Massachusetts Institute of Technology

6.038 Artificial Intelligence in Practice

Problem Set 1B: Thursday, February 5, 1998
DUE: Tuesday, February 10, 1998

1 Background

The objective of this assignment is to give you experience in using machine learning systems.

Much of this problem set (both Part A and B) is based on one developed by Tom Mitchell and Avrim Blum (of CMU) as part of their course in Machine Learning. The problem is also featured in the textbook by Tom Mitchell "Machine Learning" published by McGraw Hill, which is a good introduction to the field of machine learning.

We have appended the problem set statement from CMU since it contains key information on the code and the data (as well as some interesting project suggestions).

2 Files

You can find most of the software and data required for this assignment in `/mit/6.038/ps1/` on Athena. Let us know (soon) if you have any problems running the code.

To view image files, you can use the `xv` program. On Athena, you need to do `add graphics` before running `xv`.

In the `Makefile` for the Neural Net code, we have chosen `gcc` as the compiler, so you will need to do `add gcc` before compiling, if you don't already.

For the Identification Tree part of the assignment, you will need to use c4.5 which is installed in the 6.038 locker (for Sparcs and Sgi's). So, you need to `add 6.038` to get that. If you want to use a Linux machine, the source for the system is in `/mit/6.038/c4.5/Src/`, you can copy it and compile it on your machine. You should, however, not distribute it elsewhere.

3 Neural Nets

Read the attached "Homework 5" handout which describes the face data and code that you will be working with in this assignment. You should do questions 5 – 13 in Part I of that problem set. Note that the command strings given assume that you have compiled `facetrain` in the current directory and you have done `cp /mit/6.038/ps1/trainset/*.list .`

We have compiled the code (for Sparcs) in the 6.038 locker `/mit/6.038/faces/src/`. You will need to modify the files `facetrain.c` and `imagenet.c` for this assignment. If you can afford the space, copy all the source files (`*.c` and `*.h`) to your locker and compile there. If you're really tight on space, copy only the files you need to modify and make links the the other files.

Note that what you are being asked to do is primarily to change the structure of the neural net, the performance function and the training data. You do not need to modify the actual backprop training code itself.

4 Identification Trees

The data files you will need to run c4.5 (an identification tree builder) are in `/mit/6.038/ps1/trainset/c4.5-trainset/`. We have provided you with three datasets:

- `straighteven_userid.names`, `straighteven_userid.data`, `straighteven_userid.test`: This corresponds to the training data and the test2 data used in the experiment of question 6 in the CMU assignment.
- `all_userid.names`, `all_userid.data`, `all_userid.test`: This corresponds to the training data and the test2 data used in the experiment of question 6 in the CMU assignment, but using all the data.
- `all_pose.names`, `all_pose.data`, `all_pose.test`: This corresponds to the training data and the test2 data used in the experiment of question 10 in the CMU assignment.

Unfortunately, c4.5 insists on writing into the directory where the data is, so you will need to copy the c4.5 data files to your locker so as to run c4.5.

We have attached a description of c4.5 as well as the instructions for running it. A typical run for c4.5 is: `c4.5 -f all_pose -u`
This will run the system using `all_pose.data` for training and `all_pose.test` for testing (that's what the `-u` says). Running `c4.5rules` will operate on the tree produced by `c4.5` and construct rules and use those for classification. You should try that as well (it's very fast).

1. You should repeat the two Neural Net experiments you did for the last part using the c4.5 identificaction tree builder.
2. You should experiment with different values for the pruning options for c4.5, in particular (`-m` and `-c`) and describe their impact on the tree that is produced and the resulting performance. Can you rationalize the behavior you see?
3. Pick a (shallow) branch in the 1-of-20 tree that leads to identifying some individual, look at the pictures of that individual (using `xv` to look at the pixel values) and see if you can make sense of the tests that are being used to do the classification.

5 Reflection

Some thought questions. Please respond briefly.

1. You should have seen that both systems work pretty well on this problem. Why is that?? Make a list of the "hidden" assumptions that went into producing the data set so that the behavior would be so good.
2. Consider the problem of finding faces in an arbitrary image, that is, detecting that some piece of the image corresponds to a face (any face, not limited to one in a known set). How does that problem compare to the one you've done in this problem set? What are some of the new challenges posed by this problem.