neuralnet

COLLABORATORS

	TITLE : neuralnet	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

neuralnet

1.1 neuralnet.doc

```
NN_Clone()
NN_CreateA()
NN_Delete()
NN_GetOutputLayer()
NN_LoadA()
NN_Recall()
NN_Reset()
NN_SaveA()
NN_SetInputLayer()
NN_SetOutputLayer()
NN_Train()
```

1.2 neuralnet.library/NN_Clone

```
NAME
     NN_Clone - create a duplicate from a neural network.

SYNOPSIS
     cloned_net = NN_Clone(net);
     d0                    a0

     NEURALNET NN_Clone(NEURALNET);

FUNCTION
     create a duplicate from a neural network, including
     all its attributes and the current training status.

INPUTS
     net             - pointer to a neural net created
                       with NN_CreateA().

RESULTS
     cloned_net      - vanilla copy of the neural network,
                       or NULL if out of memory.
```

```
SEE ALSO
     NN_CreateA()
```

1.3 neuralnet.library/NN_CreateA

```
NAME
     NN_CreateA - create a neural network.
     NN_Create  - varargs stub for NN_CreateA.

SYNOPSIS
     net = NN_CreateA(input,hidden,output,taglist)
     d0               d0    d1     d2     a0

     NEURALNET NN_CreateA(ULONG,ULONG,ULONG,struct TagItem *)

     NEURALNET NN_Create(ULONG,ULONG,ULONG,tag,...,TAG_DONE)

FUNCTION
     create and initialize a neural backpropagation network.

INPUTS
     input           - number of input layer neurons.
     hidden          - number of hidden layer neurons.
     output          - number of output layer neurons.
     taglist         - pointer to an array of TagItems.

TAGS
     none defined

RESULTS
     net             - backpropagation network ready
                       for usage, or NULL if out of memory.

SEE ALSO
     NN_Delete()
```

1.4 neuralnet.library/NN_Delete

```
NAME
     NN_Delete  - dispose a neural network.

SYNOPSIS
     NN_Delete(net)
               a0

     void NN_Delete(NEURALNET)

FUNCTION
     remove a neural network and free all associated memory.
```

```
INPUTS
     net             - pointer to a neural network.

SEE ALSO
     NN_CreateA()
```

1.5 neuralnet.library/NN_GetOutputLayer

```
NAME
     NN_GetOutputLayer - read a neural network's output layer.

SYNOPSIS
     NN_GetOutputLayer(net,array)
                       a0  a1

     void NN_GetOutputLayer(NEURALNET,NEURON *)

FUNCTION
     Transfer a neural network's output layer to an array of
     the type NEURON.

     The array must allow to hold at least the number of
     output neurons the net was created with.

INPUTS
     net             - pointer to a neural network.
     array           - pointer to an array of type NEURON.

NOTES
     The type NEURON is assumed to store short (32bit) IEEE
     floating point numbers. you must set the compiler options
     accordingly. refer to the manual for further details.

SEE ALSO
     NN_CreateA(), NN_SetOutputLayer(), NN_GetOutputNeuron()
```

1.6 neuralnet.library/NN_LoadA

```
NAME
     NN_LoadA - load a neural network's training status from disk.
     NN_Load  - varargs stub for NN_LoadA.

SYNOPSIS
     success = NN_LoadA(net,filename,taglist)
     d0                 a0  a1       a2

     BOOL NN_LoadA(NEURALNET,STRPTR,struct TagItem *)

     BOOL NN_Load(NEURALNET,STRPTR,tag,...,TAG_DONE)
```

```
FUNCTION
     Load a training status from disk and insert it
     to a neural network.

     The disk data must comply to the network's topology,
     i.e. result from the same number of input, hidden,
     and output layer neurons. Otherwise this function
     returns FALSE.

INPUTS
     net             - pointer to a neural network.
     filename        - filename of a neural network's
                       training status.
     taglist         - pointer to an array of TagItems.

TAGS
     none defined

RESULTS
     success         - boolean, TRUE if the net was
                       successfully loaded.

SEE ALSO
     NN_Save()
```

