```
/* ------------------------------------------------------------------------- */
/*
  Title: LNN, Neural Network execution code.
  Automatically generated by SNN, Tue Oct 13 19:04:10 2009


  License Agreement:
  -----------------

  Copyright StatSoft Inc., 2000-2001, all rights reserved.
  This source code (Source Code Generated by STATISTICA Neural Networks,
  referred to as "CG" below) is owned by StatSoft Inc. and is protected
  by United States Copyright laws and international treaty provisions. You
  shall treat the CG like any copyrighted material.
  The CG may not be redistributed or used except in accordance with the
  conditions below.

  The licensee is granted a license to incorporate the CG as embedded
  software in their own hardware and software products, and to distribute an
  unlimited number of such embedded copies as part of this license subject to
  obtaining prior written consent from StatSoft Inc., and subject to
  the conditions listed below.

  Prior consent is required so that StatSoft Inc. can ensure that license
  conditions are not breached, and can track legitimate use of the CG. Consent
  shall not be refused unless StatSoft Inc. reasonably believes that a
  breach of license conditions will occur. Consent shall usually be granted
  within five working days of the request, providing that sufficient details of
  the intended use are given.

  Requests should be sent to SNN Project Director, StatSoft, Inc.,
  2300 East 14th Street, Tulsa OK 74104 USA, FAX: 918-749-2217,
  E-Mail: info@statsoft.com.

  The licensee may modify the CG as they see fit for embedded use, including
  recoding into alternative programming languages, altering the neural network
  architecture and weights, and otherwise modifying the CG, provided that they
  keep intact this copyright and license notice.

  The licensee may distribute products including the compiled version of CG.

  The licensee shall not:-
    Sublicense, rent, lease, or assign any portion of the CG to third parties.
    Allow compiled versions of the CG to be incorporated in products owned
    by third parties.
    Allow access to the CG to third parties.
    Use (implicitly or explicitly) any reference to StatSoft, Inc., STATISTICA,
    STATISTICA Neural Networks, or any trade names used by StatSoft, Inc.
    to describe, promote, or reference products in which CG is used, or which
    benefit from CG.

  Except as expressly stated herein, the CG is provided "AS IS." The licensee
  shall be entirely responsible for the selection of the CG and for the
  installation, integration, use of, and results obtained from, the CG.
  In particular, but without limitation, attention is drawn to the issue of
  "limited numeric accuracy," which implies that results may not be identical to
  those when executing the same network in STATISTICA Neural Networks or through
  its Application Programming Interface.

  All other warranties or conditions, either express or implied, including but
  not limited to implied warranties of merchantability or fitness for a
  particular purpose, with respect to the CG and written information
  accompanying the CG, are excluded from the license.

  No liability for Consequential Damages. To the maximum extent permitted by
  applicable law, in no event shall StatSoft Inc.,
  or the vendor be liable for any damages whatsoever
  (including, without limitation, damages for loss of profits,
  business interruption, loss of information, or any other pecuniary loss)
  arising out of the use or inability to use this product,
  even if the vendor has been advised of the possibility of
  such damages.

  This license and your right to use the CG shall terminate automatically
  if StatSoft, Inc. determines that you violate any part of the agreement
  or if you violate any part of this agreement without the knowledge of
  StatSoft, Inc. In the event of termination, you shall immediately destroy
  all copies of the CG.

  This agreement constitutes the entire agreement between you and StatSoft Inc.,
  and supersedes any prior agreement concerning the CG. It shall
  not be modified except by written agreement dated subsequent to the date of
  this agreement signed by an authorized representative of StatSoft Inc.
  StatSoft Inc. shall not be bound by any provision of any purchase order,
  confirmation, correspondence, or otherwise, unless StatSoft Inc.
  specifically agrees to the provision in writing.

  This agreement shall be considered as a contract made in the United States
  of America and according to United States Law,
  subject to the exclusive jurisdiction of the United States Courts.
