CSC 2541, Assignment #3, due in class on December 8. Worth 16% of the mark.

For this assignment, you will fit Bayesian neural network models with several architectures to data from a project I am working on with Lev Tarasov and Richard Peltier in the physics department.

The data concerns the output of a complex simulation of glaciation in North America over the last 250,000 years. The aim of the project is to reconstruct the history of glaciation over that time period. Many of the parameters of the simulator are unknown, however. These parameters control the dynamics of ice movement, aspects of the assumed climate over this time period, etc. These parameters need to be estimated on the basis of how well the simulated history of glaciation using given parameter values matches the available data on past glaciation. Primarily, the data concerns the ages of fossilized mussels found at various elevations above present sea level, from which one can infer something about how much ice was present, pushing the land down and leading to an apparent rise in sea level. For this assignment, however, we will look at another constraint on the simulator parameters, which is that they should not produce ice sheets whose thickness is less than is plausible.

In particular, for given values of the 23 simulator parameters, the simulator may or may not produce an ice sheet 25,000 years ago that is at least 200m thick over Hudson's Bay. If it doesn't, we want to discard those parameters as possible values. There is a similar constraint to eliminate parameter values that produce a hole in the South-East part of the ice sheet.

Neural networks come into the project because the simulator takes quite a long time to run — several hours on a vector supercomputer. Because of this, it's not feasible to explore possible parameters values by just running the simulator for each set of parameter values that are considered. Instead, we use past runs of of the simulator to train neural networks that predict what the simulator would do if run with given parameters. The predictions aren't going to be perfect, of course, but the hope is that they will be good enough to choose some good sets of parameters, which can then be checked using a relatively small number of actual simulator runs.

We therefore need a neural network model that will predict whether or not the Hudson's Bay constraint will be violated, given 23 parameter values as inputs, and another neural network that will predict whether or not the SE ice constraint will be violated. For this project, it is important that the model output the probability of the constraint being violated, not just a 0/1 prediction. For purposes of this assignment, I have divided the complete set of past runs into a training set of 1708 cases and a test set of 2562 cases. You should try training on the training set, and then see how well the model does on the test set. You should evaluate predictive performance primarily by the average log probability of the correct value, though you can look at average classification error and average squared error as well.

The data is available from the web page, and in the following files in /u/radford/course/csc2541 on the CSLAB and UTSTAT computer systems:

xdata.trn	Inputs for training cases
xdata.tst	Inputs for test cases
hdata.trn	Hudson's Bay indicator for training cases
hdata.tst	Hudson's Bay indicator for test cases
sdata.trn	SE ice indicator for training cases
sdata.tst	SE ice indicator for test cases

Cases are stored one per line, with each line of the xdata files containing 23 numbers, and each line of the hdata and sdata files containing one number (0 or 1).

You should fit four neural network models for each of the two prediction tasks:

1. A linear model (no hidden units)
2. A model with one hidden layer, of 5 units, with Gaussian priors
3. A model with one hidden layer, of 25 units, with Gaussian priors
4. A model with two hidden layers, of 25 units and 8 units, with Gaussian priors except for the weights on connections from the last hidden layer to the output, which should have a Cauchy prior

You should use an Automatic Relevance Determination prior for the input-to-hidden weights, to allow the model to discover which of the 23 parameters are most relevant to the prediction task. For this to work best, it may be desirable to centre the inputs to have mean zero and scale them to have standard deviation one. This can be done using a data-spec command such as the following:

```
data-spec log 23 1 2 / xdata.trn hdata.trn xdata.tst hdata.tst / +@x@ ...
```

You can play around yourself with other aspects of the models, such as whether there are direct connections from the inputs to the output and from the inputs to the second hidden layer, and whether the first hidden layer connects directly to the output when there is a second hidden layer.

You should use hybrid Monte Carlo to update the weights in the MCMC runs, along with the sample-sigmas operation to update hyperparameters. You may find that fixing the hyperparameters at reasonable values for the first few iterations helps get things started, as in the examples in the software documentation. You should expect that MCMC runs for the larger models will take all night to converge and then move around enough to produce a good sample from the posterior.

Hand in any command scripts and programs you used, along with some informative output and plots (eg, of ARD hyperparameters), and your discussion of which models worked best.