset. Report the number of correct classifications (out of the possible 16).

1.2.2. More hidden nodes

Initialize the classifier net with a learning rate of 0.5 and 4 hidden neurons. Then train it in increments of 1000 epochs. Stop training when you have completed 10000 epochs or the average error stops decreasing (changed by less than .01 in the last 1000 epochs), or if the average error is less than 0.1. Do this a few times (remember to re-initialize to reset the weights) to make sure your results are consistent and report the average error at the end of training, and the average number of epochs you trained for.

When you're done training, call `(validate-classifier-net)` to test your classifier on the dataset. Report the number of correct classifications (out of the possible 16).

2. Survey

Please answer these questions at the bottom of your `ps4b.scm` file:

- How many hours did this problem set take?
- Which parts of this problem set, if any, did you find interesting?
- Which parts of this problem set, if any, did you find boring or tedious?

(We'd ask which parts you find confusing, but if you're confused you should really ask a TA.)

When you're done, **run the tester** and submit your .scm files to your `6.034-psets/ps4b` directory on Athena. (This directory won't exist already -- you need to create it.) If the tester dies with an error when it's run on your code, or if it stops and waits for user input, your submission will not count.

3. Errata

3.1. Clarification on when to stop training

In the specification, it is somewhat unclear when you should stop training. We would like you to report the result when your neural net has reached a steady state. In the case of XOR, the neural net often gets off to a slow start and oscillates around the starting error for a few 100s of epochs, but eventually it will improve. Continue training during these initial oscillations, and report the state after the error converges to a lower value.

3.2. Submitting your code

Be sure to create a directory called `6.034-psets/ps4b` and put your code there, because this directory won't exist already. The directory will have the right permissions automatically because it's under `6.034-psets`.