*/
/* standard includes. math.h needed for exp() function. */

#include <stdio.h>
```

```
#include <math.h>
#include <string.h>
#include <stdlib.h>

#ifndef FALSE
#define FALSE 0
#define TRUE 1
#endif

#define MENUCODE -999


static double LNN10Thresholds[] =
{

/* layer 1 */
988122608754.84961

};

static double LNN10Weights[] =
{

/* layer 1 */
1164052437210.9614, -346500758243.96606, -1168388519703.0415, -
20328175742039.75,
1230977599017.5466, 20592726259760.586, 8518253104937.4355, 29511139922940.746,
-8666136497804.9941, -21571446426455, -20755266968694.52, 22467947374345.574,
509148729360.35962, 31758740026970.699, -525640550997.6864, 27359205868888.996,
-26157596607672.91, -18915029472615.629, -32863361174494.145,
2290996953994.1509,
40254284369015.266, 7709218517397.2549, -20229131000693.039, -
1165548967898.9883,
9881382613614.4121, 9692350272659.3906, 3312610953957.6768, -
5508157462925.8398,
40073373962161.594, -30063598238118.73, -9328873804630.123, -34330104188400.258,
2584471446829.5557, -25556822051776.695, 19852808255348.23, 19652665915051.309,
5049565239531.0391, -8961808643065.6367, -10279397898093.129, -
19606764656779.871,
-18882943375109.695, 9061260682959.0625, 7279352143087.4609, -
45359893049357.148,
21148208649973.398, -19842288635159.266, 17625966674227.687,
22092814787008.211,
-31432074099285.465, -24462402558075.102, 33393057246326.582, -
14548687282098.322,
9432953040102.2988, 16299092500651.521, -5680521094985.7197,
3213803769940.8018,
9204107208950.0898, -2118213400254.3159, -51520770088614.844, -
7375167718439.8467,
26383346218925.5, 55318766293993.266, -26155441646471.867, -8483956504608.2168,
-37666038992544.383, 8692860529528.2305, 26347785414349.418, -
21907148632872.766,
-26165382170947.59, 10370056147391.766, -55786375523209.312, -
9997224391818.9004,
-6907058636510.4277, 51795375719216.883, 5906918853169.9121, 13242823436132.09,
-971762280895.29431, -12762664605047.533, 23850345783983.191,
18574726632885.215,
-23430741364141.168, 14623897929309.076, 4083624200017.9487, -14260502383778.6,
8155409857867.0439, 43241890297086.008, -7421625382572.0957,
4377250972834.5703,
23338698842285.066, -6362884133579.6445

};

static double LNN10Acts[182];

/* ----------------------------------------------------------- */
/*
  LNN10Run - run neural network LNN10

  inputs - the input variables of this network.
  The variable names are listed below, together with each
  variable's offset in the data set at the time code was
  generated (if the variable is then available).
  Variable (Offset)
  µ0O0
  µ0O0avg
  µ0O0dev
  µ0O1
  µ0O1avg
  µ0O1dev
  µ0O2
  µ0O2avg
  µ0O2dev
  µ0O3
  µ0O3avg
  µ0O3dev
  µ0O4
  µ0O4avg
  µ0O4dev
  µ1O0
  µ1O0avg
  µ1O0dev
  µ1O1
```

µ1O1avg
µ1O1dev
µ1O2
µ1O2avg
µ1O2dev
µ1O3
µ1O3avg
µ1O3dev
µ1O4
µ1O4avg
µ1O4dev
µ2O0
µ2O0avg
µ2O0dev
µ2O1
µ2O1avg
µ2O1dev
µ2O2
µ2O2avg
µ2O2dev
µ2O3
µ2O3avg
µ2O3dev
µ2O4
µ2O4avg
µ2O4dev
µ3O0
µ3O0avg
µ3O0dev
µ3O1
µ3O1avg
µ3O1dev
µ3O2
µ3O2avg
µ3O2dev
µ3O3
µ3O3avg
µ3O3dev
µ3O4
µ3O4avg
µ3O4dev
µ4O0
µ4O0avg
µ4O0dev
µ4O1
µ4O1avg
µ4O1dev

```
  µ4O2
  µ4O2avg
  µ4O2dev
  µ4O3
  µ4O3avg
  µ4O3dev
  µ4O4
  µ4O4avg
  µ4O4dev
  µ5O0
  µ5O0avg
  µ5O0dev
  µ5O1
  µ5O1avg
  µ5O1dev
  µ5O2
  µ5O2avg
  µ5O2dev
  µ5O3
  µ5O3avg
  µ5O3dev
  µ5O4
  µ5O4avg
  µ5O4dev

*/
/* ----------------------------------------------------- */

void LNN10Run( double inputs[], double outputs[], int outputType )
{
  int i, j, k, u;
  double *w = LNN10Weights, *t = LNN10Thresholds;

  /* Process inputs - apply pre-processing to each input in turn,
   * storing results in the neuron activations array.