1.7 neuralnet.library/NN_Recall

```
NAME
     NN_Recall - recall a neural network.

SYNOPSIS
     NN_Recall(net)
               a0

     void NN_Recall(NEURALNET)

FUNCTION
     Recall a neural network, thus producing an output
     at its output layer.

INPUTS
     net             - pointer to a neural network.

SEE ALSO
     NN_Train(), NN_SetInputLayer(), NN_GetOutputLayer()
```

1.8 neuralnet.library/NN_Reset

```
NAME
     NN_Reset - reset a neural network.
```

```
SYNOPSIS
     NN_Reset(net)
              a0

     void NN_Reset(NEURALNET)

FUNCTION
     Clear a neural network's training status. It will behave
     as stupid as if it were freshly created with NN_CreateA().

INPUTS
     net             - pointer to a neural network.

SEE ALSO
     NN_CreateA()
```

1.9 neuralnet.library/NN_SaveA

```
NAME
     NN_SaveA - save a neural network's training status to disk.
     NN_Save  - varargs stub for NN_SaveA.

SYNOPSIS
     success = NN_SaveA(net,filename,taglist)
     d0                 a0  a1       a2

     BOOL NN_SaveA(NEURALNET,STRPTR,struct TagItem *)

     BOOL NN_Save(NEURALNET,STRPTR,tag,...TAG_DONE)

FUNCTION
     Save a neural network's current training status to disk.

INPUTS
     net             - pointer to a neural network.
     filename        - filename of a neural network's
                       training status.
     taglist         - pointer to an array of TagItems.

TAGS
     none defined

RESULTS
     success         - boolean, TRUE if the net was
                       successfully saved.

SEE ALSO
     NN_Load()
```

1.10 neuralnet.library/NN_SetInputLayer

```
NAME
     NN_SetInputLayer - set a neural network's input layer.

SYNOPSIS
     NN_SetInputLayer(net,array)
                      a0  a1

     void NN_SetInputLayer(NEURALNET,NEURON *)

FUNCTION
     Transfer an array of the type NEURON to a neural network's
     input layer.

     The array must contain at least the number of
     input neurons the net was created with.

INPUTS
     net             - pointer to a neural network.
     array           - pointer to an array of type NEURON.

NOTES
     The type NEURON is assumed to store short (32bit) IEEE
     floating point numbers. you must set the compiler options
     accordingly. refer to the manual for further details.

SEE ALSO
     NN_CreateA(), NN_SetOutputLayer(), NN_SetInputNeuron()
```

1.11 neuralnet.library/NN_SetOutputLayer

```
NAME
     NN_SetOutputLayer - set a neural network's output layer.

SYNOPSIS
     NN_SetOutputLayer(net,array)
                       a0  a1

     void NN_SetOutputLayer(NEURALNET,NEURON *)

FUNCTION
     Transfer an array of the type NEURON to a neural network's
     output layer.

     The array must contain at least the number of
     output neurons the net was created with.

INPUTS
     net             - pointer to a neural network.
     array           - pointer to an array of type NEURON.

NOTES
```

```
     The type NEURON is assumed to store short (32bit) IEEE
     floating point numbers. you must set the compiler options
     accordingly. refer to the manual for further details.

SEE ALSO
     NN_CreateA(), NN_GetOutputLayer(), NN_SetOutputNeuron()
```

1.12 neuralnet.library/NN_Train

```
NAME
     NN_Train - train a neural network.

SYNOPSIS
     NN_Train(net,errorptr)
              a0  a1

     void NN_Train(NEURALNET,float *)

FUNCTION
     Train a neural network the current input/output pattern.

     If you supply an error pointer, this value will be filled
     with the network's current degree of 'unawareness' of the
     data to be trained. The lower this value, the better the
     pattern has been recognized. An error of 0.0 indicates
     perfect recognition, but it is not always recommended to
     train a net that far.

INPUTS
     net             - pointer to a neural network.
     errorptr        - pointer to a float number.

EXAMPLE
     Train a neural network until an error value of 0.01
     has been reached:

     float err;
     do
     {
         NN_Train(net, &err);

     } while (err > 0.01);

NOTES
     The errorptr is assumed to point to a short (32bit) IEEE
     floating point number. you must set the compiler options
     accordingly. refer to the manual for further details.

SEE ALSO
     NN_Recall()
```