APPENDIX B

Interface modules
MODIFICATIONS made to CNNS (Cascade, v.1.0 by Matt White) to interface w/NNES by -rrenner 8/6/98

General:
- All modifications were made in C++
- Interface from NNES to CNNS is created by generating script files from within NNES "cc.cxx", and invoking Cascade with 'system' calls from "main.cxx".
- Interface from CNNS back to NNES is created by generating temporary data, stats, output, and prediction files from "pass2nnes.cxx", which is invoked from within modified cascade source code".

Specifics:

Modifications to "cascade.h"

- add #include "pass2nnes.h"
- add #include <fstream.h>
- add #include <new.h>
- add #include <stdarg.h>
- add #include <stdlib.h>
- add #include <stddef.h>
- add #include <unistd.h>
- add prototype "boolean del_net(char *netName);"
- add prototype "boolean del_data_file(char *filename);"
- add prototype "void free_net(net_t **net);"

Modifications to "cascade.c"

- add to function "train_net":

    //write results to "stats.tmp" and "data.tmp"
    write_stats_tmp(result,net,'r');
    write_data_tmp(dFile,'s');

- add to function "predict":

    ofstream outfile;
    write_data_tmp(outfile,cNet->outValues,cNet->Noutputs,dSet->Npts,i);
    write_data_tmp(dFile,'p');

Creation of "pass2nnes.h" and "pass2nnes.c" provides interface and implementation of functions writing temporary data from CNNS to NNES, via files (data.tmp, outputs.tmp, pred.tmp, stats.tmp)

COMPILATION:

Compile and install Cascade and its associated libraries (toolkit,parse), with C++ compiler. Install cascade executable in nnes/bin. Cascade (CNNS) can now be run independently or invoked with a 'system' call from within NNES.

NOTE:

Files written by "pass2nnes" are temporary. They will be deleted or overwritten by NNES when it has finished reading them.
#!/usr/local/bin/perl
#
rrenner 9/30/98
#
# Encodes proben formatted data into CNNS format
# - remove header info from proben file first -
# - add header info for datA and datB files after -
#
# this particular script encodes the "diabetes" data files
# - with 8 inputs and 2 outputs (ignoring the last output)
# usage: proben2cc inputfile >outputfile

while (<>) {
  @F = split (" ");
  $v = $F[0];
  printf ("%g, ", $v);
  $v = $F[1];
  printf ("%g, ", $v);
  $v = $F[2];
  printf ("%g, ", $v);
  $v = $F[3];
  printf ("%g, ", $v);
  $v = $F[4];
  printf ("%g, ", $v);
  $v = $F[5];
  printf ("%g, ", $v);
  $v = $F[6];
  printf ("%g, ", $v);
  $v = $F[7];
  printf ("%g, ", $v);
  $v = $F[8];
  printf ("%s\n", $v == 0 ? "-;" : "+;" );
}