   */

  /* Input 0: standard numeric pre-processing: linear shift and scale. */
  if ( inputs[0] == -9999 )
    LNN10Acts[0] = 0.0024664534993155234;
  else
    LNN10Acts[0] = inputs[0] * 0.00019897446715854938 + 0;

  /* Input 1: standard numeric pre-processing: linear shift and scale. */
  if ( inputs[1] == -9999 )
    LNN10Acts[1] = 0.42751501821871291;
  else
  LNN10Acts[1] = inputs[1] * 0.053441295546558708 + -0.026720647773279354;

/* Input 2: standard numeric pre-processing: linear shift and scale. */
if ( inputs[2] == -9999 )
  LNN10Acts[2] = 0.0045808935069142805;
else
  LNN10Acts[2] = inputs[2] * 0.00019823638663488865 + 0.0038085414886824057;

/* Input 3: standard numeric pre-processing: linear shift and scale. */
if ( inputs[3] == -9999 )
  LNN10Acts[3] = 0.012340613284334548;
else
  LNN10Acts[3] = inputs[3] * 0.00019897446715854938 + 0;

/* Input 4: standard numeric pre-processing: linear shift and scale. */
if ( inputs[4] == -9999 )
  LNN10Acts[4] = 0.65028748778227441;
else
  LNN10Acts[4] = inputs[4] * 0.015289164635497451 + -0.08918679370706846;

/* Input 5: standard numeric pre-processing: linear shift and scale. */
if ( inputs[5] == -9999 )
  LNN10Acts[5] = 0.016674794000276744;
else
  LNN10Acts[5] = inputs[5] * 0.0001964182634520473 + 0.013992666289833892;

/* Input 6: standard numeric pre-processing: linear shift and scale. */
if ( inputs[6] == -9999 )
  LNN10Acts[6] = 0.020037236738136597;
else
  LNN10Acts[6] = inputs[6] * 0.00019897446715854938 + 0;

/* Input 7: standard numeric pre-processing: linear shift and scale. */
if ( inputs[7] == -9999 )
  LNN10Acts[7] = 0.58270523982064493;
else
  LNN10Acts[7] = inputs[7] * 0.011461318051575931 + -0.25214899713467048;

/* Input 8: standard numeric pre-processing: linear shift and scale. */
if ( inputs[8] == -9999 )
  LNN10Acts[8] = 0.026816149353272822;
else
  LNN10Acts[8] = inputs[8] * 0.00019557911163745775 + 0.021367017946392258;

/* Input 9: standard numeric pre-processing: linear shift and scale. */
if ( inputs[9] == -9999 )
  LNN10Acts[9] = 0.033851063318347689;
```

```
else
  LNN10Acts[9] = inputs[9] * 0.00019897446715854938 + 0;

/* Input 10: standard numeric pre-processing: linear shift and scale. */
if ( inputs[10] == -9999 )
  LNN10Acts[10] = 0.40841307754993122;
else
  LNN10Acts[10] = inputs[10] * 0.0047876769358867607 + -0.27529142381348876;

/* Input 11: standard numeric pre-processing: linear shift and scale. */
if ( inputs[11] == -9999 )
  LNN10Acts[11] = 0.056105515538165458;
else
  LNN10Acts[11] = inputs[11] * 0.00019103510356941806 + 0.050885937479045205;

/* Input 12: standard numeric pre-processing: linear shift and scale. */
if ( inputs[12] == -9999 )
  LNN10Acts[12] = 0.041465767538834131;
else
  LNN10Acts[12] = inputs[12] * 0.00012242899118511264 + 0;

/* Input 13: standard numeric pre-processing: linear shift and scale. */
if ( inputs[13] == -9999 )
  LNN10Acts[13] = 0.56167130824852096;
else
  LNN10Acts[13] = inputs[13] * 0.0030087646622770675 + -0.25825230017878165;

/* Input 14: standard numeric pre-processing: linear shift and scale. */
if ( inputs[14] == -9999 )
  LNN10Acts[14] = 0.057440758276209203;
else
  LNN10Acts[14] = inputs[14] * 0.00011858704639073695 + 0.04959258720475232;

/* Input 15: standard numeric pre-processing: linear shift and scale. */
if ( inputs[15] == -9999 )
  LNN10Acts[15] = 0.99952189744719722;
else
  LNN10Acts[15] = inputs[15] * 6.7762613285860989e-011 + 0.99999965944065961;

/* Input 16: standard numeric pre-processing: linear shift and scale. */
if ( inputs[16] == -9999 )
  LNN10Acts[16] = 0.44251671863202452;
else
  LNN10Acts[16] = inputs[16] * 1.3458511080840785 + -0.020681245360892008;

/* Input 17: standard numeric pre-processing: linear shift and scale. */
if ( inputs[17] == -9999 )
```

```
  LNN10Acts[17] = 0.99952189742494102;
else
  LNN10Acts[17] = inputs[17] * 6.7762613282449192e-011 + 0.99999965944170099;

/* Input 18: standard numeric pre-processing: linear shift and scale. */
if ( inputs[18] == -9999 )
  LNN10Acts[18] = 0.99951688527456706;
else
  LNN10Acts[18] = inputs[18] * 7.7442983150293666e-011 + 0.99999961078934263;

/* Input 19: standard numeric pre-processing: linear shift and scale. */
if ( inputs[19] == -9999 )
  LNN10Acts[19] = 0.64552042702139178;
else
  LNN10Acts[19] = inputs[19] * 0.30389888455295788 + -0.06761243683162392;

/* Input 20: standard numeric pre-processing: linear shift and scale. */
if ( inputs[20] == -9999 )
  LNN10Acts[20] = 0.99951688511019143;
else
  LNN10Acts[20] = inputs[20] * 7.7442983130558765e-011 + 0.99999961080657263;

/* Input 21: standard numeric pre-processing: linear shift and scale. */
if ( inputs[21] == -9999 )
  LNN10Acts[21] = 0.99950078318963775;
else
  LNN10Acts[21] = inputs[21] * 8.7435621949295615e-011 + 0.99999956056863371;

/* Input 22: standard numeric pre-processing: linear shift and scale. */
if ( inputs[22] == -9999 )
  LNN10Acts[22] = 0.58365978067067326;
else
  LNN10Acts[22] = inputs[22] * 0.1914584927838045 + -0.19169143395002478;

/* Input 23: standard numeric pre-processing: linear shift and scale. */
if ( inputs[23] == -9999 )
  LNN10Acts[23] = 0.9995007829233189;
else
  LNN10Acts[23] = inputs[23] * 8.7435621909365358e-011 + 0.9999995606561759;

/* Input 24: standard numeric pre-processing: linear shift and scale. */
if ( inputs[24] == -9999 )
  LNN10Acts[24] = 0.99956719062957922;
else
  LNN10Acts[24] = inputs[24] * 4.3368078003008944e-011 + 0.99999978204199458;

/* Input 25: standard numeric pre-processing: linear shift and scale. */
```

```
if ( inputs[25] == -9999 )
  LNN10Acts[25] = 0.38738613907960079;
else
  LNN10Acts[25] = inputs[25] * 0.086546236950802977 + -0.37140884856252349;

/* Input 26: standard numeric pre-processing: linear shift and scale. */
if ( inputs[26] == -9999 )
  LNN10Acts[26] = 0.99956719043554709;
else
  LNN10Acts[26] = inputs[26] * 4.336807799439778e-011 + 0.99999978222810659;

/* Input 27: standard numeric pre-processing: linear shift and scale. */
if ( inputs[27] == -9999 )
  LNN10Acts[27] = 0.99953401427685873;
else
  LNN10Acts[27] = inputs[27] * 1.6086084539955739e-011 + 0.9999999191550315;

/* Input 28: standard numeric pre-processing: linear shift and scale. */
if ( inputs[28] == -9999 )
  LNN10Acts[28] = 0.58265864185011185;
else
  LNN10Acts[28] = inputs[28] * 0.058300263424248239 + -0.32243057853886198;

/* Input 29: standard numeric pre-processing: linear shift and scale. */
if ( inputs[29] == -9999 )
  LNN10Acts[29] = 0.99953401411618958;
else
  LNN10Acts[29] = inputs[29] * 1.6086084536617535e-011 + 0.99999991924399589;

/* Input 30: standard numeric pre-processing: linear shift and scale. */
if ( inputs[30] == -9999 )
  LNN10Acts[30] = 0.99952206756291384;
else
  LNN10Acts[30] = inputs[30] * 3.3881312357767097e-011 + 0.9999998297203011;

/* Input 31: standard numeric pre-processing: linear shift and scale. */
if ( inputs[31] == -9999 )
  LNN10Acts[31] = 0.43129811422089592;
else
  LNN10Acts[31] = inputs[31] * 0.90809650589230806 + -0.022021340267888476;

/* Input 32: standard numeric pre-processing: linear shift and scale. */
if ( inputs[32] == -9999 )
  LNN10Acts[32] = 0.99952206754683981;
else
  LNN10Acts[32] = inputs[32] * 3.3881312356502977e-011 + 0.99999982972112278;
```

```
/* Input 33: standard numeric pre-processing: linear shift and scale. */
if ( inputs[33] == -9999 )
  LNN10Acts[33] = 0.99951707964030245;
else
  LNN10Acts[33] = inputs[33] * 3.8721499013045177e-011 + 0.99999980539463407;

/* Input 34: standard numeric pre-processing: linear shift and scale. */
if ( inputs[34] == -9999 )
  LNN10Acts[34] = 0.61706899825374562;
else
  LNN10Acts[34] = inputs[34] * 0.20149841548973266 + -0.066467610656621315;

/* Input 35: standard numeric pre-processing: linear shift and scale. */
if ( inputs[35] == -9999 )
  LNN10Acts[35] = 0.99951707952181423;
else
  LNN10Acts[35] = inputs[35] * 3.8721499005604151e-011 + 0.99999980540740685;

/* Input 36: standard numeric pre-processing: linear shift and scale. */
if ( inputs[36] == -9999 )
  LNN10Acts[36] = 0.99950100257106156;
else
  LNN10Acts[36] = inputs[36] * 4.3717820433157167e-011 + 0.99999978028426939;

/* Input 37: standard numeric pre-processing: linear shift and scale. */
if ( inputs[37] == -9999 )
  LNN10Acts[37] = 0.57521705815925384;
else
  LNN10Acts[37] = inputs[37] * 0.13331287141762951 + -0.20641720634067695;

/* Input 38: standard numeric pre-processing: linear shift and scale. */
if ( inputs[38] == -9999 )
  LNN10Acts[38] = 0.99950100238259199;
else
  LNN10Acts[38] = inputs[38] * 4.3717820418820608e-011 + 0.99999978035196069;

/* Input 39: standard numeric pre-processing: linear shift and scale. */
if ( inputs[39] == -9999 )
  LNN10Acts[39] = 0.9995672993860123;
else
  LNN10Acts[39] = inputs[39] * 2.1684041306884304e-011 + 0.99999989102098574;

/* Input 40: standard numeric pre-processing: linear shift and scale. */
if ( inputs[40] == -9999 )
  LNN10Acts[40] = 0.3913143490349294;
else
  LNN10Acts[40] = inputs[40] * 0.057878142017694319 + -0.33832571452873189;
```

```
/* Input 41: standard numeric pre-processing: linear shift and scale. */
if ( inputs[41] == -9999 )
  LNN10Acts[41] = 0.99956729923947063;
else
  LNN10Acts[41] = inputs[41] * 2.1684041303662717e-011 + 0.99999989114773935;

/* Input 42: standard numeric pre-processing: linear shift and scale. */
if ( inputs[42] == -9999 )
  LNN10Acts[42] = 0.99953405457698974;
else
  LNN10Acts[42] = inputs[42] * 8.0430425749044379e-012 + 0.9999999595775142;

/* Input 43: standard numeric pre-processing: linear shift and scale. */
if ( inputs[43] == -9999 )
  LNN10Acts[43] = 0.57797510637666183;
else
  LNN10Acts[43] = inputs[43] * 0.039488011039245582 + -0.30595179326504013;

/* Input 44: standard numeric pre-processing: linear shift and scale. */
if ( inputs[44] == -9999 )
  LNN10Acts[44] = 0.99953405445933674;
else
  LNN10Acts[44] = inputs[44] * 8.0430425736820685e-012 + 0.99999995963983146;

/* Input 45: standard numeric pre-processing: linear shift and scale. */
if ( inputs[45] == -9999 )
  LNN10Acts[45] = 0.99952212426790088;
else
  LNN10Acts[45] = inputs[45] * 2.2587542857390072e-011 + 0.99999988648019433;

/* Input 46: standard numeric pre-processing: linear shift and scale. */
if ( inputs[46] == -9999 )
  LNN10Acts[46] = 0.43463142253924975;
else
  LNN10Acts[46] = inputs[46] * 0.72305470894856894 + -0.022920834273669637;

/* Input 47: standard numeric pre-processing: linear shift and scale. */
if ( inputs[47] == -9999 )
  LNN10Acts[47] = 0.99952212425433828;
else
  LNN10Acts[47] = inputs[47] * 2.2587542856684461e-011 + 0.99999988648091032;

/* Input 48: standard numeric pre-processing: linear shift and scale. */
if ( inputs[48] == -9999 )
  LNN10Acts[48] = 0.99951714442671258;
else
```

```
  LNN10Acts[48] = inputs[48] * 2.5814334346177584e-011 + 0.9999998702630809;

/* Input 49: standard numeric pre-processing: linear shift and scale. */
if ( inputs[49] == -9999 )
  LNN10Acts[49] = 0.62682506525409387;
else
  LNN10Acts[49] = inputs[49] * 0.15778496682212623 + -0.066369616544280363;

/* Input 50: standard numeric pre-processing: linear shift and scale. */
if ( inputs[50] == -9999 )
  LNN10Acts[50] = 0.99951714432424021;
else
  LNN10Acts[50] = inputs[50] * 2.5814334341954242e-011 + 0.99999987027393933;

/* Input 51: standard numeric pre-processing: linear shift and scale. */
if ( inputs[51] == -9999 )
  LNN10Acts[51] = 0.99950107569537128;
else
  LNN10Acts[51] = inputs[51] * 2.914521575544428e-011 + 0.99999985352283549;

/* Input 52: standard numeric pre-processing: linear shift and scale. */
if ( inputs[52] == -9999 )
  LNN10Acts[52] = 0.57905613424841007;
else
  LNN10Acts[52] = inputs[52] * 0.10147438009411504 + -0.20390431060078298;

/* Input 53: standard numeric pre-processing: linear shift and scale. */
if ( inputs[53] == -9999 )
  LNN10Acts[53] = 0.99950107552919953;
else
  LNN10Acts[53] = inputs[53] * 2.9145215747073264e-011 + 0.99999985358140042;

/* Input 54: standard numeric pre-processing: linear shift and scale. */
if ( inputs[54] == -9999 )
  LNN10Acts[54] = 0.99956733563566291;
else
  LNN10Acts[54] = inputs[54] * 1.4456028061333241e-011 + 0.99999992734732113;

/* Input 55: standard numeric pre-processing: linear shift and scale. */
if ( inputs[55] == -9999 )
  LNN10Acts[55] = 0.38491857220633779;
else
  LNN10Acts[55] = inputs[55] * 0.046152967858237262 + -0.37586823180522227;

/* Input 56: standard numeric pre-processing: linear shift and scale. */
if ( inputs[56] == -9999 )
  LNN10Acts[56] = 0.99956733551515131;
```

```
else
  LNN10Acts[56] = inputs[56] * 1.4456028059581109e-011 + 0.99999992746505062;

/* Input 57: standard numeric pre-processing: linear shift and scale. */
if ( inputs[57] == -9999 )
  LNN10Acts[57] = 0.99953406835490155;
else
  LNN10Acts[57] = inputs[57] * 5.3620284552755886e-012 + 0.99999997305167565;

/* Input 58: standard numeric pre-processing: linear shift and scale. */
if ( inputs[58] == -9999 )
  LNN10Acts[58] = 0.58407397011961093;
else
  LNN10Acts[58] = inputs[58] * 0.031314655971516628 + -0.33211436874804678;

/* Input 59: standard numeric pre-processing: linear shift and scale. */
if ( inputs[59] == -9999 )
  LNN10Acts[59] = 0.99953406825495106;
else
  LNN10Acts[59] = inputs[59] * 5.3620284545765547e-012 + 0.99999997310854394;

/* Input 60: standard numeric pre-processing: linear shift and scale. */
if ( inputs[60] == -9999 )
  LNN10Acts[60] = 0.99952215261936928;
else
  LNN10Acts[60] = inputs[60] * 1.694065761600335e-011 + 0.99999991486014339;

/* Input 61: standard numeric pre-processing: linear shift and scale. */
if ( inputs[61] == -9999 )
  LNN10Acts[61] = 0.43087811362400869;
else
  LNN10Acts[61] = inputs[61] * 0.62424393182877935 + -0.02387733039245081;

/* Input 62: standard numeric pre-processing: linear shift and scale. */
if ( inputs[62] == -9999 )
  LNN10Acts[62] = 0.99952215260768906;
else
  LNN10Acts[62] = inputs[62] * 1.6940657615543616e-011 + 0.99999991486079132;

/* Input 63: standard numeric pre-processing: linear shift and scale. */
if ( inputs[63] == -9999 )
  LNN10Acts[63] = 0.99951717681240526;
else
  LNN10Acts[63] = inputs[63] * 1.9360751374836183e-011 + 0.99999990269730765;

/* Input 64: standard numeric pre-processing: linear shift and scale. */
if ( inputs[64] == -9999 )
```

```
  LNN10Acts[64] = 0.6146835801114251;
else
  LNN10Acts[64] = inputs[64] * 0.13451574027782504 + -0.066861048705092935;

/* Input 65: standard numeric pre-processing: linear shift and scale. */
if ( inputs[65] == -9999 )
  LNN10Acts[65] = 0.99951717672400375;
else
  LNN10Acts[65] = inputs[65] * 1.9360751372049605e-011 + 0.99999990270693084;

/* Input 66: standard numeric pre-processing: linear shift and scale. */
if ( inputs[66] == -9999 )
  LNN10Acts[66] = 0.99950111224034521;
else
  LNN10Acts[66] = inputs[66] * 2.1858912597142106e-011 + 0.99999989014212276;

/* Input 67: standard numeric pre-processing: linear shift and scale. */
if ( inputs[67] == -9999 )
  LNN10Acts[67] = 0.57556805189029414;
else
  LNN10Acts[67] = inputs[67] * 0.088391738452074181 + -0.21127687963942945;

/* Input 68: standard numeric pre-processing: linear shift and scale. */
if ( inputs[68] == -9999 )
  LNN10Acts[68] = 0.99950111209813286;
else
  LNN10Acts[68] = inputs[68] * 2.1858912591736487e-011 + 0.99999989019437052;

/* Input 69: standard numeric pre-processing: linear shift and scale. */
if ( inputs[69] == -9999 )
  LNN10Acts[69] = 0.99956735374429184;
else
  LNN10Acts[69] = inputs[69] * 1.0842021235184758e-011 + 0.99999994551048987;

/* Input 70: standard numeric pre-processing: linear shift and scale. */
if ( inputs[70] == -9999 )
  LNN10Acts[70] = 0.38873412751868813;
else
  LNN10Acts[70] = inputs[70] * 0.039534966801657696 + -0.36143986807468181;

/* Input 71: standard numeric pre-processing: linear shift and scale. */
if ( inputs[71] == -9999 )
  LNN10Acts[71] = 0.99956735363773208;
else
  LNN10Acts[71] = inputs[71] * 1.0842021234028306e-011 + 0.99999994560961081;

/* Input 72: standard numeric pre-processing: linear shift and scale. */
```

```
if ( inputs[72] == -9999 )
  LNN10Acts[72] = 0.99953407505966541;
else
  LNN10Acts[72] = inputs[72] * 4.0215213658211277e-012 + 0.99999997978875665;

/* Input 73: standard numeric pre-processing: linear shift and scale. */
if ( inputs[73] == -9999 )
  LNN10Acts[73] = 0.58446308828603954;
else
  LNN10Acts[73] = inputs[73] * 0.02700825718749918 + -0.32593069622492238;

/* Input 74: standard numeric pre-processing: linear shift and scale. */
if ( inputs[74] == -9999 )
  LNN10Acts[74] = 0.99953407497269209;
else
  LNN10Acts[74] = inputs[74] * 4.0215213653644707e-012 + 0.99999997983728772;

/* Input 75: standard numeric pre-processing: linear shift and scale. */
if ( inputs[75] == -9999 )
  LNN10Acts[75] = 0.99952216963030049;
else
  LNN10Acts[75] = inputs[75] * 1.3552526324827639e-011 + 0.99999993188811354;

/* Input 76: standard numeric pre-processing: linear shift and scale. */
if ( inputs[76] == -9999 )
  LNN10Acts[76] = 0.43361026355261878;
else
  LNN10Acts[76] = inputs[76] * 0.55991848944171874 + -0.024739065258499938;

/* Input 77: standard numeric pre-processing: linear shift and scale. */
if ( inputs[77] == -9999 )
  LNN10Acts[77] = 0.99952216961981677;
else
  LNN10Acts[77] = inputs[77] * 1.3552526324499609e-011 + 0.99999993188871239;

/* Input 78: standard numeric pre-processing: linear shift and scale. */
if ( inputs[78] == -9999 )
  LNN10Acts[78] = 0.99951719624388247;
else
  LNN10Acts[78] = inputs[78] * 1.5488601399982112e-011 + 0.99999992215784461;

/* Input 79: standard numeric pre-processing: linear shift and scale. */
if ( inputs[79] == -9999 )
  LNN10Acts[79] = 0.6201201393428758;
else
  LNN10Acts[79] = inputs[79] * 0.11946062442068471 + -0.06752113593297801;
```

```
/* Input 80: standard numeric pre-processing: linear shift and scale. */
if ( inputs[80] == -9999 )
  LNN10Acts[80] = 0.99951719616354373;
else
  LNN10Acts[80] = inputs[80] * 1.5488601397973947e-011 + 0.99999992216659905;

/* Input 81: standard numeric pre-processing: linear shift and scale. */
if ( inputs[81] == -9999 )
  LNN10Acts[81] = 0.99950113416809949;
else
  LNN10Acts[81] = inputs[81] * 1.7487130461536732e-011 + 0.99999991211369621;

/* Input 82: standard numeric pre-processing: linear shift and scale. */
if ( inputs[82] == -9999 )
  LNN10Acts[82] = 0.57820299531018127;
else
  LNN10Acts[82] = inputs[82] * 0.07720488594589342 + -0.21019544898009124;

/* Input 83: standard numeric pre-processing: linear shift and scale. */
if ( inputs[83] == -9999 )
  LNN10Acts[83] = 0.99950113403724794;
else
  LNN10Acts[83] = inputs[83] * 1.7487130457575842e-011 + 0.99999991216130613;

/* Input 84: standard numeric pre-processing: linear shift and scale. */
if ( inputs[84] == -9999 )
  LNN10Acts[84] = 0.99956736460943663;
else
  LNN10Acts[84] = inputs[84] * 8.6736170822318986e-012 + 0.99999995640839145;

/* Input 85: standard numeric pre-processing: linear shift and scale. */
if ( inputs[85] == -9999 )
  LNN10Acts[85] = 0.38290819908127799;
else
  LNN10Acts[85] = inputs[85] * 0.035833266254151 48 + -0.39343194405856047;

/* Input 86: standard numeric pre-processing: linear shift and scale. */
if ( inputs[86] == -9999 )
  LNN10Acts[86] = 0.99956736451679185;
else
  LNN10Acts[86] = inputs[86] * 8.6736170814313106e-012 + 0.99999995650362361;

/* Input 87: standard numeric pre-processing: linear shift and scale. */
if ( inputs[87] == -9999 )
  LNN10Acts[87] = 0.99953407846048881;
else
  LNN10Acts[87] = inputs[87] * 3.2172188225312508e-012 + 0.99999998383099675;
```

```
/* Input 88: standard numeric pre-processing: linear shift and scale. */
if ( inputs[88] == -9999 )
  LNN10Acts[88] = 0.58847468734428787;
else
  LNN10Acts[88] = inputs[88] * 0.024334929463708174 + -0.34826893743770176;

/* Input 89: standard numeric pre-processing: linear shift and scale. */
if ( inputs[89] == -9999 )
  LNN10Acts[89] = 0.99953407838273656;
else
  LNN10Acts[89] = inputs[89] * 3.2172188222025297e-012 + 0.99999998387703981;

/*
 * Process layer 1.
 */

/* For each unit in turn */
for ( u=0; u < 1; ++u )
{
  /*
   * First, calculate post-synaptic potentials, storing
   * these in the LNN10Acts array.
   */

  /* Initialise hidden unit activation to zero */
  LNN10Acts[90+u] = 0.0;

  /* Accumulate weighted sum from inputs */
  for ( i=0; i < 90; ++i )
    LNN10Acts[90+u] += *w++ * LNN10Acts[0+i];

  /* Subtract threshold */
  LNN10Acts[90+u] -= *t++;

}

/* Type of output required - selected by outputType parameter */
switch ( outputType )
{
  /* The usual type is to generate the output variables */
  case 0:


    /* Post-process output 0, two-state nominal output */
    if ( LNN10Acts[90] >= 0.12548828125 )
      outputs[0] = 2.0;
   else
    outputs[0] = 1.0;
   break;

  /* type 1 is activation of output neurons */
  case 1:
   for ( i=0; i < 1; ++i )
    outputs[i] = LNN10Acts[90+i];
   break;

  /* type 2 is codebook vector of winning node (lowest actn) 1st hidden layer */
  case 2:
   {
    int winner=0;
    for ( i=1; i < 1; ++i )
     if ( LNN10Acts[90+i] < LNN10Acts[90+winner] )
      winner=i;

    for ( i=0; i < 90; ++i )
     outputs[i] = LNN10Weights[90*winner+i];
   }
   break;

  /* type 3 indicates winning node (lowest actn) in 1st hidden layer */
  case 3:
   {
    int winner=0;
    for ( i=1; i < 1; ++i )
     if ( LNN10Acts[90+i] < LNN10Acts[90+winner] )
      winner=i;

    outputs[0] = winner;
   }
   break;
 }
}

/*
 Test harness. Compile including this main() procedure, as
 a windows console program or a DOS program, to interactively
 test that the software functions as expected.
*/

void main(void)
{
 int i, outputType=0, noOutputs=1;
 double inputs[90], outputs[90];
```

```
  printf( "\n\nLNN test harness program. Enter inputs below\n" );
  printf( "Nominal variables should be numbered starting at 1 (0 for missing)\n" );
  printf( "(e.g. if an input is Gender={male,female}, enter 1 for male, 2 for female)\n" );

  /* Infinite (user-breakable) loop for repeated tests */
start_of_loop:
  while ( 1 )
  {
    /* Get the input pattern */
    for ( i=0; i < 90; ++i )
    {
      printf( "Enter value for input %d: ", i+1 );
      scanf( "%lg", & inputs[i] );

      /* Check for sub-menu */
      if ( inputs[i] == MENUCODE )
      {
        printf( "Control menu. Select output style, or exit:\n" );
        printf( "0. Normal output style (output variable)\n" );
        printf( "1. Output layer activations\n" );
        printf( "2. Codebook vector (usual only for Kohonen networks\n" );
        printf( "3. Winning hidden neuron (ditto only Kohonen)\n" );
        printf( "4. Exit program\n" );
        printf( "> " );
        scanf( "%d", & outputType );
        if ( outputType < 0 || outputType > 3 )
        {
          printf( "\nBye.\n" );
          return;
        }
        else
        {
          /* Determine how many outputs there are to display (depends on
           * output type)
           */
          switch ( outputType )
          {
            case 0:
              noOutputs = 1;
              break;

            case 1:
              noOutputs = 1;
              break;

            case 2:
```

```
       noOutputs = 90;
       break;

      case 3:
       noOutputs = 1;
       break;
     }
     goto start_of_loop;
    }
   }
  }

  /* Run the neural network */
  LNN10Run( inputs, outputs, outputType );

  /* Display the output of the neural network */
  printf( "\n\nOutput of neural network:\n" );

  for ( i=0; i < noOutputs; ++i )
  {
   printf( "Output %d: ", i+1 );
   printf( "%g\n", outputs[i] );
  }

  printf( "\nEnter next input pattern (for control menu inc. exit, enter %d for any input):\n",
MENUCODE );
 }
}
```